UNIVERSITY OF NOVI SAD

FACULTY OF TECHNICAL SCIENCES

Metaheuristic approaches to the green vehicle routing problem including alternative fuel vehicles

DOCTORAL THESIS

Supervisor:
dr Tatjana Davidović

Candidate:
Luka Matijević

Novi Sad, 2024

УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

Метахеуристичке методе за проблем еколошког рутирања возила укључујући возила на алтернативни погон

ДОКТОРСКА ДИСЕРТАЦИЈА

Ментор:
др Татјана Давидовић

Кандидат:
Лука Матијевић

Нови Сад, 2024. године

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА **ОБРАЗАЦ – 5а**

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Лука Матијевић
Ментор (титула, име, презиме, звање, институција):	Др Татјана Давидовић, научни саветник, Математички институт САНУ
Наслов рада:	Metaheuristic approaches to the green vehicle routing problem including alternative fuel vehicles (срп. Метахеуристичке методе за проблем еколошког рутирања возила укључујући возила на алтернативни погон)
Језик и писмо рада:	Енглески језик Латинично писмо
Физички опис рада:	Унети број: Страница 238 Поглавља 7 Референци 392 Табела 28 Слика 22 Графикона 88 Прилога 2
Научна област:	Примењена математика
Ужа научна област (научна дисциплина):	Теоријска и примењена математика
Кључне речи / предметна одредница:	Операциона истраживања, Вишекритеријумска оптимизација, Временски зависне брзине, Еколошка логистика, Делимично пуњење батерије, Меметски алгоритам, Метода променљивих околина
Апстракт на језику рада:	Фосилна горива су, по својој природи, ограничен ресурс, што подстиче истраживаче да се усмере на возила са алтернативним погонима. Међу њима, електрична возила су привукла највећу пажњу. Њихова примена је у сталном порасту, посебно због све већих такси на емисију угљен-диоксида и подстицаја за предузећа која прелазе на алтернативне погоне. Као резултат тога, многе компаније за дистрибуцију постепено укључују електрична возила у своје возне паркове. Са порастом значаја електричних возила, проблем њиховог ефикасног рутирања постаје све актуелнији. У оквиру овог проблема, неопходно је узети у обзир различита реална ограничења, као што су капацитет возила, временски зависне брзине и меки временски оквири за услуживање корисника. Циљ је минимизовати укупну пређену дистанцу

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије докторског рада и дозвола за објављивање личних података;
5г – Изјава о коришћењу.
Ове Изјаве се чувају у институцији у штампаном и електронском облику и не кориче се са радом.

	свих возила, као и пенале који настају када се корисници услуже ван дефинисаних временских интервала. За решавање овог проблема, формулисали смо мешовити целобројни линеарни програм. Ипак, с обзиром на то да је проблем НП-тежак, проналажење оптималног решења за веће инстанце је временски изузетно захтевно. Због тога смо имплементирали седам метахеуристичких метода које могу пронаћи квалитетна решења за знатно краће време. Ове методе укључују: Методу променљивих околина, Грамзиву насумичну адаптивну претраживачку процедуру, Оптимизацију колонијом мрава (са и без локалне претраге), Оптимизацију колонијом пчела, Генетски алгоритам, Меметски алгоритам и модификовану верзију меметског алгоритма. Перформансе ових метода тестиране су на два скупа инстанци из литературе, а њихова релативна ефикасност је детаљно анализирана. Формулисали смо и две нове верзије проблема које до сада нису разматране у литератури, а које имају директну примену у индустрији. Прва верзија проширује основни проблем додавањем могућности делимичног допуњавања батерије за возила. За решавање ове верзије, претходно поменуте метахеуристичке методе су модификоване и прилагођене. Друга верзија проблема иде корак даље, уводећи мешовиту флоту која укључује комбинацију електричних возила и возила са унутрашњим сагоревањем, као и асиметричне дистанце које боље осликавају реалне услове доставе у урбаним срединама. Овај проблем је постављен у контексту вишекритеријумске оптимизације. Први циљ је минимизација дистанце коју прелазе конвенционална возила, док је други циљ комбинована минимизација укупне пређене дистанце и пенала за прекорачење временских оквира. За евалуацију ових проблема креирали смо два скупа инстанци који садрже информације о временски зависним брзинама, мешовитој флоти и асиметричним дистанцама, омогућавајући реалистичну анализу и тестирање. Ове иновације у области рутирања електричних возила и мешовитих флота отварају бројне могућности за даља истраживања, али и за практичну примену у индустрији.
Датум прихватања теме од стране надлежног већа:	15.07.2021.
Датум одбране: (Попуњава накнадно институција)	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: Др Тибор Лукић, редовни професор, Факултет техничких наука, Нови Сад Члан: Др Силвиа Гилезан, редовни професор, Факултет техничких наука, Нови Сад Члан: Др Драган Урошевић, научни саветник, Математички институт САНУ

	Члан: Др Вук Богдановић, редовни професор, , Факултет техничких наука, Нови Сад Члан: Др Јелена Иветић, ванредни професор, Факултет техничких наука, Нови Сад Ментор: Др Татјана Давидовић, научни саветник, Математички институт САНУ
Напомена:	

UNIVERSITY OF NOVI SAD
FACULTY OF TECHNICAL SCIENCES

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Luka Matijević
Supervisor (title, first name, last name, position, institution)	Dr Tatjana Davidović
Thesis title in English:	Metaheuristic approaches to the green vehicle routing problem including alternative fuel vehicles
Language and script:	English Latin script
Physical description:	Number of: Pages 238 Chapters 7 References 392 Tables 28 Illustrations 22 Graphs 88 Appendices 2
Scientific field:	Applied Mathematics
Scientific subfield (scientific discipline):	Theoretical and Applied Mathematics
Subject, Key words:	Operations research, Time-dependets speeds, Green Logistics, Multiobjective optimization, Partial recharge, Memetic algorithm, Variable Neighboruhood Search
Abstract in English:	The finite nature of fossil fuels has led researchers to focus on alternative fuel vehicles, with electric vehicles (EVs) emerging as a prominent solution. Their adoption is increasing, especially in light of projected growth in carbon taxation policies, and subsidies for alternative fuel usage. Consequently, many delivery companies are integrating EVs into their fleets. This shift underscores the growing significance of efficiently routing EV fleets, considering factors like vehicle capacity, time-dependent speeds, and soft time windows. The objective is to minimize the total distance traveled by all vehicles, along with penalties for missing time windows. To address this, we developed a mixed integer linear program. However, due to the time-consuming nature of finding exact solutions for this NP-hard problem, we introduced seven metaheuristic algorithms as practical alternatives. These include General Variable Neighborhood Search, Greedy Randomized

[2] The author of the doctoral dissertation has signed the following Statements:
5б – Statement on the authorship,
5в – Statement that the printed and e-version of the doctoral dissertation are identical and authorization to use personal data,
5г – Copyright statement.
The paper and e-versions of Statements are held at the institution and are not included into the printed thesis.

	Adaptive Search Procedure, Ant Colony Optimization (with and without Local Search), Bee Colony Optimization, Genetic Algorithm, Memetic Algorithm, and Modified Memetic Algorithm. We evaluated these methods using two sets of benchmark instances, providing comprehensive results on their relative effectiveness. Additionally, we formulated two novel versions of this problem with direct industrial applications, yet unexplored in literature. The first version incorporates the option of partial battery recharging at alternative fuel stations into the problem described earlier. We adapted the aforementioned metaheuristics for this scenario, with some modifications that allow them to tackle this new type of problem. The second version we introduced is more aligned with real-world conditions, adding a heterogeneous fleet of electric and conventional vehicles and considering asymmetric distances, which better represent urban delivery scenarios. This problem was tackled through multi-objective optimization, aiming to minimize the total distance traveled by conventional vehicles and the combined distance and penalty from time window deviations for all vehicles. For this version, we also created two new sets of benchmark instances that include time-dependent speeds and a heterogeneous vehicle fleet. These innovations in routing electric and mixed vehicle fleets open new avenues for research and practical applications in efficient and sustainable logistics.
Date of endorsement by the scientific board:	15.07.2021.
Date of defence: (Filled in by the institution)	
Thesis defence board: (title, first name, last name, position, institution)	Chair: Tibor Lukić PhD, Full Professor, Faculty of Technical Sciences, Novi Sad Member: Silvia Ghilezan PhD, Full Professor, Faculty of Technical Sciences, Novi Sad Member: Dragan Urošević PhD, Research Professor, Mathematical Institute of the Serbian Academy of Sciences and Arts Member: Vuk Bogdanović PhD, Full Professor, Faculty of Technical Sciences, Novi Sad Member: Jelena Ivetić PhD, Assistant Professor, Faculty of Technical Sciences, Novi Sad Supervisor: Tatjana Davidović PhD, Research Professor, Mathematical Institute of the Serbian Academy of Sciences and Arts
Note:	

LIST OF ABBREVIATIONS

A	**A**symmetry
ABC	**A**rtificial **B**ee **C**olony
ACO	**A**nt **C**olony **O**ptimization
ACOLS	**A**nt **C**olony **O**ptimization with **L**ocal **S**earch
ACS	**A**nt **C**olony **S**ystem
AFS	**A**lternative **F**uel **S**tation
AFV	**A**lternative **F**uel **V**ehicles
AI	**A**rtificial **I**ntelligence
B	**B**ackhaul
BCO	**B**ee **C**olony **O**ptimization
BCOi	**I**mproving **BCO**
BEV	**B**attery **E**lectric **V**ehicles
BS	**B**attery **S**wapping
C	**C**apacity constraint
CC-CV	**C**onstant **C**urrent–**C**onstant **V**oltage
CW	**C**lark and **W**right Savings Algorithm
CVRP	**C**apacitated **V**ehicle **R**outing **P**roblem
D	**D**ynamic requests
DP	**D**ynamic **P**rogramming
ECR	**E**stimated **C**onsumption **R**ate
EVRP	**E**lectric **V**ehicle **R**outing **P**roblem
FA	**F**irefly **A**lgorithm
FC	**F**uel **C**onsumption (minimization)
FD	**F**lexible **D**eliveries
FDR	**F**alse **D**iscovery **R**ate
FWER	**F**amily**w**ise **E**rror **R**ate
GA	**G**enetic **A**lgorithm
GE	**G**radient **E**volution
GHG	**G**reen**h**ouse **g**as
GRASP	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
GSA	**G**reedy **S**earch **A**lgorithm
GVNS	**G**eneral **V**ariable **N**eighborhood **S**earch
GVRP	**G**reen **V**ehicle **R**outing **P**roblem
HACS	**H**ybrid **A**nt **C**olony **S**ystem
HEV	**H**ybrid **E**lectric **V**ehicles
HF	**H**eterogeneous **F**leet
IBKA	**I**mproved **B**alanced **K**-means **A**lgorithm
ICE	**I**nternal **C**ombustion **E**ngine

ICV	**I**nternal **C**ombustion **V**ehicle
ICVU	**ICV U**sage (minimization)
ILS	**I**terated **L**ocal **S**earch
LCM	**L**inear **C**onsumption **M**odel
LNS	**L**arge **N**eighborhood **S**earch
MA	**M**emetic **A**lgorithm
MB	**M**ixed **B**ackhauls
MC	**M**ultiple **C**ompartments
MD	**M**ultiple **D**epots
MILP	**M**ixed **I**nteger **L**inear **P**rogram
MIMOA	**M**embrane-**I**nspired **M**ulti-**O**bjective **A**lgorithm
ML	**M**achine **L**earning
MMA	**M**odified **M**emetic **A**lgorithm
MO	**M**ultiple **O**bjectives
MOHF-VRP-STW	**M**ulti-**O**bjective **H**eterogeneous **F**leet **V**ehicle **R**outing **P**roblem with **S**oft **T**ime **W**indows
MSCC	**M**ulti-**S**tage **C**onstant **C**urrent
MSLS	**M**ulti-**s**tart **L**ocal **S**earch
MT	**M**ultiple **T**rips
MTZ	**M**iller–**T**ucker–**Z**emlin inequalities
MXO	**Max**imum **O**vertime (minimization)
NN	**N**earest **N**eighbor Algorithm
NSGA-II	**N**on-dominated **S**orting **G**enetic **A**lgorithm-**II**
NV	**N**umber of **V**ehicles (minimization)
O	**O**pen routes
OC	**O**peration **C**ost (minimization)
OR	**O**perations **R**esearch
OX	**O**rder **C**rossover
PBA	**P**artition-**B**ased **A**lgorithm
PC	**P**recedence **C**onstraints
PD	**P**ickup and **D**elivery
PDP	**P**artial **D**ependence **P**lot
PE	**P**ollutant **E**mission reduction
PM	**P**articulate **M**atter
PR	**P**artial **R**echarge
PRP	**P**ollution **R**outing **P**roblem
QLC	**Q**uality **L**oss **C**ost (minimization)
QT	**Q**ueuing **T**ime (minimization)
RC	**R**echarging **C**ost (minimization)
RCL	**R**estricted **C**andidate **L**ist
RI	**R**oute **I**mbalance (minimization)
RM	**R**evenue **M**aximization
RT	**R**echarging **T**ime (minimization)
SD	**S**plit **D**eliveries
SL-PSO	**S**elf **L**earning **PSO**
SMG-MOMA	**S**kin **M**embrane **G**uided **M**ulti-**O**bjective **M**embrane **A**lgorithm
SoC	**S**tate of **C**harge
TD	**T**ime-**d**ependent speeds
TD-EVRP-STW	**T**ime-**D**ependent **E**lectric **V**ehicle **R**outing **P**roblem with **S**oft **T**ime **W**indows
TDPR-EVRP-STW	**T**ime-**D**ependent **P**artial **R**echarge **E**lectric **V**ehicle

	Routing **P**roblem with **S**oft **T**ime **W**indows
TDM	**T**otal **D**istance **M**inimization
TOPSIS	**T**echnique for **O**rder **P**reference by **S**imilarity to **I**deal **S**olution
TPX	**T**wo-**p**oint **C**rossover
TSACS	**T**wo-**s**tage **A**nt **C**olony **S**ystem
TSP	**T**raveling **S**alesman **P**roblem
TT	**T**ravel **T**ime (minimization)
TW	**T**ime **W**indows
TWPC	**T**ime **W**indows **P**enalty **C**ost (minimization)
VNB	**V**ariable **N**eighborhood **B**ranching
VND	**V**ariable **N**eighborhood **D**escent
VNS	**V**ariable **N**eighborhood **S**earch
VOC	**V**olatile **O**rganic **C**ompounds
VRP	**V**ehicle **R**outing **P**roblem
VRPRL	**VRP** in **R**everse **L**ogistics
WOA	**W**hale **O**ptimization **A**lgorithm
WT	**W**aiting **T**ime (minimization)
XGBoost	**E**xtreme **G**radient **Boost**ing

UNIVERSITY OF NOVI SAD

Abstract

Faculty of Technical Sciences
Department of Fundamentals Sciences

Doctor of Science - Applied Mathematics

Metaheuristic approaches to the green vehicle routing problem including alternative fuel vehicles

by Luka Matijević

The finite nature of fossil fuels has led researchers to focus on alternative fuel vehicles, with electric vehicles (EVs) emerging as a prominent solution. Their adoption is increasing, especially in light of projected growth in carbon taxation policies, and subsidies for alternative fuel usage. Consequently, many delivery companies are integrating EVs into their fleets.

This shift underscores the growing significance of efficiently routing EV fleets, considering factors like vehicle capacity, time-dependent speeds, and soft time windows. The objective is to minimize the total distance traveled by all vehicles, along with penalties for missing time windows. To address this, we developed a mixed integer linear program. However, due to the time-consuming nature of finding exact solutions for this NP-hard problem, we introduced seven metaheuristic algorithms as practical alternatives. These include General Variable Neighborhood Search, Greedy Randomized Adaptive Search Procedure, Ant Colony Optimization (with and without Local Search), Bee Colony Optimization, Genetic Algorithm, Memetic Algorithm, and Modified Memetic Algorithm. We evaluated these methods using two sets of benchmark instances, providing comprehensive results on their relative effectiveness.

Additionally, we formulated two novel versions of this problem with direct industrial applications, yet unexplored in literature. The first version incorporates the option of partial battery recharging at alternative fuel stations into the problem described earlier. We adapted the aforementioned metaheuristics for this scenario, with some modifications that allow them to tackle this new type of problem. The second version we introduced is more aligned with real-world conditions, adding a heterogeneous fleet of electric and conventional vehicles and considering asymmetric distances, which better represent urban delivery scenarios. This problem was tackled through multi-objective optimization, aiming to minimize the total distance traveled by conventional vehicles and the combined distance and penalty from time window deviations for all vehicles. For this version, we also created two new sets of benchmark instances that include time-dependent speeds and a heterogeneous vehicle fleet.

These innovations in routing electric and mixed vehicle fleets open new avenues for research and practical applications in efficient and sustainable logistics.

УНИВЕРЗИТЕТ У НОВОМ САДУ

Резиме

Факултет техничких наука
Департман за опште дисциплине у техници

Доктор наука - Примењена математика

Метахеуристичке методе за проблем еколошког рутирања возила укључујући возила на алтернативни погон

Лука Матијевић

Мотивација и основни термини

Проблем еколошког рутирања возила постаје све значајнији, посебно због растуће забринутости у вези са емисијама гасова са ефектом стаклене баште и њиховог утицаја на околину. Многе државе су преузеле обавезу да смање емисије ових гасова, посебно угљен-диоксида (CO_2), што се практично манифестује кроз субвенције компанијама које инвестирају у алтернативне изворе енергије, подршку компанијама које користе возила на алтернативни погон, као што су електрична возила (BEV) и хибридна возила (HEV), као и увођењем пореза на емисије CO_2. Као резултат ових политика, све више дистрибутивних компанија почиње да интегрише возила на алтернативни погон у своје возне паркове. Осим тога, електрична возила премештају загађење из градских подручја и смањују буку, чинећи их идеалним опцијама за испоруку робе до крајњих потрошача.

Постоји велики број возила на алтернативни погон, укључујући електрична возила, хибридна возила, возила на водоник, возила која користе биогорива попут етанола или биодизела, синтетичка горива, возила на соларни погон, течни азот, па чак и возила која користе компресовани ваздух. Сваки тип возила поседује јединствене карактеристике које их чине више или мање погодним за различите примене.

У фокусу ове дисертације биће електрична возила. Она се издвајају употребом пуњивих батерија које напајају електрични мотор, уместо традиционалног мотора на унутрашње сагоревање. Главна предност електричних возила лежи у њиховој способности да током вожње не емитују штетне гасове, уз знатно нижи ниво буке у поређењу са конвенционалним возилима. Међутим, електрична возила суочавају се с одређеним ограничењима, првенствено када је реч о капацитету батерије који директно утиче на њихов домет. Осим тога, потребно је узети у обзир и време потребно за пуњење батерије, што је веома значајан фактор приликом планирања рута за путовање.

Проблем рутирања возила (енг. Vehicle Routing Problem - VRP) први пут је формулисан 1959. године [61]. Због његове кључне улоге у логистици, проблем је привукао значајну пажњу истраживача. VRP представља генерализацију проблема трговачког путника, захтевајући одлуке о томе које возило ће посетити којег корисника и у којем редоследу. Овај проблем спада у категорију НП-тешких проблема [186], што значи да је проналажење оптималног решења често неостваривo у разумном временском оквиру.

С обзиром на све веће прихватање електричних возила од стране дистрибутивних компанија, проблем еколошког рутирања возила добија на значају и постаје предмет све веће пажње како у академским круговима тако и у индустрији. Еколошко рутирање возила има за циљ планирање рута узимајући у обзир еколошке факторе транспорта, попут емисије штетних гасова или других облика загађења. Ови проблеми се могу класификовати у три основне категорије: проблем рутирања загађења (енг. Pollution Routing Problem - PRP), који се фокусира на смањење емисија гасова при употреби возила на унутрашње сагоревање (енг. Internal Combustion Vehicles – ICVs); проблем зеленог рутирања возила (енг. Green Vehicle Routing Problem - GVRP), који разматра коришћење возила на алтернативне погоне; и проблем рутирања возила у обрнутој логистици (енг. VRP in Reverse Logistics - VRPRL), који се бави оптимизацијом прикупљања отпада и рециклажом. Посебан фокус ове дисертације биће на групи проблема везаних за рутирање електричних возила (енг. Electric Vehicle Routing Problem - EVRP), као подскупу проблема зеленог рутирања возила.

EVRP, заједно са основним VRP-ом, припада пољу операционих истраживања. Ова научна дисциплина се посвећује развијању метода за побољшање процеса доношења одлука, посебно у контекстима где су ресурси ограничени, попут ограниченог времена за извршавање метода или ограниченог меморијског простора. Поред проблема рутирања возила, операциона истраживања обухватају широк спектар проблема, укључујући распоређивање задатака, проблем трговачког путника, управљање ланцем снабдевања, теорију редова чекања, управљање залихама, као и бројне друге изазове који се јављају у индустрији.

Оптимизациони проблеми представљају темељ операционих истраживања. Они подразумевају идентификацију оптималних вредности за специфичан скуп параметара с циљем остваривања дефинисаног циља, који је формулисан кроз функцију циља. У контексту минимизације, циљ је пронаћи оне вредности параметара које резултују најмањом могућом вредношћу функције циља. Супротно, у процесу максимизације, траже се параметри који максимизирају вредност функције циља. Формулација минимизационих проблема може се извршити на следећи начин:

$$\min_{x \in S} f(x)$$

где је функција циља задата са $f : \mathbb{R}^n \to \mathbb{R}$, а $S \subseteq \mathbb{R}^n$ представља скуп свих допустивих решења. Уколико постоји тачка x^* за коју важи

$$f(x^*) \leq f(x), \ \ \forall x \in S$$

онда ту тачку називамо глобални оптимум. Са друге стране, уколико у некој околини $\mathcal{N}(\bar{x})$ постоји тачка x^* за коју важи

$$f(x^*) \leq f(x), \ \ \forall x \in \mathcal{N}(\bar{x}) \cap S$$

тада тачку x^* називамо локалним оптимумом у околини $\mathcal{N}(\bar{x})$.

Различити приступи решавању оптимизационих проблема могу се класификовати у две основне категорије: тачне методе и приближне методе. Тачне методе гарантују проналазак глобалног оптимума под условом да имају довољно времена и ресурса за извршавање. Међу овим методама издвајају се динамичко програмирање и мешовито целобројно линеарно програмирање (енг. Mixed-integer linear programming – MILP). С друге стране, због ограничења у времену које намећу тачне методе, у пракси се често користе приближне методе. Ове методе пружају довољно добра решења у знатно краћем временском периоду, иако не гарантују оптималност. Приближне методе укључују хеуристичке методе, метахеуристичке методе и апроксимационе алгоритме. Хеуристичке методе примењују специфична знања о проблему како би што ефикасније претраживале простор решења. Метахеуристичке методе, с друге стране, су општије природе и нису специфично дизајниране за одређени проблем, што их чини флексибилним алатом применљивим на широк спектар проблема. Метахеуристике често користе хеуристичке алгоритме за унапређење својих решења, а такође се могу комбиновати и са тачним методама, стварајући на тај начин хибридне методе. Што се тиче апроксимационих алгоритама, они су способни да у кратком временском периоду пруже решења са гарантованим нивоом квалитета, иако тај квалитет можда неће бити еквивалентан оптималном решењу.

Доприноси

У овој дисертацији дефинисана су три проблема релевантна за индустрију. Први проблем се бави рутирањем електричних возила са флексибилним временским оквирима и временски зависним брзинама. Друга верзија проширује овај проблем тако што омогућава делимично пуњење батерија на станицама, како би се оптимално управљало временом и корисници били опслужени на време. Трећа верзија представља додатно проширење, укључујући флоту која обухвата и конвенционална возила, што одражава најреалнији сценарио. Поред тога, разматране су асиметричне дистанце између локација, карактеристичне за урбане средине, као и вишекритеријумска оптимизација за овај проблем.

За први проблем предложен је математички модел. За сваки од проблема тестирано је осам метахуристичких метода, спроведена је статистичка анализа резултата, а методе су поређене на основу више критеријума.

Поред ових доприноса, састављена је и најкомплетнија листа метахеуристика до сада, при чему је свака метахеуристика класификована на основу инспирације. Такође, за тестирање треће верзије проблема било је неопходно креирати нове инстанце, које су јавно доступне на адреси `https://www.mi.sanu.ac.rs/~luka/resources/Instances.zip`. Коначно, развијено је неколико помоћних метода (нпр. за креирање почетних решења, додавање посета станицама или одређивање распореда полазака возила), које могу бити корисне и у будућим алгоритмима.

Из ове дисертације произашла су два рада објављена у часописима. Први рад, објављен у часопису YUJOR, представља прегледни рад на тему метахеуристичких приступа еколошком рутирању возила [221], док је други рад објављен у часопису International Journal of Industrial Engineering Computations и бави се методом променљивих околина примењеном на проблем рутирања електричних возила [220]. Иако није настао као део дисертације, рад [224] је такође тематски повезан с њом.

Метахеуристичке методе

Постоји више начина класификације метахеуристика, како је описано у литератури [91, 319]. Једна од првих класификација, представљена од стране Birattari et al. [34], категоризује метахеуристике на основу шест различитих критеријума:

- Праћење путање - Методе које прате путању систематично истражују простор претраге, постепено модификујући решење, док методе које не прате путању омогућавају велике скокове према различитим деловима простора претраге.
- Број решења - Популационе методе истражују више решења паралелно, у контрасту с другим метахеуристикама које у сваком моменту задржавају и побољшавају само једно одређено решење.
- Коришћење меморије - Метахеуристике које користе меморију користе информације о претходним решењима како би утицале на будуће правце претраге, док метахеуристике без меморије функционишу независно од претходних решења.
- Број структура околина - Метахеуристике са једном структуром околине ослањају се на стални скуп правила за претраживање простора претраге, док оне са више структура динамички бирају различите скупове правила.
- Тип функције циља - Метахеуристике са динамичким функцијама циља прилагођавају своје критеријуме у зависности од променљивих услова, док оне са статичким функцијама циља константно теже ка фиксном скупу критеријума током целог процеса оптимизације.
- Инспирација - Метахеуристике инспирисане природом су алгоритми моделовани на основу природних феномена и биолошких понашања, док су метахеуристике које нису инспирисане природом развијене из математичких или алгоритамских принципа, без директног ослањања на природне процесе.

Ниједна метахеуристика универзално не превазилази остале за све типове проблема [362]. Примећујемо значајан раст броја нових метахеуристика које се уводе сваке године, посебно након 2000. У оквиру овог истраживања, успели смо да идентификујемо 540 различитих метахеуристика, које су представљене у табели `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixA.pdf`, мада је важно напоменути да листа вероватно није комплетна. Анализом ових метахеуристика, уочава се да је већина инспирисана понашањем животиња, док су физички процеси други најчешћи извор инспирације.

У наставку ће бити описане неке од метахеуристика које су од значаја за ову дисертацију.

Метода променљивих околина

Метода променљивих околина (енг. Variable Neighborhood Search - VNS), како је представљена у [236, 238], представља итеративну метахеуристику која се ослања на систематично истраживање околине једног решења. Иницијално решење се добија помоћу неког хеуристичког алгоритма, након чега VNS методологија примењује циклус од три корака у свакој итерацији: размрдавање, локално претраживање и померање. Размрдавање доприноси диверзификацији претраживања, бирајући насумично ново решење x унутар околине $\mathcal{N}_k(x)$, где x означава тренутно најбоље решење, чиме се избегава заглављивање у локалном оптимуму. Локално претраживање се затим примењује на x с циљем унапређења решења. У фази померања, упоређује се унапређено решење x са тренутно најбољим решењем, и ако је ново решење боље, оно постаје референтно за даљу претрагу која се ресетује на прву околину. У супротном, прелази се на следећу околину. Број околина је ограничен параметром k_{max}, док алгоритам може укључити и друге параметре попут k_{min}, који означава почетну околину, или k_{step}, који дефинише величину корака за прелазак на следећу околину. Постоје различите варијанте VNS-а, неке од којих су детаљније описане у [126].

Похлепни насумично адаптивни поступак претраге

Похлепни насумично адаптивни поступак претраге (енг. Greedy Randomized Adaptive Search Procedure - GRASP), предложен од стране Feo and Resende [93], представља ефикасну итеративну метахеуристику која се састоји од две основне фазе: конструктивне фазе и фазе локалног претраживања. Током конструктивне фазе, GRASP итеративно гради допустиво решење, комбинујући похлепне принципе са елементима случајности. Ово се реализује формирањем ограничене листе кандидата (енг. Restricted Candidate List - RCL), која укључује кандидате са најбољим перформансама према одабраном критеријуму, обично функцији циља. Величина ове листе одређује се параметром α, који може дефинисати и квалитативни праг за укључивање кандидата у RCL. Након конструкције допустивог решења, примењује се локално претраживање с циљем његовог додатног побољшања. Ако је резултат локалне претраге бољи од тренутно најбољег решења, оно се ажурира и постаје нови референтни оптимум.

Оптимизација колонијом мрава

Оптимизација колонијом мрава (енг. Ant Colony Optimization - ACO) [74, 76, 77], представља метахеуристички приступ инспирисан начином на који мрави трагају за храном. Ова метода припада групи популационих метахеуристика, што значи да у свакој итерацији истражује више независних решења. Сваки "мрав" у алгоритму креира своје решење итеративно, ослањајући се на хеуристичке информације и трагове феромона остављене на путу. При конструисању решења, за сваки потенцијални корак од тачке i до тачке j, израчунава се вероватноћа p_{ij} (Формула 1), која зависи од вредности феромона τ_{ij} на тој путањи и хеуристичке информације η_{ij}, при чему су S непосећене тачке које се могу укључити у решење. Параметри α и β служе за одређивање релативног утицаја хеуристичких информација и феромонског трага.

$$p_{ij} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{k \in S}\tau_{ik}^{\alpha}\eta_{ik}^{\beta}}, \quad \forall j \in S \tag{1}$$

Након што сваки мрав формира и оцени своје решење користећи функцију циља, алгоритам приступа ажурирању феромонских трагова. У фази испаравања, концентрација феромона се редукује за одређени проценат, што се постиже применом формуле 2. Параметар ρ у овој формули представља стопу испаравања, а $\mathcal{P}$ означава скуп свих тачака. Овај процес је кључан за спречавање превремене конвергенције ка локалном оптимуму. После фазе испаравања, следи фаза појачања феромона, где се повећава концентрација феромона на путањама изабраних на основу резултата које су мрави остварили. Различите стратегије могу се применити у фази појачања; на пример, могуће је да сви мрави доприносе ажурирању феромона или да само одабрани број мрава са најбољим решењима врши ажурирање.

$$\tau_{ij} = (1-\rho)\tau_{ij}, \quad \forall i,j \in \mathcal{P} \tag{2}$$

Оптимизација колонијом пчела

Оптимизација колонијом пчела (енг. Bee Colony Optimization – BCO) [202, 203, 204] представља метахеуристику инспирисану понашањем пчела у природи, посебно њиховим методама трагања за изворима хране и међусобном комуникацијом. BCO метода обухвата две главне фазе у свакој итерацији:

- Лет унапред - У овом кораку, пчеле истражују простор претраге тако што извршавају одређен број корака, у којима се решење или мења или конструише, чиме се креирају нова комплетна или парцијална решења.
- Лет уназад - У овом кораку, пчеле деле информације о квалитету својих решења. Свака пчела пробабилистички бира да ли ће остати лојална свом решењу, у зависности од његовог квалитета. Ако пчела остане лојална свом решењу, она ће почети да га промовише. С друге стране, пчеле које нису остале лојалне свом решењу морају изабрати једно од промовисаних решења на пробабилистички начин, при чему се квалитетнија решења фаворизују.

Постоје две основне верзије BCO методе:

- Конструктивни BCO - У овој верзији, свака пчела користи хеуристичке информације како би на стохастички начин креирала комплетно решење.
- Модификациони BCO (BCOi) - У овој верзији, свака пчела већ има комплетно решење које мења примењујући на њега одређен број корака, чиме се креира ново решење.

Генетски и меметски алгоритми

Генетски алгоритми (енг. Genetic Algorithms – GA) [137] су алгоритми који су засновани на принципима еволуције. Ови алгоритми користе еволутивне механизме попут наслеђивања, мутације, селекције и укрштања. Основна верзија алгоритма представља скуп решења у облику хромозома. У свакој итерацији алгоритма, на хромозоме се примењују одређени оператори, као што су:

- Кодирање и декодирање - Ова операција укључује представљање решења у облику хромозома.
- Селекција - Селекција подразумева избор хромозома из којих ће се креирати нове јединке. Уобичајено је да је вероватноћа избора хромозома пропорционална квалитету решења које они представљају.
- Укрштање - Након што се одаберу одговарајући хромозоми, на њих се примењује оператор укрштања који ствара нове хромозоме. Постоји много врста оператора укрштања, а избор одговарајућег оператора може имати значајан утицај на квалитет креираних решења.
- Мутација - Након оператора укрштања, на креиране јединке примењује се оператор мутације са одређеном вероватноћом. Овај оператор благо мења решење на насумичан начин, чиме се диверсификује популација.
- Замена јединки - Овај оператор одређује које ће јединке бити пренете у нову генерацију.

Меметски алгоритми (енг. Memetic Algorithms – MA) представљају модификацију генетских алгоритама која комбинује еволутивне аспекте GA са локалном претрагом. На сваку јединку у популацији примењује се локална претрага, чиме се потенцијално повећава квалитет целе популације.

Проблем еколошког рутирања возила

Проблем рутирања возила (енг. Vehicle Routing Problem – VRP) представља широко истраживан проблем у области операционих истраживања, први пут предложен 1959. године [61]. Циљ VRP-а је да пронађе оптималне (или бар довољно добре) руте којима ће се кретати скуп возила како би опслужили све кориснике, узимајући у обзир ограничења специфична за сваки проблем, која називамо атрибутима. VRP спада у класу НП-тешких проблема [186], што га чини неподесним за решавање егзактним методама у случајевима већих инстанци. Због тога се за решавање овог проблема најчешће користе хеуристичке или метахеуристичке методе.

Проблем еколошког рутирања возила (енг. Green Vehicle Routing Problem – GVRP) представља надоградњу оригиналног VRP-а, с тим што додатно узима у обзир еколошке аспекте рутирања, пре свега емисије CO_2 и заштиту животне средине. У овој дисертацији пре свега се бавимо специфичном подкатегоријом овог проблема, која се може назвати problem rutiranja električnih vozila (енг. Electric Vehicle Routing Problem – EVRP), иницијално предложеном у раду [86]. Овај проблем имплицитно адресира смањење емисија CO_2 коришћењем електричних возила (EV). Међутим, употреба EV носи са собом одређена ограничења, као што су ограничен капацитет батерије (а самим тим и домета возила), као и дуже време потребно за пуњење батерије. Након одређене пређене дистанце, возило мора посетити станицу за пуњење (енг. Alternative Fuel Station – AFS) и задржати се тамо одређено време. Ово време задржавања мора се узети у обзир приликом планирања рута, како би се корисници опслужили на време, што директно утиче на задовољство корисника.

Постоји много атрибута који се могу узети у обзир за овај проблем. У овом делу издвојићемо само неке од најважнијих за ову дисертацију:

- Ограничење капацитета возила (C) - Може се задати у виду тежине, запремине или броја објеката који се могу превозити у возилу.
- Асиметричност дистанци (A) - Овај атрибут описује сценарио у којем растојање између тачке А и тачке Б није исто као растојање од тачке Б до тачке А.
- Временски оквири (TW) - Временски оквири представљају интервале времена у којима одређени корисници морају бити посећени. Могу бити стриктни (тврди) и флексибилни (меки). У случају стриктних временских оквира, посета корисника ван задатих интервала није дозвољена. Међутим, у случају флексибилних временских оквира, посете корисницима изван оквира су могуће, али уз одређене пенале.
- Хетерогена флота (HF) - Овај атрибут означава ситуацију у којој возила која чине флоту могу имати различите карактеристике, као што су капацитет или брзина. У контексту GVRP-а, овај атрибут такође може указивати на ситуацију где нека возила користе мотор на унутрашње сагоревање, док друга користе алтернативни погон, као што су електрична возила.

- Временски зависне брзине (TD) - У реалним ситуацијама, посебно у урбаним срединама, брзина возила није константна и зависи од више фактора, укључујући и доба дана. TD атрибут моделује овај сценарио тако што се сваком добу дана додељује одређени мултипликатор просечне брзине возила.
- Парцијално пуњење батерије (PR) - Када EV посети AFS, оно не мора обавезно напунити батерију до пуног капацитета како би се избегло непотребно чекање. Одлука о томе колико напунити батерију при свакој посети AFS-а додаје додатну комплексност проблему и описана је PR атрибутом.
- Вишекритеријумска оптимизација (MO) - MO атрибут омогућава оптимизацију више независних циљева истовремено. Овај приступ оптимизацији познат је и као Парето-оптимизација. За разлику од ситуације када се оптимизује само један циљ, где је потребно пронаћи само једно најбоље решење, у случају MO трага се за скупом најбољих Парето-оптималних решења. Свако решење из овог скупа најквалитетније је у контексту бар једног циља.

GVRP може оптимизовати више различитих циљева, међу којима су: минимизација утрошеног горива или енергије, минимизација емисија штетних гасова, минимизација пенала због пропуштених временских оквира, минимизација трошкова транспорта, минимизација броја возила, минимизација броја возила са мотором на унутрашње сагоревање, као и многи други циљеви.

Проблем рутирања електричних возила

Први проблем који се разматра у овој дисертацији је проблем рутирања електричних возила, који узима у обзир капацитете возила, меке временске оквире, као и временски зависне брзине возила (TD-EVRP-STW). Сва возила су идентична и полазе из истог депоа. Када EV посети AFS, његова батерија пуни се до краја, те се време задржавања на AFS рачуна у зависности од количине енергије која се може допунити и брзине пуњења задате за сва возила. Претпоставка је да се батерија пуни линеарно. Иако ова претпоставка није реалистична у стварним условима, ово поједностављење не утиче на релативне перформансе метахеуристика, што је примарни фокус ове тезе. Такође, у стварним условима, потрошња енергије зависи од више фактора, као што су пређена дистанца, ефикасност мотора, нагиб пређеног пута, временски услови, оптерећење возила, као и многи други. Ови фактори нису разматрани у оквиру ове тезе, већ је проблем поједностављен претпоставком да потрошња зависи само од пређене дистанце и просечне потрошње по јединици дистанце. Оправдање за ово поједностављење је исто као и за претпоставку о линеарном пуњењу батерије, тј. нема значајног утицаја на релативне перформансе метахеуристика.

Циљ овог проблема је двострук: минимизација пређене дистанце и минимизација пенала узрокованих опслуживањем корисника ван изабраних временских оквира. Ова два критеријума спојена су у јединствену метрику коришћењем пондерисане суме оба критеријума.

Овај проблем смо формулисали као MILP, који је затим тестиран користећи комерцијални решавач CPLEX. Као што је било очекивано, егзактном решавачу попут CPLEX-а било је потребно доста времена да реши чак и једноставније инстанце овог проблема, што додатно оправдава употребу метахеуристика.

Решења овог проблема представљена су као низ дупло повезаних листи, где свака листа представља руту која одговара једном возилу. Сваки елемент листе означава локацију коју возило треба да посети. Свака локација обухвата четири информације: индекс корисника или станице за пуњење, време доласка на ту локацију, време одласка са те локације и тип локације (корисник или AFS).

Уколико имамо решење у којем неко возило остане без батерије пре него што се врати у депо, такво решење је недопустиво. Одбацивање таквих решења може бити лоша стратегија, јер се тиме лишавамо многих обећавајућих решења. Уместо тога, ако се суочимо са решењем које је недопустиво због празне батерије, покушавамо да га поправимо тако што у руту убацујемо посете AFS пре критичне локације. То чинимо итеративно прегледајући све позиције у рути уназад, почевши од оне на којој је батерија постала празна. Ако је могуће, додајемо посету AFS и прерачунавамо времена доласка и одласка за све наредне локације. Ако то није могуће, прелазимо на претходну позицију. Овај процес настављамо док решење не постане прихватљиво или док не закључимо да се решење не може поправити, у ком случају га одбацујемо. Овај алгоритам називамо $FIX\_SOLUTION$.

За креирање почетних решења дефинисане су две хеуристике:

- $INIT\_SOLUTION\_RANDOM$ - Све док постоје нераспоређени корисници, на насумичан начин бирају се нераспоређени корисник и возило, а затим се тај корисник додаје на крај руте изабраног возила. Након што се сви корисници распореде, додају се посете AFS коришћењем методе $FIX\_SOLUTION$.
- $INIT\_SOLUTION$ - За све нераспоређене кориснике израчунава се најбоља позиција у свим рутама на коју их је могуће додати, тако да се одабере корисник чијим додавањем долази до најмање промене функције циља. Тај корисник се потом додаје на пронађену позицију. Овај процес се понавља све док постоје нераспоређени корисници. Након што су сви корисници распоређени, додају се посете AFS користећи методу $FIX\_SOLUTION$.

С обзиром на то да је дозвољено да возило сачека на некој локацији најповољнији тренутак за полазак ка следећој дестинацији, неопходно је израчунати време доласка возила на локацију и одредити када ће отпутовати. У ту сврху, дефинисали смо два алгоритма: $CALC\_SCHEDULE$ и $OPTIMIZE\_SCHEDULE$, које обично позивамо један за другим. У методу $CALC\_SCHEDULE$ рачунамо долазак и одлазак узимајући у обзир само време проведено у вожњи, време опслуживања корисника и време проведено на свакој AFS. Након тога, примењујемо $OPTIMIZE\_SCHEDULE$, који проверава да ли је возило стигло на локацију пре почетка њеног временског оквира. Да би се избегли пенали, ова метода мења време поласка са претходне локације, тако да долазак на тренутну локацију буде тачно на време. Овако израчунато време доласка и одласка није оптимално, али предложена хеуристика се брзо извршава и као таква, погодна је за укључивање у метахеуристике.

Као методу локалне претраге у свим метахеуристикама користимо методу променљивог спуста (енг. Variable Neighborhood Descent – VND). VND започиње с почетним решењем и одређеним скупом околина. Претрага почиње од прве околине, где се покушава побољшати почетно решење. Ако се побољшање не пронађе, прелази се на следећу околину. У сваком тренутку, ако се пронађе побољшање, претрага се враћа на прву околину. У нашој верзији методе користи се седам околина, од којих су четири базиране на корисницима, а три на AFS станицама. Те околине укључују:

- $\mathcal{N}_2(S)$ - Заснована на оператору који пребацује корисника на друго место у истој рути.
- $\mathcal{N}_3(S)$ - Заснована на оператору који пребацује корисника на неко место у другој рути.
- $\mathcal{N}_4(S)$ - Заснована на оператору који мења места два корисника унутар исте руте.
- $\mathcal{N}_5(S)$ - Заснована на оператору који мења места два корисника из различитих рута.
- $\mathcal{N}_6(S)$ - Заснована на оператору који избацује AFS из руте.
- $\mathcal{N}_7(S)$ - Заснована на оператору који помера AFS на друго место у рути.
- $\mathcal{N}_8(S)$ - Заснована на оператору који замењује једну AFS станицу са другом.

Након анализе утицаја сваке од ових околина на квалитет решења, закључено је да околине $\mathcal{N}_2(S)$, $\mathcal{N}_3(S)$ и $\mathcal{N}_4(S)$ имају највећи утицај. Такође је тестирана додатна околина $\mathcal{N}_1(S,\epsilon)$, заснована на оператору који мења време одласка са локације за задату вредност ϵ. Међутим, због њеног малог утицаја на квалитет решења, није даље коришћена у тестирању.

У нашем случају, VND користи принцип првог побољшања (енг. First improvement), који не захтева проверу свих суседа из одређене околине, већ се претрага околине завршава чим се наиђе на прво побољшање. Избор овог принципа извршен је због комплексности околина које се користе у VND методи.

У оквиру ове дисертације разматрано је осам метахеуристика за TD-EVRP-STW.

Прва метахеуристика коришћена за решавање овог проблема је генерална метода променљивих околина (енг. General Variable Neighborhood Search – GVNS), која се разликује од основне верзије, описане раније, тиме што користи VND као методу локалне претраге. У кораку размрдавања коришћене су две околине:

- $\mathcal{N}_9(S,l)$ - Заснована на оператору који пребацује l узастопних локација (независно од типа локације) на другу позицију унутар исте руте.
- $\mathcal{N}_{10}(S,l)$ - Заснована на оператору који пребацује l узастопних локација (независно од типа локације) на неку позицију унутар друге руте.

Избор околине, као и параметар l, бирају се насумично у сваком кораку процедуре размрдавања.

Друга коришћена метахеуристика је GRASP. Наша формулација GRASP метахеуристике прати основну верзију коју смо раније описали. Метода за конструкцију решења функционише на следећи начин: у првом кораку се рачуна цена уметања сваког нераспоређеног корисника на сваку могућу позицију у решењу. У другом кораку, бира се предефинисан број најбољих елемената за додавање, од којих се насумично бира један. Овај поступак се понавља све док постоје нераспоређени корисници. На крају, додају се посете AFS користећи $FIX\_SOLUTION$ методу.

Такође смо имплементирали две верзије ACO алгоритма за TD-EVRP-STW проблем: једну са локалном претрагом (ACOLS) и једну без локалне претраге (ACO). Верзија која користи локалну претрагу за побољшање решења користи VND методу.

Следећа коришћена метахеуристика је BCOi. Иницијална популација пчела креира се коришћењем методе $INIT\_SOLUTION\_RANDOM$. Током лета унапред, метода трансформише решење тако што извршава предвиђен број измена над тим решењем. За наше потребе, дефинисали смо пет могућих оператора за трансформацију решења, од којих се у сваком кораку бира један оператор насумично. Прва четири оператора одговарају околинама $\mathcal{N}_2(S)$, $\mathcal{N}_3(S)$, $\mathcal{N}_4(S)$ и $\mathcal{N}_5(S)$, док је пети оператор дефинисан тако да насумично изабере једног корисника и пребаци га у нову, празну руту. Након трансформације решења, оно се потенцијално поправља коришћењем $FIX\_SOLUTION$ методе.

У случају GA, иницијалну популацију генеришемо тако што једну јединку генеришемо методом $INIT\_SOLUTION$, док све остале јединке генеришемо коришћењем методе $INIT\_SOLUTION\_RANDOM$. Као оператор укрштања користили смо укрштање редоследа (енг. Order Crossover – OX). Такође смо тестирали и оператор укрштања две тачке (енг. Two-point Crossover) уз додатну поправку решења, али су резултати били значајно лошији у односу на верзију која користи OX, због чега овај оператор није даље разматран. Као оператор мутације користили смо исту методу за трансформацију решења коју користи и BCOi. У процесу селекције користили смо елитистички приступ.

MA је имплементиран на исти начин као и GA, с тим што се на све потомке примењује локална претрага, у нашем случају VND. Такође смо дефинисали и модификовани меметски алгоритам (MMA), где се VND примењује само на потомка с најквалитетнијим решењем у свакој генерацији.

За тестирање наших метода користили смо инстанце предложене у [302]. Инстанце су подељене у две групе. Прва група, коју ћемо звати I_1, садржи 35 мањих инстанци са до 15 корисника, док друга група (I_2) садржи 56 инстанци са по 100 корисника. Будући да ове инстанце нису оригинално намењене за TD сценарио, додали смо податке о просечној брзини у зависности од времена на следећи начин: цео период доставе подељен је на 13 интервала, од којих прва два интервала имају мултипликатор брзине 0.75 (моделирајући јутарњу гужву), наредних осам интервала има мултипликатор 1, док претпоследња два интервала имају мултипликатор 0.8. Последњи интервал такође има мултипликатор 1.

Сви експерименти су извршени на лаптоп рачунару са Intel i7-10750H процесором, 32GB RAM меморије и оперативним системом Ubuntu 20.04. Све метахеуристике су имплементиране у C++ програмском језику. Да бисмо показали конзистентност резултата, сваки тест је поновљен 30 пута. Као критеријум заустављања за све метахеуристике одређено је време извршења, које је ограничено на 5 минута за инстанце из скупа I_1, односно 10 минута за инстанце из скупа I_2. Хиперпараметри за сваку од метахеуристика изабрани су користећи iRace пакет за R програмски језик.

Резултати тестирања метахеуристика на оба скупа инстанци се могу преузети са адресе `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`. Након обављених статистичких тестова, закључили смо да се перформансе метахеуристика не разликују значајно када посматрамо просечан квалитет решења на скупу инстанци I_1. Међутим, за инстанце из скупа I_2 постоји статистички значајна разлика у перформансама метахеуристика. GVNS и MMA су се показали као најбољи у погледу просечног квалитета решења. Иако GRASP и MA ређе постижу најбоље резултате за поједине инстанце, статистички тестови сугеришу да ове две метахеуристике имају сличне перформансе као и GVNS и MMA. С друге стране, ACO, ACOLS, BCOi и GA су имали лошије перформансе. Из тестова се може закључити да BCOi фаворизује решења са већим бројем краћих рута. Такође можемо закључити да GVNS и GRASP брзо конвергирају ка добром решењу, након чега стагнирају, док је MMA-у било потребно више времена да пронађе решење сличног квалитета или потенцијално боље решење.

Проблем рутирања електричних возила са парцијалним пуњењем батерија

Друга верзија проблема која се разматра у овој дисертацији слична је TD-EVRP-STW, с тим што се возилима дозвољава парцијална допуна батерије. Другим речима, поред свих других аспеката проблема описаних у претходној секцији, за овај проблем је неопходно одредити колико ће се дуго свако возило задржати приликом сваке посете AFS. Ову верзију проблема означићемо са TDPR-EVRP-STW.

У ту сврху, потребно је изменити структуру решења која је коришћена за TD-EVRP-STW, тако што ће се сваком чвору додати додатни податак који одређује колико се возило задржава на тој локацији. Уколико локација представља корисника, време задржавања је унапред одређено константом задатом у самој инстанци. Са друге стране, ако локација представља AFS, време задржавања је флексибилно и одређује се током извршења алгоритма.

Такође је потребно изменити методу за додавање посета AFS. Стратегија која се користи унутар ове методе разликује се у зависности од типа чвора. Ако је чвор AFS, прво се проверава колико енергије може бити допуњено, узимајући у обзир тренутни степен напуњености и капацитет батерије. Ако возило на посматраној станици већ пуни батерију до максималног капацитета, разматра се додавање посете новој станици. Да би се избегло циклично посећивање AFS, уводи се правило да наредна посета станици мора бити ближа следећој локацији од тренутно посматране станице. Ако се таква станица не пронађе, метода прогласи решење недопустивим. Уколико је могуће додатно допунити батерију на тренутно посматраној станици, рачуна се колико би требало продужити боравак на станици. Идеално је допунити батерију тако да омогући долазак до наредне локације, плус додатних 20% капацитета, како би се повећала вероватноћа да возило може стићи до неке станице и након посете следећем кориснику. Ако то није могуће, батерија се пуни до максимума. Ако чвор представља корисника, покушавамо идентификовати најближу AFS до следеће локације која је доступна са тренутним нивоом напуњености батерије. Ако пронађемо такву AFS, додајемо је у руту и одређујемо задржавање на исти начин као и у претходном случају, тј. батерија се пуни довољно да возило стигне до наредне локације, плус додатних 20% капацитета. Ако није могуће пронаћи одговарајућу AFS, покушава се убацити станица на некој од претходних позиција у рути. Иако овај метод може потенцијално прогласити нека допустива решења недопустивим, у пракси се показао као добро решење због своје једноставности, што је неопходно због честе употребе ове методе.

Метахеуристике коришћене за ову верзију проблема су исте као оне коришћене за TD-EVRP-STW, али се користи нова структура решења и модификована верзија методе за додавање посета AFS. Поред тога, метода за конструкцију решења у ACO и ACOLS, као и репрезентација хромозома за GA, MA и MMA, су благо измењени како би се прилагодили спецификацијама проблема.

За тестирање овог проблема коришћене су исте инстанце и исто окружење као и за претходну верзију проблема.

Резултати тестирања метахеуристика на оба скупа инстанци могу се преузети са адресе `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`. На скупу I_1, све метахеуристике су имале релативно сличне перформансе, осим ACO, који је у 26 случајева имао најлошији просечан квалитет решења, а ни у једној инстанци није имао најбољи просечан квалитет решења. За MA и BCOi није показана статистички значајна разлика у погледу просечног квалитета решења у односу на ACO, као ни у односу на остале метахеуристике. Ипак, MA је имао ту предност да је током 30 извршавања пронашао решење највишег квалитета за највећи број инстанци.

У случају инстанци из скупа I_2, GVNS и MMA показали су значајно боље перформансе у односу на остале методе. Иако статистички тестови сугеришут временски зависних брз да између GVNS и MMA не постоји статистички значајна разлика, приметили смо да је MMA имао најбољи просечан квалитет решења за 36 инстанци, за разлику од GVNS-а који је то постигао за 19 инстанци.

Вишекритеријумски проблем еколошког рутирања возила

Трећа верзија проблема разматрана у овој дисертацији надограђује претходну верзију (TDPR-EVRP-STW), узимајући у обзир све већ поменуте атрибуте као што су меки временски оквири, парцијално пуњење батерије и временски зависне брзине. Ипак, ова верзија укључује и неколико додатних атрибута. Прво, уводимо претпоставку о асиметричности дистанци. Такође, за разлику од претходне две верзије проблема, ова верзија разматра флоту возила састављену од електричних и конвенционалних возила. Ова два типа возила морају се третирати на потпуно различит начин, пошто само електрична возила морају посећивати AFS. Ово повећава степен комплексности проблема, јер се за сваку руту мора одредити којим типом возила ће бити опслуживана. Како бисмо што боље адресирали еколошки аспект проблема, определили смо се за вишекритеријумску оптимизацију. Ову верзију проблема означавамо са MOHF-VRP-STW.

Као први критеријум оптимизације користили смо исту функцију циља као и за TD-EVRP-STW ($\mathcal{O}_1$). Као други критеријум, користили смо минимизацију укупне дистанце коју пређу ICV ($\mathcal{O}_2$), како бисмо смањили емисије CO_2. Може се приметити да минимизација другог критеријума може имати одређене последице по први критеријум, јер смањење коришћења ICV може повећати укупну пређену дистанцу свих возила, због чешћих посета AFS. Ипак, потребно је изабрати јединствену метрику која ће усмеравати метахеуристике. У нашем случају, одлучили смо се за следећу формулу:

$$\mathcal{F}(\mathcal{O}_1, \mathcal{O}_2) = \alpha \left(\frac{\log \mathcal{O}_1 - \log \mathcal{O}_1^{min}}{\log \mathcal{O}_1^{max} - \log \mathcal{O}_1^{min}} \right) + \beta \left(\frac{\log \mathcal{O}_2 - \log \mathcal{O}_2^{min}}{\log \mathcal{O}_2^{max} - \log \mathcal{O}_2^{min}} \right)$$

Како бисмо тестирали перформансе метахеуристика за MOHF-VRP-STW, модификовали смо инстанце коришћене за претходне верзије проблема. Увели смо асиметричност дистанци и додали податке о различитим типовима возила, као што су њихова брзина, број расположивих возила и тип мотора. Такође, убацили смо податке о мултипликаторима просечних брзина за сваки временски период директно у инстанце. Инстанце генерисане на основу оних из скупа I_1 означили смо са $\mathcal{I}_1$, док су оне генерисане на основу инстанци из скупа I_2 означене са $\mathcal{I}_2$.

Да бисмо представили решење овог проблема, модификовали смо структуру представљену за TDPR-EVRP-STW, тако што решење сада састоји од листе парова (V_i, S), где V_i означава индекс возила придруженог тој рути, док S представља листу чворова који одговарају локацијама, са истим подацима као и у случају TDPR-EVRP-STW.

Функција за додавање посета AFS је благо измењена у односу на ону коришћену у TDPR-EVRP-STW. За сваку руту прво се проверава тип возила. Уколико је возило EV, користи се исти поступак који је објашњен у одељку везаном за TDPR-EVRP-STW. Са друге стране, ако је возило ICV, све евентуалне посете AFS које се налазе у тој рути се избацују.

Због специфичности мешовите флоте возила, увели смо додатну околину која је коришћена за VND:

- $\mathcal{N}_{11}(S)$ - Заснована на оператору који размењује типове возила између две руте.

За решавање проблема MOHF-VRP-STW тестирали смо исте метахеуристике као и за TD-EVRP-STW и TDPR-EVRP-STW. У односу на TDPR-EVRP-STW, имплементације алгоритама за овај проблем разликују се у коришћењу проширене структуре решења коју смо описали раније, измењеној методи за додавање посета AFS, увођењу нове околине за VND, као и у коришћењу архиве за складиштење Парето-оптималних решења. Такође, метода конструкције решења за ACO и ACOLS је измењена тако да не додаје посете AFS у руте које се опслужују са ICV. Додатно, за BCOi увели смо још један оператор за трансформацију решења који насумично бира два возила из решења и размењује њихове типове. У случају GA, MA и MMA, главна разлика у односу на претходне верзије састоји се у мало измењеној репрезентацији хромозома, као и у измењеном оператору укрштања, који узима у обзир и типове возила.

Након спроведених тестова над инстанцама из скупа $\mathcal{I}_1$, приметили смо да GVNS и ACOLS дају најбоље резултате у односу на критеријум $\mathcal{O}_1$. Посматрајући метахеуристике у контексту критеријума $\mathcal{O}_2$, ниједна метахеуристика се није показала знатно бољом од осталих.

Резултати тестова над инстанцама из скупа $\mathcal{I}_2$ за критеријум $\mathcal{O}_1$ показују да је MMA остварио најбоље резултате. Након MMA, GVNS, GRASP и MA такође су имали релативно добре резултате. ACO, ACOLS и BCOi су показали најлошије резултате за критеријум $\mathcal{O}_1$. Слични резултати забележени су и за критеријум $\mathcal{O}_2$. Генерално, MMA је показао боље перформансе у односу на остале методе за седам од десет посматраних метрика, укључујући просечно, најбоље и најлошије решење за оба критеријума, као и број коришћених возила.

Завршне напомене

Током истраживања приметили смо неколико области везаних за EVRP које би могле добити више пажње, а које нису обухваћене овом дисертацијом. Те области и потенцијална побољшања укључују:

- Постоји много различитих VRP атрибута који нису обухваћени овом дисертацијом. Укључивањем више атрибута, проблем може постати релевантнији. Ипак, потребно је опрезно додавати нове атрибуте како проблем не би постао превише опширан и комплексан за решавање. У таквим случајевима, разумно је очекивати да ће проблеми специјализовани за одређени сценарио давати боље резултате од оних који разматрају превелики број атрибута.
- Иако нелинеарно пуњење батерије и нелинеарна потрошња енергије не утичу превише на перформансе метахеуристика, ове аспекте би требало укључити у алгоритме како би резултати били релевантнији за индустрију.

- У мултикритеријумској оптимизацији, промена вредности једног критеријума може имати позитивне или негативне последице по остале критеријуме, па би било корисно извршити анализу корелације између различитих критеријума оптимизације.
- Инкорпорација алгоритама машинског учења може бити корисна за решавање овог типа проблема, било кроз аутоматско ажурирање хиперпараметара методе током њеног извршавања, било кроз коришћење тих алгоритама за предвиђање непознатих података, чиме се метахеуристикама пружа више информација за рад.
- Иако концепт временски зависних брзина омогућава моделирање ситуација као што су јутарња или поподневна гужва, можемо приметити да брзина такође може зависити и од локације, пошто се гужва не ствара равномерно на свим локацијама у граду у исто време. Самим тим, концепт временски и просторно зависних брзина се може разматрати у контексту VRP.

Acknowledgements

From a young age, even at five, I was confidently declaring my future as a scientist. At that time, my perception of scientists was shaped by movies: esteemed individuals with lucrative careers and a captive audience for every word they spoke. While I eventually realized that this portrayal might be somewhat idealized, this revelation did not deter me. The prospect of exploring and researching topics that intrigue me continued to offer a sense of fulfillment, solidifying my decision to follow the path of a scientist.

In 2017, I embarked on my PhD journey in bioinformatics at the University of Belgrade, specifically within the Faculty of Mathematics. Not long after beginning, an opportunity at the Mathematical Institute of the Serbian Academy of Sciences and Arts (MI SANU) caught my attention. Its proximity to my residence was a compelling factor, so I applied for the position. The response was prompt, leading to an engaging discussion about the role at the institute.

During my visit, I attended a seminar on Computer Science and Applied Mathematics, where an encounter with Professor Tatjana Davidović unfolded. Our conversation delved into the intricacies of metaheuristics, marking the start of a mentorship that would shape my research trajectory. Today, as you either hold or digitally view this document, it embodies the culmination of our collaborative efforts and insights, all tracing back to that initial, fortuitous conversation years ago.

My PhD journey was a tale of evolution and unexpected turns. Initially drawn to bioinformatics, my interest gradually veered towards operations research, prompting a significant change in my academic path. This shift led me to the Faculty of Technical Sciences at the University of Novi Sad.

Upon joining the MI SANU, I had the opportunity to collaborate closely with Professor Davidović in the realm of Vehicle Routing Problems. This research venture was a concerted effort, alongside Vladimir Ilin, Panos M. Pardalos, and Tatjana Jakšić-Krüger. Together, we developed a mathematical model for our version of the problem. We then skillfully employed the Variable Neighborhood Search metaheuristic, efficiently achieving noteworthy results in a remarkably short timeframe.

However, the road to publication proved unexpectedly prolonged. As I write this, those findings are still awaiting their debut in the academic world. In the meantime, I have not been idle. My academic tenure has been marked by the authorship of four articles in scientific journals and eleven papers presented at conferences. Yet, the anticipation builds for the day when those initial results of my research journey are finally published and ready to make a significant impact in the field[1].

Throughout my early research journey, Professor Davidović was an invaluable mentor. Her insightful advice and thorough review of my papers significantly enhanced my skills in scientific writing. Each review was packed with constructive feedback, guiding me toward excellence in my publications. Although I often repeated mistakes in subsequent papers, her guidance was instrumental in shaping my approach to academic writing. I am deeply grateful for her commitment and support, which played a pivotal role in my initial incursions into the scientific community. Her dedication not only honed my technical abilities but also imbued me with the confidence and knowledge essential for a budding researcher.

[1]Since writing this section, it has been accepted for publication in a special issue of *Computers & Operations Research* (2024).

I extend heartfelt thanks to MI SANU and Professor Zoran Ognjanović for their significant role in providing the opportunities and resources necessary for the completion of my research. Their support was instrumental not only in the research process but also in enabling me to present my findings at various esteemed conferences.

I am especially grateful to my family for their unwavering support in my continuous quest for academic excellence and the pursuit of additional degrees. Their enduring encouragement and belief in my capabilities made this journey not just possible, but deeply rewarding.

I extend my heartfelt gratitude to my friends and colleagues, whose companionship in various pubs, taverns, and bistros offered much-needed relaxation during my thesis journey. Admittedly, without their delightful company, the completion of this work might have arrived much sooner.

I am also immensely thankful to everyone who contributed to my research. A special acknowledgment goes to my colleague, Professor Slobodan Jelić, whose endless stream of innovative ideas has opened numerous avenues for future exploration. Collaborating with him and Professor Davidović led to the publication of my first paper in a scientific journal, a milestone that I cherish deeply.

I extend my sincere appreciation to the members of my thesis committee for their willingness to engage with my work, which, in true fashion to my style, likely extends at least fifty pages beyond necessity.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Dedicated to my mother...

CHAPTER
ONE

INTRODUCTION

1.1 Motivation

Given the escalating concerns surrounding *greenhouse gas* (**GHG**) emissions and their adverse environmental impact, numerous countries have made significant commitments to curbing emission levels. These efforts often involve implementing measures such as imposing higher taxes on emissions or providing subsidies for environmentally friendly energy sources. Notably, carbon dioxide (CO_2) represents the most prevalent GHG, accounting for approximately 81% of total GHG emissions within the European Union [88]. The primary source of CO_2 emissions stems from the combustion of fossil fuels, which happens to be the dominant energy source [142] (Figure 1.1). It is worth highlighting that the transportation sector alone contributes to approximately 26% of all GHG emissions [88], necessitating a concerted focus on adopting strategies to reduce emissions within this sector.

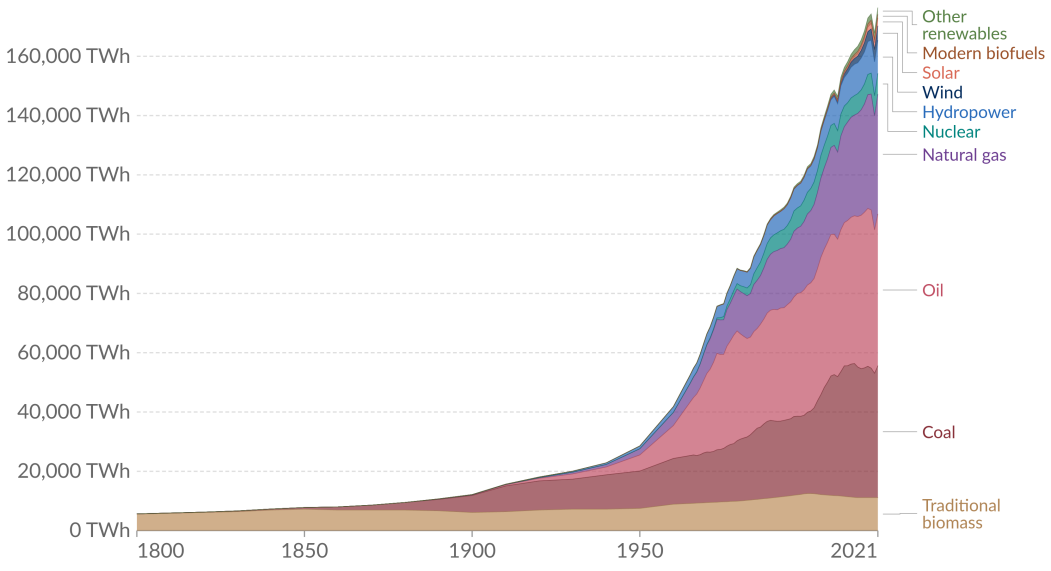

FIGURE 1.1: Global primary energy consumption by source [289].

While CO_2 emissions from *Internal Combustion Vehicles* (**ICV**s) are widely recognized, they also emit a range of other pollutants. These include *Nitrogen Oxides* (NO_x), which consist of NO and NO_2, along with *Volatile Organic Compounds*

(**VOC**s), *Carbon Monoxide* (*CO*), and *Particulate Matter* (**PM**). The particulate matter is categorized based on size, notably those less than 10 microns (PM_{10}) and those smaller than 2.5 microns ($PM_{2.5}$), which includes black carbon [359]. These pollutants contribute significantly to environmental and health concerns.

According to a 2019 report by the *Health Effects Institute* [128], the air pollution was the fourth leading cause of death in that year (Figure 1.2). A frequently suggested remedy is the use of less polluting vehicles, such as electric vehicles. While electric vehicles do not entirely resolve the issue globally, as much of the energy used still originates from fossil fuels, they do help relocate pollution away from densely populated urban areas. The combination of these benefits and the rising popularity of electric vehicles has sparked a remarkable increase in related research. While the first scholarly mention of electric vehicles can be traced to 1909 [58], the field has experienced a pronounced escalation in interest and academic focus in the contemporary era. An analysis of publications over the past 33 years, starting from 1990 up to December 3rd, 2023, reveals a marked increase in papers featuring *"electric vehicle"* in their titles or keywords, particularly since 2005[1] (Figure 1.3). This trend is a clear indicator of the escalating interest within the scientific community regarding electric vehicles.

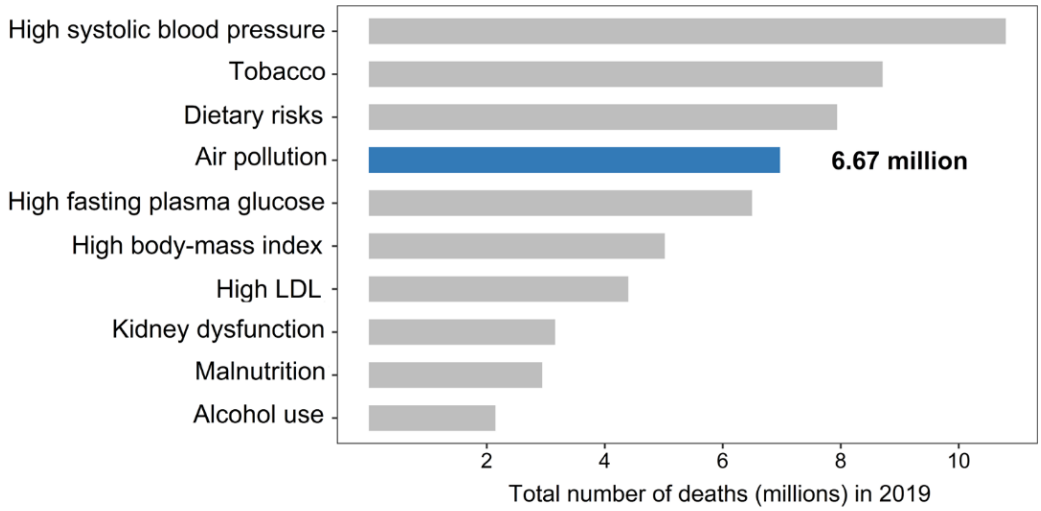

FIGURE 1.2: Worldwide ranking of risk factors based on the total death count from all causes in 2019 (Source: Health Effects Institute [128]).

Consequently, the realm of *environmentally friendly logistics* has gained substantial attention in recent years, as stakeholders seek innovative approaches to address the challenge of GHG emissions and air pollution in general.

Environmentally friendly logistics, commonly referred to as Green Logistics, encompasses the integration of environmental considerations into supply chain management to improve the environmental performance of suppliers and customers [183]. This approach involves a range of activities aimed at reducing the environmental impact of logistics operations. These activities encompass the measurement and evaluation of environmental factors associated with various distribution strategies, the

[1] Data obtained from `https://ieeexplore.ieee.org/search/searchresult.jsp?action=search&newsearch=true&matchBoolean=true&queryText=(%22Publication%20Title%22:electric%20vehicle)%20OR%20(%22Author%20Keywords%22:electric%20vehicle)`

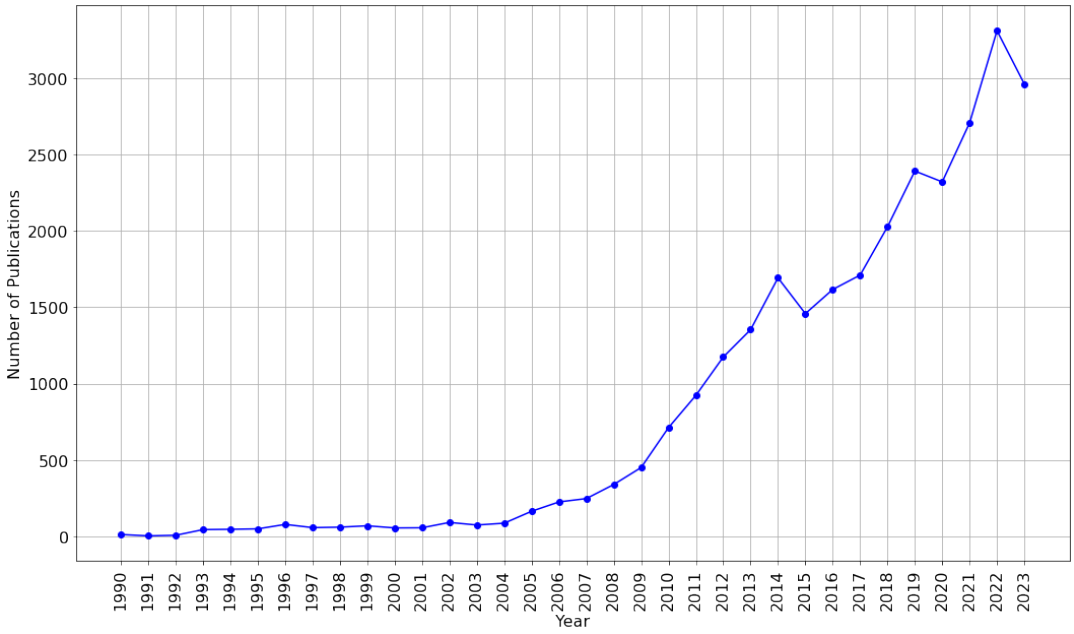

FIGURE 1.3: Number of publications per year concerning electric vehicles from 1990 to 2023.

reduction of energy consumption throughout logistics activities, the minimization and proper management of waste, and the implementation of sustainable waste treatment practices [301]. By incorporating these principles, organizations can foster sustainable practices and contribute to the preservation of the environment while ensuring efficient and effective supply chain management.

The adoption of environmental practices by a company can be influenced by a multitude of factors. These include stakeholder pressure, environmental regulations, company size, industry sector, geographical location, internationalization, position in the value chain, strategic attitude, managerial attitudes and motivations, manager's characteristics, and human resources [119].

The proactive engagement of customers plays a vital role in driving the broader adoption of environmentally friendly practices by companies. Customers have their own set of expectations and preferences when it comes to green products and services. They may specifically request products to be delivered using *alternative fuel vehicles* (**AFVs**) or in a manner that prioritizes emissions reduction. This consumer-driven demand acts as a compelling force, pushing suppliers to embrace sustainable solutions. Acknowledging the pivotal role of consumers in green logistics can serve as a powerful motivation for companies to implement eco-friendly measures. Notably, home delivery is particularly important to the customers since they directly benefit from this service. Consequently, efficiently routing a fleet of AFVs, particularly within the last-mile delivery context, becomes increasingly important as it seeks to strike a balance between the interests of both companies and customers.

However, incorporating environmentally-friendly policies in logistics often presents a unique set of challenges, as these policies can sometimes conflict with the core objectives of logistics itself. These conflicts are commonly known as the *paradoxes of green logistics*. In a study conducted by Rodrigue, Slack, and Comtois [291], various paradoxes have been identified, highlighting the intricate balance required between sustainable practices and logistical efficiency:

- **Costs**: Logistics primarily aims to reduce transportation costs and improve efficiency, but these cost-saving strategies often conflict with environmental considerations. Environmental costs, such as pollution and congestion, are typically externalized, benefiting users and consumers while burdening the environment. This situation presents a paradox in logistics, where cost reductions do not necessarily equate to decreased environmental impact.

- **Time / Speed**: In the logistics sector, the focus on time efficiency frequently results in a greater reliance on transportation methods that are highly polluting and energy-inefficient, like air freight and trucking. Take electric vehicles as an instance, while they offer the advantage of reducing local pollution, particularly in urban settings, their extended recharging durations render them less time-efficient. This limitation in time efficiency is a significant factor impeding their broader adoption in the logistics industry.

- **Reliability**: The core principle of logistics is service reliability, focusing on timely delivery with minimal risk of damage or breakage. This emphasis often leads logistics providers to favor transportation modes deemed most reliable, which, paradoxically, are also the least environmentally friendly. Modes like shipping and rail, considered less reliable in terms of punctuality and safety, have garnered a reputation for lower customer satisfaction. Consequently, the logistics industry predominantly revolves around air and truck shipments, despite their significant environmental impact.

- **Warehousing**: Logistics significantly contributes to globalization and international trade by enabling economies to minimize inventory through fast and reliable deliveries, reducing the need for extensive warehousing. However, this has led to a shift where inventories are increasingly in transit, particularly on roads and terminals, exacerbating congestion and pollution, with the environmental and societal costs falling outside the responsibilities of logistics operators.

- **E-commerce and information technologies**: The advent of information technologies has revolutionized retailing, particularly in the realm of e-commerce. E-commerce thrives on an integrated supply chain that facilitates data interchange among suppliers, assembly lines, and freight forwarders. While online transactions appear to be movement-free for customers, the distribution processes they trigger can be more energy-intensive than traditional retail activities. Moreover, in the realm of supply chain management, companies have the option to enhance their operations by equipping products with *radio frequency identification* technology and integrating it with *blockchain* technology. This advanced approach enables more precise tracing of product origins and materials, improves quality control, and simplifies information sharing [335]. However, it is important to note that such technological integration can significantly increase energy consumption.

Navigating these paradoxes is crucial for the successful adoption of environmentally-friendly logistics practices. Consequently, numerous researchers in the field of optimization are now dedicating their efforts to resolving these challenges. Their goal is to strike an optimal balance between cost-effectiveness and environmental sustainability, endeavoring to reconcile these often competing objectives.

Ultimately, setting aside environmental considerations, the finite nature of fossil fuel reserves cannot be ignored. Projecting the lifespan of these resources is an inherently complex challenge; nonetheless, some projections indicate that the peak in

global *'all-liquids'* production may occur around 2040 [177]. Beyond this point, a decrease in production is anticipated. Researchers have concluded that without swift and substantial reductions in global oil demand, facilitated by policy-driven initiatives to address climate change, the limited supply of oil is poised to have increasingly profound economic and political repercussions. Given this reduced availability of fossil fuels, the importance of alternative fuel vehicles is set to rise substantially.

1.2 Alternative fuel vehicles

Alternative fuel vehicles (AFVs) represent a dynamic shift from traditional gasoline and diesel-powered transportation, aiming to address the growing environmental and energy security concerns associated with fossil fuels. These vehicles utilize alternative energy sources such as electricity, biofuels, natural gas, hydrogen, and others, diverging from conventional petroleum-based fuels. The importance of AFVs lies in their potential to reduce harmful emissions, diminish our dependence on finite oil reserves, and foster a more sustainable and environmentally-friendly transportation landscape. As public awareness of climate change and environmental degradation increases, AFVs are attracting more attention. The core idea behind AFVs is to offer the same or improved transportation capabilities while minimizing the environmental footprint and leveraging renewable or cleaner energy sources. A comprehensive discussion on the pros and cons of various AFVs can be found in the publication by Ghadikolaei et al. [104].

Some examples of AFVs include:

- Electric Vehicles
  - *Battery Electric Vehicles* (**BEVs**): BEVs operate by using rechargeable batteries to power an electric motor, instead of a conventional internal combustion engine. One of the main advantages of BEVs is their zero-emission operation, which significantly reduces their negative environmental impact. However, a major drawback of BEVs is their relatively limited driving range and lengthy recharge time, which can hinder their practicality for long-distance travel. Additional insights into this type of vehicle are elaborated upon in references [135, 165, 299].
  - *Hybrid Electric Vehicles* (**HEVs**): HEVs combine an *internal combustion engine* (**ICE**) with an electric motor and battery system to power the vehicle. The electric motor and battery assist the ICE, improving fuel efficiency and reducing emissions. HEVs utilize regenerative braking to recharge the battery by converting kinetic energy into electrical energy. HEVs offer benefits such as reduced fuel consumption, lower emissions, and increased driving range compared to conventional vehicles. They provide a practical solution by combining the advantages of both conventional and electric vehicles, making them an attractive option for those seeking improved fuel efficiency without the limitations of pure electric vehicles. Additional resources on HEVs are available in references [81, 82, 122, 229].
  - Plug-in Hybrid Electric Vehicles (PHEVs): PHEVs operate similarly to HEVs, combining an internal combustion engine with an electric motor and battery system. However, what sets PHEVs apart is their larger battery capacity, allowing them to be charged from an electrical outlet. This feature enables PHEVs to travel longer distances solely on electric power, reducing fuel consumption and tailpipe emissions. When the battery is

depleted, PHEVs seamlessly switch to utilizing the internal combustion engine, offering extended driving range without the range anxiety associated with fully electric vehicles. The ability to recharge the battery from an external power source makes PHEVs more flexible, as they can take advantage of electricity from the grid, reducing dependence on fossil fuels. Further information on this vehicle type is detailed in references [9, 57, 218].

  - Extended-Range Electric Vehicles (EREVs): EREVs are a unique type of electric vehicle that combines the benefits of both electric and conventional vehicles. EREVs feature an electric motor powered by a large battery pack, similar to Battery Electric Vehicles (BEVs). However, what sets EREVs apart is the presence of an onboard internal combustion engine that serves as a generator to charge the battery when its energy is depleted. This means that EREVs can operate solely on electric power for a significant distance, depending on the model and battery capacity. Once the battery is depleted, the ICE kicks in to generate electricity, effectively extending the vehicle's range. This dual-power system eliminates the issue of range anxiety commonly associated with fully electric vehicles. Additional resources on this topic are available in references [279, 364].
  - Fuel Cell Electric Vehicles (FCEVs): FCEVs are a type of electric vehicle that use hydrogen as their primary fuel source. Unlike BEVs that store electricity in batteries, FCEVs generate electricity through a chemical reaction between hydrogen and oxygen in a fuel cell stack. This process produces electricity, water vapor, and heat as byproducts, making FCEVs an environmentally friendly transportation option. The electricity generated powers an electric motor, propelling the vehicle forward. One of the significant advantages of FCEVs is their long driving range and quick refueling time. Hydrogen can be refueled in a matter of minutes, similar to traditional gasoline-powered vehicles, offering a convenient and familiar experience for drivers. Additionally, FCEVs emit zero GHG, as the only byproduct is water vapor. However, the widespread adoption of FCEVs faces challenges such as the limited availability of hydrogen refueling infrastructure and the high cost of fuel cell technology. Despite these obstacles, ongoing advancements and investments in hydrogen fuel cell technology show promising potential for Fuel Cell Electric Vehicles as a clean and sustainable mobility solution. Additional references on FCEVs can be found in [249, 276, 316].

- Biofuel Vehicles

  - Ethanol Flex-Fuel Vehicles (FFVs): FFVs are engineered to operate using either ethanol, gasoline, or a mixture of both. Ethanol, a renewable fuel source mainly extracted from crops like corn and sugarcane, blends with gasoline to enhance the fuel's octane rating and reduce harmful emissions compared to gasoline alone. FFVs are equipped with an internal combustion engine and a fuel system tailored to withstand the corrosive effects of high ethanol concentrations. The use of ethanol as a fuel helps reduce reliance on imported oil and contributes to the lowering of greenhouse gas emissions. Nonetheless, the use of ethanol as a fuel source presents certain challenges, including questions regarding the energy efficiency of its production process and its influence on food commodity prices. For more

comprehensive information about this vehicle category, one can refer to specific studies indicated in references [24, 62].

- Biodiesel Vehicles: Vehicles that utilize biodiesel as fuel harness the energy from renewable, organic sources such as vegetable oils, animal byproducts, or reclaimed cooking oils. Biodiesel is versatile, allowing for usage in its entirety as B100 or in diluted forms like B5 and B20 blends, where it is combined with conventional diesel. This eco-friendly fuel stands out for its cleaner combustion process compared to regular diesel, significantly cutting down emissions of particulates, carbon monoxide, and hydrocarbons. However, it is worth noting that higher concentrations of biodiesel may not be compatible with all diesel engines and might require specific alterations. For a more comprehensive understanding of this subject, reference materials [12, 97] can provide further information.

- Propane Autogas Vehicles: Propane Autogas Vehicles operate using propane, commonly referred to as liquefied petroleum gas (LPG), as their primary fuel source. Propane autogas is a clean-burning and domestically abundant alternative to traditional gasoline and diesel. These vehicles are fitted with specialized fuel systems designed to handle propane's unique properties, which is stored under pressure as a liquid. When released, it vaporizes and can be used to power an internal combustion engine. Propane autogas vehicles tend to produce fewer GHG emissions compared to their gasoline counterparts, making them a more environmentally friendly option. Helpful resources on the topic can be found in [248, 260, 356].

- Synthetic Fuels: Vehicles using synthetic fuels operate on man-made fuel sources derived from processes that transform natural gas, coal, or biomass into liquid hydrocarbons. These synthetic fuels, often referred to as *synfuels*, aim to replicate the energy density and combustion characteristics of traditional petroleum-based fuels, like gasoline and diesel. The advantage of synfuels lies in their potential for reduced greenhouse gas emissions and diminished reliance on crude oil imports, as they can be produced from a variety of domestic resources. Moreover, they can be tailored to produce fewer impurities, resulting in cleaner combustion and reduced tailpipe emissions. While the technology and processes behind synthetic fuel production, such as Fischer-Tropsch synthesis, have been around for decades, economic viability and concerns over the full lifecycle emissions of certain feedstocks (like coal) have impacted their widespread adoption. Further information on this topic is available in references [121, 277, 278, 317].

- Dimethyl Ether Vehicles: Dimethyl Ether (DME) vehicles operate on an alternative fuel that presents itself as a promising eco-friendly option for the transportation sector. DME is a colorless, odorless gas at room temperature but can be easily liquefied under pressure, making its physical properties somewhat similar to propane. Its primary appeal is its capacity to be produced from a wide variety of renewable resources, including biogas and various organic waste materials. As DME is oxygen-rich, it reduces the emission of nitrogen oxides upon combustion. Despite its potential benefits, the adoption of DME as a mainstream vehicle fuel is still in its developing stages, primarily because of the lack of refueling infrastructure and the need for vehicle engines to be modified or specifically designed to utilize it. Further information on this topic is available in references [189, 267, 305].

- Biogas Vehicles: Biogas vehicles represent an innovative approach to sustainable transportation, harnessing the energy of biogas - a mixture of methane and carbon dioxide produced by the anaerobic digestion of organic matter. Common sources of biogas include agricultural waste, manure, municipal waste, plant material, and sewage. When purified to increase the methane content, biogas becomes biomethane, which can be used as a vehicle fuel. Vehicles running on biomethane benefit from reduced GHG emissions compared to those powered by traditional fossil fuels. Additionally, using biogas aids in waste management by converting potential pollutants into a valuable energy source. Further information on this topic is available in references [38, 265].

- Solar Powered Vehicles: Solar powered vehicles harness the energy of the sun, converting it into electricity through photovoltaic cells mounted on their surfaces, primarily on the roof. These vehicles use this electricity to power electric motors that propel the vehicle forward. While the idea of a purely solar-powered vehicle is attractive for its potential to achieve zero-emission transportation, in practical terms, many solar vehicles use the solar panels to assist in charging onboard batteries rather than as the sole power source. This is due to the limited surface area available for solar panels and the variable nature of sunlight. Though widespread commercial viability of fully solar-powered vehicles remains a challenge due to these limitations, integrating solar panels into electric and hybrid vehicles to extend their range and reduce grid charging needs is seeing increasing interest from automotive manufacturers. Additional info on this topic can be found in [211, 351].

- Liquid Nitrogen Vehicles: Liquid nitrogen vehicles operate by utilizing the rapid expansion of liquid nitrogen to produce a pressurized gas that can drive a piston or turbine, thereby generating mechanical power to propel the vehicle. When the extremely cold liquid nitrogen is exposed to ambient temperatures, it rapidly converts to nitrogen gas, expanding up to 700 times its liquid volume. This expansion can be channeled to perform work, much like steam in a steam engine. One of the primary advantages of liquid nitrogen as a propellant is its environmental benignity; the process emits only nitrogen, which is already the primary component of our atmosphere. However, the energy-intensive nature of liquefying nitrogen - requiring significant electrical input - has been a hurdle for its widespread adoption. Moreover, the storage and handling of cryogenic liquids present engineering challenges. Additional insights into this vehicle type are elaborated in references [162, 273, 375].

- Air-Powered Vehicles: Air-powered vehicles, often referred to as compressed air vehicles (CAVs), utilize compressed air to create mechanical motion that drives the vehicle. The principle is straightforward: air is compressed into tanks at high pressure and, when released, this air expands, driving a piston or turbine that in turn propels the vehicle. One of the most significant advantages of CAVs is that they emit no pollutants during operation, as the only exhaust is clean, cold air. Moreover, compressing air is a process that can use electricity from renewable sources, positioning air-powered vehicles as a potential sustainable transportation solution. However, there are challenges to overcome. The energy density of compressed air is much lower than traditional fuels, which can limit the range of these vehicles. Additionally, the process of compressing air is not always efficient, especially when compared to other energy storage methods.

Further details about this type of vehicle are available in references [89, 269, 283, 332].

1.3 Operations research

Operations Research (**OR**) represents a scientific methodology that utilizes sophisticated analytical techniques to improve decision-making, typically within the context of limited resources [360]. Utilizing the tools from the mathematical sciences, including mathematical modeling, statistical analysis, and optimization techniques, OR aims to identify the best possible solutions to complex problems involving system operations, spanning from business enterprises to intricate networks of machinery.

At the core of OR is the field of optimization, which is concerned with finding the best available values of some objective function given a defined domain of possible inputs. In practical terms, optimization helps in identifying the most efficient, least costly, or most profitable solution to a problem. OR uses a variety of optimization algorithms to tackle problems such as scheduling (finding the best order in which to perform a series of tasks), routing (determining the most efficient paths for multiple journeys), and resource allocation (distributing resources in the most effective manner).

The connection between operations research and optimization is deeply rooted, as optimization provides the mathematical foundation and tools that OR practitioners need to formulate and solve problems. Operations researchers develop and use mathematical models to simulate complex systems and scenarios. These models can be linear, nonlinear, integer, dynamic, and stochastic, each serving different types of problems and requiring different optimization techniques.

In the business world, OR translates into a host of applications such as supply chain management, where it helps in optimizing the flow of goods and services from production to consumption. In logistics, it improves the delivery routes and schedules, minimizing costs while maximizing efficiency. In finance, it assists in portfolio optimization and risk management. The versatility of OR extends to public services as well, aiding in urban planning, transportation networks, and even in healthcare management, optimizing staff schedules and patient flows.

The incorporation of OR into green logistics manifests in various ways. It includes the development of eco-friendly vehicle routing schedules that consider fuel consumption, emissions, and traffic patterns. In warehouse management, OR models help in optimizing the layout and operations to reduce waste and energy usage. Supply chain management, too, benefits from OR by optimizing product flows to reduce unnecessary transport, thus cutting down on emissions and energy expenditure.

Moreover, OR assists in strategic planning within green logistics by analyzing trade-offs between different logistical options and their environmental impacts. For instance, OR can help determine the viability of investing in alternative fuel vehicles or the long-term benefits of building greener infrastructure.

The role of OR in green logistics is not only tactical but also strategic. By integrating sustainability goals into mathematical models, OR provides a framework for making decisions that support both economic and environmental objectives. It enhances the capacity to evaluate the full cost implications of logistics, including those external costs related to environmental degradation and societal impacts.

1.4 Optimization problems

Optimization problems, prevalent in both industry and theoretical science, involve determining the most favorable values for a specific set of parameters to attain a desired objective [266]. These problems can be viewed as an extension of decision problems, where solutions are further evaluated using an *objective function*. In *minimization* scenarios, the goal is to identify a solution that yields the lowest possible value of the objective function. Conversely, in *maximization* problems, the aim is to find a solution that achieves the highest value. This approach is pivotal in optimizing outcomes across various fields and applications.

Minimization problems can be concisely formulated as:

$$\min_{x \in S} f(x) \tag{1.1}$$

Here, we define an objective function $f : \mathbb{R}^n \to \mathbb{R}$, with $S \subseteq \mathbb{R}^n$ signifying the entire spectrum of feasible solutions. The conversion between maximization and minimization problems is intuitive:

$$\max_{x \in S} f(x) = -\min_{x \in S}(-f(x)) \tag{1.2}$$

When a point x^* fulfills:

$$f(x^*) \leq f(x), \ \ \forall x \in S \tag{1.3}$$

it is termed the *global minimum* or more broadly, the *global optimum*. Conversely, if a neighborhood $\mathcal{N}(\bar{x})$ around $\bar{x}$ exists where:

$$f(x^*) \leq f(x), \ \ \forall x \in \mathcal{N}(\bar{x}) \cap S \tag{1.4}$$

the point x^* is identified as a *local minimum* or *local optimum* within $\mathcal{N}(\bar{x})$. The global optimum is invariably a local optimum with respect to all possible neighborhoods, whereas a local optimum is not necessarily a global optimum. Figure 1.4 illustrates the distinction between the global minimum and local minima using the example function $f(x) = \sin(2\pi x) + \frac{1}{2}\sin(4\pi x) + \frac{1}{4}\sin(8\pi x) - 2e^{-2(x-1.5)^2}$. The graphical depiction clearly delineates the global minimum as the lowest point on the curve, contrasted with the local minima, which are the lowest points in their immediate vicinities.

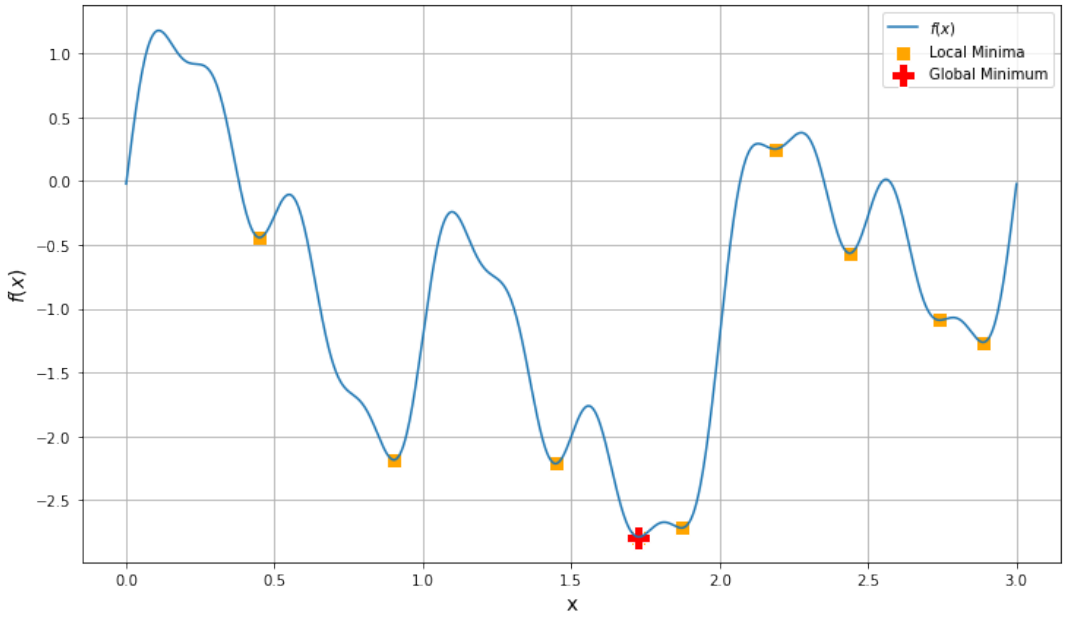

FIGURE 1.4: The difference between local minima and global minimum

Typically, optimization problems are categorized into three distinct types: discrete, continuous, and mixed [66]. Discrete optimization problems are characterized by variables that assume discrete values. In contrast, continuous optimization problems involve variables that are drawn from a continuous set, often the set of all real numbers. Mixed optimization problems present a combination of both, with certain variables assigned discrete values and others permitted to vary continuously. It is worth noting that some literature may classify mixed optimization problems under the discrete category.

Discrete optimization encompasses a variety of challenging problems, including the traveling salesman problem [219], vehicle routing [337], set cover [45], scheduling [334], and the minimum Steiner tree [290], among others. Continuous optimization encompasses tasks such as maximizing differential amplifier yields, tuning shock absorber systems in mechanics, and estimating parameters for non-linear regression models in immunology [240]. A common thread uniting these diverse problems is their inherent complexity. Discrete optimization problems are frequently NP-hard, and finding a global optimum in continuous optimization typically entails exponential complexity in the general case [103]. However, it is noteworthy that for certain global optimization scenarios, like convex optimization problems, highly efficient polynomial-time resolution methods exist, such as Interior point methods [39].

1.4.1 Solution methods

Solution methods for optimization problems are broadly categorized into two main types: exact and approximate algorithms [328]. Exact algorithms are designed to deliver optimal solutions, albeit requiring significant time and computational resources. On the other hand, approximate algorithms offer good estimates for solutions within constrained time or memory, though they do not guarantee optimality. These can be either deterministic or stochastic.

An example of an exact approach is the *Mixed Integer Linear Program* (**MILP**), which formulates the problem using optimization objectives and linear constraints, which can then be solved to optimality through techniques like the *cutting-plane method* [116], *branch and bound* [181], or *branch and cut* [261]. Typically, MILP problems are solved using ready-to-use solvers, such as *CPLEX*[2], *Gurobi*[3], or *GLPK*[4]. Another example of an exact approach is dynamic programming.

Under the umbrella of approximate algorithms, there are heuristic, metaheuristic, and approximation algorithms. Heuristic algorithms are tailored to specific problems, with constructive heuristics building up from an empty solution iteratively until completion, while improvement heuristics start with a complete solution and work towards refining it.

Metaheuristics are general, usually iterative and stochastic methods, and can be either constructive or improvement-based[5]. Their generality means they can be adapted to a wide array of optimization problems, offering frameworks to develop specific algorithms for each unique challenge. A subcategory within metaheuristics are hybrid algorithms, which merge metaheuristic techniques with some exact solver. These algorithms use metaheuristic methods to create smaller subproblems from the original problem, which are then solved using exact methods. In certain scenarios, these hybrid methods can approach the precision of exact methods given ample time.

Lastly, approximation algorithms form a class of heuristic algorithms that quickly find approximate solutions with a **guaranteed** worst-case approximation factor. As part of the heuristic category, they are also problem-specific, offering estimates of solution quality within a calculable range.

It is important to clarify once again the distinction between *Approximation algorithms* and *Approximate algorithms*. Approximation algorithms, a subset of heuristic algorithms, come with a guaranteed level of solution quality. Conversely, approximate algorithms stand in contrast to exact algorithms. These methods do not guarantee the discovery of the optimal solution, even with ample time. Within this context, approximation algorithms are considered a subclass of approximate algorithms. This thesis does not delve into approximation algorithms, readers seeking more information on this specific class of algorithms are directed to references [29, 171, 347, 358] for a more detailed exploration.

In Figure 1.5, we showcase the previously mentioned classification of methods for optimizational problems. Reflecting on our earlier statement that heuristic methods are tailored to specific problems, the examples provided here are particularly chosen with the VRP in mind. It is important to note that this classification serves more as a guideline rather than a strict rule, as certain algorithms may fall into different categories depending on the context. For instance, MIP-based methods are exact when provided with sufficient time and resources. However, under constraints of time or memory, they may be regarded as heuristic methods.

[2] `https://www.ibm.com/products/ilog-cplex-optimization-studio`

[3] `https://www.gurobi.com/`

[4] `https://www.gnu.org/software/glpk/`

[5] The classification presented here is just one approach; we elaborate on various alternative classification schemes in Section 2.1.

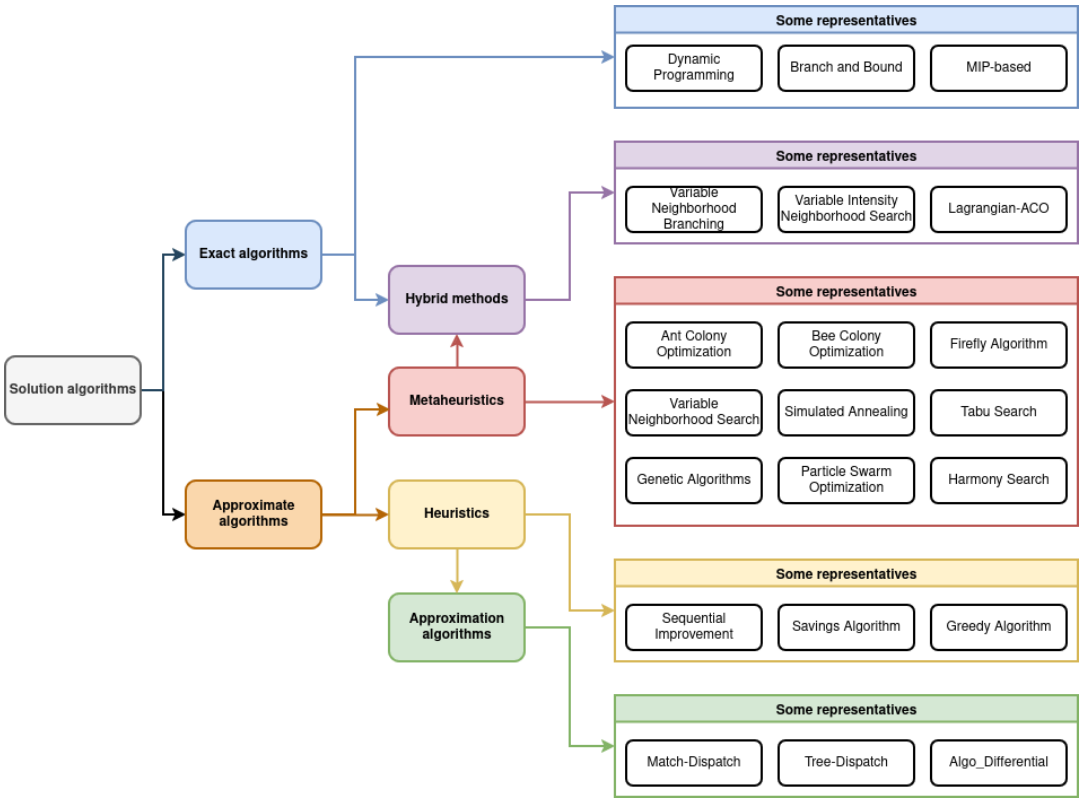

FIGURE 1.5: Various approaches to solving discrete optimization problems.

1.5 The contributions of this thesis

This thesis defines three problems relevant to the industry. The first problem deals with the routing of electric vehicles with flexible time windows and time-dependent speeds. The second version expands on this problem by allowing partial battery charging at stations, aiming for optimal time management and ensuring timely service for users. The third version represents an additional expansion, incorporating a fleet that includes conventional vehicles, reflecting the most realistic scenario. Additionally, asymmetric distances between locations, characteristic of urban environments, have been considered, along with multi-criteria optimization for this problem.

For the first problem, a mathematical model has been proposed. Eight metaheuristic methods were tested for each problem, a statistical analysis of the results was conducted, and the methods were compared based on multiple criteria.

In addition to these contributions, the most comprehensive list of metaheuristics to date has been compiled, with each metaheuristic classified based on its inspiration. Furthermore, to test the third version of the problem, it was necessary to create new instances, which are publicly available at `https://www.mi.sanu.ac.rs/~luka/resources/Instances.zip`. Finally, several auxiliary methods have been developed (e.g., for creating initial solutions, adding visits to stations, or determining vehicle departure schedules), which may also be useful in future algorithms.

Two papers published in journals have emerged from this thesis. The first paper, published in the *YUJOR* journal, is a review article on metaheuristic approaches to green vehicle routing problem:

Matijević, Luka. "Metaheuristic Approaches for the Green Vehicle Routing Problem". In: *Yugoslav Journal of Operations Research* 33.2 (2022), pp. 153–198. DOI: http://dx.doi.org/10.2298/YJOR211120016M

The second paper, published in the *International Journal of Industrial Engineering Computations*, addresses the method of *Variable Neighborhood Search* applied to the problem of electric vehicle routing problem:

Matijević, Luka. "General Variable Neighbourhood Search for Electric Vehicle Routing Problem with Time-dependent Speeds and Soft Time Windows". In: *International Journal of Industrial Engineering Computations* 14 (2023), pp. 275–292. DOI: 10.5267/j.ijiec.2023.2.001.

Although not derived from the thesis, the following paper is also thematically related to it:

Matijević, Luka et al. "General VNS for asymmetric vehicle routing problem with time and capacity constraints". In: *Computers & Operations Research* 167 (2024), p. 106630. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2024.106630. URL: https://www.sciencedirect.com/science/article/pii/S0305054824001023

1.6 The structure of this thesis

This thesis is organized as follows:

Chapter 2 offers a comprehensive overview of metaheuristic algorithms, including various classification schemes and commonly used local search procedures. We delve into several metaheuristics, particularly those relevant to this thesis, in substantial detail. Complementing this chapter, a comprehensive list of all identified metaheuristic algorithms is available in an accompanying document. This resource can be accessed and downloaded at https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixA.pdf. An additional valuable resource, featuring an extensive list of various evolutionary metaheuristics, is available at https://fcampelo.github.io/EC-Bestiary/.

In Chapter 3, we introduce the Vehicle Routing Problem and its specialized variant, the Green Vehicle Routing Problem. This chapter also provides an overview of typical scenarios and objective functions associated with these problems. Additionally, we explore specific aspects of GVRP related to electric vehicles, such as recharging and consumption functions. The final part of this chapter is dedicated to a literature review of publications focusing on GVRP. Here, we offer a statistical analysis of the common scenarios and methods employed in these studies.

The core of the thesis is presented in Chapters 4, 5, and 6, where we describe our original contributions and findings. These chapters outlines three distinct versions of the problem, proposes algorithms for solving them, and analyzes their hyperparameters. We also discuss the relative performance of these algorithms, supported by empirical results. The document complementing this chapter by showcasing all the results from our testing phase can be downloaded from https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf.

Finally, Chapter 7 concludes the thesis with some closing remarks and identifies potential areas for future research, aiming to further advance the field.

1.7 Chapter conclusion

In this introductory chapter, we delve into the driving motivation behind this thesis. We explore a range of alternative fuel vehicles, examining their respective benefits and drawbacks. Additionally, we lay the groundwork by introducing key concepts in green logistics and operations research, highlighting the interplay between these two vital fields. We also define various optimization problems, referencing some of the more renowned challenges in the field.

Our discussion extends to the diverse methodologies available to tackle these problems, differentiating between exact and approximate algorithms and categorizing the latter into distinct groups. This foundational chapter sets the stage for a deeper exploration of the topics at hand, establishing a comprehensive context for the thesis.

CHAPTER
TWO

METAHEURISTICS OVERVIEW

In the vast realm of optimization and search, metaheuristics stand out as a powerful class of algorithms designed to find high-quality solutions to complex problems, especially those for which traditional methods are either inapplicable or inefficient [50, 328]. The term '*metaheuristic*' is derived from the Greek words '*meta*' meaning '*higher*' or '*beyond*', and '*heuriskein*' meaning '*to search*' or '*to find*'. Thus, metaheuristics can be interpreted as high-level problem-independent strategies aimed at guiding other heuristics to produce the best possible solution within a reasonable computation time.

Metaheuristics are particularly useful for solving combinatorial optimization problems, where the search space is enormous, and exhaustive search is computationally expensive. These methods are characterized by their flexibility and general applicability. Unlike exact algorithms, which guarantee an optimal solution, metaheuristics provide approximate solutions, meaning that the quality of a solution can not be guaranteed. However, in many real-world scenarios, they can rapidly yield solutions that are "good enough" or close to optimal.

Popular metaheuristic approaches include Genetic Algorithms, Simulated Annealing, Tabu Search, Ant Colony Optimization, and Particle Swarm Optimization, among others. Most of these techniques is inspired by different natural or artificial processes. For instance, Genetic Algorithms draw inspiration from biological evolution, while Simulated Annealing is inspired by the annealing process in metallurgy.

2.1 Classification of metaheuristics

As previously mentioned, metaheuristics serve as versatile strategies capable of identifying satisfactory solutions for challenging optimization problems, even if they might forgo guarantees of optimality. Notably, these metaheuristics possess distinct attributes that allow for their classification into specific categories. Recognizing and understanding these categorizations is crucial, as the inherent characteristics of metaheuristics within one category might render them more suited to particular problems compared to those from another category.

Birattari et al. [34] pioneered one of the earliest classification frameworks for metaheuristics, a system which laid foundational groundwork for many subsequent classifications. Their scheme highlighted six core attributes of metaheuristics:

- Trajectory versus discontinuous methods: Trajectory methods methodically explore the solution space through a continuous path of incremental changes, while discontinuous methods jump across different regions of the search space.

- Population-based versus single-point search approaches: Population-based methods explore multiple solutions concurrently, whereas single-point methods iteratively refine a single solution.
- Memory usage versus memoryless techniques: Metaheuristics that use memory retain and utilize past solution information to influence future search directions, while memoryless metaheuristics operate independently of previous solutions, making each decision without reference to past states.
- Singular versus multiple neighborhood structures: Metaheuristics with a singular neighborhood structure rely on a consistent set of rules for solution exploration, whereas those with multiple structures dynamically switch between different rule sets.
- Dynamic versus static fitness functions: Metaheuristics with dynamic objective functions adjust their optimization criteria in response to changing conditions, while those with static objective functions consistently pursue a fixed set of criteria throughout the optimization process.
- Nature-inspired versus non-nature-inspired strategies: Nature-inspired metaheuristics are algorithms modeled after natural phenomena and biological behaviors, while non-nature inspired metaheuristics are developed from mathematical or algorithmic principles without direct reference to natural processes.

Other researchers have explored different attributes. For instance, the study by Fausto et al. [91] delved into five specific attributes affecting the performance of nature-inspired algorithms. These include:

- Exploration/exploitation mechanisms (encompassing selection mechanisms, attraction operators, and iteration dependencies)
- Computational complexity (considering population sorting, population-related measurements, and variable fitness function computation)
- Memory requirements
- Parameter tuning methodologies
- Overall implementation challenges.

A comprehensive review of metaheuristic classification is further detailed in Stegherr, Heider, and Hähner [319].

2.2 Existing metaheuristics

Over the years, a diverse array of metaheuristics has emerged, drawing inspiration from a multitude of sources. It is important to acknowledge that according to the *no-free-lunch theorem*, no single metaheuristic algorithm is universally effective for solving all optimization problems [362]. This implies that while certain algorithms may excel in specific problem classes, their effectiveness may diminish in other problem domains. Consequently, a diverse array of metaheuristics has evolved to address this variability. However, this abundance of options has simultaneously made the task of identifying the most suitable metaheuristic for a particular problem more challenging.

We have compiled a detailed catalog of metaheuristics, utilizing a variety of resources and survey publications, notably including references [1, 340]. This comprehensive catalog is available in a separate document, which can be downloaded from the following link: `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixA.pdf`. This process involved correcting inaccuracies found in these sources and incorporating numerous unlisted metaheuristics. The table from the aforementioned document presents a detailed inventory of 540 metaheuristics, specifying their names, publication years, abbreviated list of authors, and the inspirations behind each. It is important to acknowledge that our list may not be exhaustive. There are metaheuristics that might have been overlooked, and some were deliberately excluded due to their similarity to others already included. For instance, we omitted "Reduced Variable Neighborhood Search" because it is a variation of the "Variable Neighborhood Search" algorithm already listed.

The differentiation between metaheuristics can sometimes be subtle, especially when they are technically similar yet distinct in name and presentation. For example, many nature-inspired population-based metaheuristics share the Genetic Algorithm's core concepts like crossover and selection, yet are recognized as separate entities. To address this ambiguity, we followed a straightforward criterion: if an algorithm was introduced by its authors as a new metaheuristic, it was included in our list. It is also worth noting that matheuristics are not part of this compilation, as they are hybrid methods reliant on external solvers.

Figure 2.1 displays a histogram that illustrates the annual emergence of new metaheuristics. This visual representation clearly indicates an upward trend in the creation of new metaheuristics since the year 2000. This influx has been instrumental in introducing innovative concepts, significantly enriching the field of metaheuristics. However, there is a growing concern among some scholars about the proliferation of metaheuristics that primarily offer novel metaphors rather than substantial conceptual advancements [14]. This trend, while highlighting the field's dynamism, raises questions about the depth and novelty of contributions within these new metaheuristics.

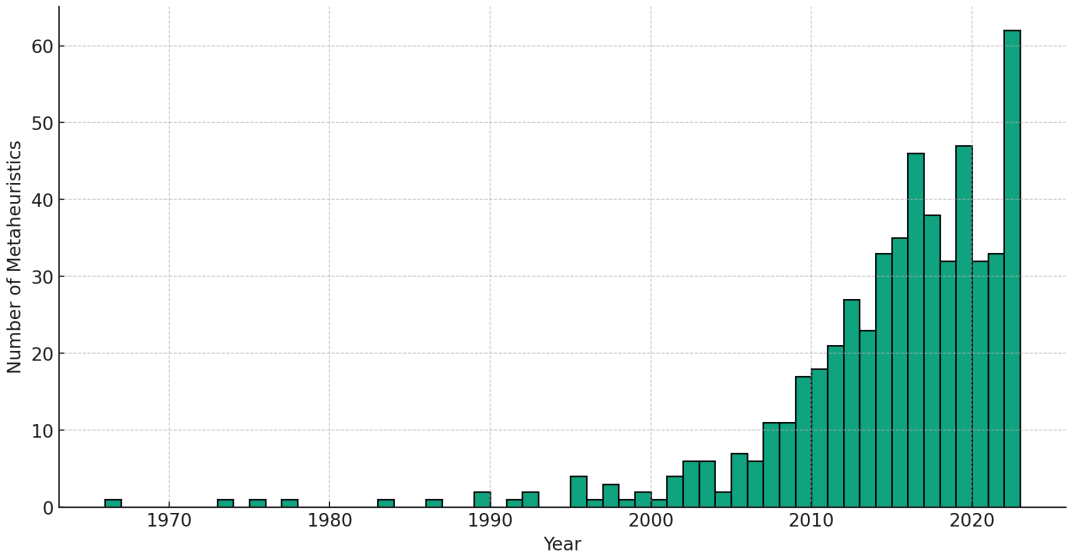

FIGURE 2.1: Number of new metaheuristics introduced per year.

In our research, we organized metaheuristics into 14 distinct categories, each defined by its source of inspiration. The **Animal** category encompasses metaheuristics inspired by animal behaviors, while the **Plant** category is dedicated to those influenced by plant characteristics. The **Organism** category includes metaheuristics inspired by entities not classified as animals or plants, such as fungi, bacteria, or viruses. The **Biology** category encompasses metaheuristics that draw inspiration from a broad spectrum of biological processes and concepts, without being specifically tied to any particular organism. **Human**-based metaheuristics derive from human behaviors or interactions, and the **Culture** category captures those influenced by specific cultural elements like cinema or traditions. **Physics** encompasses metaheuristics grounded in physical phenomena or scientific laws, and **Chemistry** covers metaheuristics based on the laws of chemistry. Other categories include: **Mathematics** for those based on mathematical principles, **Music** for those derived from music theory, **Sport/Game** for those influenced by social games, games of chance, and sports. **Technology** encompasses metaheuristics rooted in modern technological concepts like software development or transportation. **Geography** refers to those inspired by geographical elements, while **Non-metaphor** includes metaheuristics developed independently of real-world analogies.

It is important to note that categorizing metaheuristics is not always straightforward. Often, a metaheuristic may blend inspirations from multiple sources, or its categorization may be ambiguous. For instance, an algorithm inspired by the behavior of Hitchcock's birds could be classified under *Animal* due to its animalistic influence, but also under *Culture* given its cinematic origin. Similarly, a zombie-based algorithm could fall into the *Organism* category for its basis in fictional organisms, while also fitting into *Culture* due to its roots in tradition and popular media.

Figure 2.2 presents a clear distribution of metaheuristics based on their sources of inspiration. The most predominant category is that of animal-inspired metaheuristics, with physics-inspired metaheuristics being the second largest category. This trend becomes even more pronounced when examining the metaheuristics published in the last five years, as depicted in Figure 2.3.

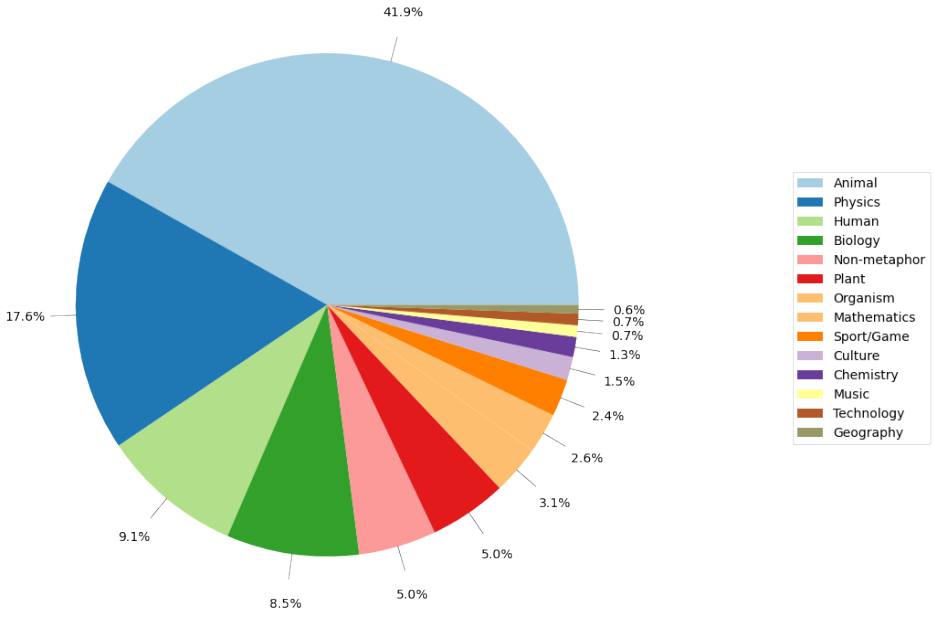

FIGURE 2.2: Distribution of metaheuristics by inspiration.

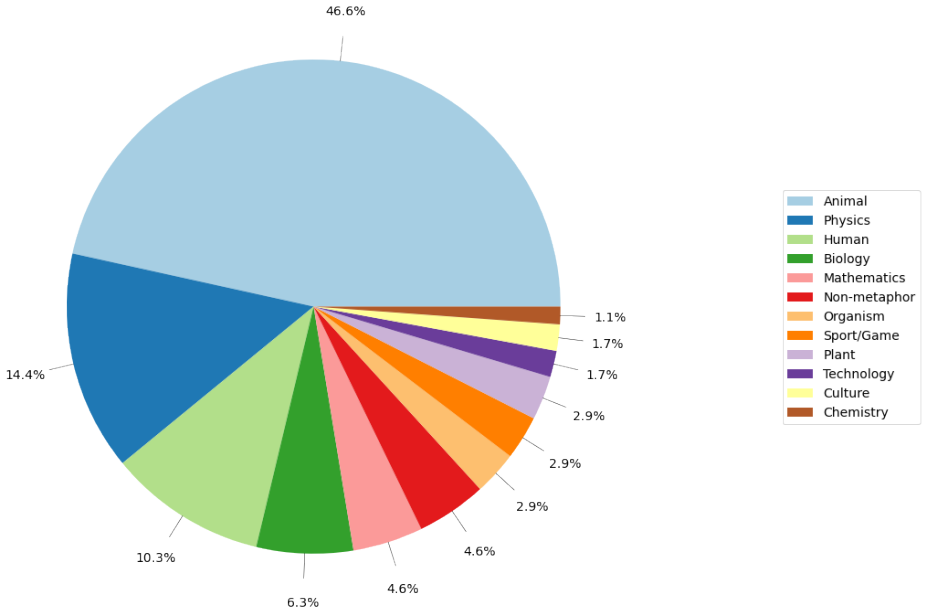

FIGURE 2.3: Distribution of metaheuristics by inspiration (Last 5 years).

2.3 Local search heuristics

Local search techniques are pivotal in the realm of optimization, forming the backbone of numerous metaheuristic strategies. Their primary function is to enhance a solution through a process of iterative improvement, examining nearby solutions to discover the ones of higher quality. These methods involve minor, step-by-step modifications to the existing solution, thereby methodically traversing the solution space in pursuit of better solutions. Two prevalent tactics employed in guiding this search process are the *first improvement* and *best improvement* methods.

- **First Improvement Strategy:** In the first improvement strategy, the algorithm searches through the neighboring solutions of the current state. As soon as it identifies a neighbor that is better than the current solution (according to the problem's objective), it moves to that neighbor without further examining the rest of the neighborhood. The advantage of this approach is speed: it can rapidly traverse the solution landscape since it does not require a comprehensive search of the entire neighborhood at each step. However, the downside is that it might miss an even better solution in the neighborhood since it stops searching after the first improvement is found.

- **Best Improvement Strategy:** On the other hand, the best improvement strategy is more exhaustive in its search. Instead of settling for the first improvement it finds, the algorithm evaluates all neighboring solutions in the current state's neighborhood. It then selects the best among them (the one that offers the most significant improvement) and moves to that state. This strategy can be computationally more intensive, especially if the neighborhood is large, as it requires a full exploration of all neighbors. However, it tends to find higher-quality local optima than the first improvement strategy since it always chooses the best available move in the neighborhood.

In practice, the choice between first improvement and best improvement often depends on the specific problem, its characteristics, and the computational resources available. Some problems may benefit more from the rapid exploration of the solution space offered by the first improvement, while others might benefit from the depth and quality of solutions found using the best improvement strategy.

These two strategies can be effectively combined, as demonstrated in various studies. For instance, in the research presented by Karakostas, Sifaleras, and Georgiadis [159], the choice of strategy dynamically depends on the instance size. Smaller instances are tackled using the best-improvement strategy, while larger instances are approached with the first-improvement strategy. This approach leverages the strengths of both strategies depending on the complexity of the problem at hand.

In another study by Ren et al. [286], the authors adopted a probabilistic approach to selecting between these strategies. Initially, the best-improvement strategy is more likely to be chosen, reflecting the higher probability of finding superior solutions in the early stages of the algorithm. As time progresses, the likelihood shifts towards the first-improvement strategy. This shift is based on the rationale that, over time, the probability of discovering significantly better solutions diminishes, making it more efficient to opt for the first-improvement strategy. This approach helps to avoid extensive searches in local optima and enhances the overall efficiency of the algorithm in navigating the solution space.

There exists a wide array of local search heuristics, each with its unique applications and methodologies. In our discussion, we present several foundational heuristics, as detailed in [294]. For an in-depth exploration of these techniques and a broader range of search methods, we highly recommend consulting this comprehensive resource.

- **Hill Climbing (Steepest Ascent/Descent)**: At each step, it evaluates all neighbors and chooses the best one as the next current solution. *Ascent* seeks to maximize the objective function, while *Descent* seeks to minimize it.
- **First-Choice Hill Climbing**: A variant of hill climbing where the first improving neighbor is chosen without checking all other neighbors.
- **Random Walk**: A local search where the algorithm picks a random neighbor to move to in each iteration, regardless of whether it improves the solution.
- **Random Restart Hill Climbing**: Hill climbing that restarts from a random position whenever it gets stuck in a local optimum.
- **Gradient Descent/Ascent (for continuous spaces)**: Uses the gradient of the objective function to guide the search. It is a first-order iterative optimization algorithm for finding a local minimum (or maximum) of a differentiable scalar function.
- **Stochastic Hill Climbing**: A variant of hill climbing where the next move is not always to the best neighbor. Instead, a random neighbor is chosen with a probability related to how much it improves the objective function.
- **Iterative Deepening**: Typically used in the context of search trees in artificial intelligence and game playing. It involves a series of depth-first searches with increasing depth limits.
- **Beam Search**: Again, often used in tree searches. It is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set or "beam" of possibilities.

- **Bidirectional Search**: Conducts two simultaneous searches: one forward from the initial state and one backward from the goal, hoping the two searches meet in the middle.
- **Constraint Propagation**: Typically used in constraint satisfaction problems. It involves refining domains of variables based on constraints, reducing the search space.

2.4 Review of some metaheuristics

In this section, we aim to present several metaheuristics relevant to this work in greater detail.

2.4.1 Variable Neighborhood Search

Variable Neighborhood Search (**VNS**) was originally introduced by Mladenović and Hansen [236, 238]. At its core, VNS is a straightforward but potent approach for finding approximate solutions to challenging optimization issues. It systematically varies neighborhoods to navigate the problem's search space, grounded in three primary principles:

- What is considered a local optimum within one neighborhood structure might not hold that status in another structure.
- The global optimum remains a local optimum across all neighborhood structures.
- For many problems, the majority of local optima tend to be in close proximity to one another [161].

The first point validates the use of multiple neighborhoods. The second observation suggests that if a solution is not a local optimum within a certain neighborhood, it cannot be the global optimum. This further supports the use of diverse neighborhoods. However, it does not assure that a solution identified as a local optimum across all utilized neighborhoods is the definitive global optimum [66]. The third point, while empirical rather than theoretical, endorses a thorough exploration of the immediate vicinity of the local optimum. There is a significant chance of discovering an even superior solution within that zone. If this principle does not hold true for a particular problem, there are VNS variations available that are not heavily dependent on this insight.

However, employing multiple neighborhoods can prompt several questions [123]:

- **Neighborhood Structure Selection**: This is influenced by the metrics suitable for the specific problem at hand.
- **Neighborhood Ordering**: The order should reflect an increasing distance between the current solution x and the solutions x_k within $N_k(x)$, where k ranges from 1 to k_{max}.
- **Neighborhood Size**: Given constraints, only neighbors with the potential to enhance the current solution should be identified and assessed.
- **Search Strategies**: Options include *first improvement* and *best improvement*, already detailed in Section 2.3.

- **Descent Approaches**: If a superior solution is not found, a decision needs to be made on accepting an ascent move, which alters the neighborhood and/or solution.
- **Exploration Directions**: This dictates the sequence in which neighborhoods are explored. In forward VNS, k progresses from 1 to k_{max}, while in backward VNS, it is the opposite: from k_{max} down to 1.
- **Search Intensity and Variety**: In the realm of local search-based metaheuristics, strategic decisions must be made about when to delve deeper into promising regions (intensification) and when to explore new areas (diversification). The neighborhoods selected for these respective processes can vary and are not bound to be identical, as demonstrated in the study by Matijević [220].

The basic VNS algorithm's pseudocode is outlined in Algorithm 1. From this pseudocode, it is evident that the VNS metaheuristic is structured around three primary components:

1. **Shaking Phase**: This component introduces randomness into the search process. By moving away from the current solution to a different region in the search space, the algorithm avoids getting trapped in local optima. By systematically changing neighborhoods, the shaking phase ensures a diversified search.
2. **Local Exploration**: After the shaking phase, this component refines the solutions by performing a more intensive and focused search within the neighborhood of the solution obtained from the shaking phase. The goal is to find a local optimum in the current neighborhood, which might potentially be better than the global best known solution.
3. **Decision on Movement**: Based on the quality of the solution found in the local exploration phase, this component decides whether to move to the new solution or stay with the current one. If the new solution is better, the algorithm updates its current position; otherwise, it might decide to shake the best-found solution again and explore a different part of the search space.

Algorithm 1 Variable Neighborhood Search

```
1: procedure VNS(k_min, k_max, stopping criterion)
2:     x_best ← initial_solution()
3:     while stopping criterion is not met do
4:         k ← k_min
5:         while k < k_max do
6:             x' ← SHAKE(x_best, k)
7:             x'' ← LOCAL_SEARCH(x')
8:             x ← MOVE_OR_NOT(x, x'', k)
9:         end while
10:    end while
11:    return x_best
12: end procedure
```

Figure 2.4 provides a visual depiction of the algorithm's flow. Initially, we begin with an initial solution, which we promptly designate as the current optimal solution, termed as X_{min}. The shaking phase is then initiated within the neighborhood

$\mathcal{N}_1(x_{min})$, symbolized by selecting a random solution X' from the circle labeled $\mathcal{N}_1$. Subsequently, we refine X' through the local search process. However, if this fails to uncover a better solution, the exploration advances to the next neighborhood. While the $\mathcal{N}_2$ neighborhood does not yield a superior solution, the $\mathcal{N}_3$ neighborhood does, leading to the discovery of a solution better than X_{min}. This new solution then becomes our updated X_{min}, and the entire procedure restarts based on this new reference point.

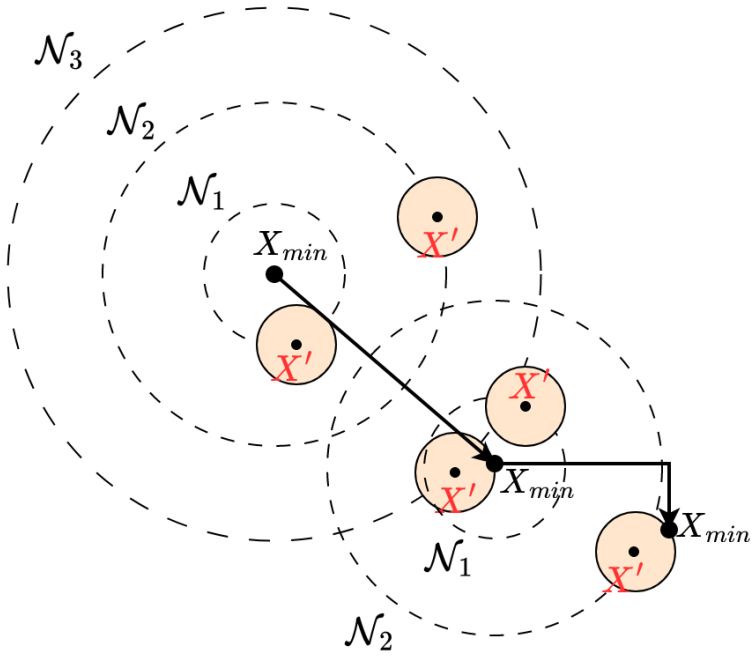

FIGURE 2.4: Graphic representation of the VNS algorithm.

VNS has been successfully applied to a wide array of problems, encompassing diverse areas such as the *p-median problem* [124], *vehicle scheduling* [13], *berth allocation problem* [167], *traveling salesman problem* [237], *maximal covering problem* [149], *patient scheduling* [280], *hub location problem* [231], *lot sizing* [311], *group steiner tree problem* [222], *capacitated clustering problem* [42], *maximum diverse grouping problem* [343], *hub location problem* [231], healthcare management [180], graph problems [344], *vehicle routing problem* [224], and many others. For more information on VNS, its variants and applications please refer to [43, 125, 126].

2.4.2 Greedy Randomized Adaptive Search Procedure

Greedy Randomized Adaptive Search Procedure (**GRASP**) is a simple, yet effective metaheuristic originally proposed by Feo and Resende [93]. GRASP operates in iterations, with each iteration consisting of two main phases:

1. **Construction Phase**: In this stage, a feasible solution is constructed piece-by-piece. Instead of following a purely greedy criterion, which might quickly lead to suboptimal solutions, GRASP adds a probabilistic element to it. During the solution assembly, a *restricted candidate list* (**RCL**) is formed. RCL is constructed by selecting a subset of the best available components based on a predefined quality criterion, thus ensuring that choices are both high-quality

and varied. The size or quality threshold of the RCL can be adjusted, offering a trade-off between exploration (larger or looser RCL, more randomness) and exploitation (smaller or stricter RCL, more greediness). By maintaining this dynamic shortlist of promising options and picking from it randomly, the RCL mechanism in GRASP encourages diverse solution construction while still guiding the search towards promising regions of the solution space. After the creation of RCL, a component is randomly selected from it and added to the solution. This process continues until a complete solution is formed. The randomized selection ensures diverse solutions across iterations.

2. **Local Search Phase**: After building a solution, GRASP refines it using local search. Starting from the initially constructed solution, neighboring solutions are explored to find improvements. The algorithm seeks to move from the current solution to a better neighboring one until no further improvements can be found, indicating a local optimum.

The algorithm repeats these two phases for a predefined number of iterations or until a stopping criterion is met, like a time limit or a satisfactory solution quality. The best solution encountered across all iterations is returned as the final solution. The pseudocode for the basic GRASP is provided in Algorithm 2, with the corresponding construction procedure presented in Algorithm 3. More information about GRASP can be found in [288].

Algorithm 2 Greedy Randomized Adaptive Search Procedure

```
1: procedure GRASP(stopping criterion)
2:     x_best ← NULL                                   ▷ f(NULL) = ∞
3:     while stopping criterion is not met do
4:         x ← greedy_randomized_construction()        ▷ Algorithm 3
5:         x' ← local_search(x)
6:         if f(x') < f(x_best) then
7:             x_best ← x'
8:         end if
9:     end while
10:    return x_best
11: end procedure
```

Algorithm 3 Basic construction procedure for GRASP

```
1: procedure GREEDY_RANDOMIZED_CONSTRUCTION
2:     solution ← an empty solution
3:     while solution is not complete do
4:         RCL ← construct_RCL()
5:         element ← choose_element_at_random(RCL)
6:         solution ← solution ∪ element
7:     end while
8:     return solution
9: end procedure
```

GRASP has been effectively utilized in solving a wide range of problems, such as *portfolio optimization* [27], *humanitarian aid distribution* [96], *flowshop scheduling* [120], *the cutting stock problem* [234], *the quadratic assignment problem* [190], *set*

packing [70], *the facility location problem* [246], *electric bus scheduling* [152], *symbolic regression* [151], and numerous others.

2.4.3 Ant Colony Optimization

Ant Colony Optimization (**ACO**) is a metaheuristic inspired by nature, introduced by Marco Dorigo in the early 1990s [74, 76, 77]. This algorithm draws from the collective intelligence displayed by ants when foraging. In their natural habitats, ants lay down chemical markers called pheromones as they move, creating pathways that fade over time. Shorter routes between the colony and food sources allow ants to move faster, leading to more frequent pheromone deposits. As a result, other ants are inclined to follow paths with stronger pheromone concentrations, reinforcing these trails even further. The ACO algorithm harnesses these principles through two primary phases: the *construction phase* and the *pheromone update phase*. In certain instances, the algorithm incorporates an additional phase known as the *improvement phase*. This phase typically conducts a local search around the optimal solution derived during the construction phase.

The *solution construction phase* is performed according to the probabilistic state transition rule. Ants can be viewed as stochastic greedy procedures that construct a solution in a probabilistic manner, by adding solution components to the partial solution until a complete solution is obtained. This iterative process takes into account:

- *Pheromone trail* - that retains the characteristics of a well-generated solution, which will guide the construction of new solutions. The pheromone trail dynamically changes during the search to reflect the acquired knowledge. It represents the memory of the entire search process.
- *Heuristic information* - problem-specific information further guides ants in the construction of a solution.

Each ant stochastically constructs a solution. Starting from a randomly selected point i, an ant chooses the next point j based on probability:

$$p_{ij} = \frac{\tau_{ij}}{\sum_{k \in S} \tau_{ik}}, \quad \forall j \in S \tag{2.1}$$

Here, S denotes the unvisited points in the search, and τ_{ij} represents the pheromone level. Pheromone values are set above zero initially, ensuring that every path has a chance to be selected. To guard against paths becoming inaccessible due to evaporation, it is important to set a minimum pheromone value, preventing values from dropping to zero. During the construction phase, ants select their starting points at random.

We can define an aditional heuristic to further guide the search depending on the problem. In general, we would define heuristic values $\eta_{ij} = 1/d_{ij}$, where d_{ij} represents the distance between points i and j. In other words, the smaller the distance between points i and j, the greater the heuristic value η_{ij} will be. Therefore, we can expand Equation (2.1) with this heuristic information:

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \eta_{ik}^{\beta}}, \quad \forall j \in S \tag{2.2}$$

where α and β are parameters that determine the relative contribution of the pheromone value and heuristic values. If $\alpha = 0$, the ACO algorithm will be similar

to the stochastic greedy algorithm, where closer points will have a higher probability of being selected. If $\beta = 0$, only pheromone values will guide the search, which may lead to a rapid convergence to the local best solution.

In the *pheromone update phase*, the values of pheromones are updated based on the quality of constructed solutions. This process consists of two steps:

1. *Evaporation phase* - the pheromone trail naturally evaporates. Each pheromone value is reduced according to the formula:

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \quad \forall i, j \in \mathcal{P} \tag{2.3}$$

   where $\rho \in [0, 1]$ represents the evaporation rate of the pheromone, and $\mathcal{P}$ is the set of all points. The purpose of evaporation is to prevent premature convergence.

2. *Reinforcement phase* - the pheromone trail is updated based on the generated solutions. Multiple strategies can be applied:

   - *Online step-by-step update*: The pheromone trail τ_{ij} is updated by the ant at every step of the solution construction.
   - *Online delayed update*: The pheromone update is done once the ant finishes generating a complete solution. Each ant updates the pheromone information with a value proportional to the quality of the found solution. The better the solution, the more pheromones the ant will leave.
   - *Offline pheromone update*: The pheromone trail is updated when all ants have finished constructing solutions.

Algorithm 4 provides a pseudocode representation of the standard ACO utilizing offline pheromone updates. For a deeper exploration of ACO, readers are encouraged to consult the extensive literature on the topic, including references [75, 78, 270].

Algorithm 4 Ant Colony Optimization

procedure ACO(Population size N, α, β, ρ, stopping criterion)
  $\tau \leftarrow$ Initialize pheromone trails
  $x_{best} \leftarrow NULL$ ▷ $f(NULL) = \infty$
  while stopping criterion is not met **do**
    $solutions \leftarrow \emptyset$
    for $i \leftarrow 1$ **to** N **do**
      $x \leftarrow construct\_solution(\tau, \alpha, \beta)$
      $solutions \leftarrow solutions \cup x$
    end for
    if $f(best(solutions)) < f(x_{best})$ **then**
      $x_{best} \leftarrow best(solutions)$
    end if
    $\tau \leftarrow update\_pheromones(solutions, \rho)$
  end while
  return x_{best}
end procedure

ACO has demonstrated successful application across a diverse array of challenges, encompassing areas such as the *traveling salesman problem* [321], *machine-part cell*

formation [192], *the shortest path problem* [107], *academic curriculum planning* [292], *virtual machine allocation and data placement optimization* [306], *image classification* [60], and several other domains.

2.4.4 Bee Colony Optimization

Bee Colony Optimization (**BCO**) is another nature-inspired metaheuristic, proposed by Lučić and Teodorović [202, 203, 204]. The basic idea is rooted in the way honey bees search for food sources, communicate with their hive mates about these sources, and decide on which sources to exploit based on their quality. Beyond BCO, numerous metaheuristics draw inspiration from honey bee foraging, with the *Artificial Bee Colony* [157] likely being the best known. For a deeper understanding of the BCO algorithm, its variations, and its use-cases, refer to references [65, 148, 330, 331].

The BCO utilizes a population of artificial bees, with each bee representing a unique solution to the problem. An iteration of the BCO algorithm encompasses two phases:

- *Forward pass* - In this stage, artificial bees traverse the search space, executing a specified number of moves that either construct or enhance partial or complete solutions, resulting in new potential solutions.
- *Backward pass* - In this stage, all artificial bees communicate the efficacy of their current partial or complete solutions. After evaluating all solutions, each bee probabilistically determines if it will remain committed to its current solution. Bees with superior solutions are more likely to retain and promote their findings. If a bee abandons its solution, it enters an uncommitted state and must choose from one of the promoted solutions. This choice is probability-driven, favoring the more promising solutions for continued exploration.

The decision of loyalty for the $i-th$ bee is influenced by the quality of its solution in comparison to all other solutions. The likelihood that it remains committed to its previously created partial or complete solution can be described as follows [65]:

$$p_i^{k+1} = e^{-\frac{O_{max}-O_i}{k}}, \quad \forall i \in B \tag{2.4}$$

In equation (2.4), O_i signifies the normalized value of the objective function for the solution devised by the i-th bee. O_{max} is the maximum value among all normalized values for solutions under comparison. The variable k represents the number the forward passes (moves), while B is the set of all bees in population.

Bees that choose to remain loyal to their solution take on the role of recruiters. In contrast, those who abandon their solution will select an alternative from loyal bees based on a probability [65]:

$$p_i = \frac{O_i}{\sum_{j=1}^{R} Oj}, \quad \forall i \in R \tag{2.5}$$

where R is the set of loyal bees.
BCO has two main variants:

- *Constructive BCO* - This is BCO in its basic form. For every bee, a solution was methodically built from the ground up using certain stochastic, problem-tailored heuristic rules. The inherent randomness of these construction methodologies ensured a diverse exploration of the search space.

- *Improvement-based BCO* (**BCOi**) - BCOi, introduced by Davidović et al. [63], deviates from the traditional BCO. Instead of constructing solutions from the ground up, bees enhance previously generated solutions. The pseudocode for BCOi is provided in Algorithm 5.

Algorithm 5 Bee Colony Optimization

```
1: procedure BCOi(Population size N, max_moves, stopping criterion)
2:     bees ← initialize_bees(N)
3:     best_solution ← find_best(bees)
4:     while stopping criterion is not met do
5:         for num_moves ← 1 to max_moves do
6:             for ∀b ∈ bees do                                  ▷ Forward pass
7:                 transform(b, num_moves)
8:                 value ← evaluate(b)
9:                 if value > evaluate(best_solution) then
10:                    best_solution ← b
11:                end if
12:            end for
13:            for ∀b ∈ bees do                                  ▷ Backward pass
14:                decide_loyalty(b)
15:            end for
16:            recruitment(bees)
17:        end for
18:    end while
19:    return best_solution
20: end procedure
```

BCO has been effectively implemented in solving a variety of problems, including the *p-center problem* [63], *task scheduling* [64], the *satisfiability problem* [320], *transit network design* [252], *backup allocation issues* [250], among many others.

2.4.5 Genetic and Memetic Algorithm

Genetic Algorithm (**GA**) [137] is a population-based metaheuristic that mimics the process of natural selection (Algorithm 6). This algorithm mirrors the process of natural evolution, leveraging principles of genetics, such as inheritance, mutation, selection, and crossover. The basic algorithm employs a collection of potential solutions (referred to as the population) for a specific problem. Each of these solutions (chromosomes) possesses certain attributes (known as genotypes). Throughout the algorithm's iterations, various genetic operations can be applied:

- *Encoding and decoding* - The first step in a GA involves encoding the potential solutions to the problem. Solutions, typically referred to as *chromosomes*, can be represented in various ways, such as binary strings, real numbers, or even more complex data structures.

- *Selection* - Selection simulates the survival of the fittest. In this phase, chromosomes are selected to form a new generation. The likelihood of a chromosome being selected is often (but not always) proportional to its fitness, which is determined by a fitness function. Several selection methods exist, such as roulette-wheel selection, tournament selection, and rank-based selection. The goal is to

give preference to the better-performing chromosomes while still retaining some diversity.

- *Crossover* - Once the selection is made, the crossover operation is applied to pairs of 'parent' solutions, producing 'offspring' that inherit features from both parents. The hope is that the combination of good features from two parents will produce an even better offspring. There are various techniques for crossover, depending on the encoding of solutions. For instance, with binary strings, one-point, two-point, or uniform crossovers might be applied.

- *Mutation* - After crossover, mutation is applied with a certain probability. This involves making small random changes to the offspring. In a binary string representation, mutation might flip a bit from 0 to 1 or vice versa. Mutation introduces variability within the population, ensuring that the genetic diversity remains high. This helps in preventing premature convergence to sub-optimal solutions.

- *Replacement* - The final step of a generation involves selecting which individuals will move to the next generation. There are different strategies for this: the generational approach, where an entirely new generation replaces the old, or the steady-state approach, where only a few new offspring replace old members.

Genetic algorithms have been extensively discussed in various books and articles. For a deeper understanding, readers are encouraged to consult references [4, 7, 168, 314].

Algorithm 6 Genetic Algorithm

```
1: procedure GA(Population size N, Crossover rate CR, Mutation rate MR, stopping criterion)
2:     Initialize a population of N individuals
3:     while stopping criterion is not met do
4:         for each individual in the population do
5:             Evaluate its fitness
6:         end for
7:         Let new_population be empty
8:         while size of new_population < N do
9:             Select two parents based on their fitness
10:            Generate a random number (r)
11:            if r < CR then
12:                Perform crossover on the parents to produce offspring
13:            else
14:                offspring = parents
15:            end if
16:            for each offspring do
17:                Generate a random number (m)
18:                if m < MR then
19:                    Mutate the offspring
20:                end if
21:            end for
22:            Add offspring to new_population
23:        end while
24:        Replace the old population with new_population
25:    end while
26:    return The best solution from the final population
27: end procedure
```

A variation of GA called *Memetic Algorithm* (**MA**) was proposed by Moscato [247]. The distinct aspect of MAs is the incorporation of a local search (or refinement) step within the evolutionary cycle. This local search is sometimes viewed as an individual learning process or the cultural transmission of knowledge. Each individual undergoes a learning process which refines its quality. The pseudocode for the MA is outlined in Algorithm 7. The key distinction from Algorithm 6 is evident in Line 21, where a local search is applied to refine each offspring. For more details about MA please refer to [251].

Algorithm 7 Memetic Algorithm

```
1: procedure MA(Population size N, Crossover rate CR, Mutation rate MR, stopping criterion)
2:     Initialize a population of N individuals
3:     while stopping criterion is not met do
4:         for each individual in the population do
5:             Evaluate its fitness
6:         end for
7:         Let new_population be empty
8:         while size of new_population < N do
9:             Select two parents based on their fitness
10:            Generate a random number (r)
11:            if r < CR then
12:                Perform crossover on the parents to produce offspring
13:            else
14:                offspring = parents
15:            end if
16:            for each offspring do
17:                Generate a random number (m)
18:                if m < MR then
19:                    Mutate the offspring
20:                end if
21:                Refine the offspring using local search
22:            end for
23:            Add offspring to new_population
24:        end while
25:        Replace the old population with new_population
26:    end while
27:    return The best solution from the final population
28: end procedure
```

GA is one of the most widely utilized metaheuristics, applied to an extensive range of problems. Its applications include *robot path planning* [179], *image processing* [118], *managing real-time control systems* [300], *job shop scheduling* [209], *optimizing bank lending decisions* [228], *automatic design of convolutional neural networks* [325], *deep reinforcement learning* [304], *open shops scheduling* [2], and numerous other areas. Similarly, the MA has been employed in various domains, such as optimizing *multilayer optical films* [310], *healthcare routing and scheduling* [69], *graph coloring* [239], *network topology optimization* [99], *job-shop scheduling* [117], *traffic signal optimization* [296], and many more.

2.5 Parameter tuning

Fine-tuning the parameters of a metaheuristic is crucial for optimizing its performance, a process often referred to as *parameter tuning*. The effectiveness of the algorithm hinges significantly on the values of these parameters. Therefore, we explore basic methodologies for setting these parameter values in the following sections. For an in-depth understanding of various parameter tuning strategies, the work of Joy, Huyck, and Yang [153] offers a thorough analysis and guidance.

2.5.1 Brute-force

Let us allocate a total time, T, for the tuning process, where each experiment takes up a duration t. Given these parameters, we can conduct $M = \lfloor \frac{T}{t} \rfloor$ experiments.

Moreover, we have a set of potential configurations, Θ, and a collection of instances, I. Each configuration from Θ is evaluated $N = \left\lfloor \frac{M}{|\Theta|} \right\rfloor$ times. This provides us with performance metrics $\hat{\mu}(\Theta_1), \hat{\mu}(\Theta_2), \ldots, \hat{\mu}(\Theta_{|\Theta|})$. From this, we can construct a brute-force algorithm. We first use a distribution P_I to randomly select N instances, which we will use to test all of the given candidate configurations. From these results, we can estimate the performance $\hat{\mu}(\Theta_i)$ of each candidate configuration Θ_i in the set Θ, in relation to the distribution P_I. Ultimately, we pick the configuration Θ_i that exhibits the top performance. Details of this method can be found in Algorithm 8.

Algorithm 8 Brute-force algorithm for parameter tuning

```
1: procedure BRUTE_FORCE(M, Θ, P_I)
2:     N ← ⌊M / |Θ|⌋
3:     A ← allocate_array(|Θ|)                ▷ A stores performance estimates
4:     for k ← 1 to N do
5:         i ← sample_instance(P_I)
6:         for θ ∈ Θ do
7:             s ← run_experiment(θ, i)
8:             c ← evaluate(s)
9:             update_mean(A[θ], c)
10:        end for
11:    end for
12:    return arg min(A)                      ▷ Returns θ with minimal value of A[θ]
13: end procedure
```

The brute-force method presents several limitations:

- The training set's size must be predetermined, posing a challenge. An undersized set might yield inaccurate results, whereas an oversized one can generate redundant computations.
- The required number of tests for each configuration and instance, essential for addressing the unpredictability inherent in metaheuristics, remains ambiguous.
- Every configuration, irrespective of its potential efficacy, is allocated an equal share of resources.

Considering these constraints, the racing approach emerges as a more viable alternative.

2.5.2 The Racing Approach

This algorithm family, initially presented in Birattari et al. [35], drew inspiration from the *Hoeffding race algorithm* featured in Maron and Moore [215]. Their core principle is the efficient allocation of resources across potential configurations. Racing algorithms assess these configurations and, through statistical analysis, eliminate the less promising ones. This approach enables the algorithm to evaluate the more promising configurations over a wider range of instances. The general structure of the racing algorithm is detailed in Algorithm 9.

Algorithm 9 Brute-force algorithm for parameter tuning

```
1: procedure RACE(M, Θ, P_I, stat_test, max_instances)
2:     ESF ← 0                                          ▷ Experiments so far
3:     ISF ← 0                                          ▷ Instances so far
4:     C ← allocate_array(max_instances, |Θ|)
5:     S ← Θ                                            ▷ Surviving configurations
6:     while (ESF + |S| ≤ M) ∧ (ISF + 1 ≤ max_instances) do
7:         i ← sample_instance(P_I)
8:         ISF ← ISF + 1
9:         for θ ∈ S do
10:            s ← run_experiment(θ, i)
11:            ESF ← ESF + 1
12:            C[ISF, θ] ← evaluate(s)
13:        end for
14:        S ← drop_candidates(S, C, stat_test)
15:    end while
16:    return select_best_survivor(S, C)
17: end procedure
```

Racing algorithms can employ various statistical tests, which we categorized based on three criteria:

- *Parametric* vs *Nonparametric* Tests: Parametric tests operate under the assumption that data follows a specific distribution, whereas nonparametric tests are not bound to any distribution assumptions. As a result, nonparametric tests are more versatile, suitable for a broader array of problems. However, parametric tests, given their structure, tend to be more potent, especially when it comes to dismissing the null hypothesis.

- *Blocking* vs *Non-blocking* Design: In statistical terms, *blocking* involves grouping experiments with similar characteristics into clusters or groups. When it comes to tuning metaheuristics, the blocking design is commonly preferred, as suggested by Mongomery [243].

- *Family-wise* vs *Pair-wise* Tests: In the family-wise approach, the aim is to verify if all results, regardless of their corresponding candidate, originate from the same distribution. If a family-wise test indicates differences between candidates, post-tests can then be conducted to compare candidates directly and determine superiority. Conversely, pair-wise tests directly compare two candidates against each other.

Some of the statistical tests that can be used include:

- *tNo-Race* - the classical paired t-test is employed as the statistical test. This test is parametric and utilizes a blocking design. Additionally, it does not adjust the p-value for multiple comparisons.

- *tBo-Race* - It mirrors the tNo-Race method, but with one notable distinction: it applies the *Bonferroni correction* to adjust p-values for multiplicity.

- *tHo-Race* - It resembles the tNo-Race method but incorporates the multiplicity correction suggested by Holm [138].

- *F-Race* - In this method, Friedman's two-way analysis of variance by ranks serves as the statistical test. This test is nonparametric and employs a blocking design with a family-wise approach. Should this test detect any statistically significant differences in the data, it moves on to post-hoc tests. These tests conduct pairwise comparisons of configurations to identify the most promising one. Various methods for executing post-hoc tests are detailed in Birattari et al. [36].

2.6 Chapter conclusion

In this chapter, we delve into the intricacies of metaheuristic algorithms. We begin by defining the general concept of metaheuristics and outlining some well-established classification schemes for these algorithms.We then delved into existing metaheuristic algorithms, offering a depiction of their increasing popularity over the years. This discussion is complemented by a accompanying document containing a list of metaheuristics (`https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixA.pdf`), which offers an extensive compilation of metaheuristics we have identified. This list, which we believe to be the most comprehensive to date, includes 540 entries and details the name, publication year, authors, and source of inspiration for each metaheuristic.

We also touch upon the fundamental principles of local search methods, providing a concise overview of some key techniques. The next segment of the chapter is dedicated to a detailed examination of specific metaheuristics that are central to our research, such as Variable Neighborhood Search, Greedy Randomized Adaptive Search Procedures, Ant Colony Optimization, Bee Colony Optimization, Genetic Algorithms, and Memetic Algorithms.

The chapter concludes with a discussion on various strategies for hyperparameter tuning of these methods, a crucial aspect for optimizing their performance. This comprehensive overview lays a solid foundation for understanding the role and potential of metaheuristic algorithms in our research.

CHAPTER
THREE

GREEN VEHICLE ROUTING PROBLEM

3.1 Vehicle Routing Problem

The *Vehicle Routing Problem* (**VRP**) is a classic and fundamental optimization challenge in the field of operations research. In 1959, the VRP was introduced by Dantzig and Ramser [61]. Subsequently, in 1964, Clarke and Wright developed the first effective heuristic for solving this problem [52]. A number of books have been written on the topic of VRP, with notable examples including [47, 115, 336, 337].

VRP entails determining the optimal routes for a fleet of vehicles to serve a set of customers or locations, subject to a range of constraints referred to as '*attributes*'. In Section 3.3, we delve into the attributes that should be considered when solving the VRP, including time windows, capacity constraints, and vehicle limitations. The problem is NP-hard, which actually means that finding an optimal solution requires exponential time in the worst case [186]. Therefore, heuristic and metaheuristic algorithms have been developed to solve the VRP efficiently in practice.

The most common objective in the VRP is to minimize the total distance traveled by vehicles while satisfying some specified attributes, but other objective functions can be utilized based on specific requirements and constraints. To cater to the diverse needs of different VRP applications, objective functions can be combined or modified accordingly. In Section 3.4, we provide an in-depth discussion of some of the common objective functions used in the VRP.

The basic version of VRP can be formally defined as follows:

Definition 1 *Let us assume that we are provided with a graph $G = (V, E)$, where V denotes a set of vertices that correspond to the depot and customers, and E represents a set of edges connecting nodes from V. Each edge $e_{ij} \in E$ is assigned a non-negative value that indicates the cost of traveling from node i to node j. The objective is to identify a set of paths (routes) starting from the depot node, such that all of the nodes from V are visited and the total cost associated with selected edges is minimized while adhering to all given constraints.*

VRP can also be defined as a *Mixed Integer Linear Program* (**MILP**) in a following way:

$$min \sum_{i=0}^{n} \sum_{j=1}^{n+1} d_{ij} x_{ij} \tag{3.1}$$

s.t.

$$\sum_{j=1}^{n+1} x_{ij} = 1, \quad 1 \leq i \leq n,\ i \neq j \tag{3.2}$$

$$\sum_{i=0}^{n} x_{ij} = \sum_{i=1}^{n+1} x_{ji}, \quad 1 \leq j \leq n+1,\ i \neq j \tag{3.3}$$

$$1 \leq \sum_{i=1}^{n+1} x_{0i} \leq M \tag{3.4}$$

$$\sum_{i=0}^{n} x_{i(n+1)} = \sum_{i=1}^{n+1} x_{0i} \tag{3.5}$$

$$x_{ij} \in \{0,1\}, \quad 0 \leq i \leq n,\ 1 \leq j \leq n+1 \tag{3.6}$$

The presented model involves decision variables x_{ij}, which indicate that edge $(i,j) \in E$ has been selected, with d_{ij} representing the distance between vertices (nodes) i and j. The depot node is represented by nodes 0 and $n+1$, while M refers to the maximum number of routes. The objective function, as represented by Equation (3.1), minimizes the total distance covered by all vehicles. Constraints (3.2) ensure that each customer is visited exactly once, while constraints (3.3) mandate that a vehicle arriving at a location must leave it (excluding depot nodes). Constraints (3.4) limit the number of routes, whereas constraints (3.5) ensure that each vehicle returns to the depot. Finally, the constraints (3.6) are responsible for regulating the nature of the decision variables.

The VRP arises in various transportation and logistics applications, including but not limited to package delivery, waste collection, and public transportation. Given its ubiquity and significance, understanding the VRP and the various techniques for solving it is essential for practitioners and researchers in the field of operations research.

In Figure 3.1 we present a generalized example of the VRP. The depot is depicted as a red rhomboid, while the unassigned customers are represented by blue circles. Upon completion of the VRP, all customers are assigned to one of the routes, distinguished by different colors.

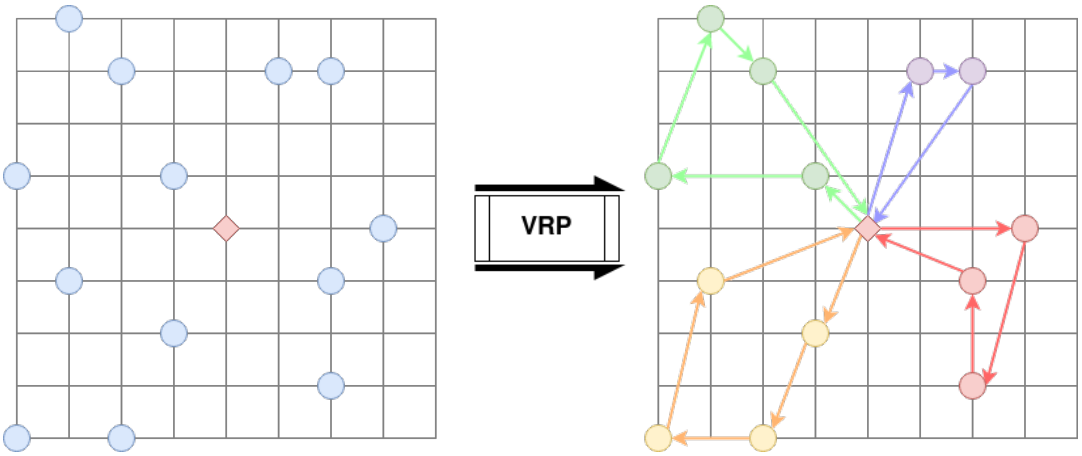

FIGURE 3.1: An example of the VRP.

3.2 Green Vehicle Routing Problem

The *Green Vehicle Routing Problem* (**GVRP**) is an extension of the traditional VRP that incorporates environmental factors related to the transportation of a fleet of vehicles. The primary focus of GVRP is to reduce the detrimental impact of transportation operations on the environment, particularly with respect to CO_2 emissions.

One way to address the negative environmental impact of transportation operations is by minimizing GHG emissions or fuel consumption of a fleet that is composed of ICVs. Another approach is to use AFVs, which do not rely on petroleum-based internal combustion engines such as electric vehicles, hybrid electric vehicles, hydrogen vehicles, and others.

The first publication to directly address the environmental impact of VRP was authored by Bektaş and Laporte in 2011 [31]. Within their research, the *Pollution Routing Problem* (**PRP**) was introduced as an extension of the classical VRP, accounting for the fleet of ICVs, while incorporating an objective function that takes into account not only the distance traveled, but also GHG emissions. The emission values depend upon several factors, such as vehicle speed, load, or road gradient. Typically, PRP is of a multi-objective nature, attempting to balance emissions with economic costs.

In 2012, Erdoğan and Miller-Hooks published a seminal paper introducing the term *Green Vehicle Routing Problem* [86]. This variant of the classic VRP seeks to minimize the fuel consumption of a fleet of vehicles, which can include both ICVs and AFVs. The GVRP implicitly addresses emissions concerns by assuming that using AFVs eliminates the GHG emissions associated with routing. It should be noted that this is a simplified perspective, as it does not account for the emissions generated during the production of AFVs or the emissions resulting from the production of electrical energy, which is predominantly derived from burning fossil fuels [142]. Nonetheless, these issues are beyond the scope of the VRP. In their study, Erdoğan and Miller-Hooks focused on a specialized version of GVRP, the *Electric Vehicle Routing Problem* (**EVRP**), which deals with the unique characteristics of electric vehicles, such as their need for recharging stations and long recharging times. Customer satisfaction can be negatively impacted by unplanned delays, so recharging visits must be carefully planned. A generalized version of the EVRP is presented in Figure 3.2. The depot is depicted as a triangle, customers as circles, and refueling stations as rhombi. At the start of the journey, each vehicle begins with a full battery level from the depot. As the vehicle services each customer, its battery level decreases until it eventually needs to visit one of the recharging stations to recharge the battery.

VRP in Reverse Logistics (**VRPRL**), which involves the optimization of waste collection or recycling of used materials, is sometimes classified as a variant of the GVRP. While VRPRL is important in addressing environmental issues, we do not consider it in this research due to its fundamental difference in the objective, which does not directly relate to vehicle emissions, unlike the PRP and GVRP.

Figure 3.3 depicts some of the key features of the GVRP that can aid in classification, as adapted from the publication [19]. The GVRP literature can be classified based on two main criteria: *problem characteristics* or *solution methodology*. With respect to problem characteristics, there are several aspects to consider:

- *Types of engines*: The types of engines in vehicles have distinct characteristics that influence the problem as a whole. For instance, charging time needs to be considered for BEVs, whereas HEVs require a determination of the point to

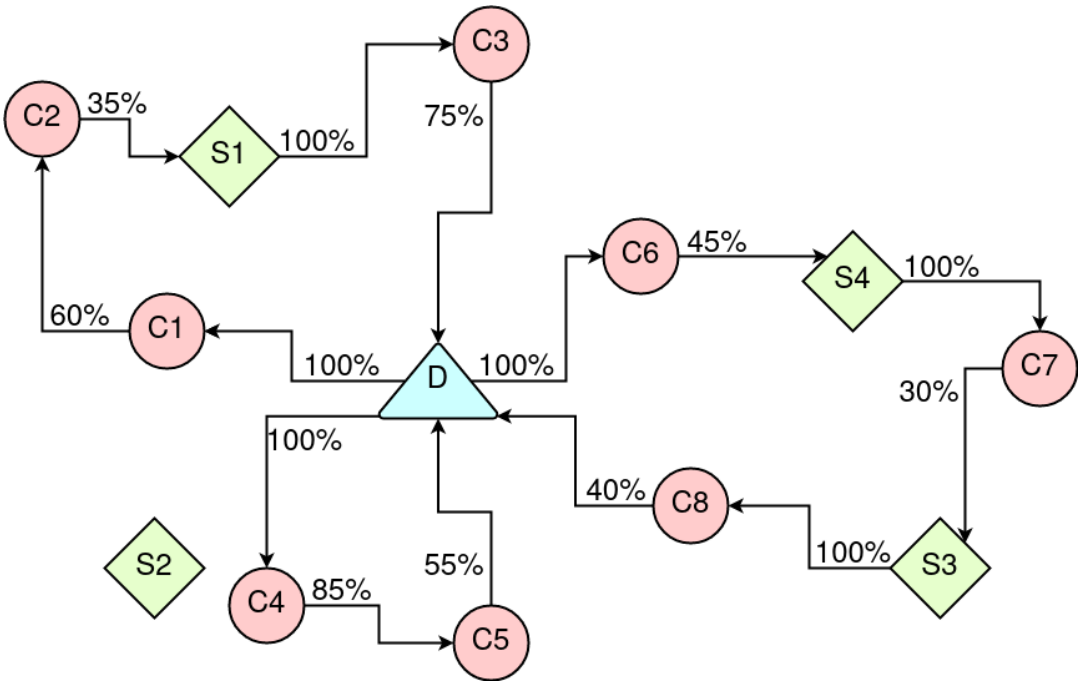

FIGURE 3.2: An example of the EVRP.

switch from one propulsion system to another. These distinct engine types have already been explained in Section 1.2.

- *Objectives*: The classical VRP primarily focuses on the economic objectives of routing a fleet of vehicles, whereas GVRP extends the scope by incorporating environmental objectives such as minimizing pollution and fuel consumption. These objectives can be addressed separately or in combination. Section 3.4 presents some of the commonly considered objectives in GVRP.

- *Scenarios*: GVRP can be applied to various real-world scenarios. It is important to consider that different vehicles have varying consumption rates (which are further elaborated in Section 3.5), ranges, and recharge speeds (Section 3.6). The number and placement of recharging stations play a significant role in the EVRP's efficiency and the feasibility of specific solutions. In addition, in actual situations, recharging stations have a capacity limit, and not all chargers may be available at any given time, resulting in some level of uncertainty. Other types of uncertainty that may arise include the time it takes to serve a customer at their location, traffic congestion, or roadwork.

- *Attributes*: The VRP and GVRP have many distinct variations, each with their own objectives and constraints. These specific characteristics of the problem are known as *attributes*, and we provided a more detailed description of many of them in Section 3.3.

In addition to classifying publications by the characteristics of the problem, they can also be categorized based on the methodology used to solve the problem. As discussed in Section 1.4.1, various approaches can be employed to obtain a solution for optimization problems, including GVRP.

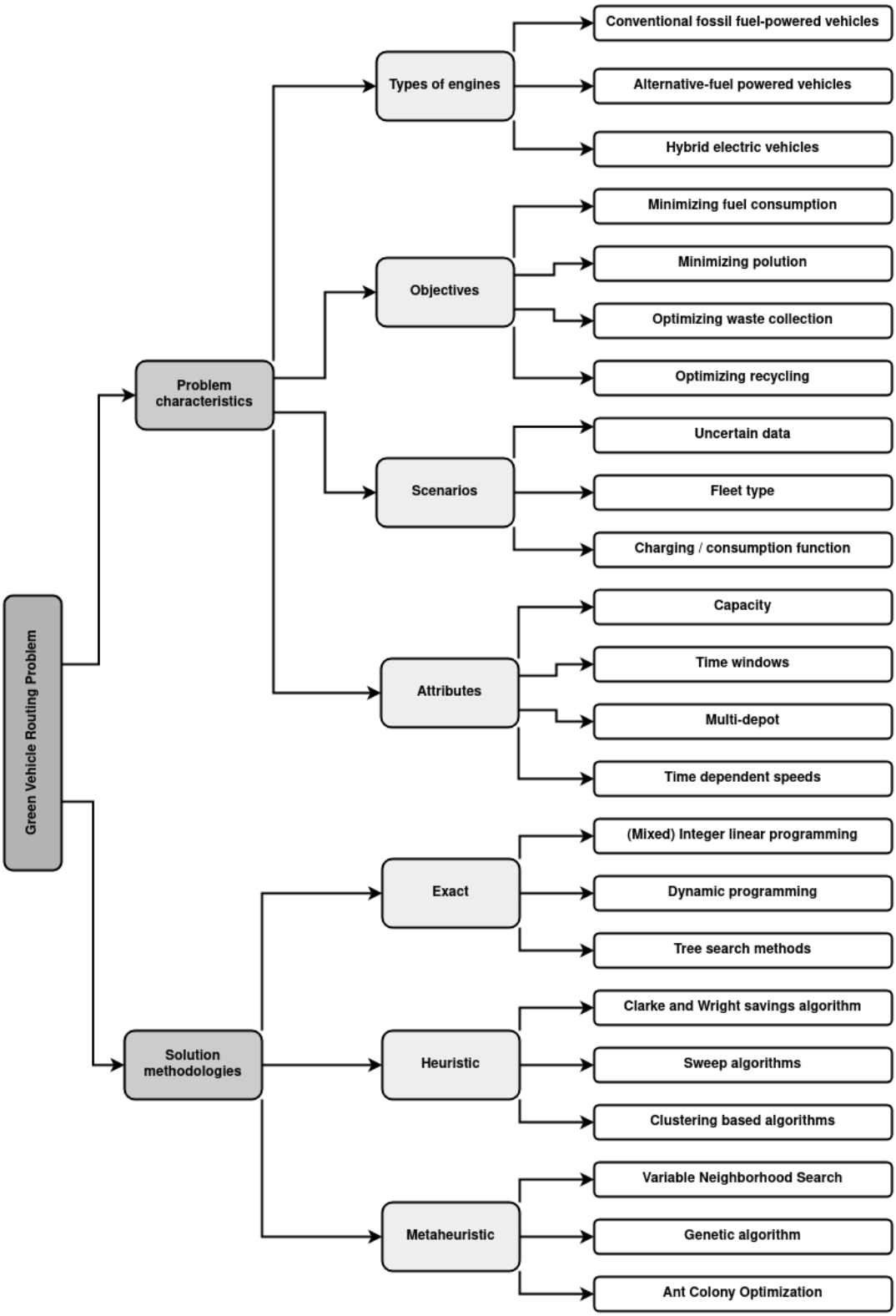

FIGURE 3.3: Different aspects of GVRP can be used for classification (adapted from [19]).

3.3 Attributes

VRP, and by extension GVRP, is a complex optimization problem that has many attributes or unique features. These attributes describe the characteristics of the problem and define the objectives and constraints that need to be considered in order to solve it effectively. Some of the attributes of GVRP include the number of vehicles available, the capacity of each vehicle, the geographical location of the customers, the time window in which each customer needs to be serviced, and the distance between the customers and the depot. Other attributes can also be considered depending on the specific problem, such as vehicle speed, traffic conditions, fuel consumption, and environmental impact. Understanding and properly defining the attributes of the GVRP is crucial in order to develop effective solutions that can optimize vehicle routes and minimize costs while meeting all the constraints and objectives of the problem. In this section, we provide a detailed description of some of these attributes.

Capacity constraint

The *Capacity Constraint* (**C**) is a key element in VRP, limiting the load that a vehicle can transport. It is typically measured in terms of maximum weight, volume, or number of items. This factor is crucial and frequently discussed in a significant portion of VRP literature, as it directly influences route planning and optimization. References [25, 53, 282] are among the many that delve into this aspect, highlighting its importance in VRP studies.

Asymmetry

Asymmetry (**A**) in the context of VRP refers to the scenario where the distance or travel cost between two locations differs depending on the direction. That is, traveling from point A to point B might not entail the same distance or cost as the return journey from B to A. This situation often arises due to factors such as one-way traffic systems, varying road conditions, or differential traffic flows. For more detailed explorations of this attribute in VRP, studies and discussions can be found in references [8, 184, 223], where various aspects and implications of asymmetry are analyzed.

Time windows

Another important attribute of the VRP is *Time Windows* (**TW**) attribute, referring to the specific time intervals within which customers must be serviced. This attribute is particularly important in scenarios where customers have specific delivery or pickup schedules. Failing to meet the time window constraints may result in additional costs or penalties. Time windows can be hard or soft constraints, with hard constraints being strictly enforced, and soft constraints allowing some flexibility. To put it differently, if the hard time window constraints are violated, the solutions become infeasible, while in the case of the soft time window constraints, some leeway is allowed with a certain penalty to the objective function. Time windows are usually defined with respect to specific events such as a customer's availability, store opening hours, or delivery restrictions in residential areas. This attribute became particularly important with the rise of e-commerce as it enables customers to specify a fixed time interval during which they can receive their shipments. Proper management of time windows can lead to significant improvements in operational efficiency, better customer satisfaction, and reduced transportation costs. However, it can also increase

the complexity of the VRP, especially when there are multiple time windows to consider. Some studies that focus on the TW attribute can be found in references [25, 182, 355].

Multiple depots

Another common attribute of the VRP is the presence of *Multiple Depots* (**MD**), which refers to the scenario where there are multiple locations from which vehicles can start and end their routes. This attribute describes scenarios where there are several warehouses, distribution centers, or service centers, and each center has its own fleet of vehicles. The main challenge in this attribute is to allocate the customers to the different depots while ensuring that the vehicle routes are efficient and the demand of each customer is met. This attribute can be further complicated by other factors such as distance between depots, vehicle capacity, and travel time. Examples of publications considering this attribute include [284, 298, 388].

Multiple trips

Another common attribute of the VRP is *Multiple Trips* (**MT**), which refers to the possibility of vehicles making more than one route to serve all the customers. In other words, a vehicle may need to return to the depot and start a new trip to serve additional customers. This attribute is particularly useful in situations where the capacity of a vehicle is not enough to serve all the customers in one trip, or where the time window constraints prevent a vehicle from serving all the customers in a single route. Multiple trips can be allowed with or without a limit, depending on the specific problem. References [140, 263, 388] provide examples of publications that explore this attribute.

Multiple compartments

The *Multiple Compartments* (**MC**) attribute is an important aspect of the VRP that can be encountered in various real-world scenarios. It is an extension of the basic VRP where each vehicle is capable of carrying multiple types of products or goods, each requiring separate compartments or containers for transportation. This attribute is often encountered in the transportation of hazardous materials, where certain products cannot be mixed and must be kept separated during transportation. This attribute is also relevant in industries such as retail, where different types of products with varying storage requirements need to be transported. For instance, a truck carrying frozen food items may have multiple compartments with different temperature controls to preserve the quality of the products. The multiple compartments attribute introduces new constraints and challenges to the problem, such as the capacity of each compartment, the compatibility of the products, and the need to handle and load/unload the products carefully. Figure 3.4 illustrates the contrast between single-compartment and multiple-compartment vehicles and their impact on the VRP. As depicted, if customers demand various types of products, utilizing multiple-compartment vehicles can result in a reduced number of routes. This is because customers no longer need to be served by different vehicles for each type of product, thereby minimizing travel time and optimizing the route planning process. Publications that have explored this attribute are referenced in [130, 285, 313].

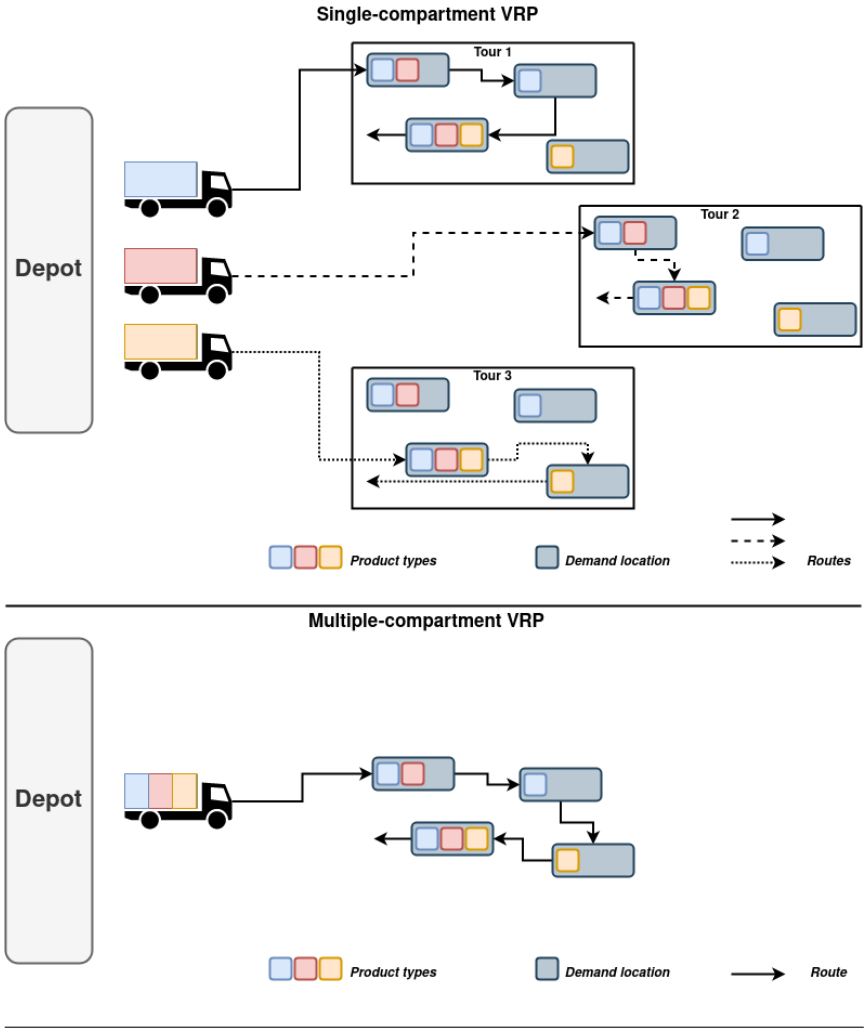

FIGURE 3.4: The difference between single-compartment and multiple-compartment VRP

Heterogeneous fleet

The *Heterogeneous Fleet* (**HF**) (also known as *Mixed Fleet*) attribute in VRP refers to a situation where a company possesses a fleet of vehicles that have varying characteristics, such as their capacity, fuel consumption, speed, and maintenance costs. In such cases, it becomes essential to consider the distinct characteristics of each vehicle while planning the routes, or else there might be some vehicles that remain underutilized or overloaded, while others might fail to complete their assigned tasks. This attribute gains more significance in the context of GVRP, as a company might have vehicles with different engine types, which require different treatment while planning routes. For instance, a company having a fleet of both ICVs and BEVs may prioritize the use of BEVs to reduce carbon tax. Additionally, the time required to recharge the BEVs needs to be taken into account when delivering products, which is not applicable to ICVs, rendering them non-interchangeable. The Heterogeneous Fleet attribute is also useful when dealing with the transportation of perishable goods and frozen food, where different types of vehicles might be needed to maintain the desired temperature conditions. Publications that explore this attribute are referenced in [150, 163, 324].

Open routes

The *Open Routes* (**O**) attribute in the VRP refers to scenarios where a vehicle can leave its depot, serve some customers, and return to any depot available in the network, without necessarily returning to its original depot. This attribute is particularly relevant in situations where customers are spread across multiple regions or cities, making it impractical for vehicles to return to their original depot after serving each customer. The open routes attribute also allows for more flexibility in route planning, as vehicles can be assigned to any depot where there is a demand for their services, making it easier to balance the workload between depots. However, this attribute also increases the complexity of the problem, as it requires a more detailed analysis of the routing options and the available depots, which can be time-consuming and computationally intensive. References [233, 287, 293] are examples of publications that delve into this attribute.

Pickup and Delivery

The *Pickup and Delivery* (**PD**) aspect of the VRP is centered around the transport of items from one location to another, necessitating both collection and delivery at distinct points. Commonly seen in distribution networks, this attribute entails handling logistics from suppliers to customers. The challenge here is not just about managing pickup and delivery points, but also considering vehicle capacity. It is a critical component in sectors like postal services, home delivery, and waste management, with the objective of minimizing the total travel distance of vehicles while adhering to pickup and delivery requirements. For detailed insights into how this attribute influences VRP, studies in references [22, 323, 354] provide thorough analysis and discussion.

Backhauls

The *Backhaul* (**B**) attribute of the VRP refers to a scenario where a vehicle can pick up a load on the return trip after making a delivery. In other words, the vehicle is utilized to carry a load from the destination back to the origin, instead of returning empty. Delivery points, also known as linehaul points, are destinations where goods are delivered from a central distribution center. On the other hand, pickup points, also called backhaul points, are locations from where the goods are collected and returned to the distribution center. Backhaul points are serviced only after all delivery points have been served, unless the problem formulation is a *Mixed Backhauls* (**MB**) version where this restriction is not applied. This strategy can be particularly useful in scenarios where the origin and destination are far apart, and it is not practical to send another vehicle to pick up the return load. By incorporating backhauls into the routing plan, companies can maximize the utilization of their vehicles, reduce transportation costs, and increase their profitability. Backhauls are commonly used in industries such as retail, food, and beverage, where vehicles deliver products to customers and then pick up empty containers or returnable items on the way back. Backhauls can also be useful when transporting goods between warehouses or distribution centers, as it allows companies to make more efficient use of their resources. Some publications that explore this attribute are referenced in [59, 102, 216].

Split Deliveries

The *Split Deliveries* (**SD**) attribute of VRP is a flexible variant that allows vehicles to visit customers multiple times, which is different from the classical VRP that

requires each customer to be visited only once. This attribute can be highly useful when a customer requires a large order that cannot be delivered in a single trip, or when a customer needs several deliveries for various items or at different times. By permitting split deliveries, the VRP can optimize the routes to minimize travel time and distance, while ensuring that all deliveries are completed on time. This attribute provides greater flexibility in the VRP, enabling businesses to cater to the specific needs of their customers more efficiently and cost-effectively. Examples of publications considering this attribute include [15, 16, 312].

Flexible Deliveries

The *Flexible Deliveries* (**FD**) attribute in VRP offers a significant enhancement to delivery flexibility by enabling customers to specify alternative delivery locations, each associated with distinct time windows. For instance, a customer could provide their workplace address for deliveries scheduled during work hours and their home address for times when they are likely to be at home. This feature requires the delivery vehicle to intelligently select the appropriate delivery location based on the actual delivery timing. An example of a study considering this attribute is [297].

Precedence constraints

In the VRP, the *Precedence Constraints* (**PC**) attribute refers to the scenario where certain deliveries must be made before others. These constraints arise in many real-world applications, such as when certain products need to be delivered before others due to their perishable nature or when the delivery of one product is dependent on the delivery of another. The Precedence Constraints attribute of VRP can also be utilized to depict scenarios where certain customers are given priority over others. To satisfy these constraints, it may be necessary to adjust the order of delivery for some of the vehicles in the fleet, or even split deliveries between multiple vehicles. Failing to account for precedence constraints can result in costly delays and disruptions to the supply chain. Some publications dealing with this attribute are [30, 201].

Dynamic requests

The *Dynamic Requests* (**D**) attribute in the VRP refers to the scenario where new orders or requests may be added to the existing routes during the course of delivery. This attribute is common in industries such as food delivery, courier services, and ride-sharing, where new orders can be received at any time. In this situation, the VRP algorithm needs to be flexible enough to adjust the existing routes and schedules to accommodate the new requests. To handle dynamic requests efficiently, the VRP algorithm needs to be integrated with a real-time tracking system that can update the location of the delivery vehicles and the status of the orders in real-time. Examples of publications considering this attribute include [20, 141, 129].

Time-dependent speeds

The VRP attribute called *Time-dependent Speeds* (**TD**) acknowledges that the speed of vehicles is not constant and can vary depending on several factors such as the time of day and location. For instance, during rush hour, vehicles may move at a lower speed due to traffic congestion, while they may move faster during off-peak hours. By incorporating this attribute into the VRP, planners can optimize routes that account for the varying speed conditions, resulting in efficient deliveries and reduced travel

time. This attribute becomes crucial, especially in urban and metropolitan areas, where traffic conditions can be highly unpredictable and can affect the delivery process significantly. With the implementation of time-dependent speeds, businesses can ensure timely deliveries, reducing costs and improving customer satisfaction. Examples of publications where this attribute was considered include [200, 262, 387].

Multiple objectives

The *Multiple Objectives* (**MO**) attribute of VRP involves optimizing multiple objectives at the same time, from reducing transportation costs to maximizing customer satisfaction and reducing carbon emissions. Some of the objectives that can be considered in the context of GVRP are presented in Section 3.4. MO recognizes that different stakeholders may have different objectives that need to be considered when planning routes. For example, a company may prioritize reducing costs while also aiming to reduce its carbon footprint by using electric vehicles, whereas a customer may prioritize fast delivery times and real-time tracking. Incorporating multiple objectives into the VRP can be challenging since different objectives may conflict with each other. However, it can lead to more comprehensive and effective route planning that considers the needs and preferences of multiple stakeholders. There are two approaches to dealing with multiple objectives: combining them into a single objective or using Pareto-optimization. The latter aims to find a set of solutions that are not dominated by any other solution in terms of all the objectives considered. In other words, a solution is Pareto-optimal if no other solution can perform better in one objective without degrading the performance in another objective. The set of Pareto-optimal solutions is called the Pareto frontier, and it represents the trade-offs between different objectives. This approach enables decision-makers to evaluate the trade-offs between objectives and make informed decisions that balance the competing objectives. While the Pareto frontier can be extensive, additional criteria such as user preferences may be used to select the optimal solution from the Pareto frontier. This attribute was considered in [101, 105, 242].

Partial recharge

The *Partial Recharge* (**PR**) attribute of GVRP involves optimizing the routes for electric vehicles that have a limited driving range and require recharging. This attribute recognizes that electric vehicles require more planning and management than traditional vehicles because of the time required to recharge their batteries. Nonetheless, recharging a battery to its full capacity every time a vehicle visits an *Alternative Fuel Station* (**AFS**) can be suboptimal, as it can be time-consuming, but ultimately not necessary. Therefore, partial recharge can be allowed, which permits a vehicle to recharge its battery to only the level needed to complete its delivery. By incorporating partial recharge into the GVRP, planners can ensure that the vehicles can complete their deliveries without running out of charge, while also minimizing the total time required for recharging. This attribute is particularly relevant in urban and metropolitan areas, where electric vehicles can have a significant impact on reducing emissions and improving air quality. The Partial Recharge attribute requires planners to consider the capacity and location of recharging stations, as well as the time required for recharging, to optimize the route and minimize downtime. Examples of studies considering partial recharge include [92, 133, 208].

Battery swapping

Battery swapping (**BS**) is an attribute of the GVRP that refers to the process of exchanging a depleted battery in an BEV with a fully charged one at a designated battery swapping station. This process enables the EV to continue its journey without the need for a prolonged charging stop, which can save valuable time and increase the efficiency of the EV. In the context of GVRP, the battery swapping attribute is considered as a potential solution for mitigating the issue of limited EV driving range and reducing the total time required for recharging or refueling, ultimately enhancing the routing efficiency and the overall performance of the EV fleet. However, the effectiveness of battery swapping depends on various factors, including the availability and accessibility of swapping stations, the battery technology and compatibility, the swapping time and cost, and the trade-offs between swapping and charging. Examples of publications in which this attribute was examined include [191, 281].

3.4 Objective functions

In optimization, the objective function is a mathematical formula used to evaluate the performance of solutions. It is usually defined using decision variables, which are the values under optimization. The aim is to find the variable values that yield the optimal performance per the objective function. Optimization problems vary, and so do their objective functions, based on the criteria being optimized. For example, in vehicle routing problems, common objectives might be to minimize travel time, reduce the number of vehicles, or cut down route costs.

The selection of an objective function is pivotal, defining the optimization's end goal and success measurement criteria. In the context of GVRP, various objective functions can be applied. It is important to note that these objectives can coexist and be analyzed simultaneously through Pareto optimization. Often, they are interlinked, meaning achieving one can affect others. For instance, lower fuel consumption might reduce operational costs.

Most research papers use a composite objective function, combining different goals like minimizing operational costs, fuel consumption, and time window penalties. In many studies, the primary aim is to minimize total cost, represented by a blend of various cost-related objectives.

Each of these considerations highlights the complexity and multi-dimensional nature of setting and achieving objectives in optimization scenarios. The choice and combination of these objectives depend on the specific nuances and demands of the problem at hand.

Operation cost minimization

In the GVRP, the goal of minimizing *Operation Cost* (**OC**) focuses on optimizing vehicle routes to reduce overall operational expenses. This includes aspects like energy usage, vehicle acquisition, maintenance, and labor costs. By targeting these areas, planners aim to cut down the costs of transportation services, thereby boosting business profitability. Additionally, reducing operational costs allows GVRP operations to offer more competitive service pricing, potentially attracting more customers. An important byproduct of this goal is its positive environmental impact, as lower operation costs often correlate with reduced fuel consumption and greenhouse gas emissions, contributing to sustainability efforts. Studies that have explored this objective include references [90, 106, 134, 193].

Fuel consumption minimization

The *Fuel Consumption* (**FC**) minimization objective in the VRP targets a reduction in the fuel used by vehicles during deliveries. This is increasingly significant due to soaring fuel prices and heightened environmental awareness. The primary benefits include lowering operating expenses and diminishing the carbon footprint. Achieving this involves route optimization to minimize travel distance and vehicle usage, and considering vehicle features like fuel efficiency, payload capacity, and engine type to further cut down fuel use. This objective is not just about reducing costs but also about bolstering the sustainability of the transport sector. It encompasses a broad range of energy sources beyond traditional fossil fuels, including electricity and hydrogen, reflecting the diverse propulsion technologies in modern vehicles. Examples of studies that considered this objective include [5, 54, 258].

Pollutant emission reduction

The *Pollutant Emission reduction* (**PE**) objective in transportation planning focuses on lessening environmental and public health impacts by curbing atmospheric pollutant emissions. This is particularly crucial in urban settings, where air quality directly affects public health. By aiming for lower emissions, planners contribute to both environmental sustainability and public health improvement. This goal encompasses a wide array of pollutants, not just those from traditional fossil fuels but also emissions associated with alternative energy sources like electricity and hydrogen. In addition, achieving this objective can offer financial advantages, as many regions enforce regulations and impose fees on high-emission vehicles, incentivizing companies to adopt cleaner transportation practices. Several studies have addressed this objective, notably references [3, 83, 341, 382].

Time windows penalty cost minimization

The *Time Windows Penalty Cost* (**TWPC**) minimization objective seeks to minimize the total cost of exceeding the customer time windows. Delivering outside the time window can lead to additional costs, such as lost sales or customer dissatisfaction. By minimizing the cost of violating time windows, planners can improve the overall quality of service, increase customer satisfaction, and reduce the overall cost of providing transportation services. This objective requires planners to consider the time windows of all customers and optimize the routes accordingly, taking into account the trade-off between minimizing time window violations and minimizing the total distance traveled by vehicles. This objective was explored in [220, 255, 366].

Travel time minimization

Minimizing *Travel Time* (**TT**) decreases the total time required to complete deliveries. This objective is of utmost importance when all customers need to be served promptly, such as in medical emergencies. It is also a factor in reducing operating costs such as fuel consumption, vehicle maintenance, and drivers' wages, as well as reducing the environmental impact of transportation. Achieving the Travel Time Minimization objective requires planners to consider several factors, including vehicle and depot capacity, traffic conditions, and the order of deliveries. This objective has been investigated in references [33, 226, 257]].

Revenue maximization

The *Revenue Maximization* (**RM**) objective in transportation route optimization is focused on increasing the total earnings of the service provider. This is particularly pertinent in scenarios emphasizing high profitability, like the delivery of valuable goods. The aim here is to ensure maximum vehicle utilization and to decrease the number of vehicles needed for all deliveries, thereby optimizing revenue generation. Key considerations for planners include analyzing the demand for transportation services, the pricing strategy, and the operational costs. By effectively balancing these elements, planners can enhance revenue while satisfying customer needs and maintaining the financial health of the service. This approach not only boosts profits but also aligns with efficient and customer-oriented service delivery. This objective has been investigated in studies [33, 197].

Total distance minimization

The goal of the *Total Distance* (**TDM**) minimization objective in to minimize the total distance traveled by all vehicles. This is the most common VRP objective and can be found in some form in most publications considering VRP. By minimizing the total distance traveled, planners can reduce the operating costs associated with transportation and increase the efficiency of the operation. Studies that have considered this objective include [10, 132, 143, 220, 224, 352].

Number of vehicles minimization

The objective of minimizing the *Number of Vehicles* (**NV**) in delivery operations centers on route optimization to reduce the total number of vehicles needed. This becomes especially crucial in scenarios where vehicle availability is a constraint, or when the costs associated with vehicle acquisition and maintenance are considerable. By achieving this goal, operations can become more efficient and cost-effective, particularly in resource-limited settings. Studies that have integrated this objective into VRP include [297, 318, 370].

Quality loss cost minimization

The goal of the *Quality Loss Cost* (**QLC**) minimization objective is to minimize the overall cost that results from quality loss during the transportation process. This objective is particularly relevant for perishable goods that experience a decline in quality over time. The goal is to reduce penalties related to quality degradation, including product damage, spoilage, and deterioration, which can result in additional expenses and dissatisfied customers. By minimizing quality loss costs, planners can lower the overall cost of providing transportation services while improving customer satisfaction. This objective requires careful consideration of factors such as product fragility and perishability, as well as handling procedures and transportation conditions that promote quality preservation. In industries such as food and pharmaceuticals, where product quality is essential, the Quality Loss Cost Minimization objective plays a critical role in ensuring the safe and efficient transportation of goods. Example studies incorporating this objective into VRP include [195, 391].

ICV usage minimization

The *ICV Usage* (**ICVU**) minimization objective tries to minimize the number of routes serviced by internal combustion vehicles, such as those that use gasoline or

diesel, and serve as many customers as possible with AFVs. By reducing the use of ICVs, planners can decrease the amount of greenhouse gases and other pollutants emitted, thereby decreasing their carbon tax and promoting sustainable transportation practices. This objective requires planners to consider factors such as the availability and location of alternative fuel sources, the range and capacity of electric or hybrid vehicles, and the scheduling and routing of vehicles to ensure that ICVs are only used when necessary. This objective was considered in a study [373].

Maximum overtime minimization

In route optimization, the *Maximum Overtime* (**MXO**) objective is defined as the greatest deviation between a route's duration and its set time limit. This minimization is crucial in contexts where driver work hours are regulated to prevent overwork. By reducing maximum overtime, planners can lower the risk of accidents due to driver fatigue and enhance driver well-being. Additionally, it can lead to significant savings in transportation costs, particularly in terms of reduced labor expenses. To meet this objective, planners might need to deploy extra vehicles, revise schedules or routes for shorter driving times, or increase vehicle capacity or speed for more efficient deliveries. This objective was explored in study [23].

Route imbalance minimization

The *Route Imbalance* (**RI**) minimization objective attempts to balance the workload of vehicles and drivers as much as possible. In other words, the goal is to minimize the difference in the number of stops and distance traveled between each vehicle or driver, ensuring that no single vehicle or driver is overloaded with work while others are underutilized. By minimizing route imbalance, planners can improve the efficiency and effectiveness of the transportation system, reduce delivery times and costs, and increase the satisfaction of both customers and drivers. Studies that have explored this objective include [94, 339].

Waiting time minimization

The *Waiting Time* (**WT**) minimization objective tries to minimize the time vehicles have to wait before serving customers, taking into account the time windows in which customers must be served. By minimizing the waiting time, planners can ensure that customers receive their deliveries within their preferred time windows while preventing vehicles from being underutilized. This objective was examined in study [44].

Recharging cost minimization

The objective of *Recharging Cost* (**RC**) minimization in GVRP is to optimize the routes of AFVs while minimizing the expenses associated with recharging them. The cost of recharging is influenced by factors such as the recharging technology used and the amount of energy required. By minimizing the costs associated with recharging, companies can reduce their overall operating costs, leading to increased profitability and sustainability. This objective requires planners to consider factors such as the location and availability of charging stations, types of chargers available at stations, the capacity and range of AFVs, and the scheduling and routing of vehicles to ensure that they can be recharged efficiently and effectively. This objective was examined in studies [92, 208].

Recharging time minimization

Recharging Time (**RT**) minimization tries to minimize the time taken for recharging AFVs. The duration of recharging depends on the charging technology, battery capacity, and the battery's current level. The aim of minimizing recharging time is to increase the productivity of AFVs, which can result in faster and more frequent deliveries. Additionally, reducing recharging time can decrease the need for more AFVs, as a fixed number of vehicles can complete more deliveries in less time. Prolonged charging times can also prevent vehicles from serving customers within their preferred time windows. Thus, minimizing recharging time is essential for efficient and timely deliveries while maintaining customer satisfaction. This objective was considered in [44].

Queuing time minimization

The *Queuing Time* (**QT**) minimization objective in GVRP aims to minimize the amount of time that AFVs spend waiting in line for a recharging station or a refueling station. Queuing time can be a significant challenge for AFVs, particularly when the number of AFVs using a given station exceeds its capacity. By minimizing queuing time, planners can ensure that AFVs are available for deliveries more frequently, which can increase the efficiency of the delivery system. The objective was considered in study [194].

3.5 Consumption models

BEVs rely solely on electric power stored in their batteries to power their propulsion system. The energy consumption of BEVs is influenced by a variety of factors. The vehicle's weight is one of the most significant factors in determining its energy consumption, as heavier vehicles require more energy to accelerate and move. Aerodynamics also plays a crucial role, with a more streamlined shape reducing drag and improving efficiency. Driving behavior, such as accelerating quickly and braking hard, can significantly impact energy consumption. Climate conditions, such as temperature and humidity, also affect BEV energy consumption, as heating and cooling systems require energy to operate. Finally, battery characteristics, such as capacity and chemistry, can impact BEV energy consumption and range. The energy consumed by a BEV is typically measured in *kilowatt-hours* (**kWh**) per unit of distance traveled, such as kWh per mile or kWh per kilometer.

Estimating energy consumption is a crucial step in creating a dependable route plan. There are two main approaches to this problem [72, 363]:

1. *Theoretical energy estimation*: This method aims to develop a mathematical model for predicting energy consumption by considering various factors such as vehicle speed, cargo weight, road slope, and other related variables. It relies on the fundamental principles and theoretical knowledge of the problem, rather than any empirical data.

2. *Empirical energy estimation*: This approach involves utilizing previously collected data on energy consumption to statistically forecast the energy consumption of a vehicle under specific conditions. This approach often employs machine learning techniques like regression or neural networks to make predictions based on historical data.

In this section, we provide a brief overview of some of the theoretical models that are applicable in estimating the energy consumption of BEVs.

Linear consumption model

The linear consumption model is a simplistic model that assumes the average energy consumption of a BEV is solely dependent on the distance traveled. Although this model oversimplifies the problem and is not suitable for accurately predicting energy consumption, it can be useful for showcasing differences between various methods used for vehicle routing problems. By simplifying the part that has little impact on the performance of some solution methods, such as metaheuristics, we can focus on improving our solution approach. However, the energy estimation function can have a considerable impact on the performance of MILP solvers.

Equation (3.7) provides the calculation formula for the *Estimated Consumption Rate* (**ECR**) based on the *Linear Consumption Model* (**LCM**). The distance traveled by the vehicle from customer i to customer j is denoted by the variable d_{ij}, and the average consumption rate of the vehicle is represented by CR_{avg}.

$$ECR_{ij} = d_{ij} \cdot CR_{avg} \tag{3.7}$$

Nonlinear consumption model

Nonlinear consumption models take into account different factors in their prediction, such as the aforementioned vehicle speed, cargo weight, road slope, etc. Several studies have examined nonlinear consumption models, including [191, 307, 368, 378, 390].

Four primary sources of energy loss in BEVs are commonly examined in research studies [1]:

- *Aerodynamic losses*: Aerodynamic losses are predominantly influenced by the driving speed of the vehicle. Such losses are determined by the relative wind speed across the vehicle, rather than the speed over the ground. The calculation of aerodynamic losses in a BEV is influenced by several factors such as air density, the frontal area of the vehicle, the coefficient of drag, and the speed at which the vehicle is traveling.

- *Tire losses*: The energy consumed to overcome the rolling resistance of a road surface on the tires of a vehicle is called tire loss, and it depends on both the weight of the vehicle and the rolling resistance of the tires.

- *Drivetrain losses*: The drivetrain losses are comprised of the energy losses that are not directly controlled by the user. These losses arise due to inefficiencies in various components, including the motor controller, the motor, the gearbox, and other factors that affect the conversion of DC electricity from the battery pack into useful torque at the wheels of the vehicle.

- *Ancillary losses*: The term "ancillary losses" refers to the energy consumed by all other electrical systems in the vehicle, such as cooling fans and pumps, the radio, interior and exterior lighting, and so on.

In addition to the four losses mentioned earlier, the study conducted by Xiao et al. [368] also examines the kinetic/potential loss. This loss considers the energy lost

[1] `https://www.tesla.com/blog/roadster-efficiency-and-range`

during the conversion of chemical energy to kinetic/potential energy due to changes in velocities and terrain heights. The authors presented a comprehensive model for estimating energy consumption in their study, which incorporates all of these losses.

The distribution of the mentioned energy losses for an electric city bus is shown in Figure 3.5.

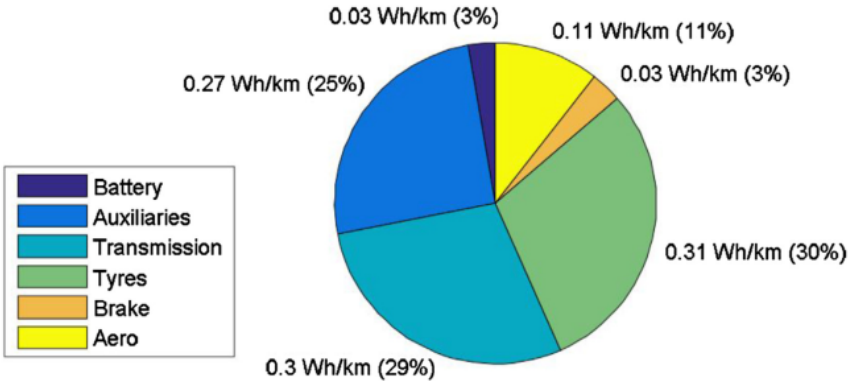

FIGURE 3.5: Energy losses distribution of an electric city bus. (Image source: [348])

For our purposes, the consumption model presented by Li et al. [191] would suffice as it incorporates significant factors such as speed, load, driving distance, road angle, among others. Therefore, we present their model in Equation (3.8).

$$ECR_{ij} = a_{ij}(w + l_{ij})d_{ij} + \gamma v_{ij}^2 d_{ij} \tag{3.8}$$

where

$$a_{ij} = a + g\sin\theta_{ij} + gC_r\cos\theta_{ij} \tag{3.9}$$

$$\gamma = \frac{1}{2}C_d A\rho \tag{3.10}$$

The vehicle acceleration is represented by the parameter a, and the gravity constant is represented by the parameter g. The road angle is denoted by θ_{ij}, and the rolling resistance coefficient is represented by C_r. The weight of the empty electric vehicle is represented by w, and l_{ij} represents the load of the vehicle. The speed of the vehicle is denoted by v_{ij}. Additionally, the symbol C_d denotes the traction coefficient, A represents the frontal area of the vehicle exposed to the wind, and ρ is the density of the air. The relation of γ to the previously mentioned aerodynamic loss is evident, and a_{ij} is associated with the tire loss.

3.6 Battery recharge models

As previously mentioned, BEVs require recharging at an AFV station once their battery level reaches a certain point. The charging speed plays a critical role in determining the duration of the vehicle's stay at the station. Accurate estimation of this time is essential in ensuring efficient route planning and timely customer service. Inaccurate predictions could result in missed customer time windows, additional worker overtime pay, or running into traffic congestion during rush hour.

In Section 3.3 we already mentioned two different policies for recharging a battery of a BEV: full recharge and partial recharge. We have also mentioned the battery swap strategy, which allows the vehicle to avoid long recharging times. However, whether we consider full or partial recharge, the charging speed is a factor that we need to consider. As with the energy consumption rate, we can differentiate between two main groups of functions that can be used for calculating charging time: linear charging function and non-linear charging function.

Linear charging function

The linear charging function is a straightforward function that usually relies on the energy required for recharging (E_{charge}) and the charging rate (CH_{rate}), which is determined by the charging technology. It can be expressed as follows:

$$T_{charge} = E_{charge} \cdot CH_{rate} \tag{3.11}$$

It has the advantage of being simple to implement, but it fails to take into account all factors that contribute to the charging speed.

Nonlinear charging rate

The charging process of batteries in the real-world is non-linear due to fluctuations in terminal voltage and current. Initially, the charge level increases faster over time, but as the battery gets closer to full charge, the rate of increase becomes lower.

Figure 3.6 illustrates the non-linear charging function in a real-world situation [136]. The graph shows the relationship between the terminal voltage (u), current (i), and *State of Charge* (**SoC**), which is equivalent to the battery level. The charge level initially increases linearly with time, but then becomes concave until it reaches full charge.

Initially, the battery undergoes charging using a constant current, which leads to a linear charging rate. Subsequently, when the SoC approaches approximately 80% of battery capacity, the charging method shifts to using constant voltage, in order to prevent damage to the battery [136]. This results in a concave charging rate. This approach is also called the *constant current–constant voltage* (**CC-CV**) charging method. In a study by Saadaoui et al. [295], the authors plotted the CC-CV method, alongside another fast-charging method called *multi-stage constant current* (**MSCC**).

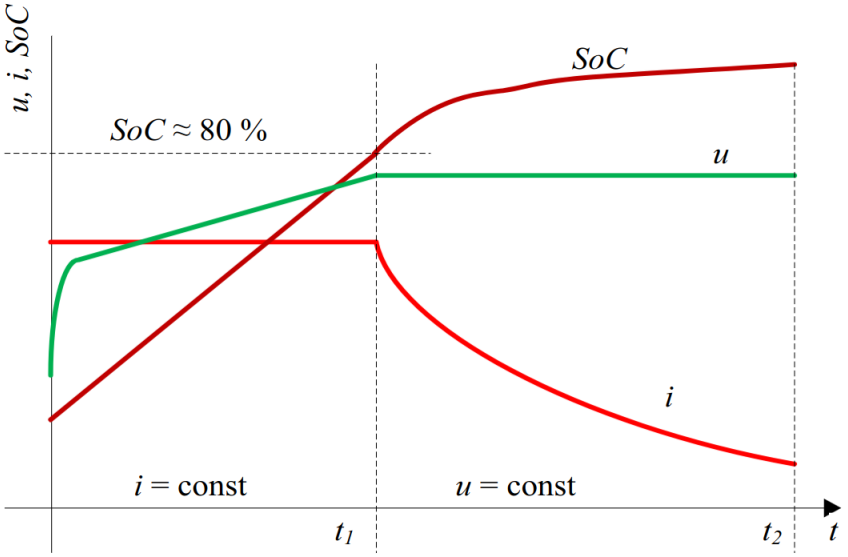

FIGURE 3.6: A typical battery charging curve where the terminal voltage is u, the current is i, and the state of charge is SoC. (Image source: [136])

Montoya et al. [245] introduced a non-linear charging function that considers the aforementioned characteristics of the charging process. It is defined as follows:

We can represent the charge level of a vehicle upon arrival at AFS i as q_i and the time spent charging at i as Δ_i. Additionally, let o_i be the charge level after the vehicle departs from AFS i. The charging function can be written as Equation (3.12).

$$o_i = g_i(q_i, \Delta_i) \tag{3.12}$$

The authors utilized the transformation proposed by Zündorf [392] to convert this two-dimensional function into a one-dimensional function, which was then estimated by approximating it with *piecewise linear functions*, using data from [342].

3.7 Literature review

3.7.1 Review papers on the topic of GVRP

Lin et al. [198] conducted one of the initial surveys on the GVRP, which also encompassed the classical VRP. The survey categorized papers based on the specific version of VRP investigated, with a focus on the most prevalent VRP attributes. The authors examined approximately 280 papers, out of which only 28 addressed the green variant of VRP. Among those 28 papers, 11 aimed to minimize fuel consumption or CO_2 emissions in VRP, while 17 were concerned with reverse logistics VRP. The study did not include VRP with AFVs. Both heuristic and exact methods were considered in the surveyed papers. The authors also outlined potential future directions for GVRP; however, some of those suggestions may have become outdated since the publication of the survey.

In their work, Park and Chae [268] provided a comprehensive review of various approaches for addressing the GVRP using ICVs. The survey encompassed a range of techniques, including exact algorithms, heuristics, and metaheuristics. A total of 40 publications were analyzed, with 21 of them employing metaheuristic methods

to tackle the problem. Additionally, the authors highlighted different fuel consumption models that were considered in the surveyed papers, providing a comprehensive overview of the modeling approaches used in ICV-based GVRP.

Eglese and Bektaş offer a comprehensive overview of the GVRP in their chapter [80]. They cover various aspects of GVRP, including emission models, fuel consumption considerations, and time-dependent GVRP. The chapter also explores topics such as speed optimization and multicriteria analysis within the context of GVRP. While AFV-based GVRP is mentioned, the chapter does not delve into a detailed review of this particular aspect. Nonetheless, the chapter provides valuable insights into the fundamental concepts and key elements of GVRP.

Another study, performed by Zhang et al. [383], presents a comprehensive analysis of swarm intelligence algorithms applied in the context of green logistics. The authors surveyed a total of 115 papers published between 1995 and 2014, exploring various problems within the domain of green logistics. While papers related to Vehicle Routing Problem (VRP) were relatively scarce, they were mainly focused on reverse logistics. Notably, papers specifically addressing PRP or VRP with AFVs were not included in the review.

In their survey, Marrekchi, Besbes, and Dhouib [217] conducted an examination of the ICV-based version of the GVRP. The authors focused on differentiating between exact, heuristic, and metaheuristic solution methodologies, providing notable examples for each of these commonly employed algorithmic approaches.

Erdelić and Carić [85] conducted an extensive analysis of EVRP. The authors provided a detailed examination of the fundamental characteristics and applications of EVRP, encompassing various energy consumption models. They explored different variations of EVRP, offering relevant examples from the existing literature. The survey encompassed a wide range of solution methodologies, including exact, heuristic, metaheuristic, and hybrid approaches. For each reviewed paper, the authors presented the problem variant, solution approach, and evaluated instances, accompanied by concise descriptions. Additionally, the study discussed potential avenues for future research in this domain, highlighting areas where further investigation could be valuable.

In the work by Konstantakopoulos, Gayialis, and Kechagias [166], authors provided an overview of the VRP in its various forms. Among the different versions of VRP, GVRP was addressed as a distinct subsection. The authors reviewed a limited number of papers concerning GVRP, five in total.

Moghdani et al. [241] examined a total of 309 papers that are relevant to the GVRP. The reviewed papers were analyzed and categorized based on various aspects such as problem classification, GVRP variant, uncertainty, solution methodology, objective function, and sustainability considerations. While not all reviewed papers were explicitly mentioned, those that were mentioned often received limited description. The study concluded by identifying several promising research directions that can shape the future of GVRP.

Ferreira, Steiner, and Canciglieri Junior [95] examined 76 papers published between 2012 and 2018 focused on the multi-objective version of the GVRP. The review primarily focused on conducting a bibliometric analysis of papers related to ICV-based GVRP, including studies on PRP, GVRP, and reverse logistics. However, papers specifically related to AFV-based GVRP were not included in the analysis.

In their comprehensive literature review, Asghari and Mirzapour Al-e-hashem [19] conducted an extensive analysis of the GVRP from 2000 to 2020. The review encompasses a total of 313 papers, which were examined and classified. The classification was based on the type of vehicle engine and further categorized according to various

attributes of the VRP and the methodologies employed. Moreover, the authors offer valuable insights into potential future research directions and identify existing gaps in the literature, highlighting areas where further investigation is warranted.

3.7.2 Metaheuristics for GVRP

In this section, we provide a literature review organized into subsections, each focusing on a specific metaheuristic as used in various publications. This review builds upon our prior work [221], and has been expanded to encompass additional publications that were not included previously.

Simulated Annealing

In the study conducted by Felipe et al. [92], the focus was on the Electric Vehicle Routing Problem incorporating multiple recharging technologies and partial recharges. This problem involves the consideration of various technologies to recharge a vehicle's battery, each with its own recharge time and associated cost. Typically, reducing the recharge time results in an increase in cost. Moreover, it is not mandatory to fully recharge the battery, allowing for a reduction in the time required to service customers. To address this problem, the authors implemented SA as well as several other heuristic methods based on local search. By conducting tests on diverse sets of instances, the researchers concluded that SA outperformed deterministic local search methods, particularly on larger instances featuring over 200 customers.

In the study conducted by Küçükoğlu et al. [173], SA was employed to address the Green Vehicle Routing Problem with Time Windows. The main objectives of this problem were to minimize fuel consumption and CO_2 emissions. The problem formulation involved a MILP approach; however, an exact solver was not employed due to its inefficiency for larger problem sizes, as concluded by the authors. To construct an initial solution for SA, the algorithm utilized Solomon's time-oriented nearest neighbor algorithm [315]. A notable feature of the proposed algorithm referred to as *Memory Structure Adapted Simulated Annealing* (**MSA-SA**), was the storage of objective function values in a special memory structure whenever a new route was generated. This approach helped to avoid redundant evaluations of the same route. Through experimentation on various test instances, MSA-SA demonstrated its ability to achieve near-optimal solutions for medium and large problem sizes within a relatively short time frame.

Xiao and Konak [367] conducted a study using SA to mitigate the CO_2 emissions generated by a fleet of homogeneous vehicles. Their research incorporated soft time windows, wherein penalties were imposed for missing deadlines, while early arrivals were encouraged to enhance customer satisfaction. The authors also took into account the TD attribute of the VRP. The main objective of their study was to minimize CO_2 emissions, which were estimated using established models available in the literature. Additionally, the authors considered several secondary objectives in a hierarchical manner, including customer satisfaction level, total route time, and total route distance. To tackle this complex problem, the researchers presented a MILP formulation and utilized the SA metaheuristic as an effective solution approach.

In their publication, Yu et al. [376] delved into the application of SA in the context of the VRP, specifically focusing on the *hybrid vehicle routing problem* (HVRP). The HVRP serves as an extension to the GVRP since hybrid vehicles possess the capability to switch between electrical energy and fossil fuel as their primary propulsion mode during the course of their routes. The problem considers two types of stations:

electric stations and fuel stations, with vehicles having the flexibility to visit these stations multiple times. To obtain an initial solution, the authors employed the *Nearest Neighbor Algorithm* (**NN**) [114], which starts with a single customer and, at each step, the nearest customer to the current one is included in the route. During each iteration, the algorithm explored one of five predefined neighborhoods, chosen randomly with a probability of $\frac{1}{5}$. To circumvent the risk of local optima, the algorithm incorporated the *restart strategy*.

In their research, Karagul et al. [158] conducted a study on the GVRP with the primary objective of minimizing both the total CO_2 emissions and the total distance traveled. Three mathematical models were examined, drawing inspiration from the model presented in [164, 369]. These models varied mainly in the formulation of the objective function, with one model derived from [83], and the other two proposed by the authors themselves [158]. To tackle the formulated problems, a basic version of SA was utilized. The algorithm's performance was evaluated using instances from the *Capacitated Vehicle Routing Problem* (**CVRP**) dataset provided in [21, 51]. The experimental evaluations concluded that the approach employing a convex composition of objectives demonstrated superior performance compared to other approaches in general.

Normasari et al. [256] also delved into the utilization of AFVs for the CVRP. The main objective was to minimize the total distance traveled by all vehicles. To address this, the researchers proposed a MILP formulation, which was tested using the CPLEX commercial solver. However, due to the computational challenges associated with exact solvers when dealing with larger instances, the authors also introduced an alternative approach using SA. In their problem representation, the researchers included a depot node, customer nodes, AFS nodes, and a set of dummy nodes. Dummy nodes were introduced to overcome the constraint that prohibits visiting the same node more than once, as the vehicles needed to be able to visit AFSs multiple times. The initial solution was generated using the NN algorithm. Four distinct neighborhood structures were utilized: *swap*, *insert*, $insert_{AFS}$, and $delete_{AFS}$. To optimize the SA approach, the parameters were tuned using a subset of benchmark instances from [86]. Additionally, the authors conducted a sensitivity analysis to assess how variations in different parameters affected the objective function's value and the required number of vehicles.

Tabu search

The paper by Kwon, Choi, and Lee [176] focuses on the heterogeneous fleet CVRP, aiming to minimize the overall cost, which includes the *carbon trading* cost. The carbon trading cost represents the expense incurred by companies for emitting a certain amount of CO_2 and is directly related to the deviation between actual emissions and a predefined upper limit. If the emissions surpass the limit, companies must purchase additional emission rights, resulting in increased transport costs. The authors presented a mathematical model for this problem and proposed a TS metaheuristic. The initial solution is constructed by assigning the customer with the highest demand to the vehicle with the largest capacity. If the partial solution is infeasible, the customer is reassigned to the vehicle with the second-highest capacity, and so on. The initial solution is further enhanced using the *3-opt heuristic* proposed in [199]. The tabu list size is determined dynamically using a tabu term rule [108]. Three different neighborhood structures were evaluated in TS algorithm: *insert*, *swap*, and *hybrid* (which randomly selects the insert or swap operator in each iteration). Experimental results

demonstrate that the TS algorithm with the hybrid neighborhood structure, yields the best outcomes.

In the study conducted by Úbeda et al. [341], an additional example of GVRP with capacity constraints focusing on minimizing CO_2 emissions using TS is presented. The CO_2 emissions of a vehicle are calculated by considering the fuel consumption associated with the vehicle's mass and load. The researchers applied the basic version of TS to seven real-world instances and observed that, in certain cases, it successfully reduced CO_2 emissions compared to TS optimization aiming to minimize total distance. However, limited information is provided regarding implementation details, particularly regarding the specific set of operators used to generate neighborhoods, which is not explicitly mentioned.

In their work, Niu et al. [254] addressed a green open CVRP with time windows, aiming to minimize both CO_2 emission costs and driver wage costs simultaneously. To approximate the CO_2 emission costs, they employed a *comprehensive modal emission model* (**CMEM**) proposed in [26]. The problem formulation also considered variable vehicle speeds throughout the trips. A mathematical model was developed, and a TS approach was utilized to generate high-quality solutions. For the initialization of the TS algorithm, a *modified NN* was employed. In each iteration, four different neighborhood operators were utilized to generate a set of neighboring solutions. A *speed improvement strategy* based on [170] was employed to determine the optimal speed and departure time for each route. The researchers analyzed various real-world scenarios, including traffic congestion, different vehicle types, diverse objectives, and the impact of empty kilometers.

Variable neighborhood search

Bruglieri et al. [44] addressed a variant of EVRP with time windows and partial battery recharge. Their objective was to simultaneously minimize the number of BEVs, total travel time, total recharging time, and total waiting time. To tackle this problem, the researchers proposed a mathematical model that was subsequently solved using a technique called *Variable Neighborhood Branching* (**VNB**) [127]. VNB is a matheuristic inspired by VNS and specifically designed for solving *0-1 Mixed Integer Programming* problems. The VNB approach demonstrated success in finding high-quality solutions for the EVRP with TW within a reasonable computational time frame.

The paper by Yavuz and Çapar [373] examined an uncapacitated variant of VRP that involves a fleet with heterogeneous vehicles (including both AFVs and ICVs), multiple trips, and limited trip durations. This problem differs from traditional delivery-focused VRPs as the vehicles also provide specific services, allowing them to spend time serving customers. Vehicles can be fully recharged at any AFS or partially recharged on-site while serving a customer. The paper presents four different objectives for the problem: minimizing total distance traveled, minimizing total CO_2 emissions, minimizing fuel costs, and minimizing total distance traveled by ICVs. To address this problem, the authors propose the use of a VNS algorithm, which utilizes a set of five different neighborhoods derived from [144, 329]. The customers are grouped and then optimized based on the aforementioned objectives. Additionally, the VNS algorithm is adapted to support Pareto optimization for the proposed problem. Experimental results demonstrate that incorporating on-site partial recharges significantly improves overall performance while increasing the number of AFSs has a smaller impact. Consequently, the proposed model with on-site recharges is more

likely to be adopted by larger companies whenever feasible, given its superior performance.

The research conducted by Affi, Derbel, and Jarboui [5] focuses on the GVRP which utilizes a fleet of AFVs, with the primary objective of minimizing the total energy cost. To tackle this problem, the authors propose a VNS algorithm that incorporates a set of 9 different neighborhood structures categorized into three types: *customer neighborhoods*, *AFS neighborhoods*, and *node neighborhoods* (which consider both customer and AFS nodes). The shaking step of the algorithm generates a random solution using only the set of node neighborhood structures. In each iteration of the shaking step, a neighborhood structure is selected at random. In the subsequent improvement step, the authors employ *Variable Neighborhood Descent* (**VND**) as a local search procedure, considering only the customer and AFS neighborhoods. Notably, unlike the classical VNS approach where the shaking step precedes the local search step, the proposed version by Affi et al. reverses this order. Consequently, the shaking step only accepts a solution if it improves upon the incumbent solution. To evaluate the algorithm's performance, benchmark instances from [86] were employed. The proposed method was then compared to solutions put forward in [86, 244, 302, 303]. The experimental results indicate that the proposed algorithm either outperforms other methods in terms of the best-found solution or achieves a comparable solution to some of the alternative approaches.

The paper by Ren et al. [286] focuses on investigating a GVRP with TW and HF. This problem variant introduces BEVs into a fleet consisting of homogeneous ICVs. The objective is to minimize both the total delay time and five different types of pollutant emissions simultaneously. Delay time pertains to the observation that the satisfaction of a customer decreases as the vehicle arrives later, even if the arrival time falls within the specified time window. In this problem, instead of recharging BEVs at AFSs, they are required to return to the depot for recharging. The authors propose a mathematical model, together with a VNS algorithm to find a good solution in reasonable amount of time. The shaking step of the algorithm employs two neighborhood operators: *1-1 exchange* and *1-0 shift*. The local search procedure is based on the VND approach, which utilizes nine different neighborhood operators. These operators include one that changes the vehicle type, four *intra-route* operators for exchanging or moving customers within a single route, and four *inter-route* operators for exchanging or moving customers between two routes. The proposed algorithm demonstrates satisfactory performance in terms of both objectives.

In their research, Yilmaz and Kalayci [374] tackled the EVRP with Simultaneous Pickup and Delivery, aiming to reduce the total distance traveled by electric vehicles. They developed a mathematical model for the problem and crafted a unique set of instances using the dataset from [302], modifying original demands to include pickup and delivery requirements. The study involved extensive testing of various neighborhood structures and five variants of the VNS, with the *Clark and Wright Savings Algorithm* (**CW**) [52] utilized for initial solution generation. Their computational experiments highlighted the critical role of neighborhood structures, solution methods, and neighborhood change steps in efficiently discovering high-quality solutions by navigating through different neighbors in the search space.

(Adaptive) large neighborhood search

The paper by Goeke and Schneider [111] addresses a CVRP with time windows and a heterogeneous fleet, including BEVs. They propose a mathematical model that considers three objectives: minimizing total travel distance, minimizing vehicle propulsion and labor costs, and minimizing battery replacement costs. To tackle this problem, they develop an ALNS algorithm. The algorithm allows for the construction of infeasible solutions but penalizes them in the objective function. The paper provides a detailed description of the destroy and repair procedures used in ALNS. Destroying entails removing customers from a dynamically selected interval, as experiments indicate that the choice and size of this interval significantly impact solution quality. Additionally, a local search method is introduced to intensify the search in promising areas. To overcome local optima, the algorithm employs a SA-based criterion for solution acceptance, which allows worse solutions to be accepted with a certain probability. The modified ALNS algorithm proves effective in finding satisfactory solutions within a reasonable timeframe, as demonstrated on a set of 56 benchmark instances from [302].

The EVRP with known TW and PR was addressed by Keskin and Çatay [160]. They introduced a mathematical model and utilized the ALNS algorithm with the objective of minimizing the total distance traveled. The acceptance criterion in the algorithm is SA-based. Two groups of destroy and repair procedures are employed: *customer-based* and *AFS-based.* To evaluate the effectiveness of their approach, the authors conducted experiments using a set of instances from [302]. The results demonstrated notable improvements over four of the best-known solutions, highlighting the positive impact of partial recharge on the overall solution quality.

The paper by Hiermann et al. [134] investigates a VRP with time windows and electric vehicles, featuring a heterogeneous fleet. In this scenario, the company leases vehicles at varying prices, each with its own load capacity, energy capacity, and acquisition cost. The objective is to minimize both the total cost of travel and the total acquisition cost of vehicles. During visits to an AFS, vehicles are fully recharged. To tackle this problem, the authors propose an ALNS approach, which incorporates a local search and a labeling procedure to enhance the solution quality. The ALNS employs a range of operators for both the destroy and repair procedures. In each iteration, an operator is selected based on the *roulette wheel strategy*, considering the weights assigned to each operator. These weights are dynamically adjusted to better suit the characteristics of the specific instance. Experimental evaluations conducted on a set of benchmark instances demonstrate the ALNS's capability to discover high-quality solutions within a reasonable computational time.

The research conducted by Macrina et al. [207] addresses a VRP with time windows, a heterogeneous fleet, and partial recharges. The fleet comprises both conventional vehicles and electric vehicles, and the objective is to minimize the overall cost, which includes fuel and electricity costs. To tackle this problem, the authors propose a matheuristic approach that combines a hybrid version of *Large Neighborhood Search* (**LNS**) introduced by Shaw [309] and the optimization solver CPLEX. The method follows a two-step process: first, an initial feasible solution is generated using CPLEX, and then random removal and insertion operators are applied to refine the current solution. Through experimental evaluation, it was observed that this combined method achieves optimal solutions more efficiently than using CPLEX alone, particularly for smaller problem instances.

The study conducted by Yu et al. [377] focuses on a GVRP with time windows, aiming to minimize carbon emissions. The authors specifically address the challenge

of solving large-scale instances with over 500 customers, which are difficult to optimize perfectly. To tackle this issue, they employ the ALNS method and introduce two novel operators: the *Forward Load Removal Heuristic* as a destroy operator and the *Fast Insertion Method* as a repair operator. The Forward Load Removal Heuristic capitalizes on the observation that carbon emissions correlate with the distance between customers and the vehicle load. Based on this insight, it is advantageous to serve high-demand customers earlier in the tour, thereby reducing the vehicle load and subsequently lowering CO_2 emissions. Consequently, the proposed operator removes customers with high demands who are served later in the tour (or earlier in the case of pickups instead of deliveries). The Fast Insertion Method aims to streamline the feasibility verification process after inserting a customer by leveraging the fact that, in certain scenarios, not all subsequent customers need to be checked to determine feasibility. By exploiting this property, the method reduces the complexity associated with assessing solution feasibility. Computational experiments were conducted on two benchmark datasets, each containing up to 1000 customers. The proposed approach, featuring the new destroy and repair operators, exhibited an average improvement of 8.49% compared to the classical ALNS method.

Ant Colony Optimization

The research conducted by Mavrovouniotis, Ellinas, and Polycarpou [225] focuses on the application of the ACO algorithm to the EVRP. The objective is to minimize the total operation time of a fleet of electric vehicles. In this study, each ant represents a complete solution for the EVRP, and during the construction phase, careful consideration is given to ensuring that sufficient energy remains for the vehicles to visit charging stations. The pheromone update policy employed in this approach follows the principles of the *MAX-MIN Ant System* ($\mathcal{MMAS}$) introduced by Stützle and Hoos [322]. This policy governs how pheromone values are adjusted based on the quality of solutions found by the ants.

Building upon their previous work [225], Mavrovouniotis et al. [226] propose a parallelization method for further enhancing the algorithm's performance. The method involves multiple independent colonies that communicate with each other using a *non-parametric migration policy*. This parallelization technique was evaluated on a set of three instances. The results demonstrate that parallelization improves solution quality compared to the sequential version of the algorithm. However, the non-parametric migration policy did not have a significant impact on solution quality. This outcome suggests that sharing the best-found solution among parallel colonies leads to convergence toward the same part of the search space.

The research conducted by Zhang, Gajpal, and Appadoo [380] focuses on the CVRP with AFVs. The authors propose two approaches to tackle this problem: the *two-phase heuristic* and the *Ant Colony System* (**ACS**). The two-phase heuristic approach involves solving the *Traveling Salesman Problem* (**TSP**) version of the problem using the NN. After obtaining a TSP solution, visits to AFSs and the depot are inserted into the solution, effectively transforming it into a solution for the initial CVRP problem. Similarly, the proposed ACS method follows a similar approach in generating new solutions. TSP solutions are generated based on *pheromone intensity* and *saving value*. Then, visits to the depot and AFSs are inserted into the solutions. This allows the ACS method to effectively address the CVRP with AFVs. Both algorithms were tested on a set of randomly generated instances. As anticipated, the ACS method outperformed the two-phase heuristic in terms of solution quality, demonstrating its effectiveness in finding improved solutions for the problem at hand.

In their research, Li, Soleimani, and Zohal [197] focus on the Multi-Depot GVRP. The authors propose a comprehensive model that considers four distinct objectives: 1) maximizing revenue, 2) minimizing costs, 3) reducing travel time, and 4) minimizing CO_2 emissions. To tackle this complex problem, the authors proposed an ACO metaheuristic as a solution method. In the ACO algorithm, the pheromone update strategy employed is the *Ant-Weight Strategy* (**AWS**), which is derived from [264].

The Multi-Depot GVRP is also investigated in a study by Zhang et al. [382]. The primary objective considered in their work is to minimize the total distance traveled by a fleet of AFVs. In their approach, Zhang et al. [382] initially assign each customer to its nearest depot. Subsequently, they apply an ACS to solve multiple single-depot GVRP. By decomposing the multi-depot problem into a series of single-depot problems, the authors effectively address the challenges posed by the multi-depot scenario. To evaluate the proposed method, the authors conduct experiments using a set of 48 instances. The results demonstrate satisfactory performance and validate the effectiveness of their approach. Building upon this idea, Zhang et al. [384] further expands the research in this domain.

In the study conducted by Bhattacharjee et al. [33], a multi-depot heterogeneous fleet GVRP is addressed. The problem aims to optimize four objectives: 1) maximizing revenue, 2) minimizing costs, 3) reducing travel time, and 4) lowering CO_2 emissions, similar to the objectives considered in [197]. The algorithm proposed in [33] shares several similarities with the approach presented in [196]. It also incorporates clustering as its initial step, where the *k-nearest neighbors algorithm* [169, 386] is utilized to assign each depot to its nearest customers. However, the exact utilization of the clustering results within the algorithm is not explicitly clarified in the original paper. It is assumed by the author of this thesis that, similar to related work such as [196], the subsequent step involves employing ACO to solve single-depot VRP for each cluster in parallel.

The paper by Li et al. [194] explores the application of *distribution sharing* with electric vehicles. Distribution sharing involves the collaboration of multiple companies to combine their distribution needs and resources, aiming to reduce costs and minimize environmental impact. This particular problem variant focuses on multiple depots with time and capacity constraints. The objective is to minimize the overall cost, encompassing factors such as electricity expenses and environmental costs. To address this problem, the authors propose the utilization of ACO. The algorithm is designed to find efficient solutions considering both separate distribution models and distribution sharing models. Various scenarios are examined, including different electricity prices and carbon taxes. The experimental results indicate that employing the distribution sharing model, coupled with higher carbon taxes, leads to a reduction in CO_2 emissions. Conversely, lower electricity prices are found to increase the total CO_2 emissions. Notably, an increase in carbon taxes results in a significant rise in the overall cost. Therefore, it is crucial to carefully balance the positive and negative effects of such measures.

Genetic algorithms

The paper authored by Ayadi et al. [23] explores the VRP with multiple trips, with a focus on minimizing total CO_2 emissions and reducing the maximum overtime of vehicles. The maximum overtime represents the largest difference between the time limit and the actual duration of each route. To address this problem, the authors propose a modified version of a GA. In each iteration of the algorithm, genetic operators are employed to generate new solutions, which are subsequently enhanced

using local search techniques (similar to MA). If these new solutions exhibit superior quality compared to the worst solutions in the population, they replace those solutions. Through experimental evaluations, the results demonstrate that this algorithm is capable of discovering high-quality solutions in terms of emissions, albeit with a potential sacrifice in terms of distance.

The research conducted by Adiba, Aahmed, and Youssef [3] focuses on addressing a CVRP with the objective of minimizing emissions. The approach involves creating an emission matrix using the methodology outlined in [131]. Solutions are represented as arrays of integers, with each integer corresponding to a specific customer, while the value zero serves as a delimiter to separate different routes. To generate the initial population, emphasis is placed on ensuring that each individual represents a feasible solution. The selection of individuals for crossover is performed using a roulette wheel strategy, with a bias toward the best solutions. A *partially-mapped crossover* operator is then applied to create new individuals. Swap mutation is employed as a means of introducing diversity. Additionally, the algorithm incorporates an elitism strategy to preserve the best solutions during the evolution process. The proposed algorithm is evaluated using a small set of instances, yielding promising results.

The study conducted by Hsueh [139] focuses on GVRP that incorporates stochastic traffic speeds and a heterogeneous fleet. To account for various road conditions and gradients that can impact vehicle speeds, the paths between customers are divided into segments. As it is difficult to precisely predict traffic conditions for each segment, the speed is treated as a random variable. The objective of the problem is to simultaneously minimize the total cost, including both emissions and fuel consumption costs. The authors propose a mathematical model to capture the problem's complexities and provide a solution approach using a genetic algorithm. In the proposed genetic algorithm, solutions are represented by two chromosomes. The first chromosome determines the order in which customers are visited, while the second chromosome indicates the index of the last visited customer in the first chromosome for each vehicle. These chromosomes consist of genes that represent specific elements of the solution. The tournament method is employed for the selection process, and the crossover operator selects two random points in the parent chromosomes and exchanges the segments between these points. However, this approach may result in children visiting the same customer multiple times. To address this issue, a mapping set derived from the exchanged segments of the parents is used to correct any duplications. Two mutation operators are applied in the algorithm. The exchange operator swaps the positions of two randomly selected genes within a chromosome, while the insert operator removes a randomly chosen gene from one position and inserts it elsewhere in the chromosome.

In their work, Tunga, Bhaumik, and Kar [339] examined a CVRP with two primary objectives: minimizing energy consumption and balancing the load across routes to achieve a more even distribution among vehicles. To address this bi-objective optimization problem, the authors proposed a genetic algorithm that incorporates the Pareto ranking scheme. The genetic algorithm introduced in the study utilizes *two tournament selection* processes. These selections randomly choose two pairs of chromosomes and conduct tournaments for each pair. The winners from both tournaments are then chosen for crossover, which helps combine favorable characteristics from the selected chromosomes. The authors employed a *greedy crossover* operator inspired by the work of Ismkhan and Zamanifar [145], while mutation is performed using the *2-opt* operator. Experimental evaluations conducted with this algorithm indicate its ability to generate a quality set of Pareto-optimal solutions, offering a range of trade-offs between the two objectives of energy consumption and route imbalance.

In their research, Cooray and Rupasinghe [54] examined a variant of the CVRP with a primary focus on minimizing energy consumption. To address this problem, the authors proposed a basic GA. However, their notable contribution lies in leveraging machine learning techniques to tune the algorithm's parameters, including the mutation rate, number of generations, and population size. The authors employed the *k-means algorithm* [206] to cluster the problem instances based on total demand and the number of customers. Each cluster corresponds to different sets of parameters, allowing for customization according to the characteristics of the instances. By using the *Freedman test* [98], the authors successfully demonstrated that different mutation rates yield varying means of minimized energy consumption. This approach was tested on a collection of 100 instances sourced from *CVRPLib*[2], a widely recognized benchmark library for CVRP.

In their study, Costa et al. [55] employed a genetic algorithm to address the minimization of CO_2 emissions per route in a basic formulation of the CVRP. The algorithm begins by generating the initial population through a combination of construction heuristics and random solutions. Specifically, the first three individuals are created using well-known construction heuristics such as the NN, CW, and *Random Insertion*. The remaining individuals are randomly generated. Notably, individuals with the same fitness value, even if they encode different solutions, are not allowed in the population. The algorithm incorporates two mutation operators, namely, the *2-Opt* and *3-Opt* operators. During each step, one of these operators is selected at random, with a predefined probability. To select parents for crossover, a binary tournament selection mechanism is employed. A new offspring replaces an existing individual only if it demonstrates a better fitness value. The algorithm terminates when it reaches a predefined number of successful offspring or a predefined number of unsuccessful offspring without improving the current best solution. Upon termination, the procedure is restarted using the *partial replacement procedure* proposed by Cheung, Langevin, and Villeneuve [49]. This restart occurs for a total of R times, providing additional opportunities to enhance the quality of the solutions.

In their study, Hiermann et al. [133] examined the VRP with time windows and a heterogeneous fleet. The fleet consisted of three classes of vehicles: conventional, hybrid, and electric vehicles, with each class encompassing multiple vehicle types distinguished by capacity, consumption, or battery capacity. The primary objective was to minimize the total cost, which included fixed costs such as vehicle acquisition and maintenance, as well as variable costs like fuel consumption. The representation of routes in this study involved sequences of customers without considering the presence of recharging stations. Route evaluation was conducted in two levels. In the first level, recharging stations were dynamically inserted into routes for BEVs and HEVs using dynamic programming techniques. In the second level, additional optimization was performed to optimize the partial recharge time and engine mode for HEVs. To address this optimization task, the authors proposed two greedy policies. To obtain the routes, the authors employed a genetic algorithm inspired by the work of Vidal et al. [349, 350]. Although the original authors referred to this algorithm as a hybrid due to the inclusion of several non-standard operators based on local search, it does not qualify as a hybrid metaheuristic since it does not combine two distinct metaheuristics. The study included extensive testing and sensitivity analysis, providing comprehensive insights into the proposed algorithm's performance. Moreover, the authors introduced a new set of benchmark instances to facilitate further research and evaluation in the field.

[2] `http://vrp.galgos.inf.puc-rio.br/index.php/en/`

In their research, Hien, Dao, and Binh [132] delves into the basic version of the EVRP, with the goal of minimizing the overall distance traversed by the fleet of EVs. To achieve this, they introduced the *Greedy Search Algorithm* (**GSA**), a technique that clusters customers into manageable subroutes. These subroutes are then optimized using a local search method, with necessary stops at AFSs incorporated strategically. Additionally, the authors developed a hybrid methodology that synergizes GSA with GA. In this approach, GSA plays a crucial role in generating the initial population, setting the stage for further refinement through GA.

Particle Swarm Optimization

The study conducted by Kumar et al. [174] aimed to optimize both the *production routing* and *pollution routing* problems. The combined problem focused on a scenario where a factory aimed to optimize its production process and efficiently deliver products to customers, considering that each customer had a limited-capacity storage unit. The authors proposed a bi-objective model that encompassed two primary objectives. The first objective involved minimizing the total operational cost, which included factors such as production costs, storage costs, distribution expenses, and more. The second objective aimed to minimize overall emissions by reducing fuel consumption. Additionally, the problem formulation incorporated time window constraints. To address this problem, the authors formulated a mathematical model and introduced a metaheuristic approach known as the *Self Learning PSO* (**SL-PSO**), initially proposed by Li et al. in 2011 [188]. Additionally, the authors applied the *non-dominated sorting genetic algorithm-II* (**NSGA-II**) [67] to the problem to assess the efficiency of SL-PSO. By comparing the Pareto fronts obtained by these two algorithms on a set of instances, the authors concluded that SL-PSO consistently outperformed NSGA-II.

The research by Norouzi, Sadegh-Amalnick, and Tavakkoli-Moghaddam [257] delves into a time-dependent version of the GVRP. In this model, travel times between customers are not static but vary depending on both the distance and the specific time of day. The study primarily aims to minimize total travel time, while also giving importance to reducing fuel consumption and carbon emissions. To tackle this issue, the researchers introduce an adapted multi-objective PSO algorithm. This innovative approach concurrently optimizes both travel time and fuel consumption, factoring in the dynamic nature of travel times. By incorporating time-dependent factors into the optimization, the algorithm is designed to efficiently reduce overall travel time and minimize environmental impacts in terms of fuel usage and emissions. This dual-focus approach presents a more holistic solution to the challenges posed by time-dependent GVRP.

The work conducted by Poonthalir and Nadarajan [274] focuses on a specific variant of the GVRP where the vehicle speed is considered as a variable, simulated using the triangular distribution. The study addresses two distinct objectives: minimizing the route cost and minimizing fuel consumption. To tackle this problem, the authors propose a PSO based approach. The initial population is generated using the NN heuristic. The algorithm incorporates various time-varying parameters when updating the particle velocities. These parameters, including inertia, cognitive acceleration coefficient, and social acceleration coefficient, dynamically change over time to enhance global search and convergence performance. Additionally, the algorithm integrates a greedy mutation operator, as proposed by Poonthalir, Nadarajan, and Geetha [275], to prevent getting trapped in local optima. This operator aids in diversifying the search process and exploring alternative solutions. To evaluate the

proposed approach, the experiments are conducted on a set of benchmark instances obtained from [86]. The results demonstrate that the proposed algorithm outperforms the previously best-known solution presented by Montoya et al. [244], leading to improved expected fuel consumption.

In the study conducted by Wang et al. [353], a multi-depot VRP is investigated, considering transportation resource sharing, time-dependent speeds, and time windows. The research addresses the simultaneous optimization of two objectives: minimizing the total CO_2 emissions and reducing the operating cost. The operating cost encompasses transportation costs, vehicle maintenance costs, and additional customer satisfaction costs incurred due to late arrivals. To tackle this problem, the authors propose a multi-objective PSO algorithm known as MLPSO. The MLPSO is combined with the CW and the *Sweep Algorithm*, both widely recognized in the VRP literature. The proposed approach is evaluated against pure MLPSO and NSGA-II using a modified set of PRP benchmark instances. The results indicate that the proposed approach outperforms the other methods in terms of both emissions and distance, providing improved solutions for the multi-depot VRP with resource sharing, time-dependent speeds, and time windows.

Other metaheuristics

In their paper, Micale et al. [230] explore an asymmetric version of the GVRP that incorporates time windows, variable delivery times, and vehicle dimensions. The inclusion of variable delivery times acknowledges the influence of various factors, such as vehicle type, road conditions, traffic, and weather, on the actual time required for deliveries. Additionally, the consideration of vehicle dimensions accounts for the limitations imposed by certain customers that may not be serviceable by vehicles of a particular size. To find a set of feasible solutions for this problem, the authors employ the *Firefly algorithm* (**FA**), initially proposed by Yang (2009) [371, 372]. The FA is adapted by introducing an elitism procedure that aims to preserve promising solutions and prevent the loss of a firefly queen. The objective at this stage is to select a subset of vehicles from the initial fleet while solely considering the total distance traveled. Subsequently, the authors employ the *Technique for Order Preference by Similarity to Ideal Solution* (**TOPSIS**) method [178] to incorporate economic and environmental factors. By evaluating the set of feasible solutions generated by the FA, the TOPSIS method selects the best solution based on multiple evaluation criteria, including total distances, utilization coefficient, carbon footprint, and fuel consumption. Following a comprehensive numerical analysis, the authors conclude that the quality of the solutions provided by their approach is closely linked to the configuration of the initial fleet.

The research conducted by Macrina et al. [208] focuses on a GVRP that involves a heterogeneous fleet, time windows, and partial recharge capabilities. The fleet consists of both BEVs and ICVs. The primary objective of the study is to minimize the total cost associated with recharging, routing, and activating the electric vehicles while also considering pollution emissions, which must be kept below a predefined threshold. To tackle this problem, the authors propose the utilization of the *Iterated Local Search* (**ILS**) algorithm. The ILS algorithm is employed to discover a set of optimized routes for two clusters of customers, where one cluster is served by BEVs and the other by ICVs. By applying the ILS algorithm, the researchers aim to find efficient and effective routing solutions for the given problem. In order to evaluate the performance of their approach, the authors conduct experimental analyses using a modified set of benchmark instances originally proposed by Solomon [315].

The paper authored by Andelmin and Bartolini [10] focuses on the utilization of BEVs in the context of VRP. The main objective of the study is to identify a collection of routes that minimize the total distance traveled. To achieve this goal, the authors introduce the *Multi-start Local Search* (**MSLS**) method, which builds upon the multigraph reformulation proposed by Andelmin and Bartolini [11]. The problem is represented on a multigraph by assigning nodes to all customers and the depot, while arcs symbolize refuel paths. A refuel path connects two nodes, such as node i and node j, and encompasses a sequence of consecutive AFSs between them. This reformulation aims to eliminate the need for explicitly modeling AFSs as distinct node types, as often observed in the literature. The proposed MSLS algorithm consists of two distinct phases. The first phase employs fast constructive operators, while the subsequent phase enhances these solutions using a broader range of operators to refine the results obtained in the first phase. This two-phase approach allows for a more comprehensive exploration of the solution space and promotes the discovery of high-quality solutions. To evaluate the effectiveness of the algorithm, the authors conducted experiments on two different sets of instances, sourced from [11] and [86]. These instances serve as standardized problem scenarios and provide a basis for performance comparison and analysis.

In their study, Peng et al. [272] focused on the EVRP with the objective of minimizing the total distance traveled while adhering to a maximum traveling time constraint. To tackle this problem, they proposed a *memetic algorithm* that incorporates an adaptive local search procedure in a local improvement phase. To generate the initial population, the authors employed the *k-Pseudo Greedy method* [92]. The adaptive local search utilized reinforcement learning to select moves from a set of eight predefined neighborhoods, enhancing the exploration and exploitation capabilities of the algorithm. Moreover, a *backbone-based crossover operator* was devised to combine genetic material from parent solutions, while the *longest-common-subsequence-based population updating strategy* [48] was used to update the population effectively. To evaluate the performance of their proposed algorithm, the authors compared it with several other algorithms from the existing literature, including the approaches presented in [302] and [10]. By conducting these comparisons, the researchers were able to demonstrate the effectiveness and competitiveness of their memetic algorithm.

In their study, Giallanza and Puma [106] focused on a GVRP within a *three-echelon supply chain*, where customer demands were uncertain. They aimed to minimize both the total cost of transport and CO_2 emissions. To tackle this problem, they utilized the NSGA-II algorithm, a popular method for multi-objective optimization. This allowed them to find a set of solutions that strike a balance between cost and emissions, providing valuable insights for decision-making in supply chain management.

In their research, Zulvia, Kuo, and Nugroho [391] investigated the multi-objective GVRP with a focus on perishable goods delivery. They considered four key objectives: operational cost, deterioration cost, carbon emissions, and service level. Deterioration cost accounted for quality loss over time, while service level measured timely deliveries within customer time windows. To tackle this problem, they employed the *Gradient Evolution* (**GE**) algorithm [175]. GE represents solutions as vectors and explores the search space using three operators: vector updating, jumping, and refreshing. The algorithm's effectiveness was demonstrated through testing on a real-world case involving a fruit delivery company.

In their study, Utama, Fitria, and Garside [345] addressed the GVRP with time windows, aiming to minimize the total cost, which encompasses fuel costs and late fees. To achieve this objective, they employed the *Artificial Bee Colony* (**ABC**)

algorithm. ABC, initially proposed by Karaboga and Basturk [156, 157], is inspired by the behavior of bees. The algorithm distinguishes three groups of bees: *employed bees*, *onlookers*, and *scouts*. Employed bees revisit previously discovered food sources, onlookers observe and select food sources based on the information exchanged by employed bees, and scouts explore new potential food sources randomly. Through the application of the ABC algorithm, superior quality solutions were obtained in comparison to the NN algorithm.

The research conducted by Niu et al. [255] investigated the GVRP with stochastic demand. In this variant, the demand of each customer is unknown until a vehicle reaches that customer. If a vehicle cannot fulfill a customer's demand, it must return to the depot for replenishment. The study aimed to minimize two objectives simultaneously: total cost, which includes fuel emission costs, and customer dissatisfaction, measured by time window violations. To address this problem, the authors proposed a *membrane-inspired multi-objective algorithm* (**MIMOA**). The algorithm consists of three subsystems: two operation subsystems and one control subsystem. The operation subsystems utilize a multi-objective evolutionary algorithm with clustering to search for solutions and transmit them to the control subsystem. The control subsystem guides the evolutionary directions of the operation subsystems. The performance of MIMOA was compared to that of NSGA-II and the *skin membrane guided multi-objective membrane algorithm* (SMG-MOMA) [385] on a set of 10 instances, yielding successful outcomes.

In their research, Woller, Kozák, and Kulich [361] delved into a basic version of the EVRP, with their primary aim being to minimize the cumulative distance traversed by the entire fleet. They developed a MILP model for this purpose and employed the GRASP to find effective solutions within a practical timeframe.

In their study, Xu et al. [370] addressed the Electric Vehicle Routing Problem with the complexities of simultaneous pickup and delivery, time windows, and the intricate dynamics of non-linear charging and load-dependent consumption functions. Their objective was to minimize both the number of electric vehicles used and the total working time of the EV fleet. To achieve this, they formulated a MILP model and employed an ALNS metaheuristic to effectively identify optimal solutions within a feasible timeframe. The authors notably contributed four novel or enhanced operators specifically designed to navigate the challenges of non-linear charging and load-dependent discharging in EVRP scenarios.

In their publication, Xiao et al. [365] examined the EVRP with time windows and mixed backhauls, focusing on reducing the total distance covered by all vehicles. They introduced a diversity-enhanced memetic algorithm, incorporating three innovative operators: genetic operators with an adaptive selection mechanism, a selection operator based on similarity degree, and modification operators for tabu search. Their experimental evaluation, which included 54 newly created instances and two classic benchmarks, demonstrated the algorithm's effectiveness in solving this specific variant of the EVRP.

Hybrid metaheuristics

In their research, Elbouzekri, Elhassania, and Alaoui [83] investigated a unique version of the Generalized Vehicle Routing Problem (GVRP) focused on estimating and minimizing CO_2 emissions rather than relying on alternative fuel vehicles. Instead of using a traditional objective function, the algorithm approximated emissions for each node pair and incorporated this information into its objective. The authors proposed an integer linear programming model and a *Hybrid ACS* (**HACS**) to address

the problem. The HACS algorithm consisted of three phases: route construction, pheromone update, and hybridization. The first two phases followed a classical ACS approach, and the hybridization phase incorporated the LNS metaheuristic to further refine the solution. Since no benchmark instances existed for this problem variant, the algorithm was evaluated on 10 randomly generated instances with varying numbers of customers. The experiments were also repeated using the traditional objective function that aimed to minimize total distance traveled. The results showed an increase in total distance when using the emission-based objective function, challenging the assumption of equivalence between the two objectives.

Schneider, Stenger, and Goeke [302] proposed a hybrid algorithm that combines TS and VNS for the CVRP with Time Windows specifically tailored for BEVs. This algorithm is primarily a VNS approach with TS employed as a local search procedure. The acceptance criterion utilized is based on SA. During the search process, infeasible solutions are permitted but penalized in the objective function. The shaking step employs neighborhoods based on the *cyclic-exchange operator* [333]. To evaluate the algorithm's performance, two sets of instances for that problem were introduced, comprising 56 large instances and 36 small instances, which were derived from instances in [315]. The algorithm's parameters were fine-tuned using a subset of 10 large instances. Additionally, the algorithm was tested on benchmark instances for three different problems: Multidepot VRP with Interdepot Routes, GVRP, and VRP with Time Windows. In each case, the algorithm demonstrated promising results within a short computational time.

Jabir, Panicker, and Sridharan [147] examined a multi-objective variant of the CVRP with multiple depots. The objective of this problem was to minimize both CO_2 emissions and the total distribution cost. The authors employed a two-step approach, starting with the ACO algorithm to generate a set of Pareto-optimal solutions. Subsequently, the VNS was applied to this set of solutions. The proposed approach was evaluated using a collection of randomly generated instances. The results revealed that minimizing the cost does not necessarily lead to reduced emissions. However, the algorithm was not compared to other existing methods, making it difficult to assess its effectiveness in comparison to alternative approaches.

In the study conducted by Ene et al. [84], a GVRP with a heterogeneous fleet and time windows was examined. The primary goal was to minimize fuel consumption while determining the optimal composition of the fleet. To address this problem, the authors proposed a hybrid metaheuristic that combines SA and TS. The effectiveness of this approach was assessed across various problem variations, including CVRP, VRP with TW, GVRP, and GVRP with TW. The algorithm consistently achieved favorable solutions in terms of fuel consumption, distance, and computational time across all tested problem instances.

In a study conducted by Suzuki [326], the combined usage of SA and TS is exemplified. The focus of the research is on the CVRP, with the objective of minimizing total fuel consumption. Recognizing that fuel consumption is directly influenced by route distance and vehicle payload, the problem is formulated as a bi-objective model. The inter-customer distances are adjusted to account for various factors impacting fuel consumption, including vehicle speed, road gradient, and congestion. It is important to note that reducing the distance traveled does not necessarily guarantee a reduction in emissions and fuel consumption [83]. Nevertheless, the assumption made in [326] suggests that the optimal solution likely resides within the Pareto front when considering payload and distance as minimization objectives. The proposed methodology consists of two steps. In the initial step, the SA algorithm is employed to approximate the Pareto front. Subsequently, a modified version of TS is applied

to each solution within the Pareto front, focusing solely on neighborhoods close to the frontier. Through this approach, the author successfully obtained solutions with reduced fuel consumption compared to the conventional single-objective formulation of the problem.

The research conducted by Jabir, Panicker, and Sridharan [146] explores a multi-depot variation of the VRP. The study investigates three distinct models that address different aspects of the problem. The first model focuses on minimizing the economic cost, which aligns with the conventional Multi-Depot VRP. The second model aims to minimize the overall emissions, while the third model seeks to strike a balance between economic cost and emissions by simultaneously minimizing both factors. To tackle this problem, the authors propose two metaheuristic algorithms: ACO and a hybrid approach combining ACO with VNS. In the hybrid approach, after each ant constructs its solution, a VNS procedure is applied to enhance the quality of the solution. Following the completion of VNS, the global pheromone matrix is updated. Comparative analysis revealed that the hybridized version outperformed the pure ACO method, particularly for larger problem instances.

The problem of time-dependent vehicle routing and scheduling, with the objective of minimizing CO_2 emissions, was explored by Xiao and Konak [366], building upon the work of Xiao and Konak [367]. The study focused on a scenario involving a fleet of heterogeneous vehicles and soft time windows, where tardiness incurred penalties in the form of a tardiness penalty in the objective function. To address this problem, the authors proposed a MILP model, which could be optimally solved by commercial solvers but was limited to small-scale instances. For larger instances, a hybrid approach combining a genetic algorithm and *dynamic programming* (**DP**) was introduced. The problem was divided into two parts: determining optimal routes for the vehicles and scheduling the travel between specific customers for each vehicle. For the scheduling aspect, a dynamic programming formulation was presented to identify the best schedule given a set of routes. The original problem was solved using a genetic algorithm focused solely on the routing component, with DP serving as a subroutine to determine the optimal schedule for each solution. The proposed approach was evaluated on 30 small-sized instances and 14 benchmark instances of the CVRP, yielding successful outcomes.

The study conducted by Zhang et al. [381] focuses on the EVRP. The algorithm devised in the research initially determines the electric energy consumption based on factors such as vehicle weight, speed, distance, and motor efficiency. Subsequently, the estimation of indirect CO_2 emissions is performed, which corresponds to the emissions resulting from electrical energy production in coal-based power plants and is proportional to the energy utilized by the BEV from the battery. Armed with this information, the authors formulated the problem as a MILP model and applied the ACO algorithm. To further enhance solution quality, ACO was hybridized with the Iterated Local Search algorithm. The pheromone matrix is updated following the elitist rule, allowing only a set of ants that have found the best solutions thus far to update the trail. Additionally, the paper introduces an ALNS for solving the EVRP at hand. The algorithms were tested on a set of generated instances, demonstrating that ACO produced solutions close to optimality, with an average gap of 3.20%. In comparison to MILP and ALNS, ACO exhibited superior performance.

Li et al. [196] proposed a two-stage algorithm to tackle the Multi-Depot GVRP with time windows. The algorithm aims to optimize multiple objectives simultaneously, including fuel consumption, carbon emissions, and other associated costs. In the first stage, the algorithm employs the *Improved Balanced K-means Algorithm* (**IBKA**) to cluster customers into groups, effectively dividing the problem into smaller

subproblems. Subsequently, in the second stage, a hybrid ACO approach is employed to address these subproblems, with the inclusion of VNS as a local search strategy. By combining these techniques, the algorithm strives to find high-quality solutions for the problem.

Li, Lim, and Tseng [195] introduced a variant of the GVRP with time windows that specifically addresses the requirements of cold chain logistics. In this context, refrigerated vehicles consume more fuel compared to regular vehicles due to the need to maintain low temperatures, resulting in increased GHG emissions. The objective of the problem is to minimize the overall cost, which encompasses various factors such as GHG emission costs (either CO_2 emissions only or all GHG types), penalties for time window violations, product freshness, quality loss, vehicle operating costs (e.g., maintenance and personnel), and energy costs. To address this problem, the authors proposed a modified version of the PSO algorithm (MPSO). MPSO incorporates TS as an intensification method to enhance its search capabilities. For comparison purposes, a standard PSO algorithm was also implemented, but the MPSO algorithm outperformed it in terms of solution quality. In addition, the study highlighted that considering all GHG emissions instead of solely focusing on CO_2 leads to solutions with improved overall cost performance.

The work by Wang and Lu [352] addresses the GVRP with a fleet of AFVs. To tackle this problem, the authors propose a hybrid algorithm that combines a MA with a competition mechanism and VNS. The proposed method in [352] represents solutions as permutation arrays of customers and AFVs, treating the problem as a TSP. These arrays are then decoded into instances of the GVRP. The initialization of solutions in the population begins with the k-nearest neighbors algorithm, starting from a randomly chosen point for each individual. To enhance the intensification process, the MA is hybridized with a VNS algorithm that employs a SA based acceptance criterion. In a competition search conducted by the authors, a specific number of solutions are selected from the population based on their quality. These selected solutions undergo further improvement through intensification procedures followed by a series of adjustments for customers and AFVs. The competitive search allocates more computational resources to the most promising solutions. Once the adjustments are completed, a crossover operator is applied. By combining the Memetic Algorithm with VNS and the competition mechanism, the proposed method offers a promising approach for solving the GVRP with a fleet of AFVs, allowing for effective optimization of the routing problem with environmental considerations.

The research conducted by Zhang et al. [384] focuses on the GVRP involving AFVs and multiple depots. The study proposes two algorithms: the *Partition-Based Algorithm* (**PBA**) and the *Two-stage ACS* (**TSACS**). The PBA algorithm adopts a partitioning strategy to divide customers into two categories: *borderline* customers, located approximately between two depots, and *non-borderline* customers. Non-borderline customers are automatically assigned to their nearest depot, and GVRP routes are generated for each depot. Subsequently, borderline customers are inserted into existing routes using the cheapest insertion criteria. Local search techniques are then employed to optimize the solution further and eliminate redundant nodes. On the other hand, the TSACS algorithm utilizes two types of ants: depot-ants responsible for assigning customers to depots and route-ants for generating routes. After the depot-ants allocate customers to depots and the route-ants generate routes, a *variable neighborhood scheme* is applied to enhance the solution's quality. This scheme bears resemblance to the variable neighborhood search algorithm, with the distinction that the neighborhood size in the perturbation (shaking) step is not dynamically determined. Finally, an additional local search phase is performed, incorporating

redundancy removal and relocation operators. Numerical evaluations demonstrate that the TSACS algorithm surpasses the PBA algorithm in terms of solution quality. Moreover, TSACS exhibits superior average speed compared to CPLEX.

The work conducted by Zhen et al. [389] focuses on the VPR with HEVs. As HEVs possess the capability to operate in two modes, utilizing both gasoline and electrical energy, it becomes crucial to determine the optimal mode for each segment of the vehicle's route in order to minimize the overall energy consumption cost. To address this problem, the authors propose a PSO algorithm with VNS employed as a local search method. Additionally, the paper introduces a labeling procedure responsible for assigning a vehicle mode to each route segment and evaluating the solution quality. The proposed approach is evaluated on three sets of instances: small, medium, and large-scale. For small-scale instances, the algorithm consistently achieves optimal solutions. Additionally, it demonstrates success in finding good solutions for a subset of the large-scale instances, indicating its effectiveness in handling real-world VRP scenarios with HEVs.

In their research, Li et al. [193] explored the optimization of a fleet of Plug-In Hybrid Electric Vehicles with the flexibility to refuel at either gas stations or AFS. The study aimed to minimize total transportation costs while adhering to constraints on the number of routes and the maximum duration of each trip. To model this problem, the authors employed a MILP framework and introduced a hybrid metaheuristic approach for efficient solution finding. This advanced hybrid method combines a memetic algorithm, a Sequential VND (utilizing five neighborhood structures), and an enhanced 2-opt method. The VND is specifically applied to refine the best individual in each generation. The proposed approach demonstrated good precision and robustness in performance.

The multi-depot GVRP is investigated by Peng et al. [271] with the goal of minimizing the total cost, which includes the cost of CO_2 emissions. The problem is formulated as the MILP model and solved using the CPLEX solver. The authors also propose a hybrid evolutionary algorithm that combines an evolutionary algorithm with VNS. In this approach, VNS is used to enhance the search process in promising areas.

In their work, Utama et al. [346] proposed a hybrid approach based on the Butterfly Optimization Algorithm (BOA) for addressing the GVRP. The objective of the GVRP is to minimize the total cost, encompassing emission costs, fuel consumption, and vehicle usage costs. BOA is a metaheuristic method introduced by Arora and Singh [17], which leverages fragrance as a means of communication between butterflies (agents). The fragrance represents the quality of a solution and can attract other butterflies towards it. To tackle the GVRP, the authors combined BOA with the TS algorithm from the study conducted by Poonthalir and Nadarajan [274]. Specifically, TS updates 10% of the initial butterfly population. Additionally, a local search method is employed in each iteration to enhance the solution quality. The proposed approach was compared to several other metaheuristics, demonstrating superior results in terms of solution quality, albeit with potentially longer computation times.

In their research, Dewi and Utama [71] investigated the application of the *Hybrid Whale Optimization Algorithm* (**HWOA**) to address the GVRP. The primary objective was to minimize the overall cost, encompassing fuel expenses and emission costs. HWOA combines the *whale optimization algorithm* (**WOA**) proposed by Mirjalili and Lewis [235] with TS and local search techniques. WOA is inspired by the behaviors exhibited by whales, such as *encircling prey*, *bubble-net attacking method* (exploitation phase), and *search for prey* (exploration phase). The proposed approach was

compared to several other metaheuristics, including TS, SA, ACO, PSO, GA, and WOA, to assess its effectiveness and performance.

In a study conducted by Olgun, Koç, and Altıparmak [258], a variant of GVRP with simultaneous pickup and delivery is examined. The primary objective is to minimize fuel consumption for a fleet composed of conventional vehicles. To obtain high-quality solutions, the authors propose a hyper-heuristic based on ILS and VND.

In their study, Sadati and Çatay [298] investigated a Multi-Depot version of the GVRP, specifically focusing on the use of AFVs. Their primary aim was to reduce the cumulative distance covered by all AFVs. To address this, the researchers developed a MILP model. Moreover, they introduced an innovative hybrid metaheuristic approach, combining VNS and TS, to solve the problem with greater efficiency.

In the research conducted by Ferreira and Steiner [94], they investigated an asymmetric bi-objective variant of the GVRP. The study aimed to minimize two objectives: total CO_2 emissions and route disbalance. To address this problem, several metaheuristics were proposed. The first algorithm was NSGA-II, developed by Deb et al. [68], followed by a *multi-objective Particle Swarm Optimization* (MO-PSO) algorithm. Additionally, two hybrid algorithms were introduced: one that combined CW with NSGA-II (CWNSGA-II), and another that combined CW, TS, and NSGA-II (CWTSNSGA-II). These four algorithms were evaluated using a case study involving newspaper distribution. The results indicated that CWNSGA-II and CWTSNSGA-II outperformed NSGA-II and MO-PSO in terms of solution quality.

In their insightful study, Sadati, Akbari, and Çatay [297] investigated a variant of the EVRP characterized by flexible delivery options. This version allows customers to choose from multiple alternative delivery locations, each with its preferred time window. The study aimed to minimize both the total distance covered by the fleet and the number of vehicles in use. To address these objectives, the authors developed a MILP model and introduced a strategic combination of VNS and TS. Additionally, they created a new set of instances specifically tailored to this flexible delivery scenario.

In their study, Farahani, Zegordi, and Kashan [90] analyzed a diverse fleet comprising both autonomous electric vehicles and traditional vehicles, factoring in constraints like time windows and multiple compartments. The research aimed at minimizing total operational costs, which include overloading, fuel consumption, and driver costs for conventional vehicles, as well as charging expenses for autonomous electric vehicles. The authors developed a MILP model and introduced an innovative hybrid algorithm that merges LNS with VNS. This algorithm incorporates various neighborhood structures, including customer and charging station adjustments, vehicle type modifications, and a mechanism to address loading constraints. The findings revealed that integrating autonomous vehicles into the fleet significantly reduces operational costs in vehicle routing problems.

In a study by Stamadianos et al. [318], authors explored the close-open EVRP, where electric vehicles have the option to conclude their routes at either an AFS or a depot. The research aimed at minimizing both energy consumption and the number of vehicles used. To achieve this, the authors introduced a hybrid metaheuristic approach. This method begins with a GRASP-like construction technique to generate initial solutions. Subsequently, it employs a synergy of VNS and SA acceptance criteria to enhance these solutions further.

3.7.3 Statistics

In this review, we analyzed a comprehensive collection of 72 studies spanning a decade, from 2013 to 2023. The green vehicle routing problems addressed in these papers are broadly classified into two distinct categories: *ICV-based* and *AFV-based*. It is important to note that the ICV-based GVRP category encompasses all problems that exclusively involve ICVs, with at least one objective centered on minimizing GHG emissions or reducing fuel consumption. Conversely, the AFV-based category focuses on problems where at least a part of the fleet is made of AFVs. A total of 48.6% of the reviewed publications focus on AFVs, while the remaining 51.4% address ICVs.

Table 3.1 provides a comprehensive overview of the attributes associated with each ICV-based problem, along with the methods employed to address these problems. Figure 3.7 presents a visual representation of the number of ICV-based papers corresponding to various VRP attributes, as well as a pie chart that illustrates the distribution of metaheuristics utilized in these studies. It is important to note that if a paper utilizes a hybrid metaheuristic approach, it is attributed to each of the constituent metaheuristics. The figure exclusively showcases metaheuristic approaches, omitting mathematical models. This figure reveals a predominant focus on capacitated versions of VRP in the literature. A prevalent trend observed is the simultaneous optimization of multiple objectives, most notably the dual focus on minimizing GHG emissions and economic costs. While attributes such as TW, MD, and HF have garnered considerable attention, attributes like MT and A remain relatively underexplored in the existing body of work.

TABLE 3.1: Overview of ICV-Based publications

Publication	Attributes							Methods
	C	TW	MD	MT	HF	MO	A	
Adiba, Aahmed, and Youssef [3]	✓							GA
Ayadi et al. [23]	✓			✓				GA
Bhattacharjee et al. [33]	✓		✓		✓	✓		ACO
Cooray and Rupasinghe [54]	✓							GA
Costa et al. [55]	✓							GA
Dewi and Utama [71]	✓							HWOA
Elbouzekri, Elhassania, and Alaoui [83]	✓							ACS-LNS
Ene et al. [84]	✓	✓			✓			SA-TS
Ferreira and Steiner [94]	✓					✓	✓	NSGA-II, PSO, CWNSGA-II, CWTSNSGA-II
Giallanza and Puma [106]	✓		✓			✓		NSGA-II
Hsueh [139]	✓				✓			GA
Jabir, Panicker, and Sridharan [147]	✓		✓			✓		ACO-VNS
Jabir, Panicker, and Sridharan [146]	✓		✓			✓		ACO-VNS, LINGO
Karagul et al. [158]	✓							SA
Küçükoğlu et al. [173]	✓	✓						SA
Kumar et al. [174]	✓	✓				✓		PSO, NSGA-II
Kwon, Choi, and Lee [176]	✓				✓			TS
Li, Soleimani, and Zohal [197]	✓		✓			✓		ACO
Li et al. [196]	✓	✓	✓					ACO-VNS
Li, Lim, and Tseng [195]	✓	✓				✓		PSO-TS
Micale et al. [230]	✓	✓				✓	✓	FA
Niu et al. [254]	✓	✓						TS
Niu et al. [255]	✓	✓				✓		MIMOA
Norouzi, Sadegh-Amalnick, and Tavakkoli-Moghaddam [257]	✓					✓		PSO
Olgun, Koç, and Altıparmak [258]	✓							ILS-VNS
Peng et al. [271]	✓		✓					EA-VNS, CPLEX
Poonthalir and Nadarajan [274]	✓					✓		PSO
Suzuki [326]	✓					✓		SA-TS
Tunga, Bhaumik, and Kar [339]	✓					✓		GA
Úbeda et al. [341]	✓							TS
Utama et al. [346]	✓							BOA
Utama, Fitria, and Garside [345]	✓	✓						ABC
Wang et al. [353]	✓	✓	✓					PSO
Xiao and Konak [367]		✓				✓		SA, CPLEX
Xiao and Konak [366]	✓	✓			✓			DP-GA, CPLEX
Yu et al. [377]	✓	✓						ALNS, CPLEX
Zulvia, Kuo, and Nugroho [391]	✓	✓				✓		GE

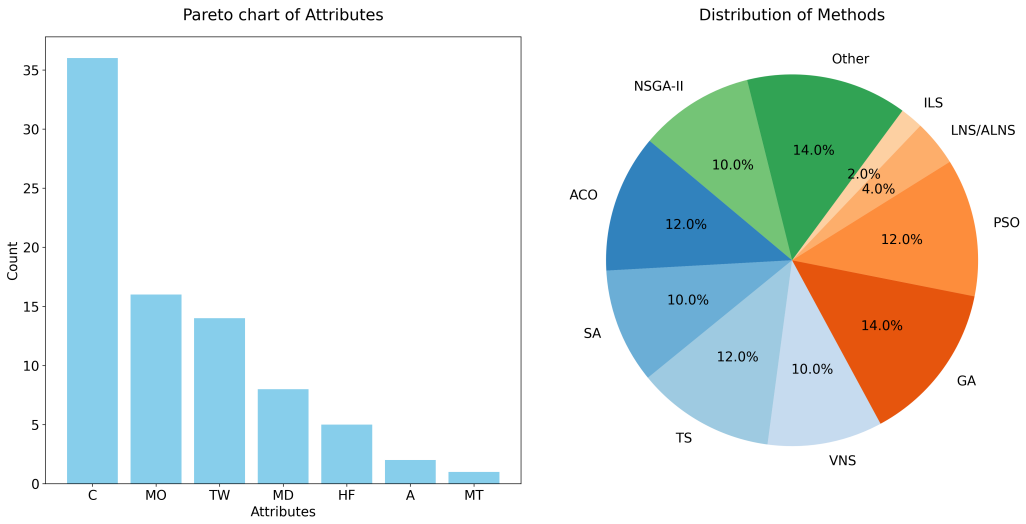

FIGURE 3.7: Graphical overview of attributes and methods in ICV-Based GVRP publications.

Table 3.2 provides a detailed summary of AFV-based GVRP, analogous to the one for publications considering ICV-based GVRP. Figure 3.8 displays both the frequency of each attribute's consideration in the papers and the proportion of metaheuristic

approaches utilized. In line with the trends observed in ICV-based GVRP research, the capacitated version of VRP remains a primary focus. However, a notable divergence is observed in the lesser prominence of the MO attribute in AFV-based studies. This difference is likely attributable to the inherent GHG reduction benefits of using AFVs. The pie chart further reveals a distinct trend in methodological preferences for AFV-based problems: VNS and ALNS are more prevalent, while GA and TS see reduced usage compared to their usage in ICV-based GVRP.

Table 3.2: Overview of AFV-Based publications

Publication	**Attributes**						**Methods**
	C	**TW**	**MD**	**HF**	**MO**	**PR**	
Affi, Derbel, and Jarboui [5]	✓						GVNS
Andelmin and Bartolini [10]	✓						MSLS
Bruglieri et al. [44]	✓					✓	VNB, CPLEX
Farahani, Zegordi, and Kashan [90]	✓	✓					LNS-VNS
Felipe et al. [92]	✓			✓		✓	SA, CPLEX
Goeke and Schneider [111]	✓	✓		✓			ALNS
Hien, Dao, and Binh [132]	✓						GA
Hiermann et al. [134]	✓	✓		✓			ALNS, CPLEX
Hiermann et al. [133]	✓	✓		✓		✓	GA
Keskin and Çatay [160]	✓	✓				✓	ALNS, CPLEX
Li et al. [194]	✓		✓				ACO
Li et al. [193]	✓						MA-VND
Macrina et al. [207]	✓	✓		✓		✓	LNS, CPLEX
Macrina et al. [208]	✓	✓		✓		✓	ILS, CPLEX
Mavrovouniotis, Ellinas, and Polycarpou [225]	✓						ACO
Mavrovouniotis et al. [226]	✓						ACO
Normasari et al. [256]	✓						SA, CPLEX
Peng et al. [272]							MA
Ren et al. [286]	✓	✓		✓	✓		VNS
Sadati and Çatay [298]	✓		✓				VNS-TS
Sadati, Akbari, and Çatay [297]	✓	✓					VNS-TS
Schneider, Stenger, and Goeke [302]	✓	✓					VNS-TS, CPLEX
Stamadianos et al. [318]	✓						VNS-SA
Yu et al. [376]	✓						SA
Wang and Lu [352]	✓						MA-VNS
Woller, Kozák, and Kulich [361]	✓						GRASP
Xiao et al. [365]	✓	✓					MA
Xu et al. [370]	✓	✓					ALNS
Yavuz and Çapar [373]			✓	✓	✓		VNS, CPLEX
Yilmaz and Kalayci [374]	✓						VNS
Zhang et al. [381]	✓						ACO-ILS, ALNS, CPLEX
Zhang, Gajpal, and Appadoo [380]	✓						ACS, CPLEX
Zhang et al. [382]	✓		✓				ACS
Zhang et al. [384]	✓		✓				ACS, PBA, CPLEX
Zhen et al. [389]	✓						PSO, CPLEX

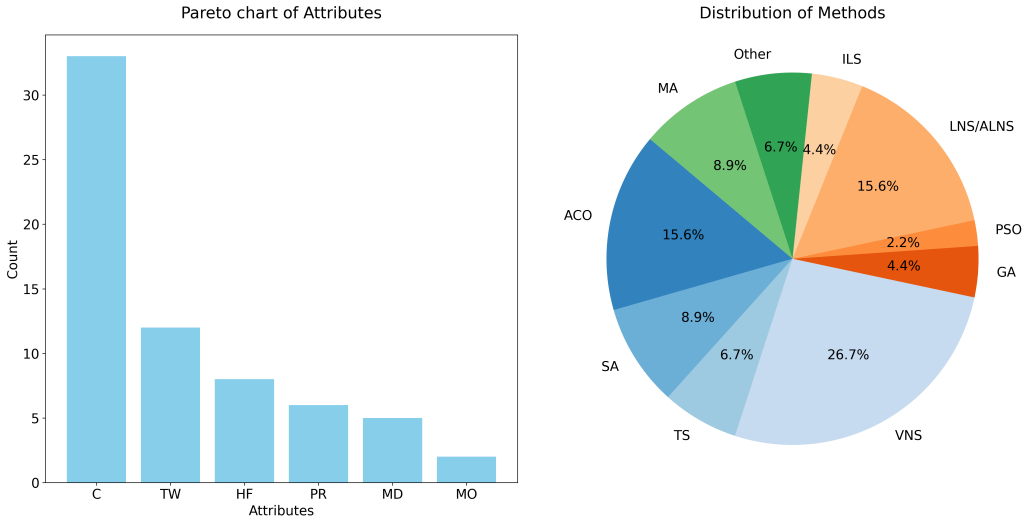

FIGURE 3.8: Graphical overview of attributes and methods in AFV-Based GVRP publications.

Figure 3.9 presents a comprehensive summary across all reviewed publications dealing with GVRP, revealing that attributes such as TW, MD, and HF are the most researched in GVRP studies. This trend suggests their significant practical relevance for numerous delivery companies. The analysis indicates that ACO and VNS are the predominant metaheuristics in GVRP research, closely followed by SA, TS, GA, and LNS/ALNS. It is worth noting that ACO and VNS are popular solutions for both ICV-based and AFV-based problems.

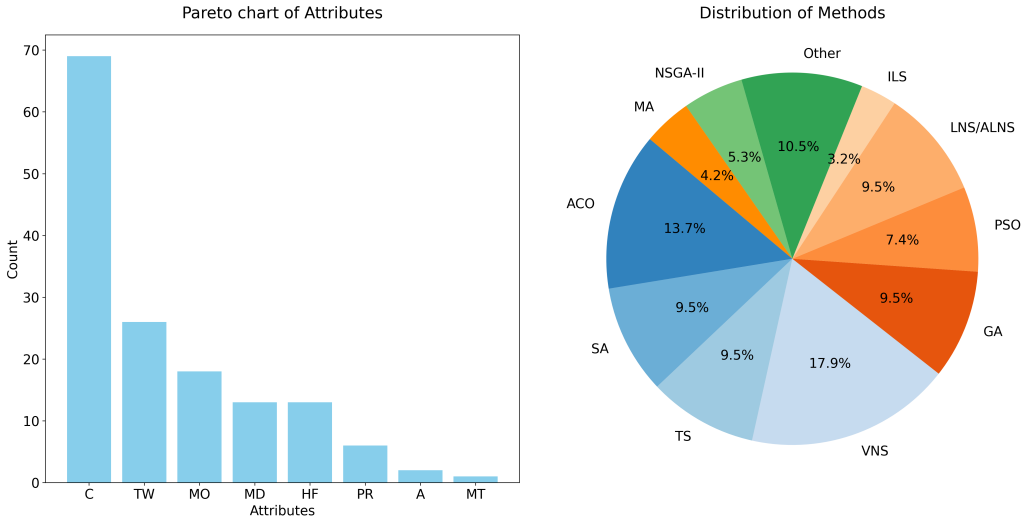

FIGURE 3.9: Graphical overview of attributes and methods in all GVRP publications.

Table 3.3 offers a detailed enumeration of the objectives targeted in each reviewed publication. These objectives were either amalgamated into a singular goal or addressed individually using multi-objective optimization techniques. Figure 3.10

illustrates the frequency of each objective across the reviewed papers, examining ICV-based and AFV-based problems both individually and collectively. From this figure, it is evident that the most prevalent objectives in GVRP research are PE, FC, OC, and TD. Notably, TD appears significantly more frequently in studies focusing on AFV-based problems.

Table 3.3: Summary of objectives addressed in each research paper

ICV-based		AFV-based	
Publication	**Objectives**	**Publication**	**Objectives**
Adiba, Aahmed, and Youssef [3]	PE	Affi, Derbel, and Jarboui [5]	FC
Ayadi et al. [23]	PE, MXO	Andelmin and Bartolini [10]	TDM
Bhattacharjee et al. [33]	RM, TT, PE, OC	Bruglieri et al. [44]	NT, TT, RT, WT
Cooray and Rupasinghe [54]	FC	Farahani, Zegordi, and Kashan [90]	OC
Costa et al. [55]	PE	Felipe et al. [92]	RC
Dewi and Utama [71]	FC, PE, OC	Goeke and Schneider [111]	TDM, OC, FC
Elbouzekri, Elhassania, and Alaoui [83]	PE	Hien, Dao, and Binh [132]	TDM
Ene et al. [84]	FC	Hiermann et al. [134]	OC
Ferreira and Steiner [94]	PE, RI	Hiermann et al. [133]	FC, OC
Giallanza and Puma [106]	OC, PE	Keskin and Çatay [160]	TDM
Hsueh [139]	FC, PE, OC	Li et al. [194]	OC, TWPC, FC, PE, QT
Jabir, Panicker, and Sridharan [147]	PE, OC, FC	Li et al. [193]	OC
Jabir, Panicker, and Sridharan [146]	PE, TD, NT	Macrina et al. [207]	FC
Küçükoğlu et al. [173]	FC, PE	Macrina et al. [208]	OC, EC, RC
Karagul et al. [158]	PE, TD	Mavrovouniotis, Ellinas, and Polycarpou [225]	TT
Kumar et al. [174]	OC, FC, TWPC	Mavrovouniotis et al. [226]	TT
Kwon, Choi, and Lee [176]	OC, PE	Normasari et al. [256]	TD
Li, Lim, and Tseng [195]	PE, TWPC, OC, QLC, FC	Peng et al. [272]	TD
Li, Soleimani, and Zohal [197]	PE, TT, OC, RM	Ren et al. [286]	PE, TWPC
Li et al. [196]	FC, PE, OC	Sadati and Çatay [298]	TD
Micale et al. [230]	TD, PE, FC	Sadati, Akbari, and Çatay [297]	TD, NV
Niu et al. [254]	PE, OC	Schneider, Stenger, and Goeke [302]	NT, TD
Niu et al. [255]	TWPC, OC	Stamadianos et al. [318]	FC, NV
Norouzi, Sadegh-Amalnick, and Tavakkoli-Moghaddam [257]	TT, FC	Yu et al. [376]	OC
Olgun, Koç, and Altıparmak [258]	FC	Wang and Lu [352]	TD
Peng et al. [271]	FC, PE, OC	Woller, Kozák, and Kulich [361]	TD
Poonthalir and Nadarajan [274]	FC, OC	Xiao et al. [365]	TD
Suzuki [326]	FC	Xu et al. [370]	TT, NV
Tunga, Bhaumik, and Kar [339]	FC, RI	Yavuz and Çapar [373]	TD, PE, FC, ICVU
Úbeda et al. [341]	PE	Yilmaz and Kalayci [374]	TD
Utama et al. [346]	EC, FC, OC	Zhang et al. [381]	PE
Utama, Fitria, and Garside [345]	FC, TWPC	Zhang, Gajpal, and Appadoo [380]	TD
Wang et al. [353]	PE, TWPC, OC	Zhang et al. [382]	PE
Xiao and Konak [367]	PE, TWPC, TT, TD	Zhang et al. [384]	PE
Xiao and Konak [366]	PE, TWPC	Zhen et al. [389]	FC
Yu et al. [377]	PE		
Zulvia, Kuo, and Nugroho [391]	OC, QLC, PE, TWPC		

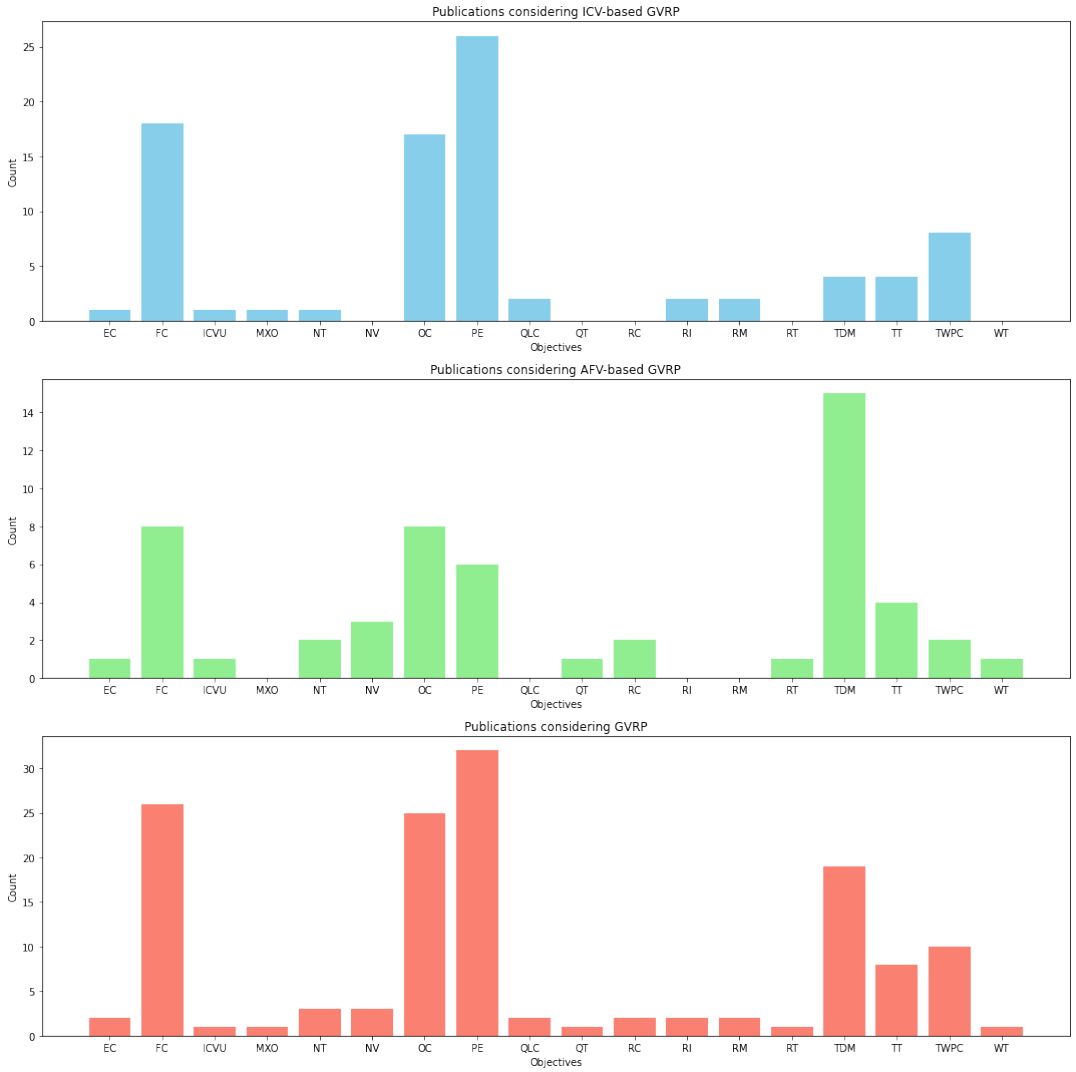

FIGURE 3.10: Bar charts depicting objectives addressed in GVRP-related publications.

3.8 Chapter conclusion

In this chapter, we focus on the central theme of this thesis - the Green Vehicle Routing Problem. We initiate the discussion by exploring the fundamentals of the Vehicle Routing Problem, including the presentation of a MILP model and a graphical illustration of the problem. This sets the stage for a deeper dive into the GVRP, where we examine three distinct variations of the problem.

Particularly relevant to this thesis is the Electric Vehicle Routing Problem, which we explore as a subset of one of these GVRP categories. As the chapter progresses, we delve into various scenarios (or attributes) of the GVRP, highlighting some of the most prevalent ones. We then shift our focus to the potential objectives that can be integrated into the GVRP framework.

The chapter includes an overview of different energy consumption and recharging models for electric vehicles, an aspect crucial to understanding and solving the GVRP effectively. This comprehensive exploration lays a robust foundation for addressing the complexities of green vehicle routing in the context of this thesis.

Finally, we offer a thorough literature review on the GVRP, intentionally excluding publications focused on reverse logistics due to their irrelevance to this thesis. Our review encompasses a total of 72 publications specifically pertaining to the GVRP. Additionally, we analyze existing survey papers on the topic, elucidating the distinctions among them. To provide a comprehensive overview, we also include statistics related to the attributes, methods, and objectives discussed in these reviewed papers, painting a detailed picture of the current state of research in the field of GVRP.

CHAPTER
FOUR

ELECTRIC VEHICLE ROUTING PROBLEM

4.1 Problem definition

Let us consider a distribution company equipped with a fleet of uniform electric vehicles. Daily, this firm faces numerous delivery requests from its clientele, aiming to meet these demands efficiently. Customers specify both the quantity of goods and a preferred delivery timeframe. While dispatching, the firm ensures no vehicle is overloaded beyond its capacity. Given the limited range of EVs, occasional stops at refueling stations are inevitable. Unlike internal combustion vehicles, EVs take longer to recharge, requiring strategic scheduling to ensure timely deliveries. Deliveries outside a customer's preferred window risk diminishing client satisfaction, leading them to seek services elsewhere. Both early and late arrivals present challenges. Some suggest making vehicles wait to align with the start of the delivery window, but this can jeopardize punctuality for subsequent deliveries. Hence, our model permits early arrivals, albeit with penalties. Late arrivals are also permitted with penalties. Moreover, real-life refueling stations can only service a limited number of EVs simultaneously and offer varied charging speeds and prices due to differing technologies. In our study, we have simplified several real-world characteristics that can influence EVRP. We have assumed that refueling stations can accommodate an infinite number of vehicles and have a uniform recharging rate. While factors like road gradient, vehicle weight, and distance typically result in nonlinear energy consumption models (detailed in Section 3.5), we have not treated consumption as a direct proportion of distance for this study. Our rationale is to prioritize the development of robust metaheuristic methods that can quickly yield satisfactory results. Similarly, though real-world charging rates vary based on battery levels, we have represented them linearly. Incorporating the true nonlinear characteristics into our metaheuristic would not alter our algorithm's structure. We have also presumed that EVs recharge to full capacity whenever they stop. However, simply considering distances between nodes falls short in real-world scenarios. A critical limitation is the assumption that vehicle speed remains constant throughout the day, which is particularly unrealistic in urban settings. Factors like construction work and rush hour traffic significantly alter the average vehicle speed. While some of these external influences are unpredictable, others, like traffic volume variations by hour, can be anticipated and factored in using traffic counter data. A study by Cardelino [46] has already explored hourly traffic volume variations. To account for varying road conditions and traffic patterns, we have divided the entire service timeframe into smaller segments, each having its average travel speed. This segmentation ensures that fluctuating speeds, affecting timely

deliveries, are considered. Incorporating AFS visits, soft time windows for early and late deliveries, vehicle scheduling, and time-dependent speeds into the VRP results in a highly complex problem. Achieving optimal solutions is extremely challenging except for very small instances.

A more formal definition of EVRP is given in Definition 2.

Definition 2 *Given a weighted graph $\hat{G} = (\hat{N}, A, d)$ where vertices $\hat{N} = \{D\} \cup V \cup F$ represent a central depot (D), customer locations (V) with specified goods demand and delivery time windows, and refueling stations (F) for electric vehicles, and where the edges A represent roads between two vertices, and the function d assigns the travel distance to the edges, the objective is to determine optimal paths for a fleet of m uniform-capacity EVs. These paths should originate and conclude at the depot, ensure each customer is visited exactly once, adhere to vehicle capacity and EV battery constraints, respect customer time windows while minimizing penalties for early or late deliveries (calculated by some function $\mathcal{P}$), and incorporate refueling stops as necessary. The duration a vehicle must remain at an AFS depends on the amount of energy it requires for recharging and the recharging rate g. The energy a vehicle consumes when traveling from one location to another depends on the distance between the locations and the vehicle's consumption rate h. The average speed a vehicle can achieve at any given time is influenced by factors such as traffic volume, street congestion, weather conditions, etc. To improve the optimization process, this data on average speed should be estimated in advance and provided as part of the instance data.*

Henceforth, we designate this problem as the *Time-Dependent Electric Vehicle Routing Problem with Soft Time Windows* (**TD-EVRP-STW**).

4.2 Mathematical model

The TD-EVRP-STW problem is structured as a directed graph $G = (N, A)$. Here, V represents customers, F stands for refueling stations, and D indicates the depot. Given that refueling stations might see multiple visits, we introduce dummy nodes (denoted as F') by replicating entries from (F). Therefore, N encompasses the depot, customers, and refueling stations as $N = \{D\} \cup V \cup F' \cup \{D'\}$, with the depot duplicated (D') to distinguish each route's start and end. Arcs A in the graph link these vertices as $A = \{(i, j) \mid i, j \in N, i \neq j\}$, with their weights signifying distances.

Delivery time is chunked into intervals, each with its unique start, end, and average speed, reflecting real-world speed fluctuations due to traffic or other factors. The time to cover each arc considers these intervals, making the model more dynamic and realistic.

Customers come with specific demands and time windows, with penalties imposed for early or late arrivals to maintain service satisfaction. Vehicles are uniform in capacity, consumption rate, and speed per time interval. While each customer's service time is defined as s_i, $i \in V$, refueling service time adjusts based on battery level and refueling rate. Only full battery recharges are factored in.

Our MILP formulation, as presented in our work [220] and influenced by works like [302] and [379], adopts their naming conventions. A comprehensive list of symbols and their explanations can be found in Table 4.1 and Table 4.2.

TABLE 4.1: Parameter definitions

Symbol	Meaning
$0, N+1$	Depots
F	Recharging stations set
F'	Replicated nodes from F (Dummy nodes)
F'_0	Nodes in F' with the depot node 0, i.e., $F' \cup 0$
V	Customer set $V = \{1, ..., N\}$
V_0	Customers and depot: $V_0 = V \cup \{0\}$
V'	Customers combined with recharging stations: $V' = V \cup F'$
V'_0	Customers, recharging stations, and depot 0: $V'_0 = V' \cup \{0\}$
V'_{N+1}	Customers, recharging stations, and depot $(N+1)$: $V'_{N+1} = V' \cup \{N+1\}$
$V'_{0,N+1}$	Customers, recharging stations, with depots 0 and $(N+1)$: $V'_{0,N+1} = V' \cup \{0\} \cup \{N+1\}$
K	Time intervals set
ψ_k	Start of time interval k
υ_k	End of time interval k
L_k	The speed of the vehicle within the time interval $k \in K$.
c_1, c_2	Objective function coefficients
d_{ij}	Distance between node i and node j
t_{ijk}	Travel time from node i to node j in interval k
C	Vehicle's max capacity
g	Rate of recharging
h	Vehicle's consumption rate
Q	Max battery capacity
q_i	Node i demand (0 if $i \notin V$)
s_i	Service time at node i (0 if $i \notin V$)
e_i	Time window start for node i
l_i	Time window end for node i
α	Early arrival penalty coefficient
β	Late arrival penalty coefficient
M	A large constant (upper bound)

TABLE 4.2: Decision variables

Symbol	Meaning
x_{ij}	Binary variable indicating that arc (i, j) is used
z_{ijk}	Binary variable indicating that arc (i, j) is used in time interval k
τ_i	Arrival time at node i
ϕ_i	Departure time from node i
u_i	Remaining cargo upon arriving at node i
y_i	Remaining battery upon arriving at node i
φ_{ijk}	Distance traveled from node i to node j during interval k
μ_i	Early arrival penalty at node i
η_i	Late arrival penalty at node i
p_i	Total penalty at node i[1]

We are now ready to formulate model of the considered TD-EVRP-STW as:

$$min \left(c_1 \sum_{i \in V'_0} \sum_{j \in V'_{N+1}, i \neq j} d_{ij} x_{ij} + c_2 \sum_{i \in V} p_i \right) \tag{4.1}$$

s.t.

$$\sum_{j \in V'_{N+1},\, i \neq j} x_{ij} = 1 \ , \forall i \in V \tag{4.2}$$

$$\sum_{j \in V'_{N+1},\, i \neq j} x_{ij} \leq 1 \ , \forall i \in F' \tag{4.3}$$

$$\sum_{i \in V'_{N+1},\, i \neq j} x_{ji} - \sum_{i \in V'_0,\, i \neq j} x_{ij} = 0 \ , \forall j \in V' \tag{4.4}$$

$$\tau_i + \sum_{k \in K} t_{ijk} + s_i - l_0(1 - x_{ij}) \leq \tau_j \ , \forall i \in V_0, \forall j \in V'_{N+1}, i \neq j \tag{4.5}$$

$$\tau_i + \sum_{k \in K} t_{ijk} + g(Q - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j, \forall i \in F', \forall j \in V'_{N+1}, i \neq j \tag{4.6}$$

$$\mu_i = max(0, e_i - \tau_i) \ , \forall i \in V \tag{4.7}$$

$$\eta_i = max(0, \tau_i - l_i) \ , \forall i \in V \tag{4.8}$$

$$p_i = \alpha \mu_i + \beta \eta_i \ , \forall i \in V \tag{4.9}$$

$$0 \leq u_j \leq u_i - q_i x_{ij} + C(1 - x_{ij}) \ , \forall i \in V'_0, \ \forall j \in V'_{N+1}, \ i \neq j \tag{4.10}$$

$$0 \leq u_0 \leq C \tag{4.11}$$

$$0 \leq y_j \leq y_i - (h \cdot d_{ij}) x_{ij} + Q(1 - x_{ij}), \forall i \in V, \ \forall j \in V'_{N+1}, \ i \neq j \tag{4.12}$$

$$0 \leq y_j \leq Q - (h \cdot d_{ij}) x_{ij}, \ \forall i \in F'_0, \forall j \in V'_{N+1}, \ i \neq j \tag{4.13}$$

$$d_{ij} \cdot x_{ij} = \sum_{k \in K} \varphi_{ijk}, \ \forall i \in V'_0, \ \forall j \in V'_{N+1}, \ i \neq j \tag{4.14}$$

$$\phi_i \geq \tau_i + s_i, \ \forall i \in V'_{0,N+1} \tag{4.15}$$

$$\phi_i \geq \tau_i + g(Q - y_i), \ \forall i \in F' \tag{4.16}$$

$$x_{ij} - z_{ijk} \geq 0, \ \forall i \in V'_0, \ \forall j \in V'_{N+1}, \ \forall k \in K \tag{4.17}$$

[1]This is a dependent variable of μ_i and η_i, introduced solely for the sake of simplicity in the model.

$$\sum_{k \in K} z_{ijk} \geq x_{ij},\ \forall i \in V_0',\ \forall j \in V_{N+1}' \tag{4.18}$$

$$\varphi_{ijk} - d_{ij} \cdot z_{ijk} \leq 0,\ \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.19}$$

$$z_{ijk} - M \cdot \varphi_{ijk} \leq 0,\ \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.20}$$

$$t_{ijk} = \frac{\varphi_{ijk}}{L_k},\ \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.21}$$

$$t_{ijk} \leq v_k - \psi_k,\ \forall i \in V_0', \forall j \in V_{N+1}',\ \forall k \in K \tag{4.22}$$

$$\phi_i + t_{ijk} \leq v_k + v_{N+1}(1 - z_{ijk}), \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.23}$$

$$\psi_k + t_{ijk} \leq \tau_j + v_{N+1}(1 - z_{ijk}) \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.24}$$

$$\phi_i + \sum_{k \in K} t_{ijk} \leq \tau_j + v_{N+1}(1 - x_{ij}) \forall i \in V_0',\ \forall j \in V_{N+1}' \tag{4.25}$$

$$x_{ij} \in \{0, 1\},\ i \in V_0',\ j \in V_{N+1}' \tag{4.26}$$

$$z_{ijk} \in \{0, 1\},\ \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.27}$$

$$\tau_i,\ \phi_i,\ u_i,\ y_i,\ \mu_i,\ \eta_i,\ p_i \geq 0,\ \forall i \in V_{0,N+1}' \tag{4.28}$$

$$\varphi_{ijk} \geq 0,\ \forall i \in V_0',\ \forall j \in V_{N+1}',\ \forall k \in K \tag{4.29}$$

The objective function (referenced in (4.1)) has two primary goals: first, to reduce the overall distance covered by all vehicles and second, to lessen the penalties incurred from missing the customers' time windows (i.e., arriving at customer locations either early or late). The balance between these two objectives is maintained by the constants c_1 and c_2, that are given as the input parameters.

The constraints outlined in (4.2) dictate that every customer must be serviced once and only once. Additionally, (4.3) specify that dummy nodes, which represent refueling stations, can be approached maximum one time.

For the continuity of every route, (4.4) state that all nodes, barring the depot, should possess an equal count of incoming and outgoing arcs. The constraints in (4.5) set the arrival times for every node, ensuring the time-appropriateness of arcs departing from the depot and customer locations. Similarly, (4.6) ensure the same for arcs that originate from refueling stations.

Constraints (4.7), (4.8), and (4.9) are tasked with computing the penalties for missing the time windows at individual nodes (essentially computing the value of the function $\mathcal{P}$ mentioned in Definition 2). Given that this penalty is factored into the objective function's minimization, the *max* function can be straightforwardly depicted through a series of linear constraints, as seen in (4.30), without the need to introduce fresh variables.

$$y = max(x_1, x_2, ..., x_n) \Leftrightarrow y \geq x_i,\ \forall i = 1, ..., n \tag{4.30}$$

Bear in mind that this transformation is effective only when the objective function aims to minimize y. For a broader approach to linearization, see [18].

Constraints (4.10) and (4.11) ascertain that there is ample cargo to meet every customer's demand. The provisions in (4.12) and (4.13) ensure the vehicle's battery does not deplete during deliveries.

In (4.14), it is mandated that the distance traversed across all periods should match the distance between nodes i and j, provided the arc (i, j) is utilized. The next set of constraints, (4.15), establish the departure timings from customers and the depot, considering both service and arrival times. In a similar vein, (4.16) set out the exit time from fueling stations.

Constraints (4.17) and (4.18) bridge the variables x and z. They mandate that for any arc (i, j) to be traveled, it must correlate with at least one period. Conversely, in the absence of any traveled arc, no periods should be linked.

As per (4.19), the distance journeyed within any given period should not surpass the arc's full span if the arc (i, j) is in use. The stipulations in (4.20) confirm that if an arc is in play during a period k, the distance covered therein must be non-zero.

Constraints (4.21) spell out the time taken in each interval, while (4.22) cap this duration, ensuring it does not go beyond the period's length. (4.23) then adjust this time, taking the vehicle's departure time into account.

(4.24) see to it that the cumulative travel time within a specific period, combined with the period's start, does not overstep the subsequent node's arrival time. Meanwhile, (4.25) set a ceiling on the total travel time, combining the departure from node i and all periods, ensuring it remains below or equals the arrival at node j.

Lastly, constraints from (4.26) to (4.29) lay out the characteristics of the decision variables.

Given the high complexity of the proposed model, with $N((K+1)N+7)$ variables and $N(2F'+5N+2V+12)+F'+9KN^2+3V-5$ constraints, it is reasonable to expect that approximate approaches, such as metaheuristics, will be better suited for tackling this problem.

4.3 Test instances

There are not many benchmark instances for evaluating EVRP. The first set of instances was introduced together with EVRP itself, by Erdoğan and Miller-Hooks [86]. This set contains 52 instances, each instance containing coordinates of customers and AFSs, together with additional information like the number of vehicles, average velocity, vehicle capacity, battery capacity, etc. The distance between nodes can be calculated using their coordinates by applying the Euclidean distance formula. Building on instances proposed by Solomon [315], Schneider, Stenger, and Goeke [302] introduced a set of 86 instances, with additional information concerning time windows and service time. Another dataset was proposed by Andelmin and Bartolini [11], which contains 40 instances, each with coordinates for customers and AFSs, but without any additional information about vehicles. Koç and Karaoglan [164] proposed 52 instances, in the same format as the instances from [86]. In a preprint by Goeke [110], 92 instances were introduced. They were in a similar format as those from [86] and [164] but with additional information about service time and time windows for each customer. Another 25 instances were introduced by Mavrovouniotis et al. [227], but these instances do not accommodate time windows.

For the purposes of developing and testing our algorithms, we utilized the instances presented in [302]. To facilitate a structured approach to experimentation,

the dataset's instances are methodically segregated into two distinct groups based on the size and complexity of each instance. The initial group encompasses 35 instances, each with a size catering to a maximum of 15 customers. We refer to this subset of instances as I_1. On the other hand, the latter group comprises 56 larger instances, each with 100 customers. We refer to this subset as I_2. This division is crucial to ensuring precise evaluation and comparison of performance across varied instance sizes.

For our research, we divided the day into five distinct periods, each marked by a comparable level of traffic volume:

- *Early Morning (12 AM to 6 AM)*: During these off-peak hours, roads are typically less crowded, and traffic flow is smooth, resulting in higher average vehicle speeds.
- *Morning Rush Hour (6 AM to 9 AM)*: As people commute to work and schools, traffic volume increases significantly, leading to lower average vehicle speeds. Depending on the city, these hours can see some of the lowest speeds of the day.
- *Mid-Day (9 AM to 5 PM)*: There is usually a slight increase in speed during these hours as the morning rush subsides. However, lunchtime activities can still cause moderate traffic in some areas.
- *Afternoon Rush Hour (5 PM to 7 PM)*: This is another period of heavy traffic as people return from work and school. The average speed during this time is typically lower, similar to the morning rush hour.
- *Evening (7 PM to 12 AM)*: As activities wind down, the traffic volume decreases and average vehicle speeds increase again.

This segmentation aimed to approximate peak traffic hours. It is necessary to recognize that this classification primarily reflects typical weekday traffic patterns, with weekend volumes potentially differing significantly.

While this approach to segmentation is quite simplified, it proved to be adequate for the purposes of evaluating our methods. A more nuanced and accurate division could be achieved by analyzing historical traffic data specific to an area. However, it is out of the scope of this work.

In our study, we examined a scenario involving a goods transportation company based in Belgrade, which operates from 7 AM to 8 PM, totaling 13 hours of daily operation. To simulate this, we divided the delivery period into 13 equal subintervals in our models, each characterized by a distinct speed multiplier. These multipliers are inversely related to traffic volume, as per the day division discussed earlier: the first two intervals have a multiplier of 0.75, indicating higher traffic; the next eight intervals are set at a multiplier of 1, signifying normal traffic conditions; the following two intervals have a multiplier of 0.8, again indicating higher traffic; and the final interval returns to a multiplier of 1, reflecting regular traffic.

4.4 Solution representation

In our TD-EVRP-STW, the solution, denoted as S, is visualized as an array made up of doubly linked lists. Each of these lists corresponds to a vehicle, labeled as H_a for a ranging from 1 to m. Within these lists, every element signifies a customer and is characterized by a four-component sequence, S_{ab}, which represents the b-th location visited by the a-th vehicle. This sequence includes:

- S_{ab}^{index}: Designates the index of a customer, depot, or recharging station represented by node S_{ab}.
- $S_{ab}^{arrival}$ and $S_{ab}^{departure}$: Indicate the times of arrival and departure at the node S_{ab}.
- S_{ab}^{type}: Specifies the node's nature, distinguishing between a depot, customer, or recharging station.

The order of nodes within each list showcases the route's traversal sequence. For a comprehensive visualization of our solution's design, refer to Figure 4.1.

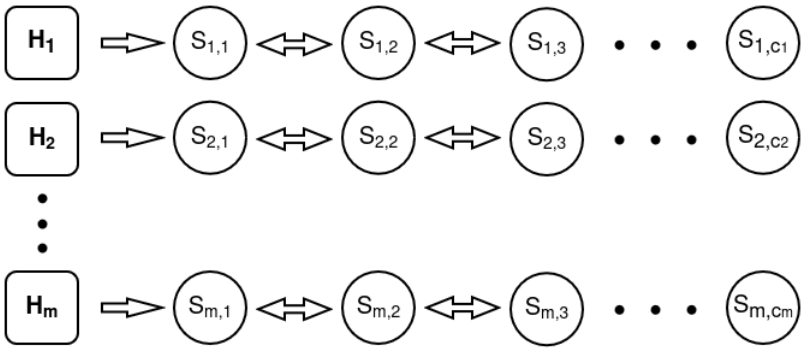

FIGURE 4.1: Representation of a solution

4.5 Solution feasibility

Combinatorial optimization problems often have a large search space of potential solutions, but not all of these solutions satisfy the predefined constraints or criteria of the problem. A solution is deemed "feasible" if it adheres to all such constraints, making it a valid candidate for consideration. Conversely, "infeasible" solutions violate one or more constraints and are typically discarded or repaired. Ensuring feasibility while navigating the search space efficiently is a central challenge in many optimization algorithms.

In the context of EVRP presented in Section 4.1, a solution can be infeasible for two main reasons:

1. The load of one or more vehicles exceeds its capacity.
2. The battery of one or more vehicles was depleted during the trip.

Checking and fixing the feasibility of a solution that violates the first constraint is relatively straightforward. By removing some customers from the problematic route and reinserting them into another, less congested route, a feasible solution can be achieved. However, even though it may be important to fix such solutions while trying to generate initial feasible solutions, our empirical analysis suggests that it is more effective to outright dismiss such solutions whenever possible, for example, during the local search.

On the other hand, solutions that are infeasible because of the second reason have to be fixed, as discarding them would deprive us of too many promising solutions. However, addressing infeasibilities tied to the battery level constraint can be somewhat more intricate than rectifying those related to capacity or customer limit

constraints. Therefore, we propose Algorithm 10 for fixing the infeasibility of such solutions, by adding AFSs wherever necessary.

In this algorithm, we traverse every node for each vehicle, adjusting the current battery level based on the node's type. Specifically, if a node signifies a refueling station, we replenish the battery to its full capacity (Line 8). Otherwise, we deduct the energy expended from the battery (Line 10). Should the battery's current level be insufficient to reach the upcoming customer, we contemplate the addition of a refueling station to rectify the shortfall (Lines 12-24). At every juncture, our objective is to pinpoint the optimal station for insertion at the current position (Line 14). If station insertion is not feasible (Line 15), meaning the battery does not have enough charge to access any stations, we regress to the preceding node and evaluate station placement there. When a station is successfully integrated, we recalibrate battery levels and resume the algorithm from this freshly incorporated station (Line 23). In the worst-case scenario, the algorithm has a complexity of $\mathcal{O}(\frac{N^2}{R})$, where N represents the total number of nodes and R denotes the number of routes in the solution.

Algorithm 10 The algorithm for fixing the feasibility of a solution

```
1: procedure FIX_SOLUTION(Solution S)
2:     for ∀ vehicle ∈ S do
3:         battery ← BATTERY_CAPACITY;
4:         for j = 1 to size(S[vehicle]) do
5:             spent ← calculate consumption between nodes j and (j − 1);
6:             if battery ≥ spent then
7:                 if Node j is a refueling station then
8:                     battery ← BATTERY_CAPACITY;
9:                 else
10:                    battery ← battery − spent;
11:                end if
12:            else
13:                for k = j downto 1 do
14:                    station ←Find the best station to add at position k;
15:                    if No station can be added then
16:                        battery ← battery+ consumption from nodes k and (k − 1);
17:                    else
18:                        Insert station at position k in S;
19:                        break;
20:                    end if
21:                end for
22:                battery ← recalculate battery level;
23:                j ← k;
24:            end if
25:        end for
26:    end for
27:    return S
28: end procedure
```

4.6 Solution construction

Metaheuristics are high-level algorithms designed to tackle complex optimization problems by providing a general strategy to find optimal or near-optimal solutions.

These algorithms typically start their search from a specified initial solution. In some cases, such as with Simulated Annealing or Variable Neighborhood Search, the algorithm operates on a single solution at a time, requiring just one initial point to start. In other cases, like with Genetic Algorithms or Particle Swarm Optimization, the algorithm works with a set of solutions, necessitating multiple initial solutions to create an initial population. The choice of the initial solution or solutions is crucial. If chosen wisely, it can bring the algorithm closer to the optimal region of the search space, allowing for faster convergence and higher-quality solutions. Moreover, a strategically chosen initial solution can help the algorithm avoid regions with many local optima, which are solutions better than their immediate neighbors but not necessarily the best overall. This ensures the algorithm does not get trapped in suboptimal regions and has a better chance of finding the global optimum. So, while metaheuristics inherently possess the capability to explore and exploit a problem's search space, commencing from a favorable initial position significantly amplifies their performance and efficiency.

In this section, we introduce two distinct strategies to create initial solutions. The first strategy involves randomly allocating customers to vehicles. After this assignment, we modify the solution by incorporating trips to refueling stations whenever necessary. While the randomness of this method may be suitable for algorithms that work on a population of solutions, it may not be refined enough for those that focus on a single solution. The method's pseudocode can be found in Algorithm 11. In this algorithm, we begin by incorporating the depot node into each route (Line 2) and populating the set of unassigned customers (A) with all available customers (Line 3). As we proceed, for any remaining unassigned customers, we randomly select a customer from this set (Line 5) and then pick a route at random (Line 6). If the selected route is already at its customer capacity, we bypass that cycle (Lines 7- 9). If not, the chosen customer is appended to the route's end (Line 10) and is subsequently removed from the unassigned set (Line 11). The algorithm cannot get stuck in an infinite loop, as it will eventually find a route where adding a customer is possible. However, it may be useful to exclude fully occupied routes from consideration in Line 6. Ultimately, we refine the solution by scheduling visits to AFSs (Line 13), leveraging the method outlined in Algorithm 10. Although the main algorithm has a complexity of $\mathcal{O}(N)^2$, the solution refinement method (Line 13) has a complexity of $\mathcal{O}(\frac{N^2}{R})$, as discussed earlier. Consequently, the overall complexity of the algorithm is $\mathcal{O}(\frac{N^2}{R})$.

[2]More specifically, the complexity is $\mathcal{O}(|V|)$, where $|V|$ represents the number of customers. However, since $|V| \leq N$, where N is the total number of nodes, we can consider the complexity to be $\mathcal{O}(N)$.

Algorithm 11 The algorithm for generating the initial solution

```
1: procedure INIT_SOLUTION_RANDOM(Instance data D, customer_limit)
2:     S ← initialize the solution by adding depot instances to each route
3:     A ← set of all customers
4:     while A ≠ ∅ do
5:         C ← randomCustomer(A)
6:         vehicle ← randomVehicle()
7:         if size(S[vehicle]) ≥ customer_limit then
8:             continue
9:         end if
10:        add customer C to S[vehicle]
11:        A ← A \ C
12:    end while
13:    S ← FIX_SOLUTION(S)
14:    return S
15: end procedure
```

In Algorithm 12 we present our greedy approach to generating an initial solution, inspired by the procedure presented in [379]. Same as in Algorithm 11, two depot instances are incorporated into every vehicle, symbolizing the starting and concluding points of each route (Line 2). We also set up a collection of customers who are yet to be designated to any vehicle (Line 3). Following this, we determine the optimal placement in terms of the least increase to the objective function value for each of these unassigned customers (Line 7), bearing in mind the maximum number of customers per route. This limit of customers per route was introduced to methods for generating initial solutions to create more balanced routes, as the empirical evidence suggests that more balanced initial solutions typically enhance the performance of our metaheuristics. We then include the customer that has the minimal impact on the objective function into the solution at this ideal position (Line 12), subsequently removing them from the pool of unassigned customers (Line 13). This process is repeated until every customer is allocated. Finally, the solution undergoes a feasibility check, with essential refueling stops being integrated as required (Line 15). The complexity of this algorithm is $\mathcal{O}(|V|^2)$, where $|V|$ is the number of customers.

Algorithm 12 The algorithm for generating the initial solution

```
1: procedure INIT_SOLUTION(Instance data D, customer_limit)
2:     S ← initialize the solution by adding depot instances to each vehicle
3:     A ← set of all customers
4:     while A ≠ ∅ do
5:         best_customer ← −1
6:         for ∀customer ∈ A do
7:             Calculate the best position of customer in S
8:             if customer is better than best_customer then
9:                 best_customer ← customer
10:            end if
11:        end for
12:        Insert the best_customer into the best position in S
13:        A ← A \ {best_customer}
14:    end while
15:    S ← FIX_SOLUTION(S)
16:    return S
17: end procedure
```

4.7 Determining a schedule

To reduce the penalty cost of a solution, it is crucial to ascertain the arrival and departure times for every node. We outline our scheduling method in Algorithm 13. Notably, this algorithm is not designed to find the absolute best scheduling but aims to quickly and deterministically find a high-quality schedule. Pursuing an optimal schedule would significantly complicate the problem due to the expansive search space. Empirical findings suggest that our proposed algorithm typically yields satisfactory results.

The essence of our methodology is quite straightforward. For every route, we navigate through each node (which can be either a depot, customer or an AFS), determining their respective arrival and departure times. Deriving a node's arrival time is simple: it is the sum of the preceding node's departure time and the transit duration between the two nodes (Line 7). An exception is made for the depot node at each route's inception, where the arrival time is designated as zero (Line 5). Travel time is calculated by considering the distance between two nodes, the average vehicle speed (provided as a parameter), and speed multipliers corresponding to different time intervals. The vehicle is not required to cover the entire distance within a single interval, therefore, if a new interval begins while the vehicle is en route, the multiplier for the subsequent time interval is immediately applied to the remaining distance. Regarding a node's departure time, we define it as the sum of the arrival time and the service time (Line 8). For customers, this service time is predefined, whereas for AFSs, it is calculated based on the required energy and the recharge rate, which is provided as a parameter. The complexity of Algorithm 13 is $\mathcal{O}(N)$, where N is the total number of nodes across all routes.

Algorithm 13 The algorithm for calculating a schedule for a given solution

```
1: procedure CALC_SCHEDULE(Solution S)
2:     num_routes ← number of routes in S
3:     for i = 1 to num_routes do
4:         num_nodes ← number of nodes in route S_i
5:         S_i0 ← 0                                          ▷ Depot instance
6:         for j = 2 to num_nodes do
7:             S_ij^arrival ← S_i(j−1)^departure + travel_time(S_i(j−1), S_ij)
8:             S_ij^departure ← S_ij^arrival + service_time(S_ij)
9:         end for
10:    end for
11:    return S
12: end procedure
```

In Algorithm 14, we provide a deterministic method for optimizing the schedule after it has been calculated by CALC_SCHEDULE algorithm. The process involves examining each node of every route to determine if the vehicle arrived at the customer's location earlier than expected (Line 5). If this is the case, then it indicates room for improvement, so we add $diff$ to the departure time of the previous node (Line 7), thus eliminating the penalty for arriving early. We then need to fix arrival and departure times for all of the following nodes, taking into account node types. The worst-case complexity of this algorithm is $\mathcal{O}(N)$, where N is the number of nodes in the solution.

Algorithm 14 The algorithm for optimizing the schedule for a given solution

```
1: procedure OPTIMIZE_SCHEDULE(Solution S)
2:     R ← number of routes in S
3:     for i = 1 to R do
4:         for ∀node ∈ S[i] do
5:             diff ← ready_time(node) − arrival_time(node)
6:             if diff > 0 then
7:                 S_i,node^departure ← S_i,node^departure + diff
8:                 Fix arrival and departure times of all subsequent nodes
9:             end if
10:        end for
11:    end for
12:    return S
13: end procedure
```

4.8 Neighborhood structures

The selection of appropriate neighborhoods for both local search and shaking processes significantly influences an algorithm's effectiveness. In our study, we pinpointed ten distinct neighborhood configurations, with some of them already have been discussed in [5]. In the literature, VRP neighborhoods are commonly categorized as **inter-route** or **intra-route**. Intra-route neighborhoods involve modifications within a single route, while inter-route neighborhoods involve changes that affect multiple routes simultaneously. Additionally, it is helpful to classify neighborhoods based on

the types of nodes involved: **customer-based** neighborhoods modify nodes representing customers, **AFS-based** neighborhoods modify nodes representing AFSs, and **combined** neighborhoods modify any type of node, regardless of its type.

- $\mathcal{N}_1(S, \varepsilon)$: Intra-route customer-based neighborhood that adjusts a customer's departure time by a factor of ε, either increasing or decreasing it. In the worst-case scenario, this neighborhood has $\mathcal{O}(|V| + |F|)$ neighbors.
- $\mathcal{N}_2(S)$: Transfers a customer within the same route to a different position (Figure 4.2). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

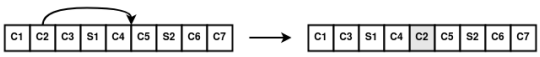

FIGURE 4.2: Illustration of the $\mathcal{N}_2(S)$ neighborhood structure

- $\mathcal{N}_3(S)$: Inter-route customer-based neighborhood that relocates a customer to a different route (Figure 4.3). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

FIGURE 4.3: Illustration of the $\mathcal{N}_3(S)$ neighborhood structure

- $\mathcal{N}_4(S)$: Intra-route customer-based neighborhood that exchanges the spots of two customers within a single route (Figure 4.4). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

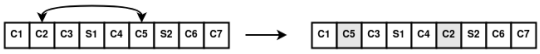

FIGURE 4.4: Illustration of the $\mathcal{N}_4(S)$ neighborhood structure

- $\mathcal{N}_5(S)$: Inter-route customer-based neighborhood that switches the places of two customers between two different routes (Figure 4.5). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

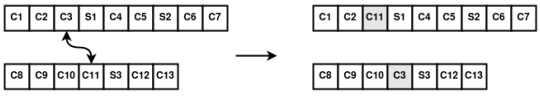

FIGURE 4.5: Illustration of the $\mathcal{N}_5(S)$ neighborhood structure

- $\mathcal{N}_6(S)$: Intra-route AFS-based neighborhood that excludes a refueling station from its current route (Figure 4.6). In the worst-case scenario, this neighborhood has $\mathcal{O}(|F|)$ neighbors.

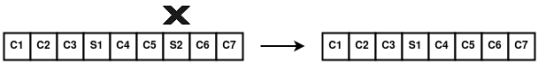

FIGURE 4.6: Illustration of the $\mathcal{N}_6(S)$ neighborhood structure

- $\mathcal{N}_7(S)$: Intra-route AFS-based neighborhood that shifts a refueling station to a different spot within its route (Figure 4.7). In the worst-case scenario, this neighborhood has $\mathcal{O}(|F| * (|V| + |F|))$ neighbors.

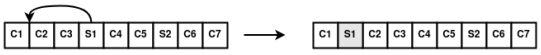

FIGURE 4.7: Illustration of the $\mathcal{N}_7(S)$ neighborhood structure

- $\mathcal{N}_8(S)$: Intra-route AFS-based neighborhood that substitutes one refueling station with another (Figure 4.8). In the worst-case scenario, this neighborhood has $\mathcal{O}(|F|^2)$ neighbors.

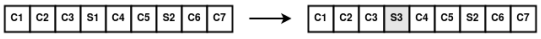

FIGURE 4.8: Illustration of the $\mathcal{N}_8(S)$ neighborhood structure

- $\mathcal{N}_9(S,k)$: Intra-route combined neighborhood that relocates a sequence of k nodes within their current route (Figure 4.9). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

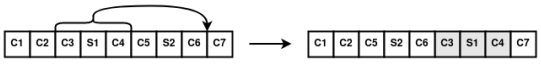

FIGURE 4.9: Illustration of the $\mathcal{N}_9(S)$ neighborhood structure

- $\mathcal{N}_{10}(S,k)$: Inter-route combined neighborhood that moves a group of k nodes to an entirely different route (Figure 4.10). In the worst-case scenario, this neighborhood has $\mathcal{O}(|V|^2)$ neighbors.

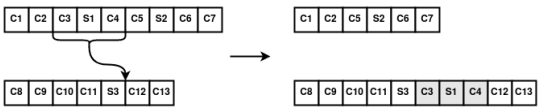

FIGURE 4.10: Illustration of the $\mathcal{N}_{10}(S)$ neighborhood structure

As we can see, many neighborhoods have $\mathcal{O}(V^2)$ neighbors in the worst-case scenario. However, the number of neighbors in inter-route neighborhoods is most often larger than in intra-route neighborhoods. Additionally, evaluating intra-route neighbors is faster than evaluating inter-route neighbors, as we only need to recalculate the schedule for one route instead of two.

4.9 Local search procedure

Many different local search methods can be applied to the VRP, and by extention to EVRP. A few studies that discuss local search methods for VRP can be found in references [40, 100]. While a comparison of these local search methods is not covered in this work, it offers a promising avenue for subsequent research.

For our purposes, we employ the Variable Neighborhood Descent as the local search technique within each metaheuristic that utilizes local search. The main idea behind VND is that a local minimum with respect to one neighborhood structure might not be a local minimum for another. Thus, by systematically changing the neighborhood and diving deep into each one, the algorithm can escape local optima and potentially discover better solutions. The VND procedure begins with an initial solution and a predetermined set of neighborhood structures. It explores the first neighborhood, seeking for improvements using a descent method. If a better solution is found, the search reverts to the first neighborhood and continues. If no improvement is detected within a particular neighborhood, the algorithm transitions to the next neighborhood structure. This process repeats until no improvements can be found in any of the neighborhoods, at which point the algorithm terminates.

The neighborhood structures employed in VND are detailed in Section 4.8. However, we do not make use of all these structures. We excluded the node-based neighborhoods ($\mathcal{N}_9(S)$ and $\mathcal{N}_{10}(S)$) since they often overlap with the neighbors explored by earlier structures, leading to unnecessary redundancy. Such repetitive evaluations can hinder the efficiency of the method. Consequently, our focus is on measuring the impact of the initial eight neighborhoods ($\mathcal{N}_1(S, \varepsilon)$ - $\mathcal{N}_8(S)$) exclusively. To conduct our testing, we set the value of $\varepsilon = 5$. This decision is made using the *iRace*[3] package for the R programming language, and we allocated a budget for 2000 tests. To analyze the gathered data, we employed a *gradient boosting regressor* from the *Extreme Gradient Boosting* (**XGBoost**) library in the Python programming language. More details on this can be found in the official XGBoost documentation[4].

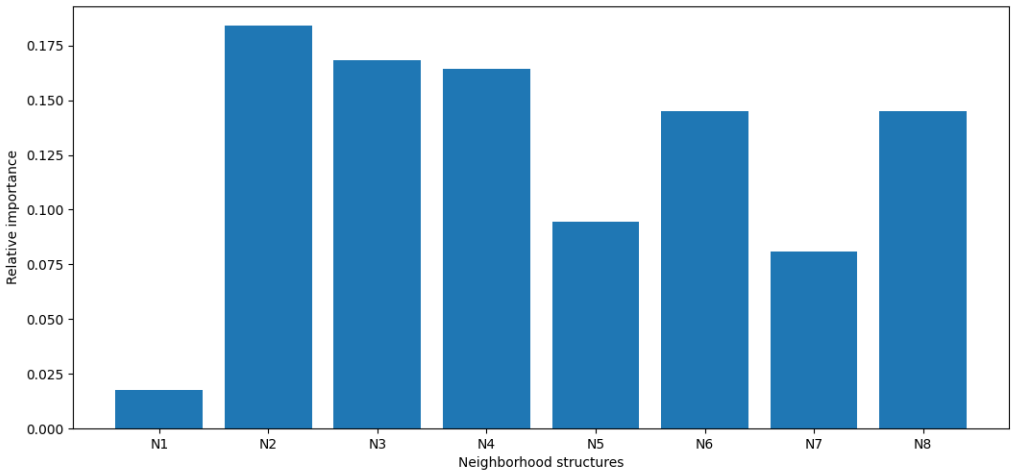

FIGURE 4.11: Relative influences of each neighborhood structure tested.

[3] https://cran.r-project.org/web/packages/irace/index.html
[4] https://xgboost.readthedocs.io

Figure 4.11 illustrates the relative impact of each neighborhood structure incorporated in the local search phase, as calculated by the XGBoost regressor. It is important to recognize that the selection of neighborhood structures is not the sole determinant of VND procedure's efficacy. The sequence in which these structures are applied can noticeably influence the quality of solutions. However, this aspect was not delved into in our study. From Figure 4.11, we observe that the most influential neighborhood structures for the algorithm's performance are $\mathcal{N}_2$ and $\mathcal{N}_3$, which are centered around customers. This prominence was anticipated, particularly for $\mathcal{N}_3$, as it uniquely alters the customer count in a vehicle. While $\mathcal{N}_4$ also focuses on customers and significantly affects solution quality, its relative impact is lesser since its effect can be mirrored by $\mathcal{N}_2$. Actions like omitting a refueling station visit ($\mathcal{N}_6$) or substituting one recharging station with another ($\mathcal{N}_8$) moderately affect solution quality. In contrast, adjusting the recharging station's visitation order ($\mathcal{N}_7$) offers limited influence, likely because it often leads to infeasible solutions. It is interesting that swapping customers between two vehicles ($\mathcal{N}_5$) does not significantly affect solution quality, possibly due to the influence of the $\mathcal{N}_3$ structure. Notably, the $\mathcal{N}_1$ structure provided negligible enhancement to the solution's quality. The likely reason is the efficiency of the scheduling procedure outlined in Algorithm 13, coupled with the optimization approach in Algorithm 14. Given its minimal contribution and high computational demand for larger instances, we opted to exclude this neighborhood from VND entirely.

With all of this in mind, we present the pseudocode for our VND method in Algorithm 15. Considering the fact that proposed neighborhood for our problem are rather complex, we utilized the *first improvement* neighborhood search strategy.

Algorithm 15 Variable neighborhood descent

```
1: procedure VND(Solution S, Set of neighborhoods 𝒩)
2:     i ← 2
3:     while i ≤ 8 do
4:         improved ← LS(𝒩_i, S, S′)      ▷ LS is the local search in neighborhood 𝒩_i
5:         if improved then
6:             S ← S′
7:             i ← 2
8:             continue
9:         else
10:            i ← i + 1
11:        end if
12:    end while
13:    return S
14: end procedure
```

In Algorithm 16, we outline the general structure of our local search procedure. This procedure takes a neighborhood structure as its input parameter, along with the incumbent solution and a structure to store an improved solution if found. The procedure returns *True* if an improvement is found, and *False* otherwise.

The procedure iterates over all neighbors of the solution S within the neighborhood structure $\mathcal{N}_i$. For each neighbor, the schedule is recalculated and optimized before evaluation. It should be noted that in our pseudocode, we call the methods *CALC_SCHEDULE* and *OPTIMIZE_SCHEDULE* on the entire solution for clarity. In practice, only the routes that have undergone changes need their schedules

recalculated. Specifically, for intra-route neighbors, only one route's schedule needs recalculating, while for inter-route neighbors, the schedules for two routes must be recalculated.

Algorithm 16 Local search procedure

```
1: procedure LS(Neighborhood structure N_i, incumbent solution S, improved solution S')
2:     for ∀ neighbours S_n ∈ N_i(S) do
3:         S_n ← CALC_SCHEDULE(S_n)
4:         S_n ← OPTIMIZE_SCHEDULE(S_n)
5:         if f(S_n) < f(S) then
6:             S' ← S_n
7:             return True
8:         end if
9:     end for
10:    return False
11: end procedure
```

4.10 VNS

Certain foundational elements need to be established before we can effectively outline our VNS algorithm. Foremost among these is the *shaking* procedure, a pivotal component that plays a determinative role in how our algorithm functions and evolves.

In the traditional VNS algorithm, the shaking procedure identifies a random neighboring solution that's precisely k units distant from the current best or incumbent solution. However, our adaptation deviates from this convention. Our shaking method seeks out a neighbor that is distanced *at least* k units from the incumbent, not strictly k. This nuanced change, when put to the test, empirically demonstrated improved solution quality. After thorough evaluations and testing, we selected two specific neighborhood structures from those discussed in Section 4.8 ($\mathcal{N}_9$ and $\mathcal{N}_{10}$) to incorporate into our shaking procedure. During each cycle of the algorithm, one of them is stochastically selected and applied to the current solution. Our shaking procedure is laid out in Algorithm 17. Navigating the core loop of the algorithm (from Lines 4 to 12), one first encounters the stochastic selection of a neighborhood structure (Line 5), followed by determining the size of that neighborhood (Line 6). This size is bound by the threshold l_{max}. Subsequently, the incumbent solution undergoes a transformation, leveraging the selected neighborhood type and its size (as seen in Lines 9 and 11).

Algorithm 17 Shaking

```
1: procedure SHAKE(Solution S, k, l_max)
2:     i ← 0
3:     S' ← S
4:     for i = 1 to k do
5:         p ← random integer between 0 and 1
6:         l ← random integer between 1 and l_max
7:         switch (p) do
8:             case 0 :
9:                 S' ← random neighbor from N_9(S', l)
10:            case 1 :
11:                S' ← random neighbor from N_10(S', l)
12:    end for
13:    return S'
14: end procedure
```

In most instances, the shaking procedure generates a feasible solution. However, in the rare event that it yields a solution which the *FIX_SOLUTION* method cannot repair (specifically, when it fails to add a necessary AFS visit to address insufficient battery charge) the solution is discarded, and the shaking procedure is repeated.

Since the main loop of the algorithm executes k times and the minimum alteration to the solution is 1, it is likely that the resulting solution will typically be at least k units away from the incumbent solution. However, there exists a small possibility that the selected transformation may be the inverse of a prior one, effectively nullifying its impact. Given that the size of each transformation has a probability $\frac{l_{max}-1}{l_{max}}$ of being greater than 1, the resulting solution will usually have a distance greater than or equal to k. Nevertheless, because the probability of this not occurring is quite low, we do not specifically verify that the distance meets or exceeds k, instead, we simply accept the new solution.

In Algorithm 18, we outline the structure of our *General Variable Neighborhood Search* (**GVNS**) algorithm. The input parameters encompass the problem instance, a neighborhood set used in VND, along with the standard VNS parameters: k_{min}, k_{step}, and k_{max}. These parameters control the number of transformations in the shaking process. Additionally, we introduce an extra parameter for shaking, represented as l_{max}. As for the termination criterion, we have opted for the maximum allowable CPU time. It should also be noted that the function f assesses the quality of a solution by calculating the value of the objective function.

The procedure starts by constructing an initial solution, using the method illustrated in Algorithm 12 (as seen in Line 2). Keep in mind that the *FIX_SOLUTION* method may not always resolve issues related to battery charge, especially if the solution involves two subsequent locations being so distant that adding a visit to an AFS does not rectify the problem. Additionally, if the solution is infeasible due to exceeding capacity constraints, it cannot be corrected using the *FIX_SOLUTION* method. In such cases, this method can be skipped altogether to save time. If the solution turns out to be infeasible, it undergoes the shaking process until feasibility is attained (Lines 3-6). While this step might seem redundant, considering that VND could potentially pinpoint a feasible solution during its local search, our empirical results revealed enhanced performance using this approach.

After initializing the solution, we calculate the schedule using Algorithm 13 and then optimize it with Algorithm 14. As a reminder, these methods adjust specific

fields within the solution structure, corresponding to departure and arrival times at each location. Our aim then shifts to refining the initial solution, which is achieved by applying VND (Line 9). Within the core loop of the algorithm, we engage the shaking process on the current best solution, resulting in a fresh, perturbed solution. Given that this new solution might not be feasible, the shaking step is reiterated until a feasible solution emerges. Although it is highly unlikely, there is a chance the algorithm may fail to find a feasible solution within the allotted time. Therefore, we check whether the stopping criterion is met after each attempt. Once a feasible solution is found, the schedule is reconfigured, and then VND is applied to the updated solution. Should the VND-processed solution yield a superior objective function value in comparison to the best recorded solution, it is embraced as the new best solution (Line 23), and concurrently, the parameter k is reset to k_{min} (as illustrated in Line 24). Otherwise, k is incremented by the k_{step} value (Line 26).

Algorithm 18 General variable neighborhood search

```
1: procedure GVNS(Instance data D, Set of neighborhoods 𝒩, k_min, k_step, k_max, l_max, vnd_limit)
2:     S ← INIT_SOLUTION(D)
3:     while S is infeasible do
4:         S ← SHAKE(S, 1, l_max)
5:         S ← FIX_SOLUTION(S)
6:     end while
7:     S ← CALC_SCHEDULE(S)
8:     S ← OPTIMIZE_SCHEDULE(S)
9:     S ← VND(S, 𝒩, vnd_limit)
10:    while the stopping criterion is not met do
11:        k ← k_min
12:        while k ≤ k_max ∧ the stopping criterion is not met do
13:            S' ← SHAKE(S, k, l_max)
14:            S' ← FIX_SOLUTION(S')
15:            while S' is infeasible and the stopping criterion is not met do
16:                S' ← SHAKE(S, k, l_max)
17:                S' ← FIX_SOLUTION(S')
18:            end while
19:            S' ← CALC_SCHEDULE(S')
20:            S' ← OPTIMIZE_SCHEDULE(S)
21:            S'' ← VND(S', 𝒩, vnd_limit)
22:            if f(S'') < f(S) then
23:                S ← S''
24:                k ← k_min
25:            else
26:                k ← k + k_step
27:            end if
28:        end while
29:    end while
30: end procedure
```

4.10.1 Hyperparameter analysis

Our GVNS algorithm has five hyperparameters:

- k_{min} - This hyperparameter controls the minimum number of steps involved in the shaking procedure.
- k_{step} - This refers to the hyperparameter that controls how much the shaking procedure's steps increase in each iteration.
- k_{max} - This hyperparameter controls the maximum number of steps involved in the shaking procedure.
- l_{max} - This hyperparameter controls the maximum step size during each phase of the shaking process.
- vnd_{limit} - Hyperparameter manages the time constraint for each invocation of the VND procedure, measured in seconds. Upon reaching this threshold, the VND process concludes, yielding the best solution discovered thus far. This parameter is particularly crucial for larger instances where VND may require extensive time to finalize, as it incrementally refines the current solution. This gradual improvement process risks the algorithm becoming entrenched in a local optimum.

To accurately assess the relative significance of our hyperparameters, we initially employ a *Random Forest regressor*, which is well-suited for capturing complex relationships in the data. This regressor uses hyperparameter values as inputs and the observed solution quality as output. Following this, we undertake a sensitivity analysis via *permutation feature importance*. This method quantifies the importance of a feature by measuring the reduction in the model's performance when the values of that feature are randomly permuted [41].

Figure 4.12 illustrates the relative impact of each hyperparameter, measured using the *mean decrease of impurity* metric [41]. This metric is a staple in tree-based machine learning models such as decision trees, random forests, and gradient boosting machines, and it is instrumental in assessing the significance of individual features within the model. The figure clearly highlights that hyperparameters k_{min} and k_{max}, along with l_{max}, are particularly influential. The prominence of l_{max} is attributed to its role in defining the magnitude of each step in the shaking process, thereby jointly determining the neighborhood size with k_{min} and k_{max}. Conversely, the vnd_{limit} hyperparameter exhibits minimal impact on the solution quality. This aligns with expectations, especially for smaller instances where the VND tends to complete rapidly without ever reaching the time threshold set by vnd_{limit}.

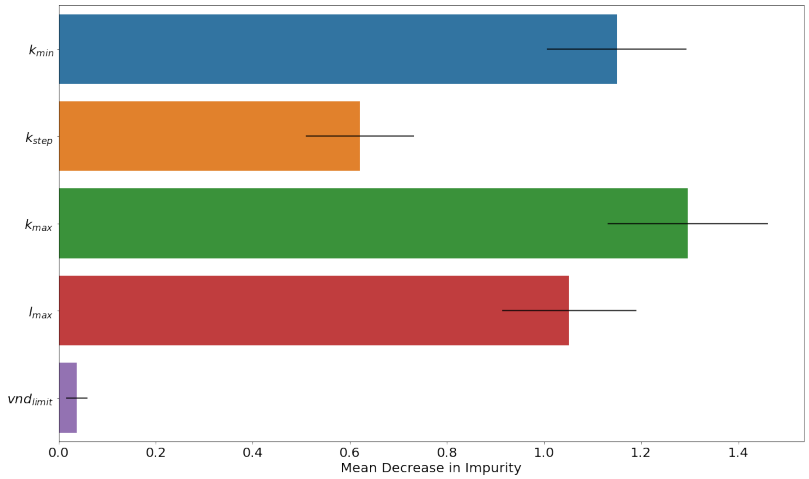

FIGURE 4.12: Relative importance of GVNS hyperparameters for dataset I_1

While understanding the relative importance of hyperparameters provides valuable insights into which ones warrant more careful tuning, this alone does not reveal the nature of their correlation with the achieved solution quality. To delve deeper into this relationship, we used *Partial Dependence Plots* (**PDP**s) for the instance *r202C15* from dataset I_1, as seen in Figure 4.13. PDPs show the connection between the target function (solution quality, in our context) and selected *target* features (hyperparameters), while accounting for the effects of other features. Essentially, PDPs clarify whether the relationship between the target and a feature is linear, monotonic, or of a more intricate nature.

The insights from Figure 4.13 complement our findings from Figure 4.12. The more pronounced variations in solution quality within these plots suggest that the corresponding hyperparameter exerts a significant impact on the solution's quality. Moreover, these plots offer a general understanding of **how** these hyperparameters influence the outcome. For instance, it is observable that excessively increasing k_{min} could negatively affect solution quality, whereas setting the k_{max} hyperparameter around 15 appears to yield optimal performance. However, it is crucial to interpret these findings cautiously, as they represent optimal hyperparameter values for a single instance rather than for the entire dataset. Hence, a more comprehensive analysis is necessary to identify the best hyperparameter settings across all dataset instances.

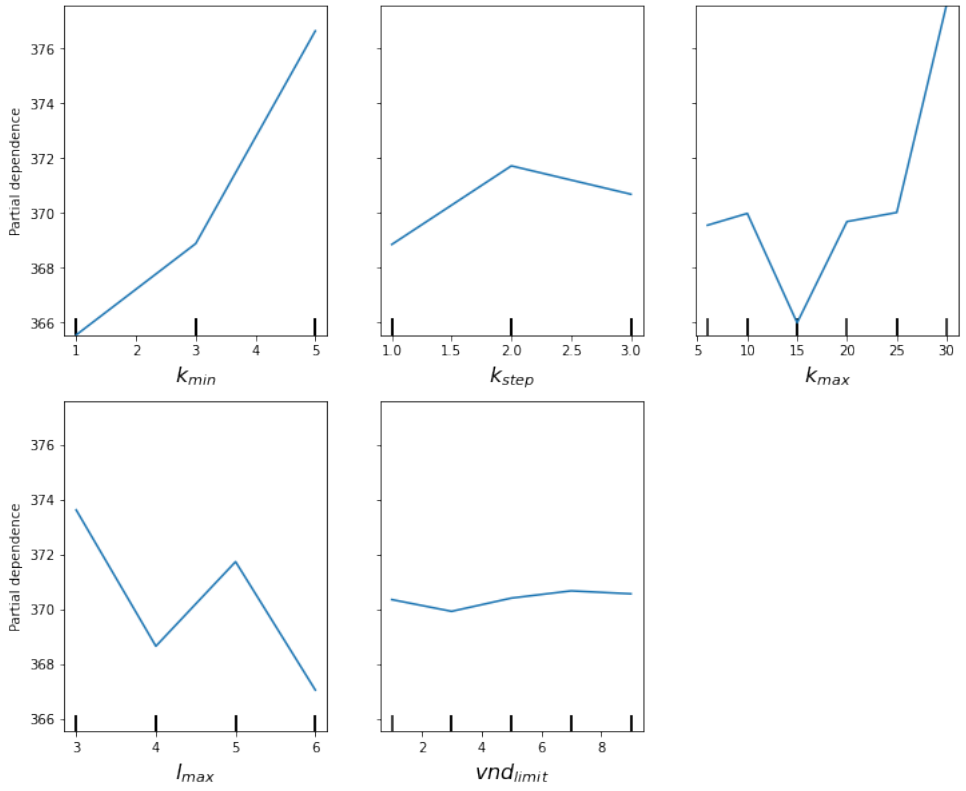

FIGURE 4.13: Impact of GVNS hyperparameters on the performance on instance *r202C15*

In Figure 4.14, we showcase a similar analysis for the dataset I_2 with larger instances. This figure reveals a stark contrast in the relative impact of hyperparameters compared to the smaller instances. Most notably, the vnd_{limit} hyperparameter emerges as the most influential for I_2, a significant shift from its minimal impact observed in the I_1 dataset. This change can be attributed to the extended duration that VND often requires for larger instances, making the execution time limit a critical factor.

Additional exploration is provided in Figure 4.15, where we present PDPs for instance *c101_21* from the I_2 dataset. These plots distinctly illustrate that extending the VND time limit enhances solution quality, as evidenced by a reduction in the objective function. Additionally, the plots indicate a detrimental effect on solution quality with an increase in the k_{min} hyperparameter, underlining its nuanced influence in the context of larger instances.

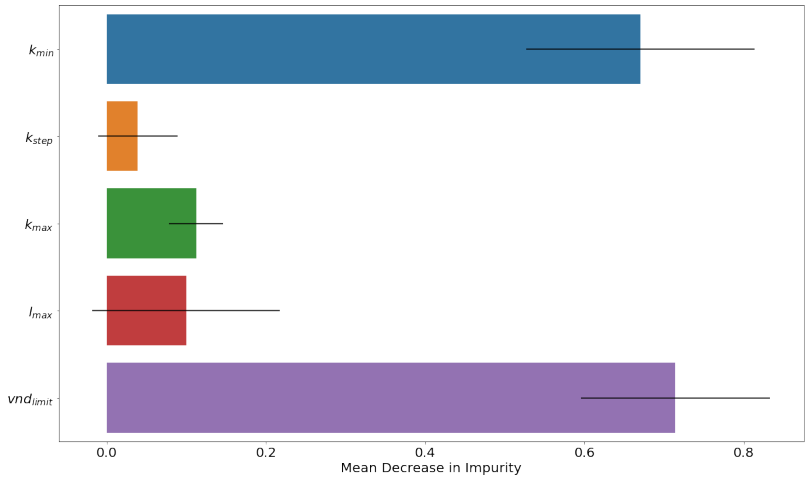

FIGURE 4.14: Relative importance of GVNS hyperparameters for dataset I_2

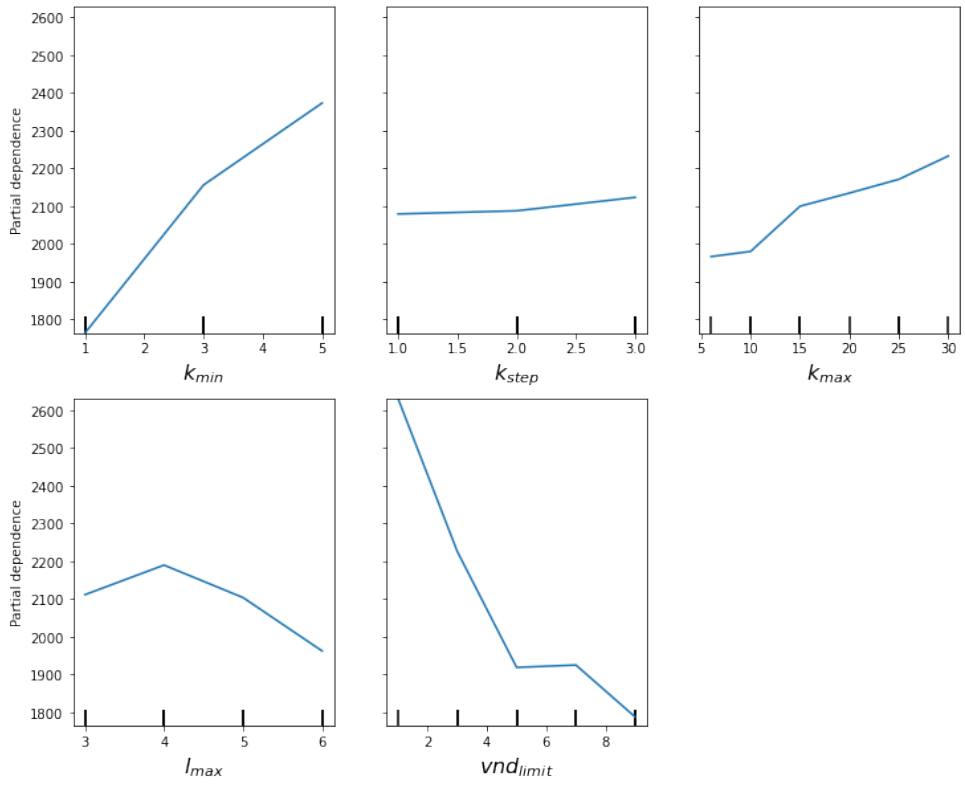

FIGURE 4.15: Impact of GVNS hyperparameters on the performance on instance *c101_21*

4.11 GRASP

Our GRASP algorithm follows the straightforward steps presented in Algorithm 2. The primary component we need to clarify here is our greedy randomized construction procedure, which is detailed in Algorithm 19.

To begin with, our method takes in a set of customer locations, marked as C, along with the *RCL_LIMIT* parameter. This parameter plays a crucial role in the stochastic nature of the method by dictating the upper limit of top candidates that we evaluate at each stage.

The process begins by creating an empty solution, designated as S (Line 2). A new array is generated during every iteration (Line 5), which stores the costs of integrating each unassigned customer to potential route positions, expressed as ordered triplets. The route positions are essentially ordered pairs consisting of a route index and a position within the route. Once these costs are fully computed, the array is sorted in ascending order based on cost (Line 12). To further clarify, we evaluate all unassigned customers against every position within the solution, not merely the positions at the end of each route.

Subsequently, we determine the size of the *Restricted Candidate List* (**RCL**), which is the smaller of the values *RCL_LIMIT* and the size of the *cost_array* (Line 14). Then, the initial *RCL_SIZE* elements are extracted to shape the RCL (Line 14). A random selection is made from this RCL (Line 15) and the result is stored in the structure *chosen*, which is an ordered triplet consisting of {*customer, position, cost*}. The chosen customer is then incorporated into the partial solution (Line 17). This customer is then categorized as *assigned* and removed from the set of unassigned customers (Line 18).

This iterative process continues until every customer finds its placement. Finally, the solution is fixed by including visits to AFSs, as detailed in Algorithm 10 (Line 20).

Algorithm 19 Greedy Randomized Construction procedure

```
1:  procedure GREEDY_RANDOMIZED_CONSTRUCTION(V, RCL_LIMIT)
2:      S ← create an empty solution
3:      V_temp ← V
4:      while V_temp ≠ ∅ do
5:          cost_array ← empty array
6:          for ∀customer ∈ V_temp do
7:              for ∀position ∈ S do
8:                  cost ← cost of adding customer at position in S
9:                  Add {customer, position, cost} to the cost_array
10:             end for
11:         end for
12:         Sort cost_array by cost ascending
13:         RCL_SIZE ← min(RCL_LIMIT, size of cost_array)
14:         RCL ← first RCL_SIZE elements from cost_array
15:         r ← random(1, RCL_SIZE)
16:         chosen ← get element from RCL at position r
17:         Add chosen.customer to chosen.position in S
18:         V_temp \ chosen.customer
19:     end while
20:     S ← FIX_SOLUTION(S)
21:     return S
22: end procedure
```

4.11.1 Hyperparameter analysis

Our GRASP algorithm has two hyperparameters:

- rcl_{limit} - Hyperparameter controlling the size of the RCL list.
- vnd_{limit} - Hyperparameter managing the time constraint for each invocation of the VND procedure, measured in seconds.

Leveraging the Random Forest regressor alongside permutation feature importance, we have determined the relative impact of GRASP hyperparameters, the details of which are depicted in Figure 4.16. Mirroring our observations with GVNS, the hyperparameter vnd_{limit} showed negligible effect on the algorithm's performance. This phenomenon is further illustrated in Figure 4.17, where we present Partial Dependence Plots (PDPs) for the instance *r202C15*, offering a visual confirmation of the minimal influence of vnd_{limit}. On the other hand, RCL_{limit} had a significantly greater influence, with the most favorable values for this parameter ranging between 3 and 7.

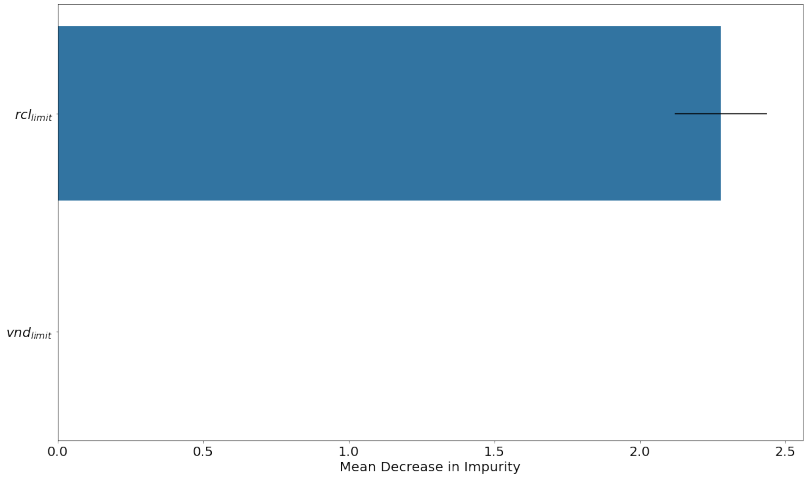

FIGURE 4.16: Relative importance of GRASP hyperparameters for dataset I_1

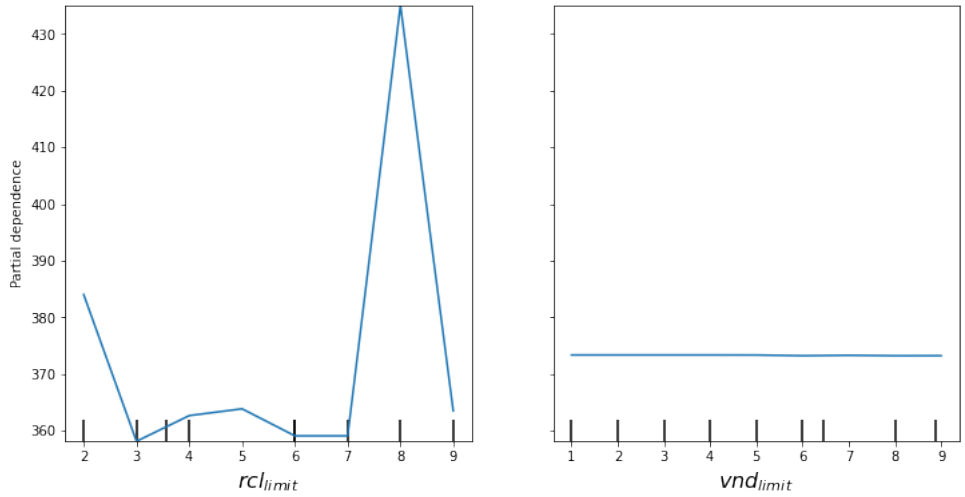

FIGURE 4.17: Impact of GRASP hyperparameters on the performance on instance *r202C15*

Likewise, Figure 4.18 demonstrates the relative impact of the two GRASP hyperparameters as tested on instances from dataset I_2. Interestingly, both hyperparameters exerted roughly the same influence on the quality of the solution. Complementing this, Figure 4.19 offers PDPs for the instance *c101_21*, providing a deeper insight into their effects. From this figure, it is evident that for the *c101_21* instance, the method preferred lower values for the RCL_{limit} hyperparameter, and that increasing the VND_{limit} resulted in higher quality solutions.

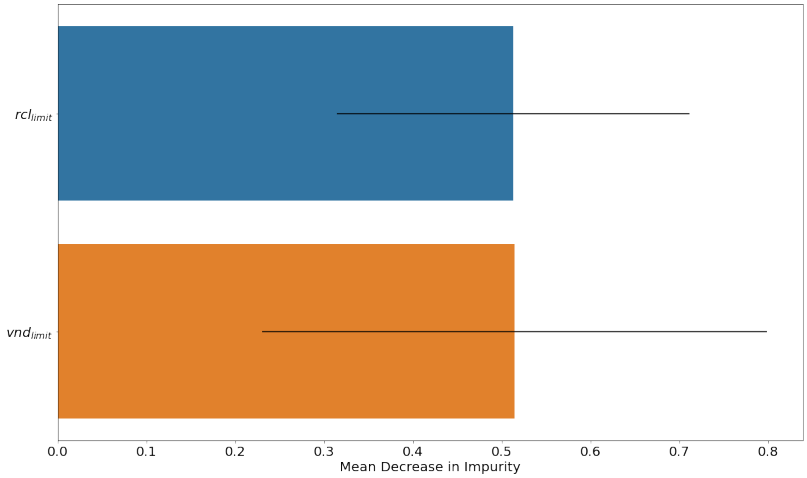

FIGURE 4.18: Relative importance of GRASP hyperparameters for dataset I_2

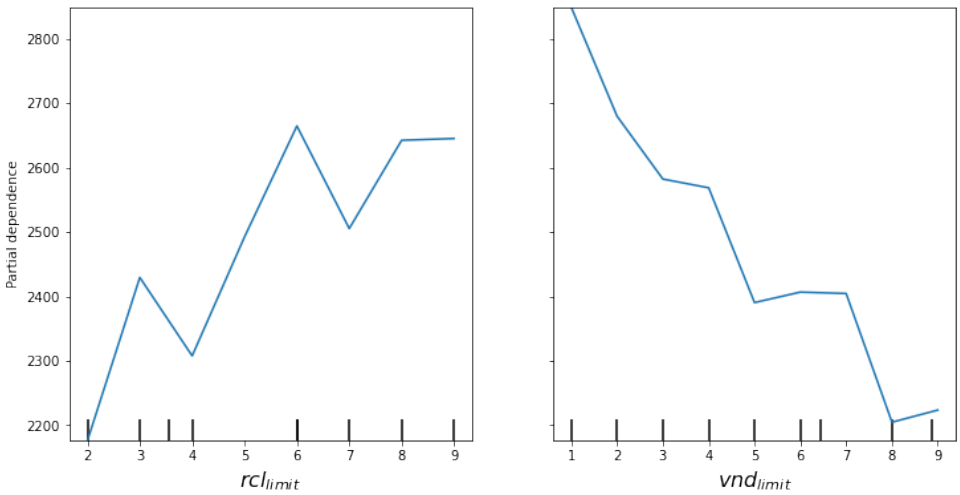

FIGURE 4.19: Impact of GRASP hyperparameters on the performance on instance *c101_21*

4.12 ACO

We have designed our ACO algorithm to closely adhere to the basic algorithm that is outlined in Algorithm 4.

We commence by setting the pheromone values, drawing upon the initial solution produced by Algorithm 12. Once the crafted solution undergoes evaluation, we

calibrate the initial pheromone levels to be the reciprocal of $f(S_{init})$, where S_{init} represents the generated solution. This methodology takes its cue from the strategy outlined in [154], with its accompanying pseudocode clarified in Algorithm 20.

We employ the *offline pheromone update* mechanism to reinforce the pheromones. Specifically, only the best solution in the current generation is used to strengthen the pheromone trail. If an arc is incorporated within the best solution, then its pheromone trail is augmented by $\frac{1}{f(S_{best})}$, where S_{best} represents the best solution. For heuristic insights, we have used the distance metrics between locations.

Algorithm 20 Procedure for initializing pheromone values

```
1: procedure INIT_PHEROMONES(Instance data D)
2:     S ← INIT_SOLUTION(D)
3:     val ← 1/f(S)
4:     Set all pheromone values to val
5: end procedure
```

The method presented in Algorithm 21 is used to systematically construct a solution for each ant. This method is employed in Line 7 of Algorithm 4, with additional parameters specific to the problem, primarily the set of customers V and the set of AFSs F. The process begins by initializing an empty solution S (Line 2). For each iteration, the algorithm first identifies all available positions in the current solution S (Line 5). It is important to note that only positions at the end of each route are considered. The rationale behind this approach is that earlier positions have already been evaluated when adding previous elements to the solution. Although the stochastic nature of this method means that checking previous positions would not be redundant, doing so would significantly slow down the creation of solutions by increasing the number of neighbors to be checked. Additionally, it would increase the likelihood of the method generating new solutions in a deterministic, greedy manner, as elements with a higher probability of being placed in certain positions would have multiple opportunities to be placed there. This contrasts with our approach in the GRASP construction phase (Algorithm 19), where we achieved better results by evaluating all unassigned customers against every position in the route, not just at the ends of the routes. With the aforementioned positions in hand, the next objective is to determine all feasible moves (Line 6), which are essentially combinations of adding specific customers to particular positions. To help in this, the algorithm employs the GET_MOVES function, which uses the calculated list of available positions, the unassigned customers from set C, and the current state of the solution S, to return a set of ordered pairs associated with potential moves, where the first element in each pair represents a position in the solution, and the second represents the node (customer or AFS) that could be inserted at that position. Once the potential moves are identified, the algorithm calculates the probability for each move using a specified formula (Line 10), ensuring that the solution construction is not deterministic and has a degree of randomness. This stochastic nature of move selection allows for diverse solution exploration. Based on these probabilities, a move is selected (Line 11), and the associated node is added to the solution at the designated position. If the inserted node represents a customer, that customer is then marked as assigned, removing them from further consideration in set C (Line 12). The loop continues until all customers are assigned. Upon completion, the fully constructed solution S is returned, marking the successful creation of a solution for the ant.

Procedure for finding a set of potential moves based on a set of available positions in the solution is presented in Algorithm 22. To start, the algorithm creates an empty

Algorithm 21 Procedure for constructing solutions

```
1: procedure CONSTRUCT_SOLUTION(Customers V, AFSs F, α, β, τ)
2:     S ← empty solution
3:     V_temp ← V
4:     while V_temp ≠ ∅ do
5:         positions ← get all available positions in S
6:         list_of_moves ← GET_MOVES(positions, V_temp, S, F)
7:         if list_of_moves is empty then
8:             continue from line 2
9:         end if
10:        Calculate probabilities for each move using equation the (2.2)
11:        move ← choose one move based on calculated probabilities
12:        Perform the move by adding move.node at move.position to S
13:        if move.noderepresentsacustomer then
14:            V_temp \ move.node
15:        end if
16:    end while
17:    return S
18: end procedure
```

set called *moves* intended to hold pairs of positions and nodes (customers or stations) (Line 2). Given a set of positions, a set of available customers, and the set of AFSs, the main loop begins by iterating over each position, denoted as *pos*. Within this loop, another nested loop assesses every available customer, denoted as *a*.

Two primary constraints guide the addition of customers to the solution:

- The solution should not exceed the given capacity. If adding customer *a* at position *pos* would breach this capacity constraint, the algorithm continues to the next available customer without adding this one (Line 6).
- After visiting a customer, there must be a station reachable with the remaining battery level (keep in mind that the depot is also considered an AFS). If no station can be accessed after visiting customer *a*, then this customer is not considered for this position (Line 9). This constraint significantly reduces the likelihood of the solution becoming infeasible due to a lack of battery charge, as a vehicle can always reach the next AFS. In the unlikely event that an infeasible solution is generated, we can simply discard the partial solution and start over from the beginning (Line 8, Algorithm 21).

Whenever a move is deemed feasible by meeting both constraints, the algorithm pairs the current position *pos* with customer *a* and adds this pair to the *moves* set (Line 11). If, after considering all available customers for a specific position, no customers were added to *moves*, the algorithm then identifies all reachable stations from the current position (Lines 13-18). Each of these stations is paired with the current position *pos* and added to the *moves* set. This ensures that if no feasible customer can be visited from a position, the algorithm considers possible stations. By the end of this process, the *moves* set is populated with all valid moves the ant can make from the provided positions, ensuring a comprehensive set of possibilities for solution construction.

Beyond the basic ACO algorithm, we also developed *Ant Colony Optimization with Local Search* (**ACOLS**). This enhanced version invokes the VND procedure on

Algorithm 22 Procedure for finding a set of potential moves based on a set of available positions in the solution

```
1: procedure GET_MOVES(positions, available, S, F)
2:     moves ← empty set o pairs
3:     for ∀pos ∈ positions do
4:         for ∀a ∈ available do
5:             if adding a to S as position pos violates capacity constraint then
6:                 continue
7:             end if
8:             if no station can be reached after visiting a then
9:                 continue
10:            end if
11:            moves ← moves ∪ {(pos, a)}
12:        end for
13:        if no customers have been added to moves for pos then
14:            stations ← find stations from F that can be visited from pos
15:            for s ∈ stations do
16:                moves ← moves ∪ {(pos, s)}
17:            end for
18:        end if
19:    end for
20:    return moves
21: end procedure
```

each ant's solution following the completion of the construction phase. Although applying VND to every solution may appear computationally expensive, this issue is mitigated by the vnd_{limit} hyperparameter, which controls the amount of time spent on each solution. While it might be a viable strategy to apply VND to only a subset of solutions, or even just a single solution, this approach was not explored in this thesis and warrants further investigation.

4.12.1 Hyperparameter analysis

The basic version of the our ACO algorithm is characterized by four key hyperparameters:

- *population size* - The number of artificial ants employed in each iteration by the ACO algorithm.
- α - The hyperparameter governing the relative influence of pheromone values during the construction phase, as outlined in Section 2.4.3.
- β - The hyperparameter governing the relative influence of heuristic values during the construction phase, as outlined in Section 2.4.3.
- ρ - The hyperparameter responsible for regulating the evaporation rate in the ACO algorithm.

In addition to these four, ACOLS features an extra hyperparameter:

- vnd_{limit} - Hyperparameter managing the time constraint for each invocation of the VND procedure, measured in seconds.

Employing the Random Forest regressor in conjunction with permutation feature importance, we estimated the relative significance of hyperparameters for both ACO and ACOLS.

Figure 4.20 details the relative impact of each hyperparameter, while Figure 4.21 showcases the PDPs for the instance *r202C15*. These figures reveal that the most significant positive influence on the quality of the solutions was exerted by the population size. The evaporation rate also had a considerable impact, though not uniformly positive. This outcome for the evaporation rate aligns with our expectations, given that we tested a much broader range of values than typically used in ACO algorithms. Specifically, we explored values for ρ up to 0.9, which essentially reduced our ACO algorithm to being marginally better than a random search, as the majority of the pheromone values would be lost after each iteration.

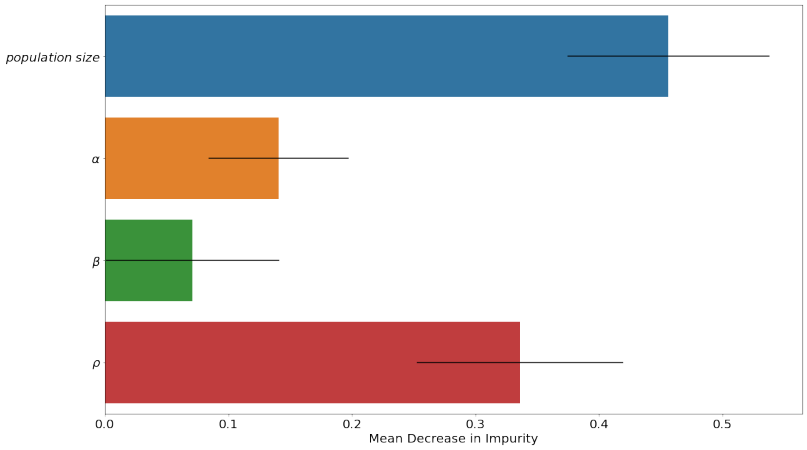

FIGURE 4.20: Relative importance of ACO hyperparameters for dataset I_1

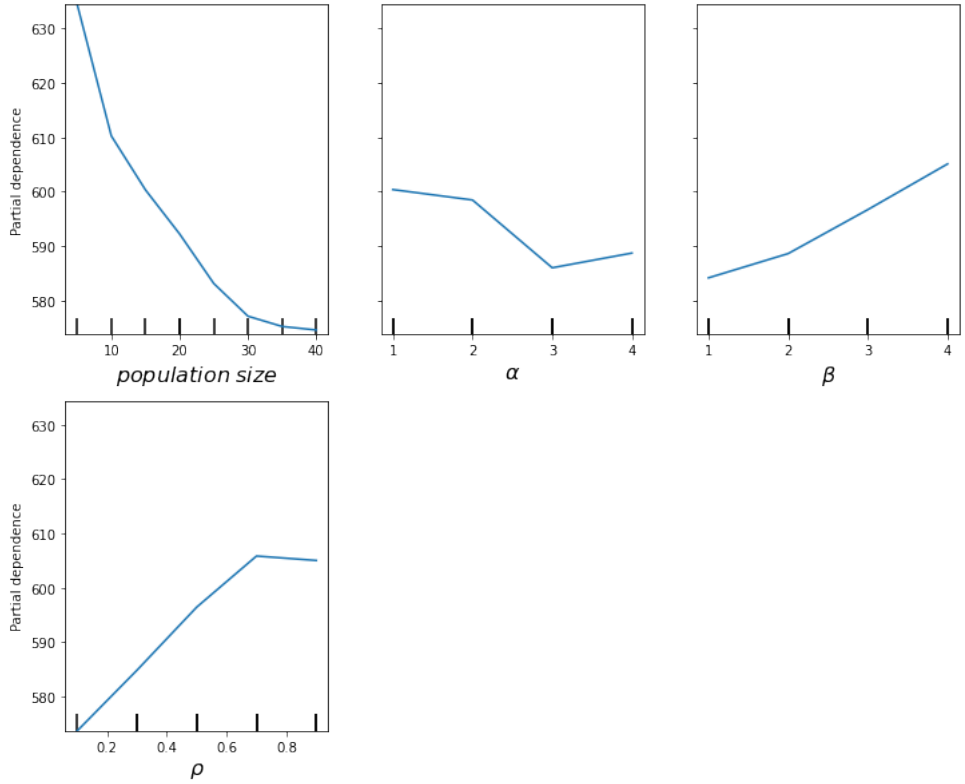

FIGURE 4.21: Impact of ACO hyperparameters on the performance on instance *r202C15*

In the case of ACOLS, Figure 4.22 illustrates the relative influence of hyperparameters for the I_1 dataset, while Figure 4.23 displays the PDPs for the instance *r202C15*. Mirroring the findings from ACO, the ρ hyperparameter was found to have a detrimental effect on the quality of the solutions obtained. The population size remained a critical factor, however, as indicated in Figure 4.23, there is a point beyond which increasing the number of ants ceases to be advantageous. This is likely due to the time consumption during the local search phase performed by each ant, especially when exploring less promising solution candidates. Notably, the hyperparameters α and β exhibited a significantly greater impact in ACOLS compared to their influence in the ACO algorithm.

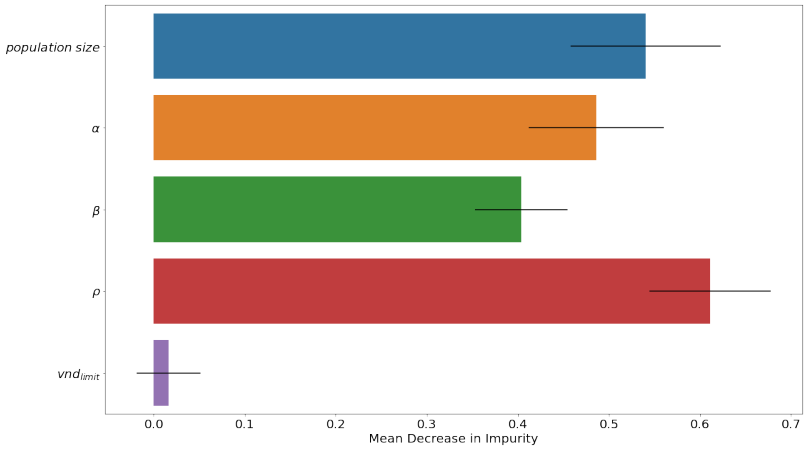

FIGURE 4.22: Relative importance of ACOLS hyperparameters for dataset I_1

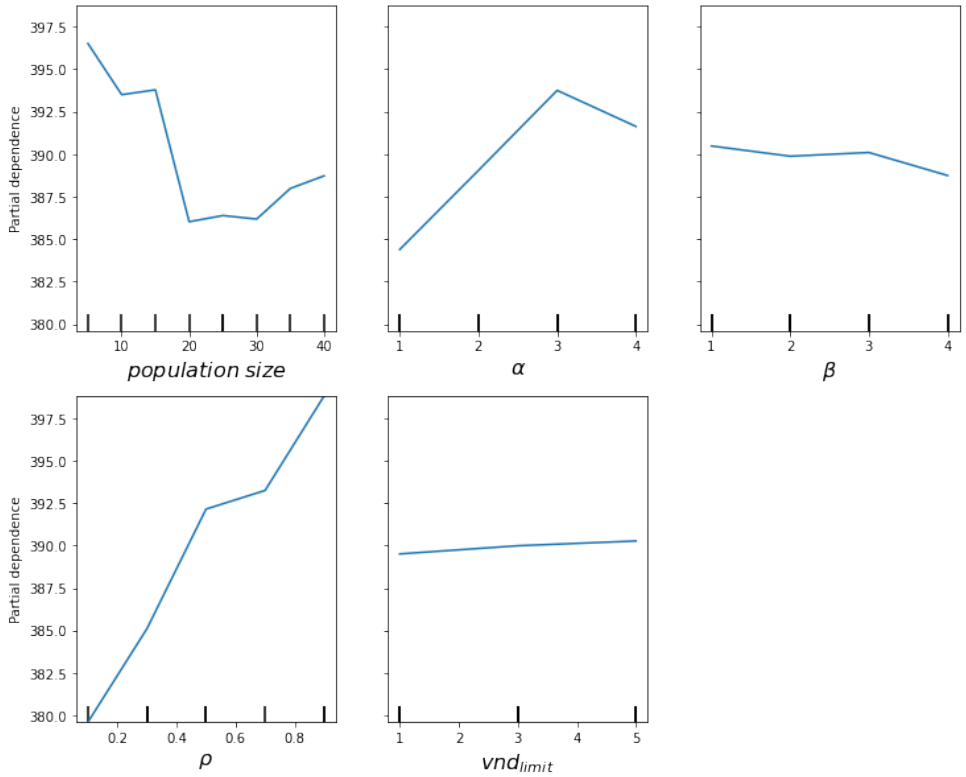

FIGURE 4.23: Impact of ACOLS hyperparameters on the performance on instance *r202C15*

This analysis was replicated for instances from the I_2 dataset for both ACO and ACOLS. For the ACO algorithm, Figure 4.24 delineates the relative impact of each hyperparameter, and Figure 4.25 presents the PDPs for the instance *c101_21*. In this scenario, the α hyperparameter emerged as the most positively influential, indicating that pheromone values play a critical role in larger instances. Notably, the population size had a reduced impact in comparison to the smaller instances.

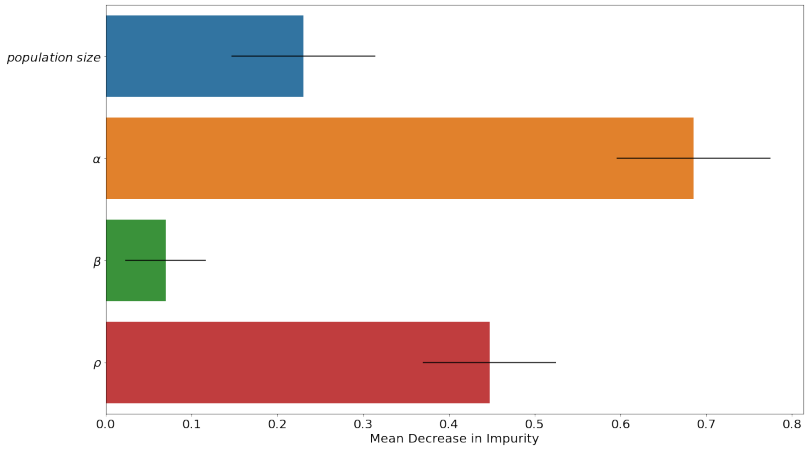

FIGURE 4.24: Relative importance of ACO hyperparameters for dataset I_2

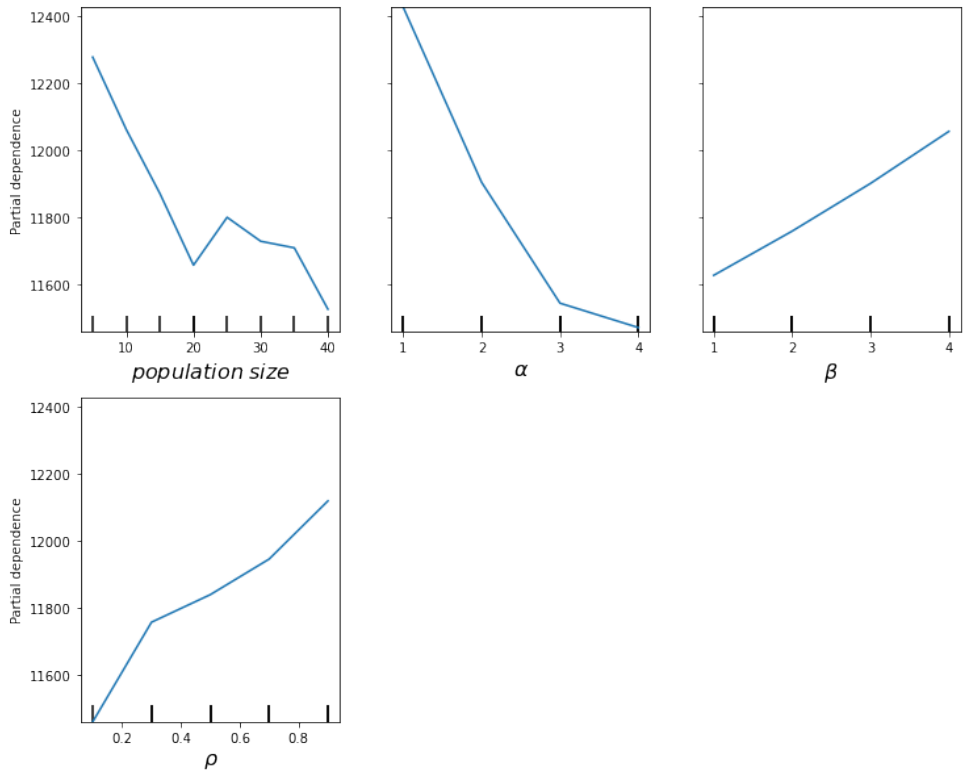

FIGURE 4.25: Impact of ACO hyperparameters on the performance on instance *c101_21*

Turning to the ACOLS algorithm, Figure 4.26 illustrates the relative influence of its hyperparameters on the instance set I_2, with Figure 4.27 offering the PDPs for instance *c101_21*. Here, all hyperparameters demonstrated significant influence. The β hyperparameter, in particular, showed a decline in solution quality with increasing values, suggesting that heuristic information might be less pivotal in navigating the search for larger instances. An interesting observation was that the algorithm achieved

optimal performance at $\rho = 0.5$ for the *c101_21* instance, highlighting a unique characteristic in the behavior of ACOLS in this context.

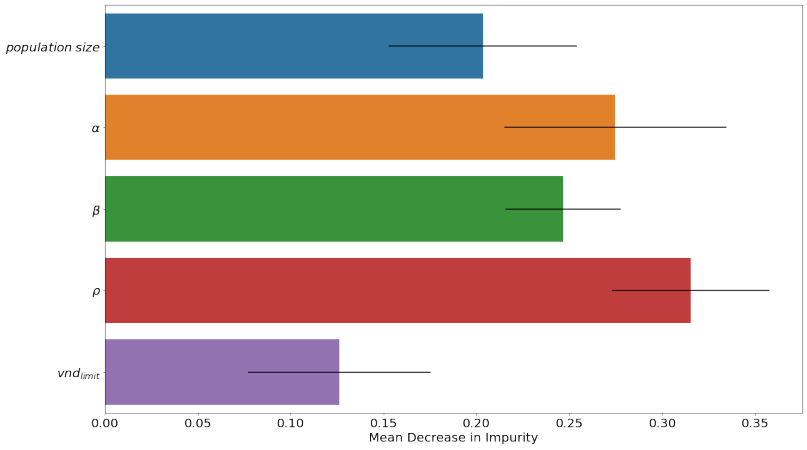

FIGURE 4.26: Relative importance of ACOLS hyperparameters for dataset I_2

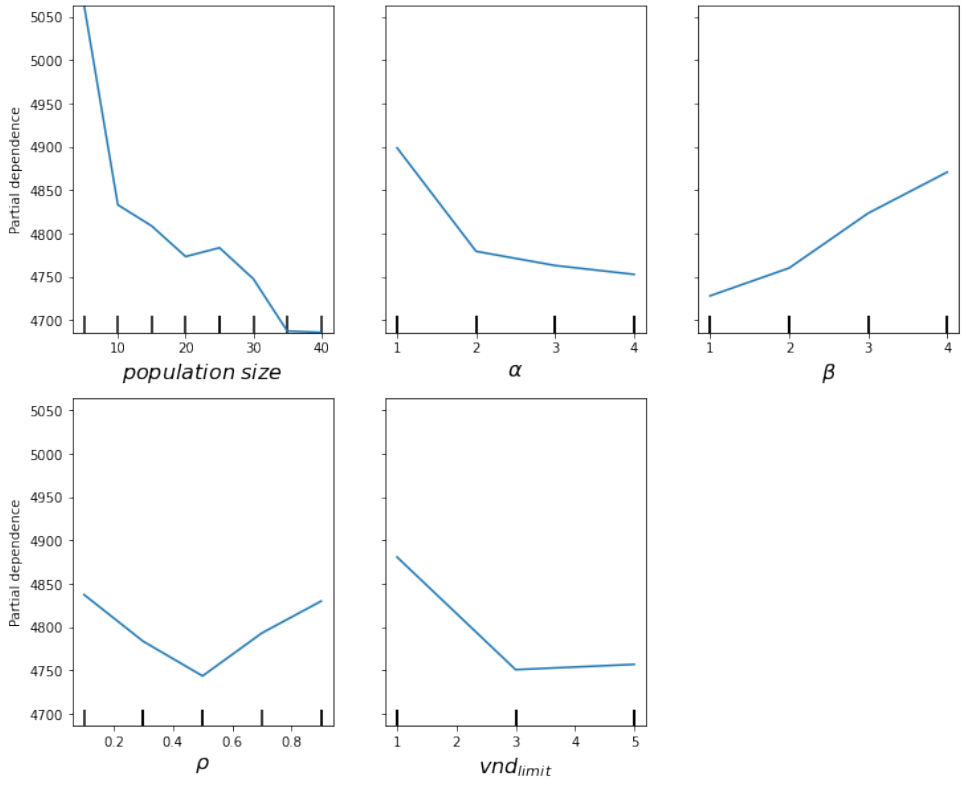

FIGURE 4.27: Impact of ACOLS hyperparameters on the performance on instance *c101_21*

4.13 BCO

For our purposes, we have opted to use the improving version of BCO (BCOi) described in Section 2.4.4. BCO has seen limited application in the field of VRP, with

only a few publications, specifically [205, 214, 253], addressing its use. To the best of our knowledge, BCOi has not yet been applied to EVRP. Our implementation closely follows the steps outlined in Algorithm 5. There are a few implementation details that are specific to our problem.

The method starts by creating the initial population of bees, using the method described in Algorithm 11. Following this initialization, the algorithm's primary loop starts, successively executing forward and backward passes.

In the forward pass, we call the procedure that transforms the current solution by applying a predefined number of changes to it. There are several possible types of transformational moves which the method can utilize to change the current solution, most of them corresponding to the neighborhood structures outlined in Section 4.8. The specific transformations available are:

- $M_1(S)$ - Select a single customer at random and relocate them to a different position within the same route (corresponds to $\mathcal{N}_2(S)$).
- $M_2(S)$ - Select a single customer at random and relocate them to a different route, placing them at a random position (corresponds to $\mathcal{N}_3(S)$).
- $M_3(S)$ - Select two customers at random in the same route and swap their positions (corresponds to $\mathcal{N}_4(S)$)
- $M_4(S)$ - Select two customers from different routes at random and swap their positions (corresponds to $\mathcal{N}_5(S)$)
- $M_5(S)$ - Randomly choose a customer and transfer it to an empty route, provided the maximum number of routes has not been exceeded.

Algorithm 23 presents the method to transform the given solution. To begin with, the algorithm runs a loop for a specified number of iterations (*num_moves*). In each iteration, it selects a transformation function randomly from the list of provided moves and applies it to S (Lines 3-5). After applying all the transformations, the algorithm adds the visits to AFSs to the solution. If the solution remains infeasible after this step, the algorithm undoes all the transformations and retraces the transformational moves, essentially repeating the process (Lines 8-15). To avoid the algorithm getting stuck in an infinite loop, we monitor the number of times we revert the transformed solution back to its original state, tracking this count in the variable *tries*. If *tries* reaches a predefined limit, the method will return the original, unchanged solution (Line 12). In our implementation, this limit, *TRIES_LIMIT*, is set to 100. If a feasible solution is achieved, the algorithm attempts additional refinement by removing any redundant visits to AFSs (Line 16). It then calculates a schedule for S using the *CALC_SCHEDULE* function (Algorithm 13) and optimizes it using the *OPTIMIZE_SCHEDULE* function (Algorithm 14). It is important to distinguish between the *num_moves* parameter in this algorithm and the *max_moves* parameter referenced in Algorithm 5 (See Line 7). While *num_moves* sets the extent of a single transformation, *max_moves* defines the total number of such transformations to be carried out within each iteration.

During the backward phase, each bee has to decide whether it wants to stick to its current solution or opt for a solution offered by another bee. This decision is based on Equation (2.4), while the process of recruiting new solutions follows Equation (2.5).

Algorithm 23 Procedure for transforming the current solution

```
1: procedure TRANSFORM_SOLUTION(S, num_moves, moves)
2:     tries ← 1
3:     for i = 1 to num_moves do
4:         mode ← random integer between 1 and size(moves)
5:         S ← moves[mode](S)
6:     end for
7:     FIX_SOLUTION(S)
8:     if S is not feasible then                    ▷ Solution could not be fixed
9:         Revert all changes to S
10:        tries ← tries + 1
11:        if tries ≥ TRIES_LIMIT then
12:            return S
13:        end if
14:        Continue from line 3
15:    end if
16:    Remove unnecessary visits to AFSs in S
17:    CALC_SCHEDULE(S)
18:    OPTIMIZE_SCHEDULE(S)
19:    return S
20: end procedure
```

4.13.1 Hyperparameter analysis

Our BCO algorithm has two hyperparameters:

- *population size* - The number of artificial bees employed in each iteration by the BCO algorithm.
- *max_moves* - The maximum number of transformational moves in the forward pass of the algorithm.

Figure 4.28 clarifies the relative influence of each hyperparameter, complemented by Figure 4.29 which displays the PDPs for the instance *r202C15*. These figures highlight the crucial importance of selecting the optimal number of moves over determining the precise population size for this algorithm, the conclusion also confirmed by Maksimović and Davidović [210]. It is interesting to note that Figure 4.29 indicates that lower values for both hyperparameters are more favorable compared to higher ones. A similar trend was observed for instances from dataset I_2, as depicted in Figure 4.30 and Figure 4.31. However, for the instance *c101_21*, a somewhat larger population size proved to be more preferable.

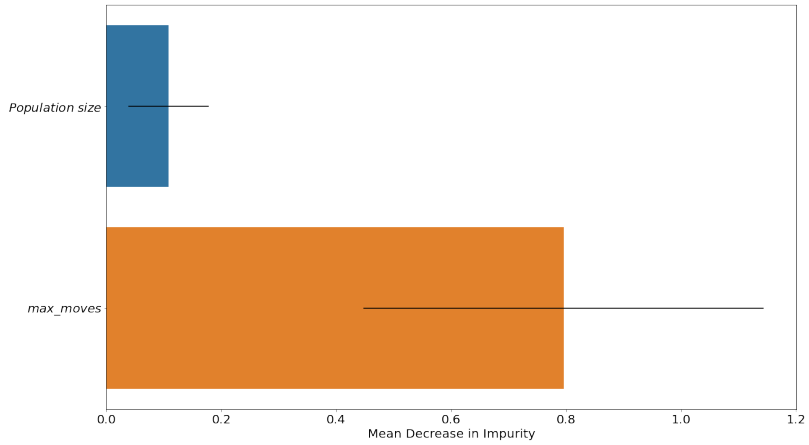

FIGURE 4.28: Relative importance of BCO hyperparameters for dataset I_1

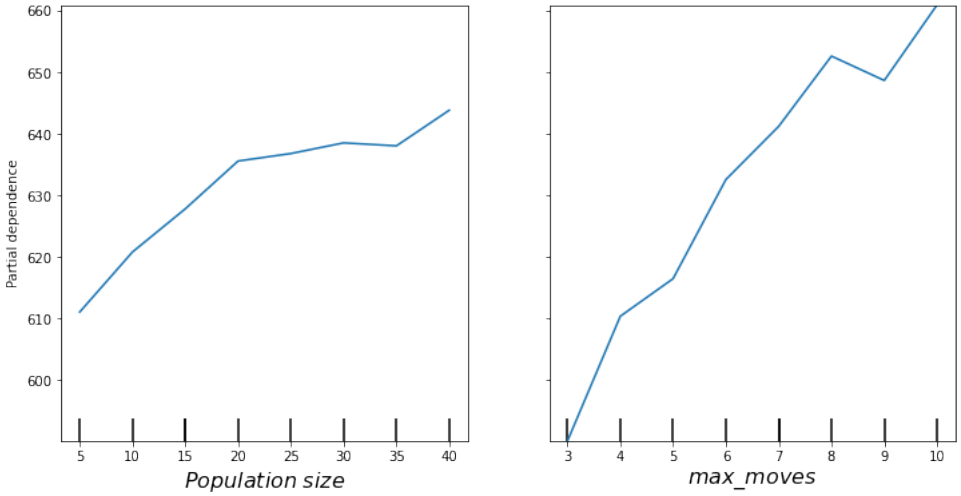

FIGURE 4.29: Impact of BCO hyperparameters on the performance on instance *r202C15*

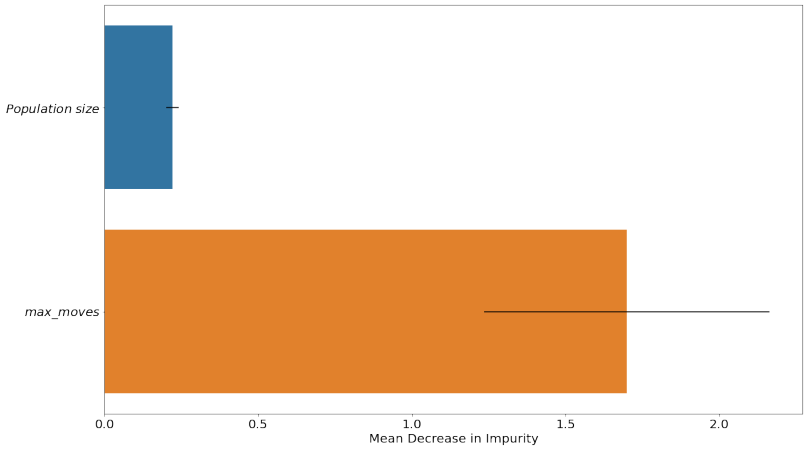

FIGURE 4.30: Relative importance of BCO hyperparameters for dataset I_2

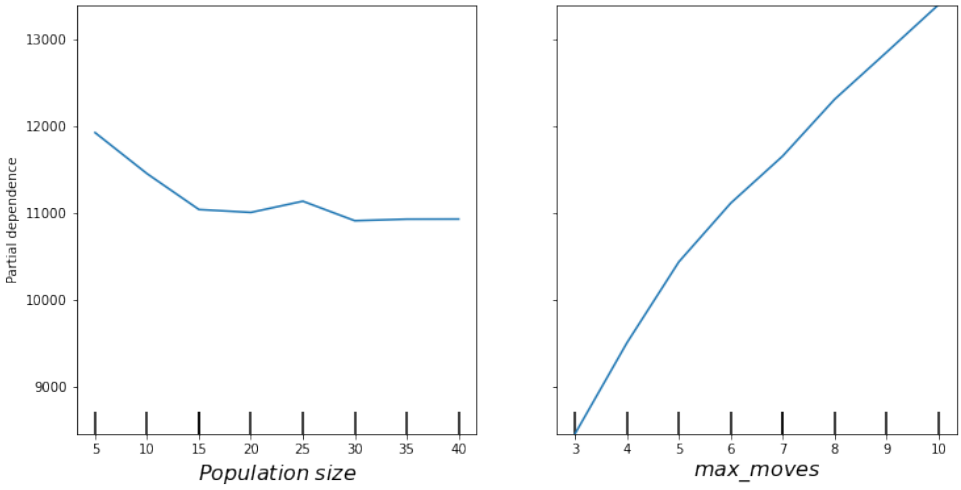

FIGURE 4.31: Impact of BCO hyperparameters on the performance on instance *c101_21*

4.14 GA and MA

The genetic algorithm, known for its versatility, includes various forms based on the specific genetic operators it integrates. In our method, we initiate the process by generating an initial population of solutions. For crafting a single solution, we utilize Algorithm 12, while the rest of the population is created through Algorithm 11. This approach leverages the strengths of both algorithms: Algorithm 12 is adept at providing a robust initial solution, and Algorithm 11 introduces a range of diversity into the population.

Figure 4.32 illustrates a transformation from our solution structure described in Section 4.4 to a chromosome for genetic algorithm. Distinct color groupings (blue, green, and orange) denote individual routes taken by separate vehicles. Within this chromosome, special markers, numbered as ”0”, delineate the end of one vehicle’s route and the start of another’s. For instance, the sequence ”1 5 6” between two

zeros signifies the first vehicle's route, visiting customers 1, 5, and 6 consecutively. This compact representation is better suited for the crossover operation in genetic algorithms than our standard solution representation.

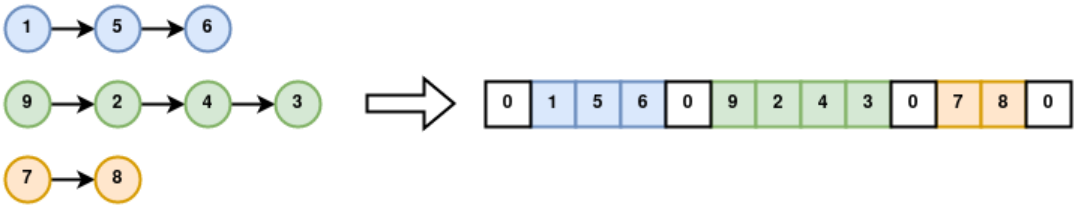

FIGURE 4.32: Representation of a chromosome

Numerous crossover operators are documented in literature, such as Single-point Crossover [137], Multi-point Crossover [112], Uniform Crossover [327], Partially Matched Crossover [113], Cycle Crossover [259], Edge Recombination Crossover [357], among others.

For this research, we explored the capabilities of two crossover methods: *Two-point Crossover* (**TPX**) and *Order Crossover* (**OX**) [87].

Two-point crossover is a specific instance of Multi-point crossover. In this method, two points are selected on the parent chromosomes, and everything between these points is exchanged, resulting in two offspring. Although this approach may not be suitable for problems based on permutations, as it can produce offspring with redundant or missing elements, we wanted to investigate whether its inherent randomness might improve or hinder the algorithm's effectiveness. This additional randomness occurs because the offspring must be corrected after creation by removing any duplicate customers and then randomly placing absent customers within the solution, or by removing or adding some depot visits to reduce or increase the route count.

Order Crossover is a specialized crossover technique that is primarily used for permutation-based problems in genetic algorithms. One of the most popular applications of this technique is for solving the Traveling Salesman Problem. The main idea behind OX is to preserve the relative order of genes (or elements) from the parent chromosomes while creating offspring that are valid permutations. To apply OX, two crossover points are randomly chosen on the parent chromosomes. The section between these points in one parent is directly copied to the offspring. To complete the offspring, genes from the second parent are considered in their order of appearance, but only if they are not already present in the offspring. This guarantees that the offspring inherits genes from both parents while maintaining the gene's relative ordering and producing valid permutations. Since depot visits (designated as 0) are also being copied, it is possible that the number of routes in the new solution may exceed the available number of vehicles. In such cases, we remove some depot visits from the chromosomes, merging the shortest routes to resolve the issue. The example of OX is presented in Figure 4.33.

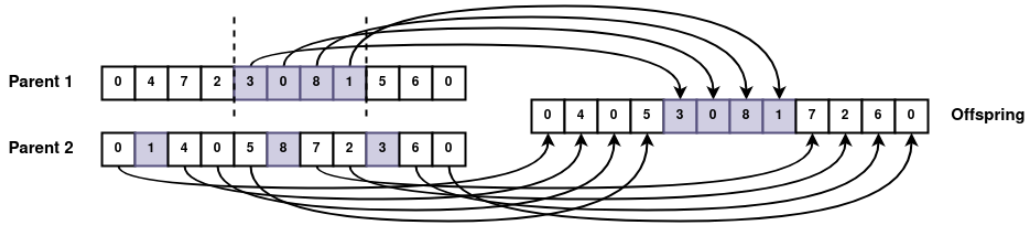

FIGURE 4.33: A diagram illustrating the OX crossover operator.

Both operators underwent testing across both instance sets described in Section 4.3, utilizing hyperparameters identified by iRace with an identical budget allocated for each algorithm. Within the I_1 instance set, TPX marginally surpassed OX concerning the average solution quality. Specifically, TPX outperformed OX in 17 instances, with differences ranging from a minimum of 0.25% to a maximum of 6.48%, and an average difference of 1.98%. In contrast, both operators achieved identical results in 11 instances, while TPX fell short in 8 instances, registering differences between 0.05% and 8.47%, and an average shortfall of 2.54%. Figure 4.34 showcases both the Q-Q plot and a histogram illustrating the differences. An assessment of this figure highlights a deviation from a normal distribution for the differences, a finding further endorsed by the *Shapiro–Wilk test*. Consequently, to determine any statistically significant discrepancies between the two algorithms, we employed the *Wilcoxon signed-rank test*, which operates without the presupposition of normality and assesses the significance of variations between two paired groups. Based on this test, at a conventional significance threshold of 0.05, the p-value stood at 0.109, indicating no statistically significant difference between the two algorithms.

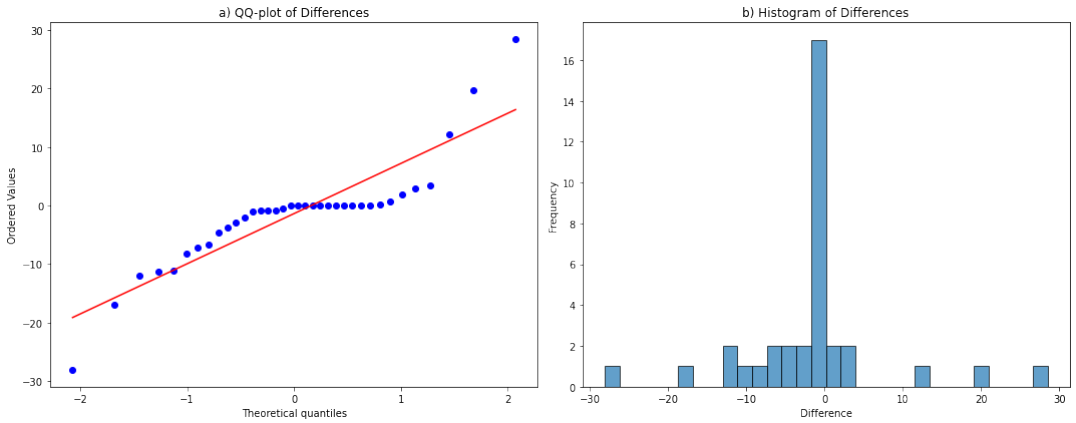

FIGURE 4.34: Distribution of differences for TPX and OX presented as: a) Q-Q plot, b) Histogram

Conversely, upon evaluating the operator on the set of larger instances (I_2), OX decidedly outperformed TPX. OX achieved superior solutions in all 56 instances. The performance enhancements ranged from a minimal improvement of 17.34% to a peak of 74.49%, with an average improvement of 51.40%. Given this clear disparity in outcomes, we deemed additional statistical tests redundant.

In our mutation procedure, each individual in the population has a chance to undergo mutation based on a specified probability termed the *mutation_rate*. The procedure operates by iterating over each individual and generating a random number

between 0 and 1. If this number is less than or equal to the *mutation_rate*, the individual is selected for mutation. The extent of this mutation is determined by a randomly chosen integer, *transformation_size*, which can be between 1 and 3. Once selected, the TRANSFORM function is invoked to mutate the individual based on the defined *transformation_size*. We considered two functions as the TRANSFORM function: the SHAKING procedure described in Algorithm 17, and the TRANSFORM_SOLUTION procedure described in Algorithm 23. We tested these two variations on a subset of instances from I_2, but did not find any statistically significant difference. As a result, we have decided to use the TRANSFORM_SOLUTION function in all future experiments. After considering every individual in the population for mutation, the procedure is completed. This procedure is described in Algorithm 24.

Algorithm 24 Mutation procedure

```
1: procedure MUTATION(population, mutation_rate)
2:     for ∀individual ∈ population do
3:         r ← random number in range [0,1]
4:         if r ≤ mutation_rate then
5:             transformation_size ← random integer in range [1,3]
6:             TRANSFORM(individual, transformation_size)
7:         end if
8:     end for
9: end procedure
```

In our version of the algorithm, we replace the *crossover rate* hyperparameter from Algorithm 6 with the *number of offspring* hyperparameter. This new parameter dictates the number of offspring generated through crossover. During each step, two parents are selected using the *Roulette wheel selection*. This process is repeated $\frac{number\ of\ offspring}{2}$ times to produce the predetermined number of offspring, which are subsequently incorporated into the original population. Selection adopts an *elitist approach*. After incorporating the offspring into the population, only the top *population size* individuals advance to the subsequent generation.

Our memetic algorithm aligns with the GA in terms of selection, mutation, and crossover operations. However, it distinguishes itself by incorporating the VND as its local search procedure.

In addition, we also propose another algorithm for solving our problem, that we call the *Modified Memetic Algorithm* (**MMA**). The MMA represents an adaptation of the traditional Memetic Algorithm. Both algorithms combine elements of genetic algorithms with local search procedures, which allows them to harness the explorative powers of genetic algorithms and the exploitative strengths of local searches. This hybrid approach offers a balance between global exploration of the solution space and local refinement.

However, the distinction between MMA and classic MA lies in their local search application. In traditional MA, local search is typically applied uniformly across selected individuals or a significant portion of the population. This ensures that not only the best solutions but also other potential solutions in the population undergo refinement, allowing for diverse exploration and exploitation.

In contrast, the MMA is more selective in its approach. It applies the local search procedure exclusively to the offspring with the best solution in each generation. By focusing solely on the most promising solution in each generation, MMA emphasizes intensive refinement of the best solutions over the exploration of the broader solution landscape. This approach might speed up convergence to a high-quality solution, but

there is a trade-off—it could also potentially risk getting trapped in local optima if the broader population's diversity is not maintained. In essence, while classic MA maintains a broader exploration and exploitation balance, MMA leans more towards intensive exploitation of the top-performing solutions.

4.14.1 Hyperparameter analysis

Our implementation of the GA is configured with three key hyperparameters:

- *Population size* - The hyperparameter that specifies the size of the population in terms of the number of individuals.
- *Number of offspring* - The hyperparameter that dictates the number of new offspring generated in each generation.
- *Mutation rate* - The hyperparameter that sets the likelihood of the mutation operator being applied to an individual.

For the MA and MMA algorithms, there is an additional hyperparameter:

- vnd_{limit} - Hyperparameter managing the time constraint for each invocation of the VND procedure, measured in seconds.

It should be emphasized that our assessment of the relative influence of hyperparameters was specifically conducted on versions of GA, MA, and MMA that utilize the *order crossover* operator.

The relative impact of each GA hyperparameter on instances from dataset I_1 is depicted in Figure 4.35. Additionally, Figure 4.36 presents PDPs that illustrate the performance of GA on the instance *r202C15* across various hyperparameter values. From the insights obtained from these figures, it is evident that the number of offspring is the most influential hyperparameter, with an increase in offspring correlating to improved solution quality. In comparison, the population size and the mutation rate exerted a relatively lesser impact. Upon examining instances from dataset I_2, as illustrated in Figures 4.37 and 4.38, the number of offspring continued to be the most influential hyperparameter. However, in this case, the population size and the mutation rate assumed a more prominent role. Interestingly, elevating these two parameters tended to diminish the quality of the solutions.

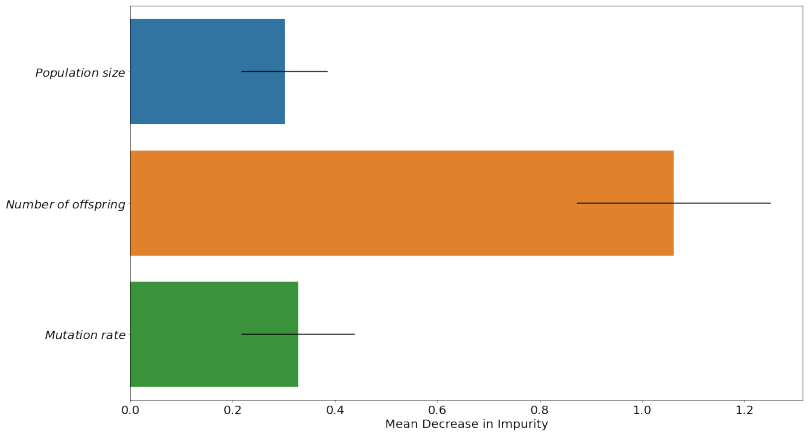

FIGURE 4.35: Relative importance of GA hyperparameters for dataset I_1

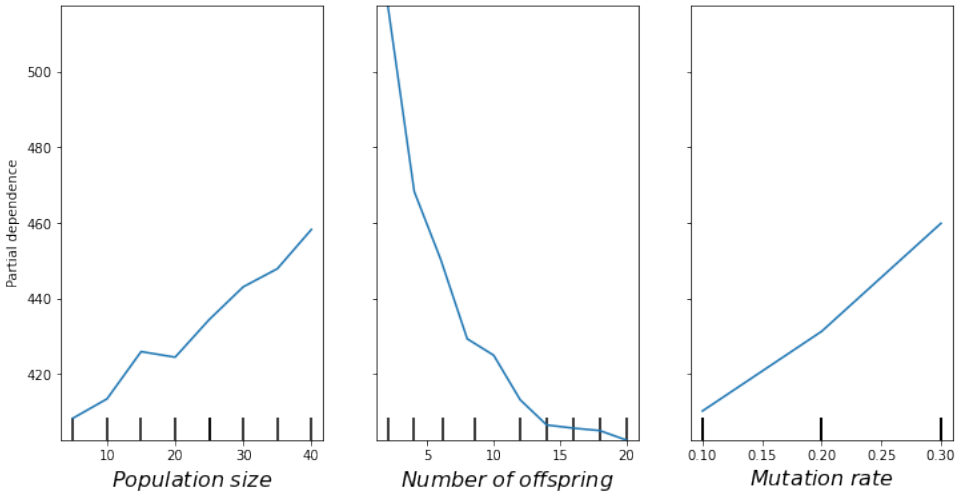

FIGURE 4.36: Impact of GA hyperparameters on the performance on instance *r202C15*

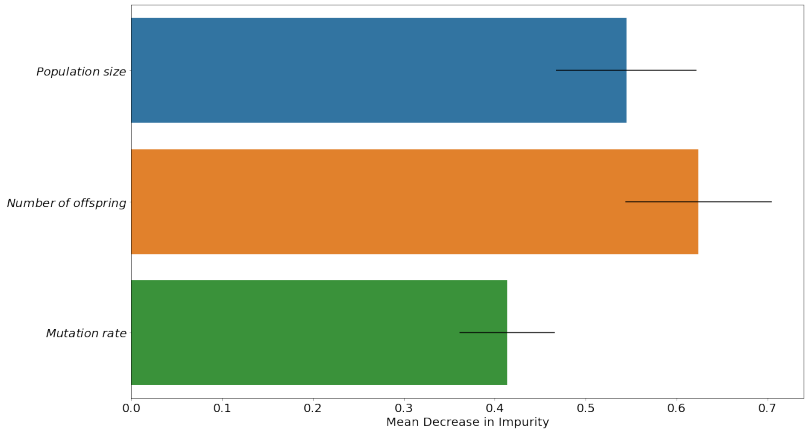

FIGURE 4.37: Relative importance of GA hyperparameters for dataset I_2

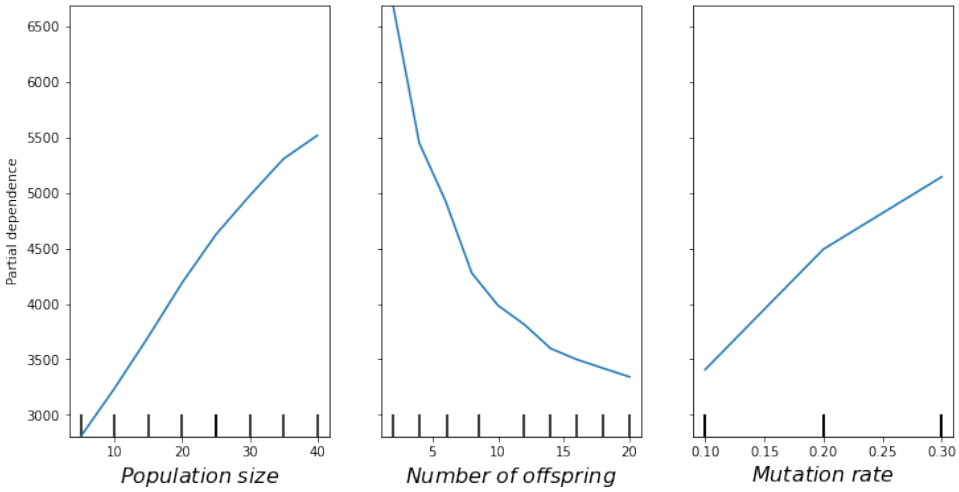

FIGURE 4.38: Impact of GA hyperparameters on the performance on instance *c101_21*

For the MA algorithm, a similar analysis was conducted, with the relative influence of hyperparameters on instances from dataset I_1 depicted in Figure 4.39 and the corresponding PDPs shown in Figure 4.40. The outcomes here contrast significantly with those for GA as nearly all hyperparameters, except vnd_{limit}, exhibited a substantial impact. The findings for MA on instances from dataset I_2 are particularly interesting (Figures 4.41 and 4.42). The population size emerged as the sole significantly influential parameter, where its increase led to inferior solution quality. It is worth noting that the situation becomes even more interesting due to the fact that, unlike in the cases of GVNS, GRASP, and ACOLS, where an increased VND limit generally improved solution quality, for MA, a higher VND limit adversely affected solution outcomes. This likely stems from the fact that the population size in MA can be quite substantial, and conducting VND on each solution with an extended time limit may result in considerable time spent exploring less promising solutions. In contrast, while ACOLS may also involve a large number of ants, it tends to benefit from prolonged VND runs. This difference can be attributed to the GA component of MA, which appears more effective in steering the search towards optimal solutions independently, as compared to the guidance provided by ACO, at least in the context of our specific problem.

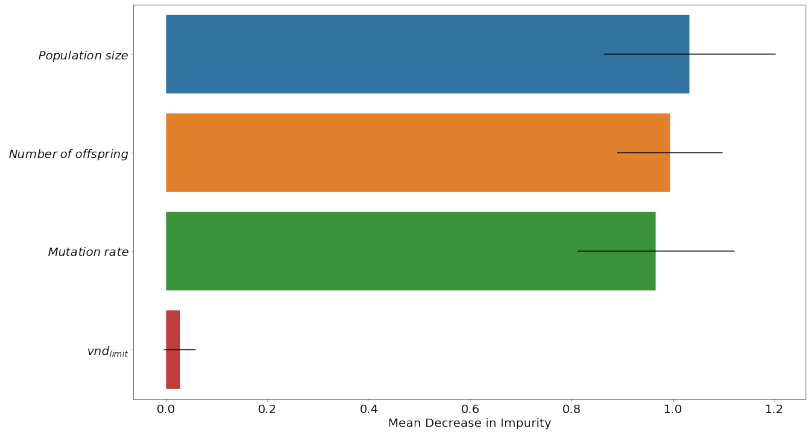

FIGURE 4.39: Relative importance of MA hyperparameters for dataset I_1

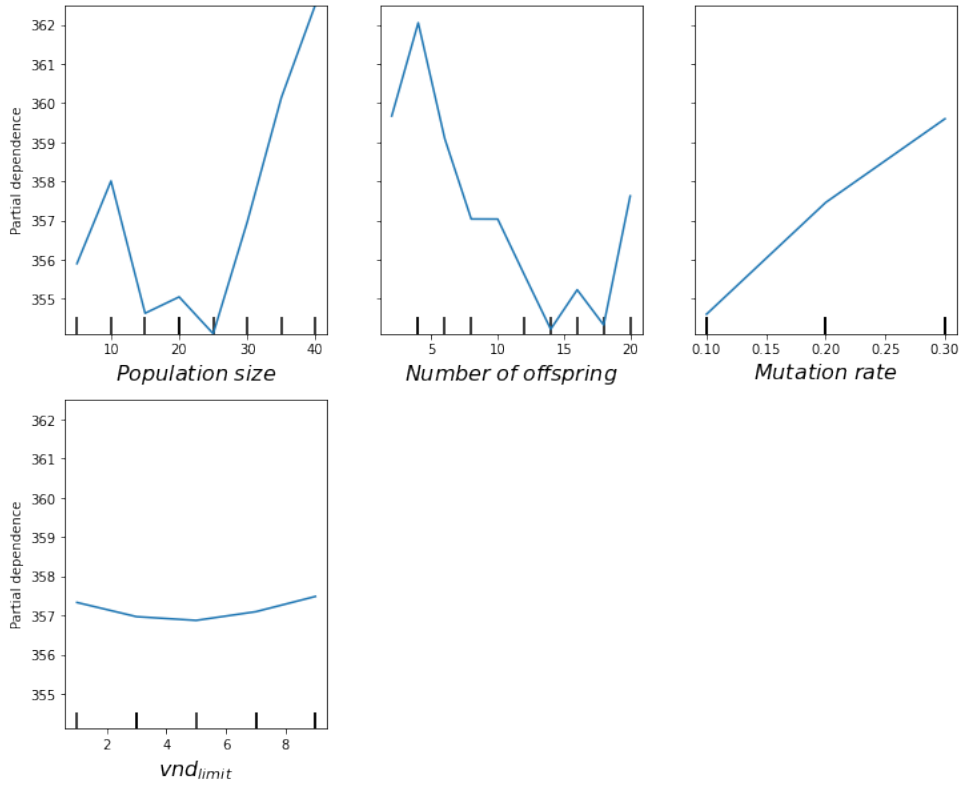

FIGURE 4.40: Impact of MA hyperparameters on the performance on instance *r202C15*

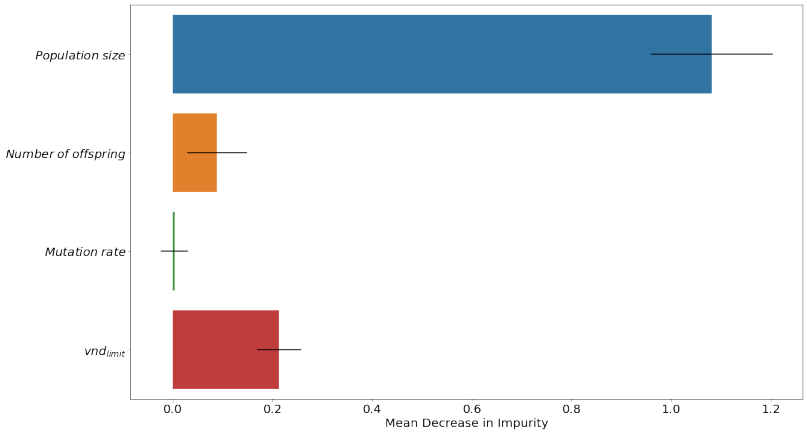

FIGURE 4.41: Relative importance of MA hyperparameters for dataset I_2

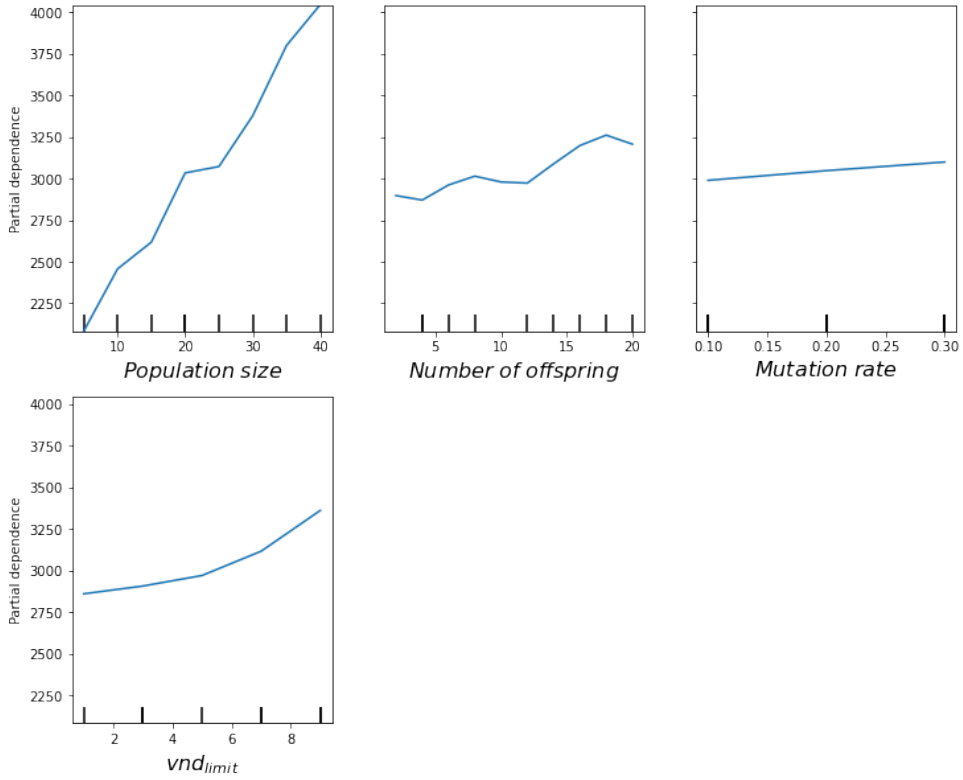

FIGURE 4.42: Impact of MA hyperparameters on the performance on instance *c101_21*

MMA demonstrated a hyperparameter influence pattern similar to MA for instances from dataset I_1, as depicted in Figures 4.43 and 4.44. However, when evaluated against instances from dataset I_2 (presented in Figures 4.45 and 4.46), the number of offspring assumed a more pivotal role in MMA compared to MA, with a higher count correlating to improved solution quality. Additionally, similar to the trends observed in GVNS, GRASP, and ACOLS, increasing the vnd_{limit} in MMA positively influenced the quality of the solutions.

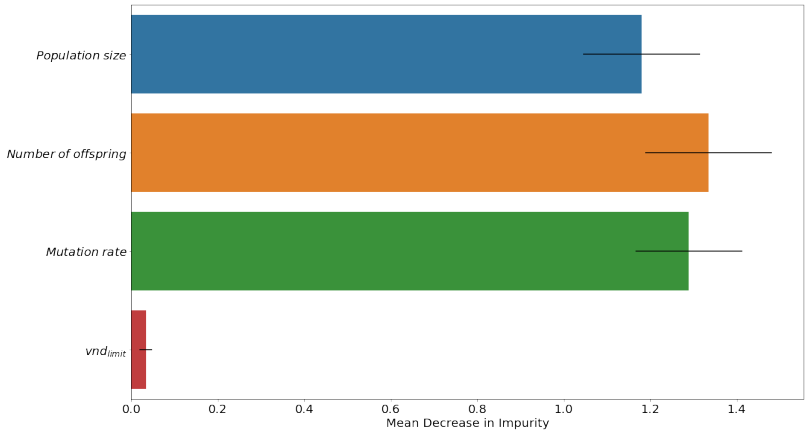

FIGURE 4.43: Relative importance of MMA hyperparameters for dataset I_1

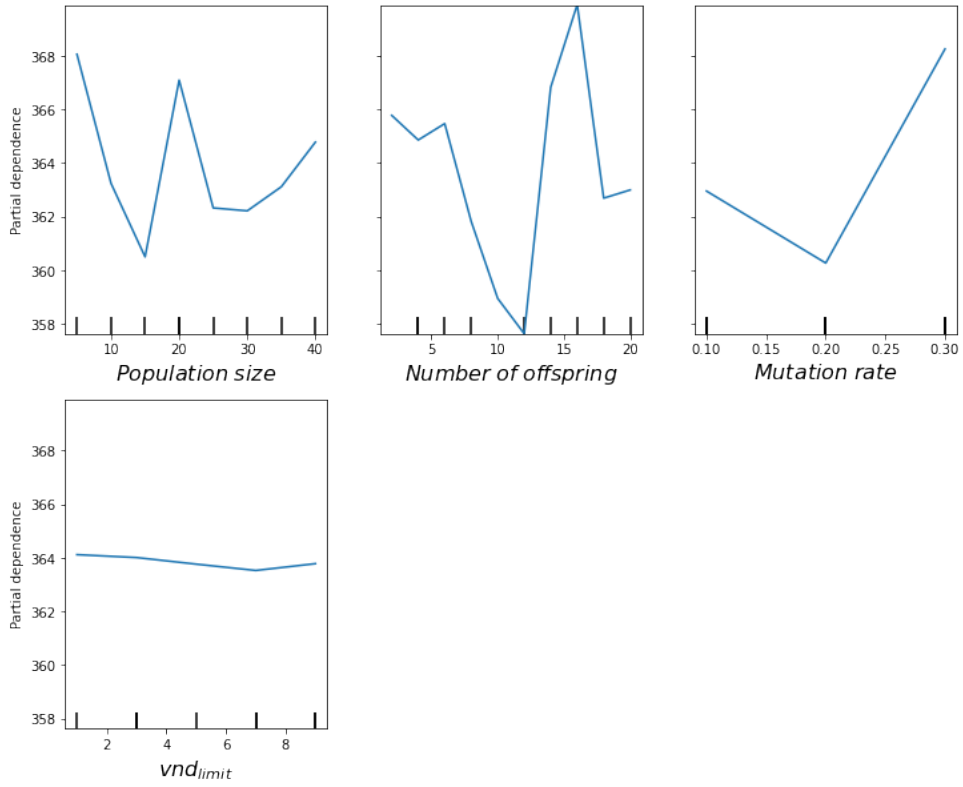

FIGURE 4.44: Impact of MMA hyperparameters on the performance on instance *r202C15*

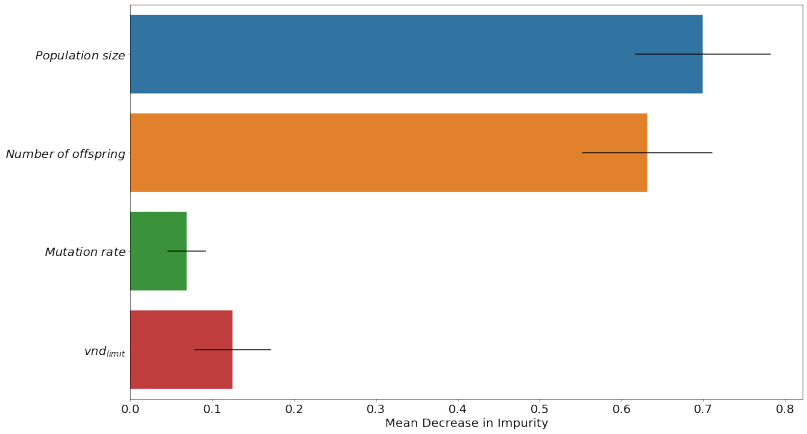

FIGURE 4.45: Relative importance of MMA hyperparameters for dataset I_2

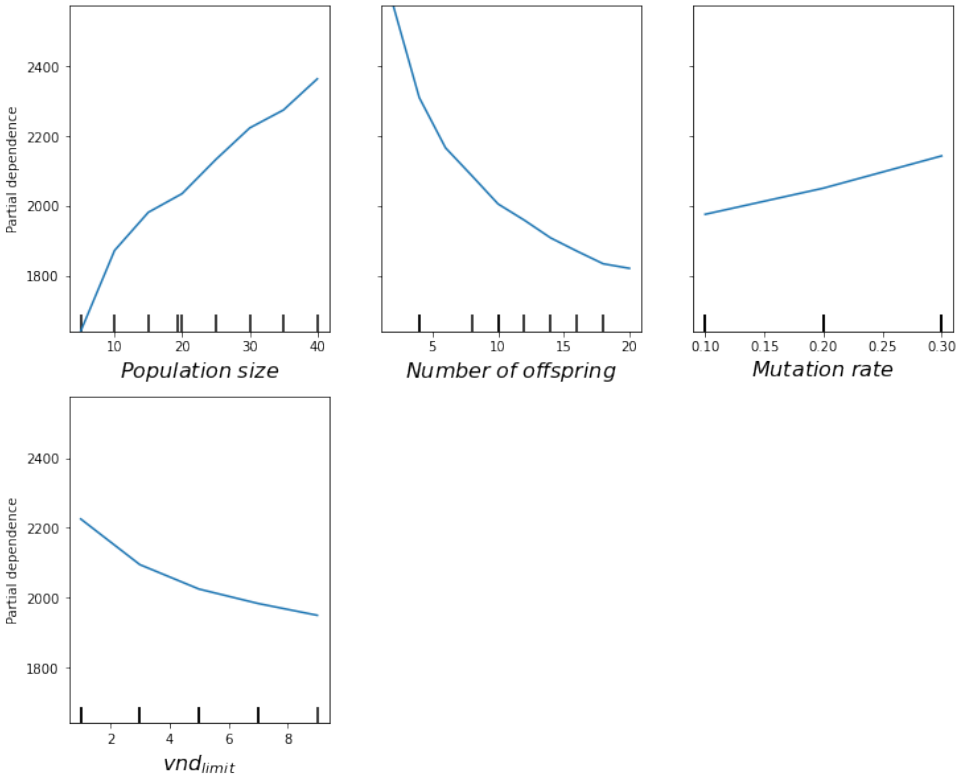

FIGURE 4.46: Impact of MMA hyperparameters on the performance on instance *c101_21*

4.15 Experimental evaluation

4.15.1 MILP performance

In order to evaluate the effectiveness of our MILP formulation, as outlined in Section 4.2, we employed the *CPLEX 20.1.0* commercial solver. A time constraint of 60 minutes was imposed for each test. As detailed in Section 4.2, our MILP model

necessitates the creation of 'dummy' nodes to represent refueling stations. This is achieved by duplicating existing station nodes, thereby enabling multiple visits to these stations. The replication frequency of these station nodes is a critical parameter; setting it too low might exclude potentially high-quality solutions by rendering them infeasible [1], whereas too high a frequency can significantly escalate the complexity of the model. For our experiments, we replicated station nodes threefold for instances with up to 10 customers, and sixfold for those with 15 customers. These replication rates were determined through experimental trials and yielded the most favorable outcomes.

The results derived from dataset I_1 are showcased in Table 4.3. However, the instances from dataset I_2 presented a formidable challenge for CPLEX, as it failed to identify a first feasible solution within a reasonable timeframe. This difficulty may stem from our model's reliance on *Miller–Tucker–Zemlin* (**MTZ**) inequalities [232]. While MTZ inequalities are advantageous due to their limited quantity, they also tend to produce weaker continuous relaxations, resulting in less tightly constrained models [73].

4.15.2 Experimental setup

The experiments were carried out on a personal laptop equipped with an Intel i7-10750H processor and 32GB of RAM, running the Ubuntu 20.04 operating system. We developed all proposed algorithms using the C++ programming language and compiled them using the GCC 9.4.0 compiler with the O3 optimization flag for enhanced performance. To ensure the reliability of our results, each test was conducted 30 times, demonstrating the consistency of the outcomes.

For the instance set I_1, we imposed a five-minute time limit per method during testing. In contrast, for the instance set I_2, the time limit was extended to ten minutes, accommodating their increased complexity. The selection of hyperparameter values for each method was done with precision and consideration, and these values will be detailed in the following section. This structured approach to testing and parameterization ensures a robust and comprehensive evaluation of the algorithms.

4.15.3 Hyperparameters tuning

To identify optimal values for our algorithms, we employed the *iRace*[5] package within the R programming language. Each algorithm was allocated a budget of 1000 experiments for every dataset. The resulting values for GVNS, GRASP, ACO, ACOLS, BCOi, GA, MA, and MMA are presented in Tables 4.4 through 4.11, respectively. The third row of each table lists the range of possible values for each hyperparameter from which iRace could make selections. The fourth and fifth rows detail the best-performing hyperparameter values identified for datasets I_1 and I_2, respectively. These determined values were then consistently used for all further testing of the algorithms.

4.15.4 Overall comparison

In Table 4.12, we present the average solution quality for each algorithm across a range of instances from the dataset I_1, with the top-performing results for each instance highlighted in bold. In the bottom row, we display the number of times each algorithm surpassed the others in terms of average solution quality. The comprehensive results

[5] `https://cran.r-project.org/web/packages/irace/index.html`

TABLE 4.3: Results obtained by CPLEX commercial solver using the model described in Section 4.2.

Instance	**C**	**A**	**Results**		
			Objective	**Gap (%)**	**Time (s)**
c101C10	10	5	401.5689	28.1	3600
c101C5	5	3	247.1497	0	9.37
c103C15	15	5	242.6993	38.3	3600
c103C5	5	2	165.6667	0	0.69
c104C10	10	4	279.9331	3.58	3600
c106C15	15	3	275.1332	0	239.73
c202C10	10	5	243.2035	0	444.9
c202C15	15	5	376.7893	12.02	3600
c205C10	10	3	228.2812	0	21.97
c206C5	5	4	236.579	0	85.3
c208C15	15	4	300.5485	0.04	3600
c208C5	5	3	158.4807	0	203.6
r102C10	10	4	256.0954	9.89	3600
r102C15	15	8	456.0582	44.13	3600
r103C10	10	3	175.3568	0	1251.41
r104C5	5	3	136.6897	0	9.32
r105C15	15	6	365.8703	33.41	3600
r105C5	5	3	156.0821	0	13.88
r201C10	10	4	222.4339	0	2597.21
r202C15	15	6	362.6361	28.37	3600
r202C5	5	3	128.7771	0	5.66
r203C10	10	5	218.2135	0	1996.53
r203C5	5	4	179.0559	0	140.95
r209C15	15	5	293.2004	12.29	3600
rc102C10	10	4	423.5102	8.29	3600
rc103C15	15	5	415.1275	42.93	3600
rc105C5	5	4	238.0522	3.19	3600
rc108C10	10	4	345.9273	12.34	3600
rc108C15	10	5	396.9885	21.98	3600
rc108C5	5	4	253.9307	12.31	3600
rc201C10	10	4	310.0573	0	729.84
rc202C15	15	5	391.6175	30.25	3600
rc204C15	15	7	310.8969	22.52	3600
rc204C5	5	4	176.394	0	534.11
rc205C10	10	4	325.9774	0	3118.21
rc208C5	5	3	167.9835	0	337.36

TABLE 4.4: The best-found values for each GVNS hyperparameter.

	Hyperparameters				
	k_{min}	k_{step}	k_{max}	l_{max}	vnd_{limit}
Interval	[1, 5]	[1,3]	[6, 30]	[3,6]	[1, 9]
I_1	3	3	20	5	8
I_2	1	2	12	5	7

TABLE 4.5: The best-found values for each GRASP hyperparameter.

	Hyperparameters	
	rcl_{limit}	vnd_{limit}
Interval	[2, 9]	[1, 9]
I_1	6	3
I_2	3	9

TABLE 4.6: The best-found values for each ACO hyperparameter.

	Hyperparameters			
	population size	α	β	ρ
Interval	[5, 40]	[1, 4]	[1, 4]	{0.1, 0.2, … , 0.9}
I_1	33	1	3	0.2
I_2	34	4	1	0.1

TABLE 4.7: The best-found values for each ACOLS hyperparameter.

	Hyperparameters				
	population size	α	β	ρ	vnd_{limit}
Interval	[5, 40]	[1, 4]	[1, 4]	{0.1, 0.15, 0.2, 0.25, 0.3}	[1, 9]
I_1	10	1	4	0.1	5
I_2	34	2	1	0.25	3

TABLE 4.8: The best-found values for each BCO hyperparameter.

	Hyperparameters	
	max_moves	*population size*
Interval	[3, 10]	[5, 40]
I_1	3	33
I_2	3	11

TABLE 4.9: The best-found values for each GA hyperparameter.

	Hyperparameters		
	population size	*Number of offspring*	*Mutation rate*
Interval	[5, 40]	[2, 20]	{0.1, 0.2, 0.3}
I_1	16	20	0.1
I_2	6	10	0.1

TABLE 4.10: The best-found values for each MA hyperparameter.

	Hyperparameters			
	population size	*Number of offspring*	*Mutation rate*	vnd_{limit}
Interval	[5, 40]	[2, 10]	{0.1, 0.2, 0.3}	[1, 9]
I_1	25	17	0.3	4
I_2	6	3	0.1	7

Table 4.11: The best-found values for each MMA hyperparameter.

	Hyperparameters			
	population size	*Number of offspring*	*Mutation rate*	vnd_{limit}
Interval	[5, 40]	[2, 10]	{0.1, 0.2, 0.3}	[1, 9]
I_1	18	17	0.3	3
I_2	5	13	0.1	8

for all algorithms, encompassing all metrics, are detailed in an external document provided at `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`. To ascertain whether the differences in performance among these algorithms are statistically significant, it was necessary to evaluate the feasibility of conducting an ANOVA analysis. ANOVA requires three key conditions to be valid [213]:

- The data should be normally distributed.
- The variances across groups should be equal.
- Each sample must be collected independently from the other samples. In our case, this assumption is met by default.

To verify the first two conditions, we employed the Shapiro-Wilk test [308] to check for normal distribution of each algorithm's results and the Levene's test [187] to examine the equality of variances across algorithms. The Shapiro-Wilk test yielded p-values below 0.05 for certain algorithms (ACO, ACOLS, and BCOi), suggesting their data do not adhere to a normal distribution. Additionally, the Levene's test produced a significantly low p-value, further indicating unequal variances among the algorithms. These findings imply that the criteria for a traditional ANOVA are not satisfied.

Consequently, we opted for the Kruskal-Wallis test [172], a non-parametric alternative that does not assume normal distribution. This test is suitable for determining statistically significant differences between two or more groups on a continuous or ordinal dependent variable. The Kruskal-Wallis test yielded a p-value of approximately 0.119, exceeding the conventional threshold of 0.05. This indicates that the performance differences among the algorithms, when evaluated on instances from dataset I_1, are not statistically significant.

In Table 4.13, we provide a comprehensive overview of each algorithm's performance across a variety of metrics. The table details the number of instances in which each algorithm achieved the best value for a given metric. It is important to note that in cases where two or more algorithms attained the same optimal value, this achievement is attributed to each of those algorithms. This table reflects a similar pattern observed in the comparison of average solution quality, where no single metaheuristic consistently outperforms the others across any specific metric. This suggests a balanced competitive landscape, with no definitive leader emerging in any particular performance category.

In the case of smaller test instances, we evaluated our metaheuristic approaches against the results produced by CPLEX, rounding all outcomes to two decimal places. Instances where CPLEX surpassed other methods are detailed in Table 4.14, including the percentage differences in solution quality. We omitted 17 cases from this table where CPLEX achieved the same solution quality as some other method. Conversely, Table 4.15 showcases instances where metaheuristic methods outperformed

TABLE 4.12: Results for all metaheuristics on the instance set I_1.

Instance	**C**	**A**	**Metaheuristics**							
			GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
c101C10	10	5	**385.31**	401.13	449.37	412.33	479.91	397.92	396.23	396.58
c101C5	5	3	**247.15**	250.04	259.73	**247.15**	254.23	249.17	**247.15**	248.59
c103C15	15	5	**356.52**	371.8	460.68	371.8	605.77	381.08	366.17	372.7
c103C5	5	2	**165.67**	479.05	**165.67**	873.81	173.6	**165.67**	322.36	323.87
c104C10	10	4	304.03	295.55	373.99	**273.93**	348.24	**273.93**	323.49	357.09
c106C15	15	3	277.51	275.83	365.37	**275.13**	487.69	301.83	**275.13**	281.89
c202C10	10	5	**251.94**	264.46	309.71	**251.94**	351.51	258.53	**251.94**	253.76
c202C15	15	5	373.25	**369.56**	512.54	378.45	623.89	380.3	369.61	**369.56**
c205C10	10	3	**228.28**	**228.28**	322.97	**228.28**	365.49	232.95	**228.28**	**228.28**
c206C5	5	4	**236.21**	241.49	258.04	239.94	237.4	**236.21**	**236.21**	**236.21**
c208C15	15	4	**300.55**	**300.55**	478.34	**300.55**	600.77	317.52	**300.55**	**300.55**
c208C5	5	3	**158.48**	205.0	210.19	195.7	213.74	**158.48**	**158.48**	**158.48**
r102C10	10	4	270.24	267.56	298.14	270.49	327.01	**250.14**	267.56	272.16
r102C15	15	8	438.67	539.29	476.85	459.57	546.21	**405.94**	539.24	521.9
r103C10	10	3	**175.36**	**175.36**	209.45	**175.36**	197.79	182.88	**175.36**	182.12
r104C5	5	3	**136.69**	**136.69**	137.01	**136.69**	137.23	**136.69**	**136.69**	**136.69**
r105C15	15	6	353.4	407.93	434.11	382.47	531.84	346.61	384.93	**337.04**
r105C5	5	3	181.02	172.45	195.31	169.03	163.46	**156.08**	161.6	158.11
r201C10	10	4	**222.43**	**222.43**	260.0	**222.43**	274.59	**222.43**	**222.43**	**222.43**
r202C15	15	6	380.24	359.84	577.37	381.64	623.74	384.84	351.87	**350.72**
r202C5	5	3	**142.65**	147.12	157.55	**142.65**	152.93	**142.65**	**142.65**	**142.65**
r203C10	10	5	**232.68**	**232.68**	307.88	**232.68**	342.73	233.44	**232.68**	**232.68**
r203C5	5	4	**179.06**	201.78	227.94	195.34	196.81	184.03	**179.06**	**179.06**
r209C15	15	5	**307.68**	347.07	456.3	334.15	536.89	307.83	**307.68**	**307.68**
rc102C10	10	4	455.09	465.48	486.37	461.15	479.25	450.87	**450.47**	**450.47**
rc103C15	15	5	426.98	426.21	476.33	513.04	604.68	**399.92**	486.51	560.1
rc105C5	5	4	273.34	276.17	237.25	278.38	231.22	**227.0**	259.43	275.23
rc108C10	10	4	400.0	398.8	417.8	464.96	429.12	**348.89**	438.67	449.48
rc108C15	10	5	421.02	467.44	512.09	505.41	678.98	**383.83**	514.22	556.82
rc108C5	5	4	399.07	399.07	271.74	341.02	268.1	**253.93**	370.04	388.81
rc201C10	10	4	**310.06**	**310.06**	369.73	**310.06**	378.69	**310.06**	**310.06**	**310.06**
rc202C15	15	5	**397.2**	445.12	532.11	416.59	680.99	433.98	405.87	403.68
rc204C15	15	7	351.24	353.22	554.22	353.85	619.69	340.54	319.93	**315.46**
rc204C5	5	4	**176.39**	185.16	187.69	178.61	185.64	177.22	**176.39**	178.61
rc205C10	10	4	357.13	344.95	446.84	460.58	449.8	**329.75**	364.92	396.08
rc208C5	5	3	**167.98**	203.94	190.47	168.93	196.53	**167.98**	168.79	**167.98**
Count best			20	8	1	12	0	18	17	17

TABLE 4.13: Performance comparison of metaheuristics for TD-EVRP-STW across various metrics (Dataset I_1).

Metric	**Method**							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality	**20**	8	1	12	0	18	17	17
Best-found solution	30	16	1	26	6	24	**33**	28
Worst-found solution	**20**	9	1	13	0	18	**20**	17
Penalty	17	17	**27**	17	26	24	17	18
Average route duration	1	2	6	0	**27**	2	0	0
Maximum route duration	6	16	17	10	14	**18**	11	15
Number of routes	**29**	12	3	20	3	17	28	25

CPLEX. It is worth noting that these tables reflect the best solutions found across all 30 runs of the metaheuristic algorithms. The scenario remains similar when considering average solution quality, with CPLEX leading in 10 instances, underperforming in 11, and matching solution quality in 15 cases. Given the time constraints (five minutes for metaheuristic methods versus one hour for CPLEX) these results suggest that metaheuristic algorithms are quite effective. Additionally, the superiority of the metaheuristic approach becomes even more evident in larger instances, where CPLEX failed to find even the first feasible solution.

TABLE 4.14: Instances where CPLEX was the best excluding ties with the second best

Instance	CPLEX Value	Second Best Value	Second Best Algorithm(s)	Difference
c103C15	242.70	348.54	GVNS	30.37%
c202C10	243.20	251.94	GVNS, ACOLS, MA, MMA	3.47%
r202C5	128.78	142.65	GVNS, GRASP, ACOLS, BCOi, GA, MA, MMA	9.72%
r203C10	218.21	232.68	GVNS, GRASP, ACOLS, GA, MA, MMA	6.22%
r209C15	293.20	307.68	GVNS, ACOLS, GA, MA, MMA	4.71%
rc102C10	423.51	450.47	GVNS, GRASP, ACOLS, BCOi, GA, MA, MMA	5.98%
rc202C15	391.62	397.20	GVNS, MA, MMA	1.40%
rc204C15	310.90	315.46	MA, MMA	1.45%

TABLE 4.15: Instances where CPLEX was outperformed by other algorithms

Instance	CPLEX Value	Best Non-CPLEX Value	Best Non-CPLEX Algorithm	Difference
c101C10	401.57	385.31	GVNS, ACOLS, MA, MMA	-4.05%
c104C10	279.93	273.93	ACOLS, GRASP, ACOLS, GA, MA, MMA	-2.14%
c202C15	376.79	369.56	GVNS, GRASP, ACOLS, GA, MA, MMA	-1.92%
c206C5	236.58	236.21	GVNS, ACOLS, GA, MA, MMA	-0.16%
r102C10	256.10	249.19	GVNS, GRASP, ACOLS, GA, MA	-2.70%
r102C15	456.06	397.99	GVNS	-12.73%
r105C15	365.87	323.79	GVNS	-11.51%
r202C15	362.64	347.21	MA, MMA	-4.25%
rc103C15	415.13	397.67	GVNS, GRASP, ACOLS, MA, MMA	-4.21%
rc105C5	238.05	227.00	BCOi, GA, MA	-4.64%
rc108C15	396.99	369.55	GVNS, GRASP, ACOLS, MA	-6.91%

In Table 4.16, we showcase the average solution quality of each algorithm across various instances from the dataset I_2. Values highlighted in bold indicate the top performance among all algorithms for a given instance. Analysis of this table reveals that the GVNS algorithm consistently achieved the highest solution quality, leading in 29 instances. It is closely followed by MMA, which secured the best results in 20 instances. The MA and GRASP algorithms, while competitive, were comparatively less successful in achieving the top spot, securing the best solution in 4 and 3 instances, respectively.

To assess whether there are statistically significant differences among the algorithms, our analysis begins with the *Shapiro-Wilk test*, applied to each algorithm's results to evaluate the normality of their distributions. The outcomes of this test reveal that the algorithms GVNS, GRASP, ACO, BCOi, GA, and MA do not conform to a normal distribution. In contrast, the results for ACOLS and MMA align more closely with a normal distribution. This divergence from normality in several algorithms suggests that the prerequisites for conducting a conventional ANOVA are not satisfied.

Consequently, we turn our attention to a non-parametric alternative, namely the *Kruskal-Wallis test*. This test is particularly fitting as it does not rely on the

TABLE 4.16: Results for TD-EVRP-STW obtained on the instance set I_2.

Instance	**C**	**A**	**Value**							
			GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
c101_21	100	21	1398.52	1965.53	9750.78	4264.11	8085.37	1978.9	1681.72	**1342.4**
c102_21	100	21	**1288.67**	1614.33	6668.71	3403.06	5788.21	1863.19	1523.02	1314.18
c103_21	100	21	**1217.04**	1484.72	4296.01	2967.35	4755.04	1776.04	1514.5	1275.14
c104_21	100	21	**1172.94**	1320.1	3216.92	2598.31	4258.82	1508.9	1231.57	1173.17
c105_21	100	21	**1356.9**	1664.26	7833.56	3658.4	7100.96	1894.01	1474.51	1358.14
c106_21	100	21	**1274.24**	1584.87	6252.16	3380.82	6439.67	1775.53	1434.86	1335.62
c107_21	100	21	**1174.64**	1550.86	6331.87	3162.13	6609.96	1761.08	1396.84	1336.93
c108_21	100	21	**1164.35**	1438.37	4984.24	3059.58	5605.57	1662.59	1326.15	1308.36
c109_21	100	21	**1144.19**	1333.11	3969.14	2820.16	4922.45	1608.73	1401.04	1238.89
c201_21	100	21	**1029.94**	1096.08	16532.46	3577.55	6892.61	1570.31	1365.1	1174.7
c202_21	100	21	**1009.1**	1090.49	9208.88	3040.07	4988.78	1509.53	1231.18	1087
c203_21	100	21	**1035.19**	1064.7	4622.01	2791.45	4482.58	1435.02	1116.07	1061.55
c204_21	100	21	1006.27	1022.79	3431.11	2541.96	3987.69	1324.9	1052.14	**1005.81**
c205_21	100	21	**1055.63**	1115.19	11863.45	3064.64	6007.85	1426.51	1182.13	1165.98
c206_21	100	21	**1018.77**	1127.59	8844.86	2975.73	5196.69	1397.39	1234.86	1122.22
c207_21	100	21	**1038.21**	1114.9	6649.66	2874.04	4750.37	1321.94	1236.52	1085.72
c208_21	100	21	**1007.64**	1113.1	6780.33	2811.37	4861.4	1327.02	1252.16	1083.48
r101_21	100	21	1855.86	2137.73	5289.69	3733.9	4969.5	2135.9	1945.79	**1796.81**
r102_21	100	21	1730.98	1862.49	4274.61	3292.63	4357.63	1929.85	1790.05	**1647.96**
r103_21	100	21	1553.79	1603.01	3745.98	2851.69	3856.42	1743.02	1557.31	**1470.93**
r104_21	100	21	**1286.6**	1342.15	2877.23	2453.63	3466.14	1516.37	1327.47	1373
r105_21	100	21	1676.42	1779.99	4433.89	3326.81	4482.85	1883.3	1753.99	**1631.73**
r106_21	100	21	1568.45	1659.83	3916.88	2962.04	4069.58	1780.22	1567.92	**1566.06**
r107_21	100	21	1423.25	1502.42	3159.36	2723.56	3668.11	1688.25	**1410.08**	1442.94
r108_21	100	21	1317.02	1331.48	3143.9	2613.72	3382.82	1474.45	**1277.37**	1295.67
r109_21	100	21	1521.86	1528.56	3268.92	2818.52	3906.53	1690.4	1511.39	**1444.34**
r110_21	100	21	**1342.52**	1379.79	2915.06	2511.39	3540.85	1528.91	1361.38	1375.01
r111_21	100	21	**1341.27**	1436.63	2956.94	2531.26	3557.94	1582.46	1378.18	1463.41
r112_21	100	21	1298.21	1319.3	2772.27	2375.66	3263.3	1467.38	1284.05	**1283.44**
r201_21	100	21	**1298.3**	1360.29	9188.95	2832.02	4076.65	1471.91	1335.95	1622.56
r202_21	100	21	**1183.33**	1238.73	6831.98	2502.23	3645.91	1351.47	1215.06	1328.25
r203_21	100	21	1130.49	1118.48	4009.43	2318.25	3397.04	1275.55	1120.35	**1063.43**
r204_21	100	21	989.73	1002.74	3263.97	1976.21	2942.22	1200.28	1003.8	**984.35**
r205_21	100	21	1193.47	1209.56	6196.51	2457.84	3693.02	1316.65	**1191.93**	1254.08
r206_21	100	21	**1124.19**	1145.22	4281.01	2461.1	3518.2	1307.75	1168.62	1200.59
r207_21	100	21	**1026.1**	1057.2	3629.04	2077.4	3157.88	1239.05	1054.14	1033.08
r208_21	100	21	1041.01	**981.11**	3071.61	1913.41	2800.78	1208.54	1024.9	996.12
r209_21	100	21	1145.16	**1120.53**	4094.65	2393.74	3426.64	1279.5	1136.62	1252.71
r210_21	100	21	1111.36	1119.55	3848.74	2419.17	3402.6	1264.38	1098.92	**1096.46**
r211_21	100	21	**964.85**	1063.75	3439.86	2153.35	2851.4	1201.12	1047.07	1046.74
rc101_21	100	21	**2013.8**	2176.51	5584.32	3943.16	5852.98	2244.7	2159.93	2040.86
rc102_21	100	21	1994.2	1998.45	4725.28	3638.03	5369.58	2181.04	1941.02	**1826.31**
rc103_21	100	21	1825.17	1827.44	4375.07	3331.43	4899.39	2018.25	1767.09	**1621.43**
rc104_21	100	21	1695.11	1702.19	3729.14	3077.9	4512.52	1765.64	1488.88	**1476.75**
rc105_21	100	21	1949.79	1965.76	4720.27	3419.8	5321.63	2108.41	1932.1	**1740.49**
rc106_21	100	21	1760.9	1919.35	4664.7	3630.67	5194.9	2035.84	1802.99	**1713.77**
rc107_21	100	21	1611.65	1766.19	3969.32	3187.4	4698.42	1830.97	1604.11	**1488.94**
rc108_21	100	21	1668.58	1743.76	3696.77	2989.64	4510.1	1857.97	1568.43	**1457.14**
rc201_21	100	21	**1618.54**	1639.38	9754.37	3737.58	5394.11	1759.15	1642.03	1633.31
rc202_21	100	21	1545.23	1501.56	7415.01	3235.27	4751.73	1589.22	**1488.77**	1565.08
rc203_21	100	21	**1245.04**	1349.18	4803.52	2958.78	4386.65	1472.15	1325.35	1404.95
rc204_21	100	21	1184.96	1214.43	3763.28	2579.8	3846	1457.33	1209.11	**1148.66**
rc205_21	100	21	**1415.37**	1467.57	6286.59	3266.2	4867.6	1618.84	1502.6	1464.91
rc206_21	100	21	1441.6	**1411.53**	6649.22	3269.32	4756.54	1547.73	1438.78	1561.74
rc207_21	100	21	**1211.6**	1322.35	4734.6	3120.88	4220.85	1437.93	1310.05	1271.28
rc208_21	100	21	**1085.55**	1203.04	3706.53	2770.57	3575.27	1398.79	1246.43	1181.17
Count best			29	3	0	0	0	0	4	20

assumption of normal distribution and is designed to identify statistically significant differences across two or more independent groups on a continuous or ordinal dependent variable. The Kruskal-Wallis test yielded a highly significant result (p-value < 0.0001), indicating pronounced differences among the algorithms.

Given that the Kruskal-Wallis test establishes the presence of differences but does not pinpoint their specific locations, we typically follow up with post-hoc tests. These tests help in discerning which exact pairs of algorithms differ significantly. A commonly employed post-hoc test following the Kruskal-Wallis are *Dunn's test* [79] or *Mann-Whitney U test* [212], which are used to further investigate and clarify these differences.

In the realm of multiple statistical testing, the likelihood of encountering at least one false positive, or *type I error*[6], escalates with the number of tests conducted. This cumulative error probability is termed the *familywise error rate* (**FWER**)[7]. To mitigate the FWER, various correction strategies are employed. These strategies are designed to adjust the p-values from individual tests, taking into consideration the total number of comparisons. The primary objective is to diminish the chance of mistakenly identifying a significant difference when, in fact, it is merely a product of random variability.

A well-known method for this adjustment is the *Bonferroni correction* [37], which is notably conservative. It adjusts the p-values by multiplying them with the number of comparisons, or alternatively, by dividing the significance threshold (commonly 0.05) by the number of comparisons. This approach is effective in ensuring that the FWER does not exceed the set threshold, such as 0.05.

However, the Bonferroni method can be excessively inflexible, particularly with a substantial number of comparisons, potentially leading to an increased risk of *type II errors*, where genuine differences are overlooked. Consequently, alternative methods, like the *Holm-Bonferroni procedure* [138] or the *False Discovery Rate* (**FDR**) [32] controlling methods, are often favored. These approaches strive to strike a balance between minimizing type I and type II errors.

In our analysis, we have opted to utilize Mann-Whitney U tests, together with FDR correction, to provide a more nuanced interpretation of the data. The obtained results are presented in Figure 4.47. This figure reveals that a significant number of these pairs exhibit statistically significant differences. However, certain algorithm pairs, such as ACO and BCOi, do not demonstrate a statistically notable distinction in performance. While this specific pair might not be of primary interest due to their overall lesser performance, the absence of significant differences between more proficient algorithms like GVNS, GRASP, MA, and MMA is particularly noteworthy.

Despite these observations, it is prudent to interpret these results cautiously. For instance, our in-depth comparative analysis in Section 4.15.6, which focused on MA and MMA, highlighted MMA's superior performance in numerous aspects when evaluated independently. This disparity suggests that the apparent lack of difference between MA and MMA, as indicated in Figure 4.47, could potentially be a Type II error. Such an error might arise as a consequence of the corrections applied following the post-hoc test, underscoring the need for a nuanced understanding of these findings.

[6]For a better understanding of *Type I* and *Type II* errors, please refer to Akobeng [6].

[7]For a more comprehensive understanding of the family-wise error rate, please see references [185, 338].

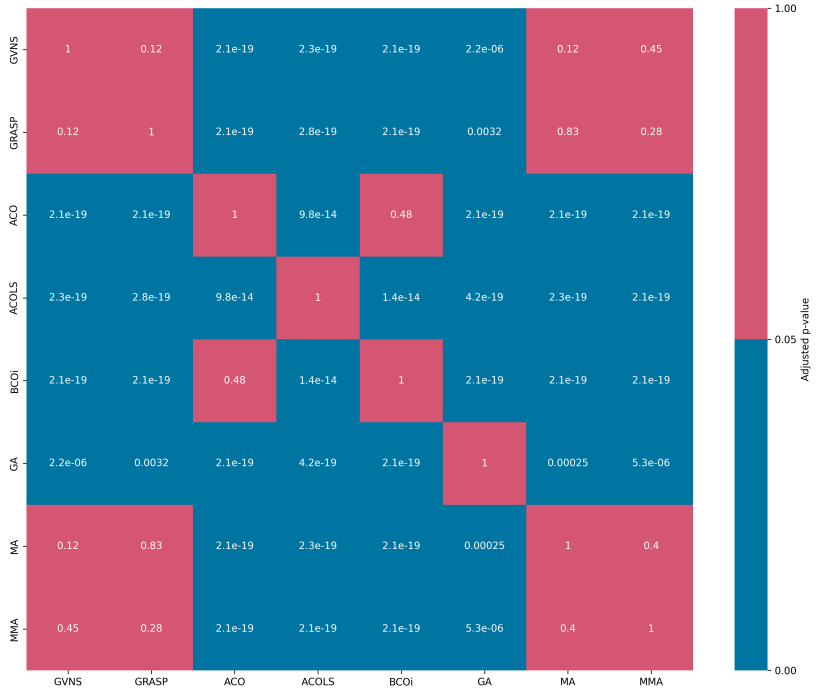

FIGURE 4.47: Heatmap of pairwise comparisons for TD-EVRP-STW on the instances from I_2 dataset (Mann-Whitney U test with FDR correction)

Given our suspicion that the outcomes of the Mann-Whitney U tests for MA may be influenced by a Type II error, coupled with the observation that GRASP demonstrates performance comparable to MA, it is logical to undertake a separate, focused comparison between GVNS and GRASP. This tailored analysis will enable us to scrutinize their performances in greater detail, providing a clearer understanding of their relative strengths and weaknesses.

In our detailed comparison between GVNS and GRASP, focusing on dataset I_2, we find that GVNS surpasses GRASP in average solution quality in 51 instances. The respective average, minimum, and maximum margins of superiority for GVNS are 8.43%, 0.12%, and 40.54%. Conversely, GRASP outshines GVNS in 5 instances, with the differences averaging at 2.88%, and ranging from a minimum of 1.07% to a maximum of 6.1%. These distinctions are clearly depicted in Figure 4.48, which illustrates the average solution quality achieved by each algorithm across individual instances. Here, instances plotted above the line indicate where GVNS has an edge over GRASP in terms of average solution quality across 30 runs.

Notably, despite the Mann-Whitney U test with FDR correction indicating no significant performance discrepancy between these algorithms, the Wilcoxon signed-rank test tells a different story, revealing a substantial difference with a p-value smaller than 0.0001. Moreover, GVNS consistently outperforms GRASP in all 56 instances concerning the best-found solutions, boasting an average advantage of 12.81%, and reaching up to 41.91%. The scenario changes slightly when examining the worst-found solutions; GVNS excels in 22 cases, while GRASP leads in 34 instances. It is

also observed that solutions derived from GVNS generally incur a higher penalty and longer average route durations, which aligns with its tendency to favor fewer number of routes.

Overall, this analysis underscores GVNS's frequent superiority over GRASP. It also strengthens the hypothesis that the lack of significant difference reported by the Mann-Whitney U test might indeed be a Type II error. Nonetheless, while GVNS exhibits a discernible edge in solution quality over GRASP, this advantage is not as pronounced as its superiority over other algorithms like ACO, ACOLS, and BCOi.

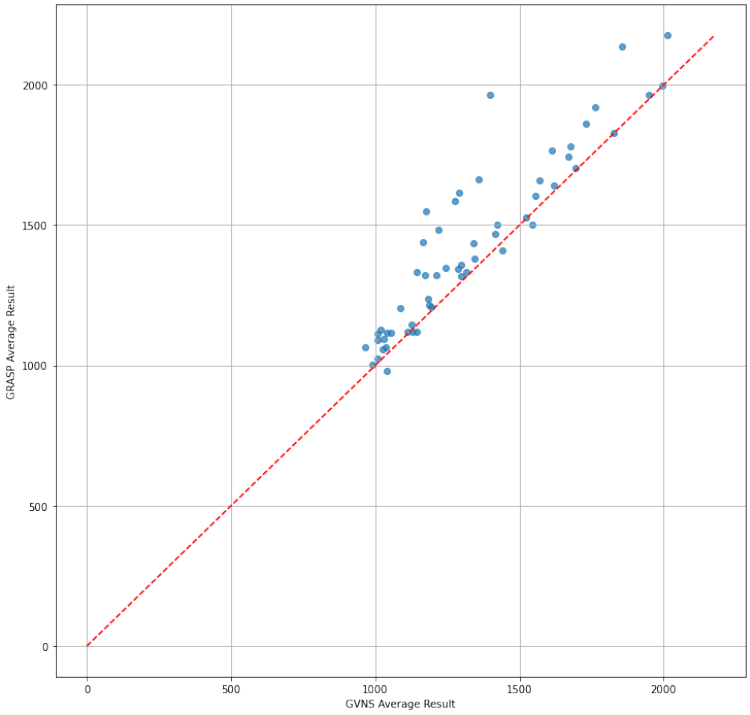

FIGURE 4.48: GVNS vs GRASP performance in terms of average solution quality on instances from I_2

In our final analysis, we turned our focus to the two top-performing algorithms: GVNS and MMA. Employing the Wilcoxon signed-rank test for a comparative assessment in terms of average solution quality, we corroborated our previous observation that these algorithms are closely matched in performance, with no significant differences between them. However, a notable divergence was identified when examining the best-found solutions across all 30 runs: GVNS consistently outperformed MMA in this regard. Additionally, GVNS demonstrated a inclination for generating solutions involving fewer routes.

This finding is particularly relevant in scenarios where the availability of vehicles is a critical constraint. In such cases, GVNS emerges as the potentially more advantageous choice due to its efficiency in route utilization. Outside of this specific context, both algorithms exhibit comparable levels of performance, making either a viable option depending on the specific requirements of the task at hand.

In Table 4.17, we provide a comprehensive analysis of the performance of various metaheuristics across multiple metrics, detailing the frequency with which each algorithm outshines its counterparts for specific metrics. This side-by-side comparison effectively illuminates the unique strengths and capabilities of each algorithm within diverse evaluative criteria. Notably, GVNS demonstrates superior performance in

TABLE 4.17: Performance comparison of TD-EVRP-STW metaheuristics across various metrics (Dataset I_2).

Metric	Method							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality	**29**	3	0	0	0	0	4	20
Best-found solution	**42**	0	0	0	0	0	0	14
Worst-found solution	13	14	0	0	0	0	8	**21**
Penalty	7	17	0	0	0	**20**	19	13
Average route duration	0	0	0	0	**56**	0	0	0
Maximum route duration	11	10	1	0	1	6	**23**	8
Number of routes	**41**	2	0	0	0	0	3	10

three distinct metrics, standing out from other metaheuristics. However, it is crucial to recognize that GVNS's higher frequency of outperformance does not necessarily imply a statistically significant superiority over others, as exemplified in the GVNS versus MMA comparison for average solution quality.

A particularly intriguing observation is BCOi's dominance in the average route duration metric. This can be attributed to BCOi's tendency to opt for solutions involving a greater number of routes, thereby reducing the average duration per route. As the matter of fact, BCOi consistently generated the highest route count in every test instance, a strategy that distinctly influenced its performance in this specific metric.

Figure 4.49 illustrates the convergence patterns of various algorithms towards their best-found solutions, using the *r101_21* instance as an example. This graph suggests that the previously set time limit of 600 seconds may not be necessary since most algorithms appear to converge within approximately 200 seconds. To provide a clearer comparison of the algorithms' performance in this early convergence phase, Figure 4.50 has been adjusted. Here, the maximum y-axis value is set to 10000, focusing exclusively on the algorithms' behavior during the initial 200 seconds of execution. This adjustment more effectively highlights the distinctions among the algorithms.

Observations from Figures 4.49 and 4.50 reveal that both GVNS and GRASP rapidly converge to a robust solution, showing minimal improvement thereafter. In contrast, MMA takes the full 200 seconds to achieve what GVNS accomplishes in about 25 seconds. Although these findings are based on a single instance, similar patterns have been noted across other instances, underscoring the consistent behavior of these algorithms.

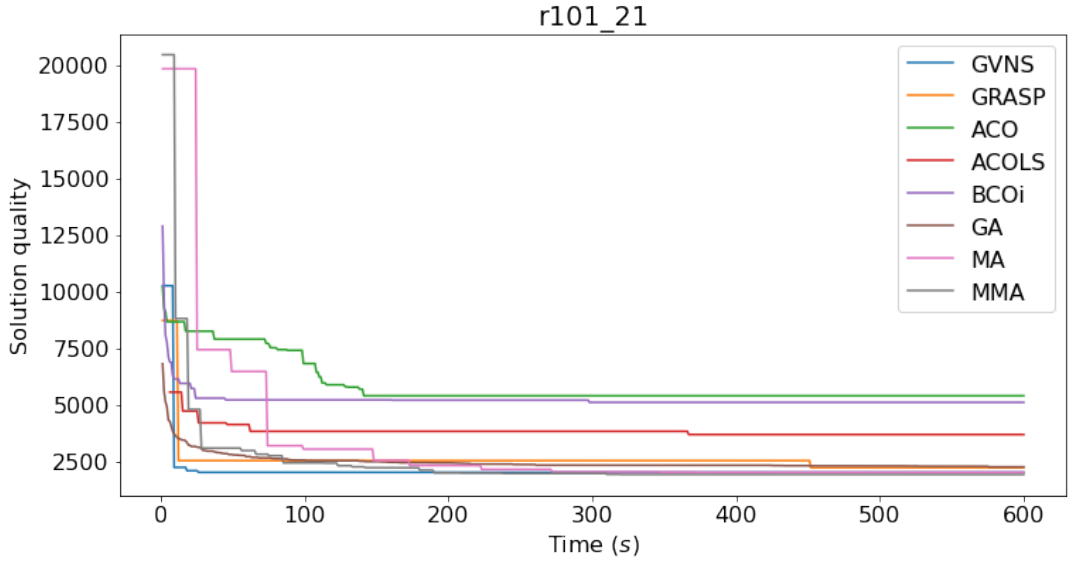

FIGURE 4.49: An illustration depicting the convergence of TD-EVRP-STW algorithms over a 600-second period (Instance: *r101_21*)

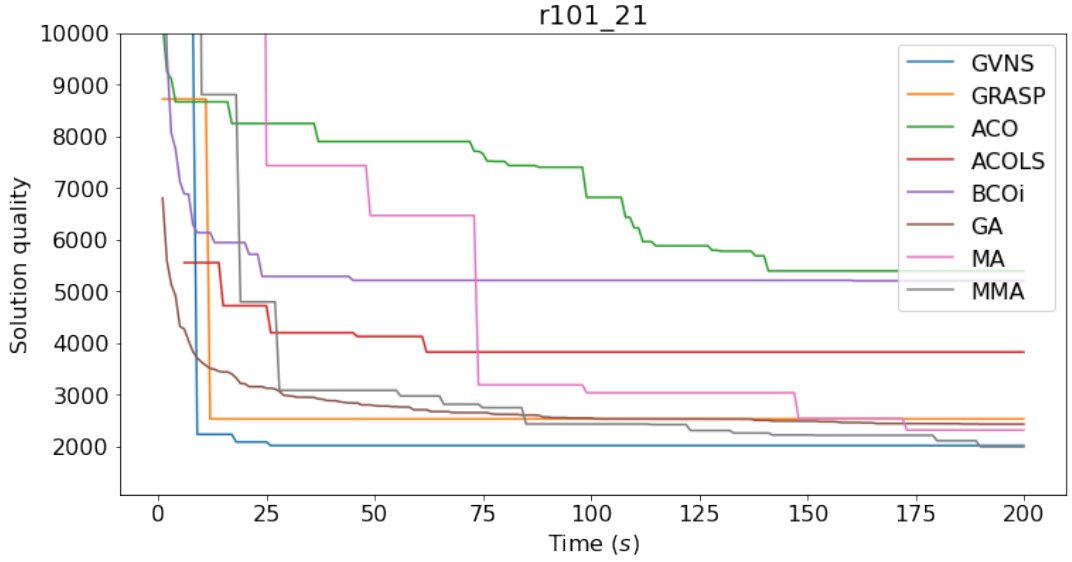

FIGURE 4.50: An illustration depicting the convergence of TD-EVRP-STW algorithms over a 200-second period (Instance: *r101_21*)

Based on our analysis in this section, we can draw several key conclusions. Firstly, for the smaller-sized instances from dataset I_1, no single algorithm demonstrated a clear advantage over the others. However, this does not preclude the selection of certain metaheuristics that frequently outperformed others in specific metrics, particularly when aligned with specific objectives.

Turning our attention to the results from the larger dataset I_2, GVNS and MMA stand out as the leading methods, with GRASP and MA following closely behind. While the difference between performances of these two pairs of algorithms is not definitive, empirical evidence slightly favors GVNS and MMA. GA, on the other hand, delivered moderately inferior results compared to GRASP and MA but was

still generally effective. In contrast, algorithms like ACO, ACOLS, and BCOi did not manage to match the performance level of the aforementioned metaheuristics. In the following two sections, we delve deeper into comparing similar versions of algorithms to better understand their strengths and weaknesses. Specifically, we compare ACO to ACOLS, and MA to MMA.

4.15.5 ACO and ACOLS comparison

In this section, we conduct a comparative analysis of the outcomes achieved by the basic version of ACO against those obtained by the ACOLS on datasets I_1 and I_2.

In the assessment carried out on dataset I_1, ACOLS demonstrated superior performance to ACO in 30 instances, while lagging behind in 6 instances, based on the average solution quality across 30 iterations (See supplementary material `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`). In scenarios where ACOLS outshone ACO, it registered an average enhancement in solution quality of 21.21%, with the variation in performance ranging from a minimal increase of 0.23% to a maximum of 59.15%. Conversely, in the cases where ACO exceeded ACOLS, the average improvement in solution quality reached 82.06%, with the differential spanning from 3.07% to an extensive 427.44%. Notably, this substantial leap in ACO's performance over ACOLS can be largely attributed to an exceptional outlier instance (*c103C5*). To validate the statistical significance of this performance discrepancy, a Wilcoxon signed-rank test was conducted, resulting in a p-value < 0.001, which decisively affirms the statistical relevance of the observed differences between the two algorithms. The disparity in performance between the two algorithms is further illustrated in Figure 4.51. This figure presents a scatter plot where each dot symbolizes an individual input instance, positioned according to its respective results from ACO and ACOLS. A key feature of this plot is the diagonal red dashed line, which serves as a visual benchmark for comparison. Points that fall below this line signify instances where ACOLS achieves superior results, evidenced by lower values compared to those of ACO.

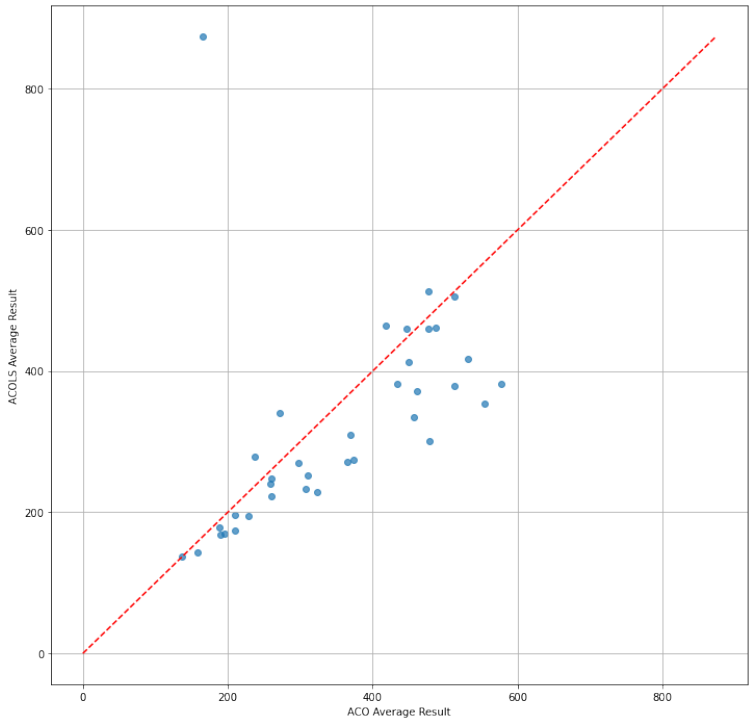

FIGURE 4.51: ACO vs ACOLS performance in terms of average solution quality on instances from I_1

The pattern of ACOLS outperforming ACO is also evident in the analysis of the best-found solutions across all runs for both algorithms. ACOLS delivered superior results in 33 instances, underperformed in 2, and achieved the same best-found solution as ACO once. The average improvement in the best-found solution quality where ACOLS excelled was 22.25%, with the highest and lowest differences being 55.79% and 0.23%, respectively. In scenarios where ACO outshone ACOLS, the greatest difference observed was 19.67%, and the smallest was 8.87%.

Regarding the worst-found solutions, ACOLS again outperformed ACO, albeit by a narrower margin, finding better solutions in 27 instances against 9 instances where it fared worse. Notably, in terms of penalties, ACOLS was less efficient, incurring higher penalties in 17 instances, lower in 5, and matching ACO (often zero) in 14 instances. Moreover, ACOLS tended to result in longer route durations, with an average increase of 30.07% in 32 out of 36 cases. This trend also extended to the longest routes, which were typically longer in ACOLS, possibly explaining the higher average penalties.

In terms of vehicle usage, ACOLS generally favored solutions involving fewer routes, thus leading to longer routes. Specifically, it found solutions with fewer routes in 32 instances (an average reduction of 70.6%), while producing the same number of routes as ACO in 4 instances.

These results are visually represented in Figure 4.52 through scatter plots, illustrating the best-found solutions, worst-found solutions, average route duration, and average route count.

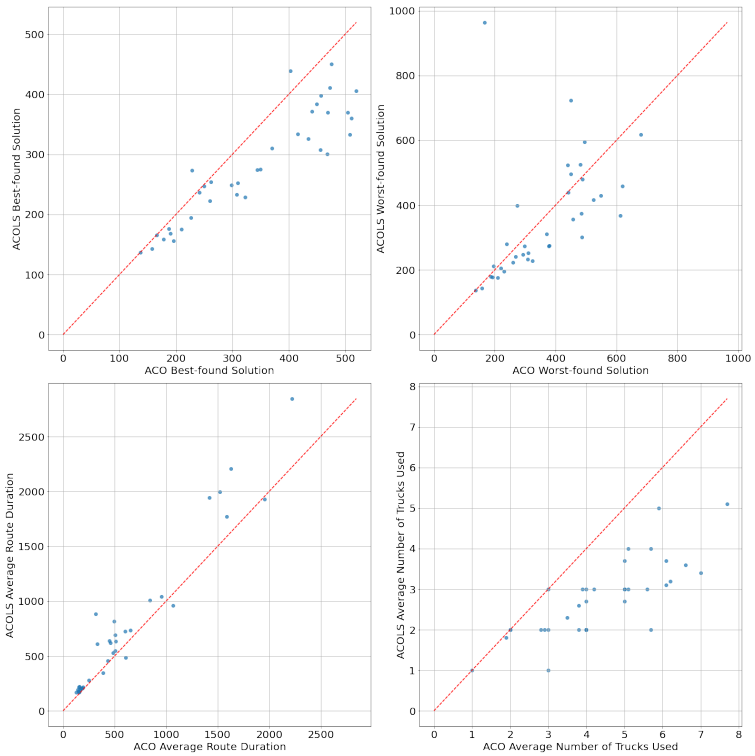

FIGURE 4.52: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from I_1

In the comparison between ACO and ACOLS on the larger instance set I_2, the performance disparity is starkly apparent. ACOLS exceeded ACO in all 56 instances, boasting an average improvement in solution quality of 78.77%, with the range of improvement spanning from 15.98% to 362.11%. This significant difference is evident in Figure 4.53. Further validation through the Wilcoxon signed-rank test confirmed the statistical significance of this disparity, yielding a p-value < 0.0001.

Although the average route duration typically remains longer with ACOLS compared to ACO (in 43 out of 56 cases), ACOLS demonstrates a substantially lower average penalty. Specifically, the penalty for solutions derived from ACO is, on average, 3520.23% higher, reaching up to an astonishing 18645.88% in the most extreme cases.

Figure 4.54 visually presents a comparative analysis of these two algorithms across various parameters, including the best-found solution, the worst-found solution, average route duration, and route count.

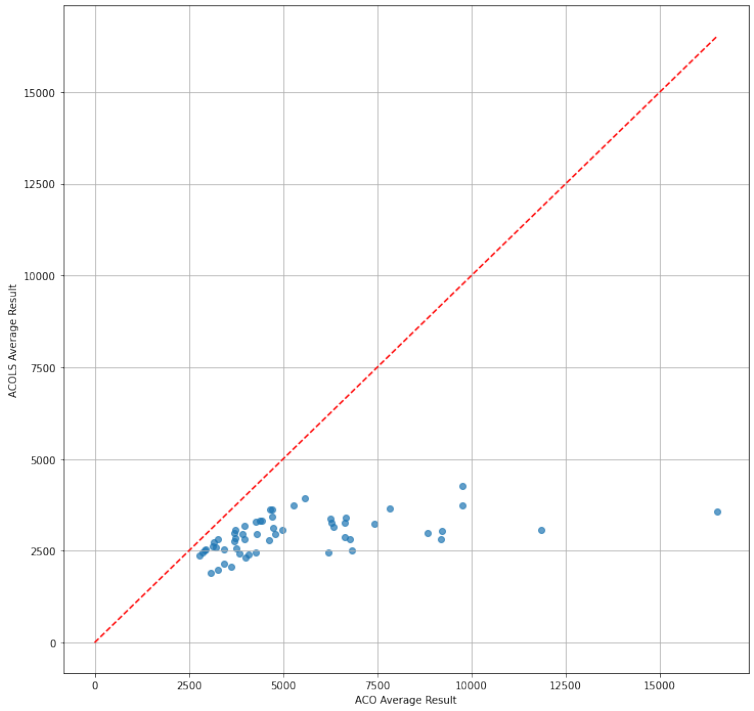

FIGURE 4.53: ACO vs ACOLS performance in terms of average solution quality on instances from I_2

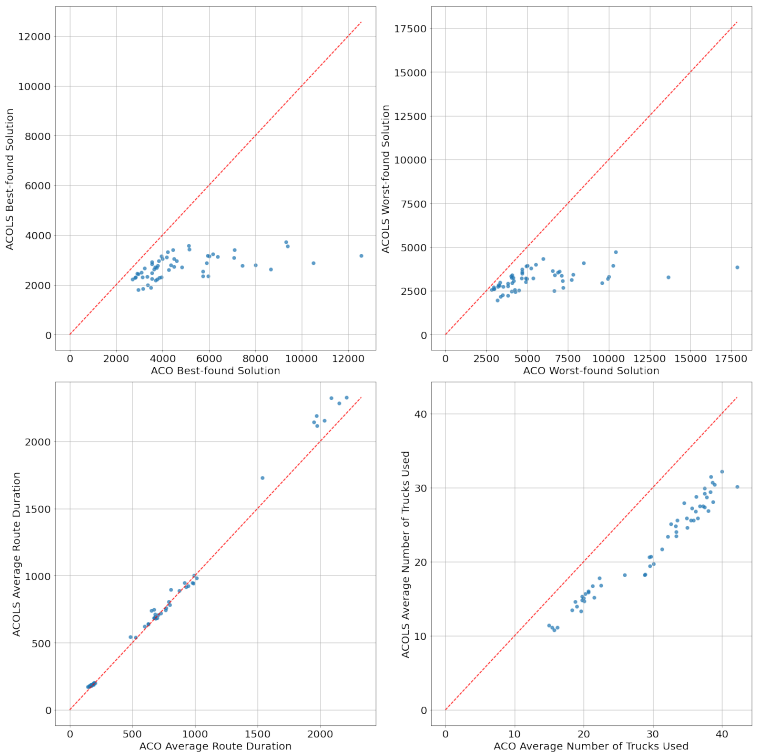

FIGURE 4.54: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from I_2

4.15.6 MA vs MMA comparison

In this section, we conduct a comparative analysis of the MA and MMA results to determine if reducing the frequency of VND calls yields any advantages in terms of solution quality.

In the evaluation using instances from dataset I_1, MA demonstrated slightly superior performance compared to MMA. Specifically, it outperformed MMA in 16 cases (with an average improvement in solution quality of 4.3%, and the minimum and maximum improvements at 0.09% and 15.13%, respectively), underperformed in 8 cases (with an average decline in solution quality of 2.81%, and the minimum and maximum declines at 0.01% and 14.2%, respectively), and achieved identical results in 12 instances. These outcomes are clearly presented in Figure 4.55. Applying the Wilcoxon signed-rank test at a 0.05 significance level, we concluded that the performance difference between these two algorithms is statistically significant in terms of average solution quality, as indicated by a p-value of 0.0397.

In the context of the best-found solution, a statistically significant performance disparity was also observed. On average, the best-found solution quality improved by 11.93% when using MA, with a maximum improvement of 26.83%. However, no statistically significant differences were found in terms of the worst-found solutions, average penalties, route count, or the average and maximum route duration.

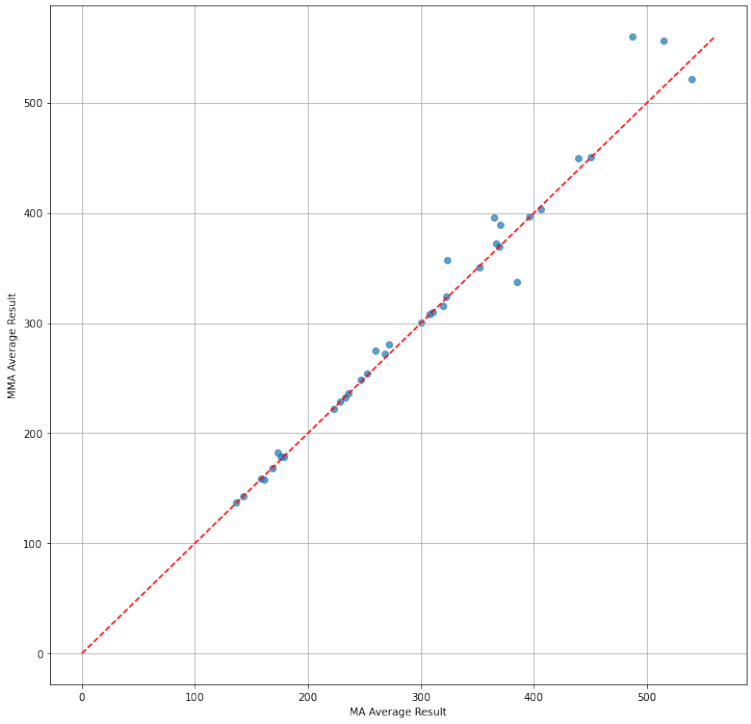

FIGURE 4.55: MA vs MMA performance in terms of average solution quality on instances from I_1

In testing on instances from dataset I_2, MMA generally surpassed MA. Specifically, MMA outperformed MA in 43 instances, achieving an average improvement in solution quality of 6.93%, with the minimum and maximum differences at 0.03% and 25.28%, respectively. Conversely, MMA underperformed in 13 instances, with an average decrease in solution quality of 6.38%, and the minimum and maximum decreases at 1% and 21.45%, respectively. The results for each instance are detailed

in Figure 4.56. Through the Wilcoxon signed-rank test, we confirmed a statistically significant difference in average solution quality between the two algorithms, with a p-value < 0.001.

Regarding the best-found solutions, MMA's superiority over MA was even more pronounced. MMA excelled in 46 instances, with an average, minimum, and maximum improvement in the best-found solution quality of 5.91%, 0.43%, and 23.46%, respectively. Although MMA's worst-found solution was superior in 32 instances compared to MA, the Wilcoxon signed-rank test yielded a p-value of 0.4628. Therefore, at the standard significance level of 0.05, we must reject the hypothesis of a statistically significant difference between the two algorithms in terms of the worst-found solution. Interestingly, the penalties were typically lower in solutions generated by MA (in 42 instances, with 4 instances matching MMA's values), suggesting that while MMA is more effective in minimizing total distance, MA performs better in terms of penalty reduction. Additionally, the average route duration was often shorter in MA's solutions, though this difference was not statistically significant. However, MMA frequently found solutions with a lower route count (in 37 instances). The comparative performance of MA and MMA across different metrics, including the best-found solution, worst-found solution, average route duration, and average route count, is illustrated in Figure 4.57.

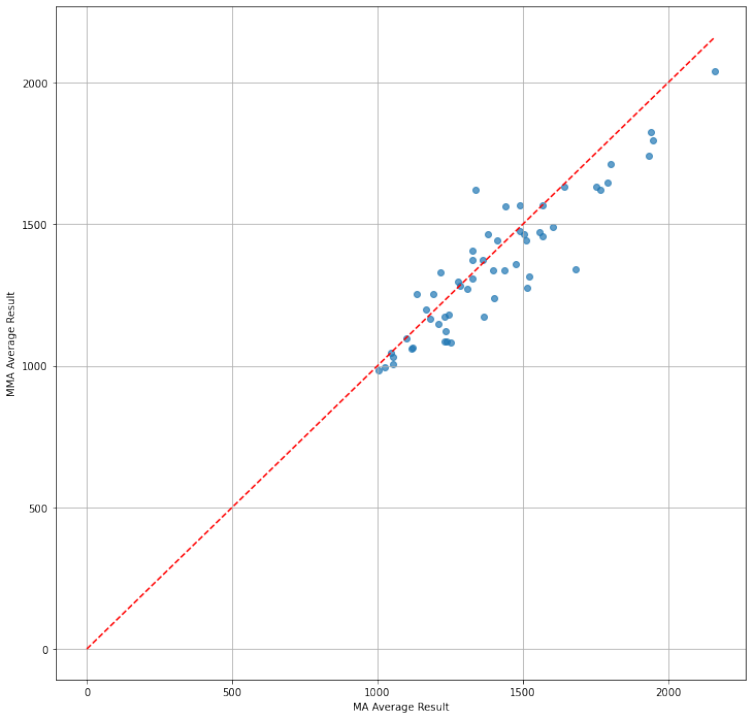

FIGURE 4.56: MA vs MMA performance in terms of average solution quality on instances from I_2

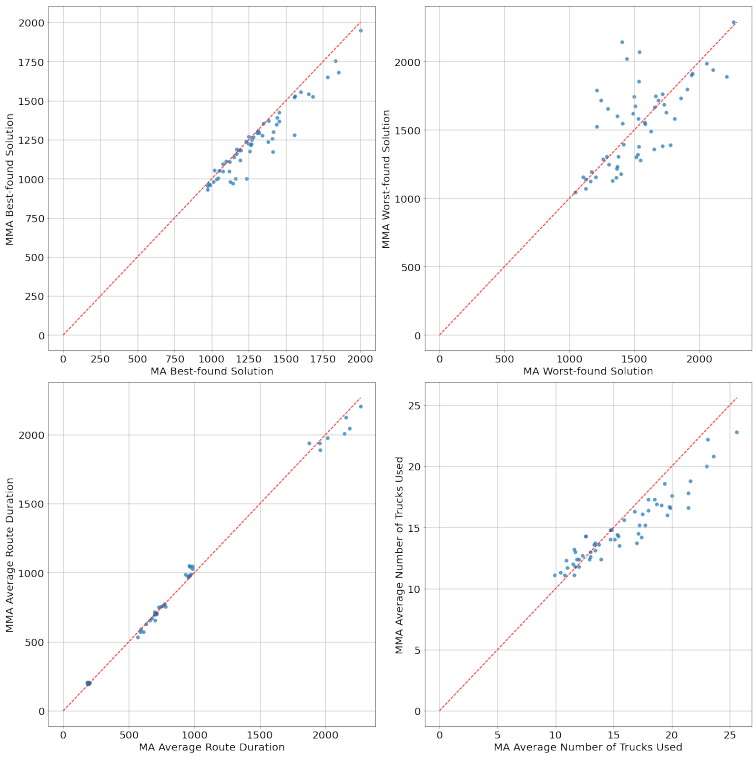

FIGURE 4.57: Comparative scatter plots showcasing various performance metrics for MA and MMA on instances from I_2

4.16 Chapter conclusion

In this chapter, we present our research findings on the Electric Vehicle Routing Problem with soft time windows and time-dependent speeds. We first formulate the problem as a MILP, followed by a detailed presentation of metaheuristic algorithms developed for solving this version of the problem. Our discussion includes algorithms for generating initial solutions, adjusting route feasibility with additional AFS visits, and the structure of these solutions.

Moreover, we elaborate on algorithms designed for calculating and refining vehicle schedules. We delve into the neighborhood structures integral to some of our methods and describe the VND algorithm, which served as our primary local search procedure.

A significant part of our analysis focuses on the hyperparameters of each algorithm, pinpointing those that are most critical. We present the results of our extensive testing, pinpointing the most effective methods for all three GVRP variants. Our analysis includes a statistical examination of the test results across various metrics to identify any significant differences.

To ensure clarity and enhance comprehension, we employ graphical representations alongside our written findings, allowing for a more intuitive understanding of the results and their implications.

CHAPTER
FIVE

ELECTRIC VEHICLE ROUTING PROBLEM WITH PARTIAL RECHARGE

5.1 Problem definition

In this section, we explore a problem closely related to the one discussed in Section 4.1. The primary distinction lies in the battery charging strategy of electric vehicles. In our previously outlined problem, a fundamental assumption was made: every time electric vehicles dock at an AFS, they recharge their batteries to the fullest extent possible. This implies that vehicles are always geared up to their maximum energy potential before setting off again, regardless of their next journey's length.

On the other hand, the current problem variation introduces a more flexible approach to recharging. Instead of a full battery top-up, vehicles are given the liberty to partially recharge based on their immediate energy requirements. This nuanced strategy stems from a practical observation: if vehicles are allowed to recharge only the exact amount of energy they need to comfortably reach their subsequent destination, it could potentially lead to more efficient route planning.

The benefits of this revised approach are multifaceted. For one, it can eliminate the superfluous waiting time at AFSs for vehicles that do not necessarily need a full battery charge. This time-saving can result in optimized route plans, which in turn can have two-fold advantages: firstly, it can abbreviate the overall time taken for deliveries, ensuring goods reach their destination faster. Secondly, with more efficient route planning, vehicles might find it feasible to cater to a larger number of customers, especially tending to those who have specific time windows for delivery. Such a tailored approach could greatly enhance customer satisfaction rates and operational efficiency.

Nevertheless, while this method undeniably offers several advantages, it also brings with it increased challenges to the problem at hand. Specifically, determining the precise amount of energy to recharge every time a vehicle stops at an AFS becomes a significant challenge. This added layer of complexity requires careful consideration and planning, as it necessitates a fine balance between optimizing routes and ensuring that vehicles have sufficient energy for their journeys.

Moving forward, this problem will be known as the *Time-Dependent Partial Recharge Electric Vehicle Routing Problem with Soft Time Windows* (**TDPR-EVRP-STW**).

5.2 Solution representation

The solution structure for TDPR-EVRP-STW, closely mirrors that of TD-EVRP-STW, as detailed in Section 4.4. Yet, in this context, S_{ab} encompasses an additional component:

- S_{ab}^{stay}: In the standard TD-EVRP-STW, the duration a vehicle spends at each location can be readily determined. At customer locations, this duration is predefined and provided within the instance data, whereas at AFS locations, it corresponds to the time required to fully recharge the battery. However, the TDPR-EVRP-STW variant introduces a nuance. While the service time at a customer's location remains predefined, the refueling duration at an AFS can vary from a brief moment up to the full battery recharge time. Consequently, recording the exact time spent at each AFS becomes essential for this problem variant, as it directly influences the remaining battery charge at any given moment.

5.3 Solution feasibility

In order to add AFS visits to a solution, we apply the *FIX_ROUTE* method described in Algorithm 25 to each route of that solution.

The procedure initiates by identifying a point within the route that is beyond the reach of the current battery capacity (as seen at Line 2). Should there be no such point, the solution is considered viable and the procedure is concluded. If, however, such a point is found, the procedure then proceeds into its primary loop (spanning Lines 3-32). Within the main loop of the algorithm, we conduct a reverse examination of all locations, commencing from the identified infeasible spot and moving towards the start of the route. The strategy we employ at each specific waypoint is dependent upon the type of that location, leading us to adopt one of two distinct approaches. When examining a position that corresponds to an AFS, our initial step is to determine the potential additional charge that can be administered at this station, taking into consideration the present state of the battery's charge and the vehicle's maximum battery capacity (referenced in Line 6). If the vehicle is receiving a full recharge at the specified AFS, we must consider integrating an additional AFS into the route (mentioned in Line 8). To prevent a perpetual cycle of adding stations without end, we apply a rule: any new AFS to be incorporated should be nearer to the subsequent point on the route than the current AFS. Should there be no AFS that meets this criterion, the algorithm acknowledges its inability to rectify the route, thereby concluding the process (as stated in Line 10). If the vehicle can receive additional charging at the current AFS (Lines 15-19), we forgo the addition of another AFS and instead extend the charging duration. While calculating the required charge for the vehicle, we factor in the existing battery level and the additional charge necessary to render the route viable. Ideally, our aim is to replenish the battery sufficiently to not only reach the upcoming position but also to add an extra 20% of the battery's capacity. This additional charge enhances the probability that the vehicle will have enough power to reach the subsequent position as well. In instances where this approach is not feasible, we opt to fully charge the vehicle to its maximum battery capacity.

In the alternative case, where the current node is a customer location, our initial step is to identify the AFS nearest to the subsequent position in the route that is accessible with the existing battery charge (Line 21). Subsequently, this AFS

is integrated into the route, and charging times are allocated in accordance with the previously outlined strategy. If a suitable AFS is not available, we backtrack to the preceding stop in the journey with the aim of recalibrating the route. This recalibration may enable us to approach the current customer stop with a greater battery reserve, thereby increasing the likelihood of locating an accessible AFS.

The process continues until there are no infeasible positions in the route, or until the method concludes it can not fix the route in this way. Although this method is not infallible, potentially labeling certain routes as unfeasible when more advanced techniques could rectify them, its simplicity is beneficial given its frequent application. Despite the risk of overlooking some viable solutions, this method is designed to avoid imposing excessive computational demands.

The worst-case time complexity of this algorithm is $\mathcal{O}(n^2)$, where n represents the number of nodes in the route.

Algorithm 25 Procedure for adding visits to AFSs to a route

```
1: procedure FIX_ROUTE(route)
2:     infeasible ← find the position in route that breaks battery level constraint
3:     while ∃ infeasible do
4:         for ∀ location from the infeasible to the start of the route do
5:             if current node is AFS then
6:                 canBeRecharged ← Calculate potential recharge
7:                 if canBeRecharged = 0 then
8:                     Find the best AFS to add
9:                     if No AFS is found then
10:                        return Failure to fix the route
11:                    end if
12:                    Determine how much to charge
13:                    Insert AFS into route with calculated charging time
14:                    break
15:                else
16:                    Determine how much to charge
17:                    Update charging time of the current node
18:                    break
19:                end if
20:            else                                  ▷ Current node is customer
21:                afs ← find closest AFS to current customer
22:                Calculate energy needed to reach afs
23:                if battery is insufficient to reach afs then
24:                    continue
25:                end if
26:                Determine required recharge amount at the afs
27:                Insert AFS into route with calculated charging time
28:                break
29:            end if
30:        end for
31:        Update infeasible for next iteration
32:    end while
33:    return route
34: end procedure
```

5.4 Metaheuristics

In addressing the TDPR-EVRP-STW, we examined the foundational algorithms previously introduced in Section 4. While the core principles of the algorithms remain largely consistent with the foundational work we have presented, they are adapted to tackle the unique challenges presented by the TDPR-EVRP-STW. This complex problem demands a nuanced approach to solution representation and the strategic insertion of AFS into existing routes. This distinction in solution representation, which is previously elaborated in Section 5.2, involves a more intricate data structure that allows for dynamic adjustments in vehicle recharge times to accommodate for the PR attribute of this problem.

Additionally, the insertion of AFS is executed with a tailored method, described in Section 5.3. This method is not merely a matter of adding stops for recharging but requires a careful consideration of the vehicle's remaining charge to ensure that the addition of any AFS is both logical and practical within the route's timeline.

Unless we specify additional modifications, it is safe to assume that the algorithms used for the TDPR-EVRP-STW are variations of those we described earlier, modified only in their solution representation and the AFS integration method.

One of the methods that require some additional modification is our ACO algorithm and by extension ACOLS. More precisely, in the solution construction phase, after an AFS is chosen to be added to the partial solution, we need to determine the charging time at that AFS. This is done by utilizing the *CALCULATE_STATION_STAY* method, outlined in Algorithm 26.

In this method, we start by identifying the final node within the route (refer to Line 2). Subsequently, we compute the remaining battery charge at this node (see Line 3), followed by an estimation of the energy required to travel from this node to the specified AFS (Line 4). Should the current battery charge be insufficient to reach the targeted AFS, the function will yield a designated error return indicator (as noted in Line 6).

The collection of unvisited customers signifies those who have yet to be included in the current solution. In scenarios where this collection is empty, the strategy shifts to recharging the vehicle minimally, ensuring it possesses just enough power to safely return to the depot (addressed in Lines 8-11).

Conversely, when there are customers yet to be visited, the strategy adapts. We look for a customer positioned beyond the specified AFS, one that lies within the range of the current battery capacity. Additionally, we aim to identify another AFS that is reachable post-visit to this customer with the available battery charge (Line 13). The goal is to charge the battery sufficiently to ensure travel to this next AFS is possible. This guarantees the presence of at least one subsequent reachable point in the route, thereby maintaining the feasibility of the proposed solution.

In the GA, MA, and MMA, we employ a nuanced variation in chromosome representation compared to what is described in Section 4.14. Rather than using a simple vector of integers to signify the indices of nodes, chromosomes are encoded as a sequence of pairs; the first component of each pair signifies the node index, and the second denotes the duration of stay at that node. Apart from this distinction in chromosome structure, all other elements of these algorithms are consistent with the original description.

Algorithm 26 Procedure for calculating recharge time at an AFS

```
1: procedure CALCULATE_STATION_STAY(route, afs, unvisited_customers)
2:     node ← get the last node in route
3:     battery ← calculate battery level at node
4:     cost ← calculate energy consumption when traveling from node to afs
5:     if (battery − cost) < 0 then
6:         return −1                                  ▷ Predefined failure indicator
7:     end if
8:     if size(unvisited_customers) = 0 then
9:         consumption ← calculate consumption between afs and depot node
10:        return (consumption − (battery − cost))/REFUELING_RATE
11:    end if
12:    for ∀customer ∈ unvisited_customers do
13:        station ← find a station that can be visited after customer
14:        if no station can be visited after customer then
15:            continue
16:        end if
17:        C ← calculate consumption between afs and station through customer
18:        return (C − (battery − cost))/REFUELING_RATE
19:    end for
20: end procedure
```

5.5 Experimental evaluation

5.5.1 Experimental setup

The experimental framework for this specific version of the problem mirrored the setup outlined in Section 4.15.2. Although, in theory, the optimal hyperparameter values for this version could differ from those identified as optimal for TD-EVRP-STW, our empirical analysis indicated that the hyperparameter values detailed in Section 4.15.3 were also effective in this context. Consequently, these values were retained for all subsequent testing phases.

5.5.2 Overall comparison

In Table 5.1, we showcase a comparative analysis of the average solution quality, derived from thirty iterations of each algorithm applied to instances from dataset I_1. The values highlighted in bold indicate the superior average solution achieved across all evaluated algorithms. The comprehensive results for all algorithms, encompassing all metrics, are detailed in tables available for download at `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`.

To assess if the performance differences between our algorithms are statistically significant, we initially verified their eligibility for ANOVA by employing the Shapiro-Wilk and Levene's tests. These tests aimed to check the normality of each algorithm's results and the homogeneity of variances across them. However, since the prerequisites for ANOVA were not fulfilled, we opted for the Kruskal-Wallis test. This test yielded a p-value 0.01, leading us to confirm the presence of significant performance variations among the algorithms.

TABLE 5.1: Results for all TDPR-EVRP-STW metaheuristics on the instance set I_1.

Instance	**C**	**A**	**Metaheuristics**							
			GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
c101C10	10	5	**374.95**	367.79	595.62	405.82	473.23	411.66	511.83	382.99
c101C5	5	3	263.59	236.34	290.6	**234.82**	249.44	243.28	532.7	**234.82**
c103C15	15	5	367.68	414.28	818.52	**365.89**	609.52	416.66	487.32	383.93
c103C5	5	2	165.67	635.75	170.79	557.4	167.17	**161.26**	1000.37	**161.26**
c104C10	10	4	290.56	298.12	425.57	332.07	317.38	279.33	**263.9**	268.71
c106C15	15	3	**267.81**	281.28	676.13	286.83	491.38	328.92	301.19	384.76
c202C10	10	5	265.48	264.4	485.29	244.88	339.92	257.98	**239.88**	248.66
c202C15	15	5	421.53	387.17	845.88	409.18	627.95	405.06	**377.08**	384.44
c205C10	10	3	247.25	228.28	453.8	241.25	348.49	267.83	235.63	**228.27**
c206C5	5	4	227.48	233.94	278.83	**219.34**	243.37	239.05	423.8	**219.34**
c208C15	15	4	309.82	315.31	644.87	382.69	594.67	315.45	**297.87**	306.39
c208C5	5	3	158.48	205	216.54	197.11	203.73	158.48	**158.37**	160.44
r102C10	10	4	296.71	288.4	387.65	291.43	308.27	**262.93**	273.84	292.84
r102C15	15	8	555.18	569.14	722.06	459.05	532.13	**429.24**	502.81	447.89
r103C10	10	3	**158.16**	160.02	315.58	176.86	213.02	201.46	160.52	177.21
r104C5	5	3	**136.69**	**136.69**	140.23	140.93	136.72	136.59	139.31	**136.69**
r105C15	15	6	376.52	430.97	764.88	386.79	501.64	378.63	450.32	**340.56**
r105C5	5	3	175.19	224.1	207.14	173.11	177.83	**169.31**	184.58	171.42
r201C10	10	4	220	**205.86**	358.72	247.47	272.5	246.55	223.69	215.02
r202C15	15	6	364.19	371.26	709.52	389.93	568.78	401.42	**347.15**	349.1
r202C5	5	3	141.03	157.5	166.98	141.22	142.01	140.49	143.92	**128.78**
r203C10	10	5	259.27	262.42	400.18	279.26	285.65	241.67	**230.63**	232.05
r203C5	5	4	197.66	200.73	237.47	194.05	199.83	196.18	183.88	**179**
r209C15	15	5	298.38	301.03	689.45	359.86	476.44	360.7	**285.2**	291.22
rc102C10	10	4	436.82	436.98	517.41	429.89	464.6	**428.46**	495.6	430.4
rc103C15	15	5	469.38	520.42	640.15	452.19	584.77	**400.65**	488.73	446.69
rc105C5	5	4	257.81	287.14	256.91	263.08	243.15	**239.04**	275.52	268.17
rc108C10	10	4	474.38	**394.27**	446.85	432.17	428.03	394.7	535.13	529.68
rc108C15	10	5	423.42	400.32	748.47	466.96	614.64	455.62	482.81	**396.97**
rc108C5	5	4	366.82	344.44	315.15	625.95	300.01	300.27	**293.75**	321.61
rc201C10	10	4	308.56	304.39	468.66	308.93	375.79	309.6	442.25	**304.33**
rc202C15	15	5	**415.1**	419.38	771.45	463.37	643.71	437.51	426.77	426.13
rc204C15	15	7	371.12	372.98	655.91	405.24	433.11	362.02	337.96	**326.49**
rc204C5	5	4	183.72	184.13	191.2	227.68	180.06	175.71	220.53	**172.31**
rc205C10	10	4	**346.14**	354.68	490.82	374.45	395.65	363.39	540.04	360.61
rc208C5	5	3	181.24	196.55	202.44	170.71	178.64	167.98	**162.81**	167.97
Count best			4	3	0	3	0	8	10	11

TABLE 5.2: Performance comparison of TDPR-EVRP-STW metaheuristics across various metrics (Dataset I_1).

Metric	Method							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality	4	3	0	3	0	8	10	**11**
Best-found solution	6	7	0	3	1	2	**28**	6
Worst-found solution	6	4	0	2	1	**12**	8	**12**
Penalty	19	15	13	12	21	**31**	6	17
Average route duration	3	4	11	0	**17**	0	0	1
Maximum route duration	**18**	17	4	14	8	**18**	12	13
Number of routes	11	0	0	2	0	4	**31**	11

Subsequently, we conducted Mann-Whitney U tests, complemented by FDR correction, to pinpoint specific algorithms that exhibited distinct performances. Figure 5.1 displays these findings. Interestingly, while most algorithms showed similar effectiveness, the ACO stood out, demonstrating notable differences in performance when compared to the majority, except for BCOi and MA. Notably, the ACO algorithm frequently delivered the worst average solution quality, underperforming in 26 out of 36 instances. However, its distinct performance did not emerge as a particularly remarkable aspect of our study.

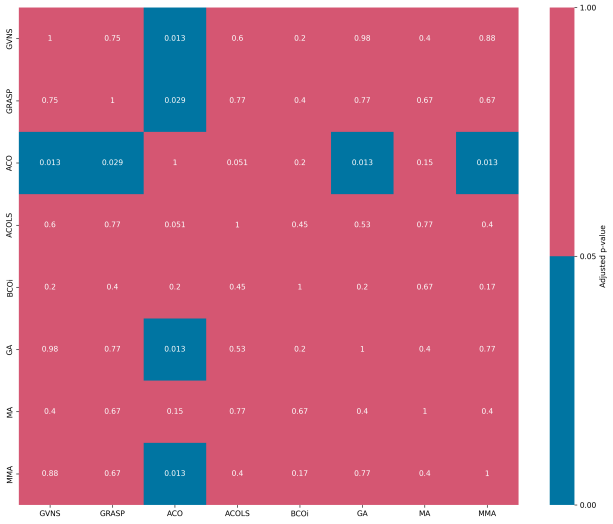

FIGURE 5.1: Heatmap of pairwise comparisons of TDPR-EVRP-STW algorithms for I_1 dataset (Mann-Whitney U test with FDR correction)

In Table 5.2, we offer a detailed breakdown of our algorithms' performance across various metrics. For each metric, the table indicates the number of instances in which a specific algorithm surpassed all others in effectiveness.

In Table 5.3, we showcase the outcomes of the proposed algorithms, focusing on the average solution quality achieved for instances within dataset I_2.

Given that the data outlined in Table 5.3 did not satisfy the criteria for conducting ANOVA, we opted for the Kruskal-Wallis test instead. The test yielded a p-value $<$ 0.0001, leading us to conclude that there is a statistically significant difference among

Table 5.3: Results for TDPR-EVRP-STW obtained on the instance set I_2.

Instance	C	A	Value							
			GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
c101_21	100	21	1725.4	1967.57	18404.84	5884.31	7508.52	2286.64	1968.39	**1425.37**
c102_21	100	21	1596.69	1690.59	13079.81	5168.16	5676.94	2115.56	1766.97	**1301.88**
c103_21	100	21	1555.91	1633.22	8799.49	4093.41	4785.92	1954.08	1758.56	**1273.47**
c104_21	100	21	1450.49	1523.92	5123.53	3594.52	4324.33	1702.52	1328.32	**1174.17**
c105_21	100	21	1494.98	1779.02	16226.69	5195.15	6694.43	2088.6	1983.55	**1452.56**
c106_21	100	21	1507.06	1702.55	14228.96	5027.91	5996.66	2040.51	1886.75	**1381.99**
c107_21	100	21	1484.23	1683.96	14087.29	5131.79	6028.31	1962.65	1743.57	**1303.6**
c108_21	100	21	1564.1	1629.69	11612.77	4985.79	5462.89	1872.58	1671.02	**1268.24**
c109_21	100	21	1355.7	1542.92	8650.05	4279.54	4965.35	1786.83	1805.17	**1164.75**
c201_21	100	21	1423.71	1463.71	23990.48	4874.2	5870.55	1704.62	1746.81	**1123.97**
c202_21	100	21	1386.15	1410.59	13470.17	4602.75	4782.19	1543.81	1671.93	**1099.12**
c203_21	100	21	1265.13	1376.38	7676.74	3770.33	4512.77	1567.53	1521.1	**1055.16**
c204_21	100	21	1238.87	1277.31	4792.96	3502.36	3926.94	1390.15	1188.97	**1049.69**
c205_21	100	21	1398.87	1440.62	20394.67	4723.4	5290.35	1539.26	1676.97	**1196.42**
c206_21	100	21	1452.98	1454.76	15463	4944.68	4727.31	1402.26	1634.4	**1155.25**
c207_21	100	21	1387.47	1390.33	12643.52	4288.43	4647.45	1426.71	1598.87	**1076.9**
c208_21	100	21	1439.81	2249.11	11227.08	4446.96	4685.66	1386.74	1584.42	**1064.99**
r101_21	100	21	2481.16	2390.91	7634.29	4399.03	5316	2478.94	2478.03	**1923.23**
r102_21	100	21	2356.55	2038.19	6120.34	4056.38	4691.6	2295.95	2320.75	**1800.77**
r103_21	100	21	1804.71	1812.42	5047.06	3540.02	4172	2104.93	2177.99	**1609.85**
r104_21	100	21	1543.97	1580.9	3934.73	3103.28	3814.47	1860.78	1724.71	**1375.44**
r105_21	100	21	1980.19	1986.01	6347.82	3849.73	4743.34	2300.66	2308.82	**1788.34**
r106_21	100	21	1755.86	1820.25	5378.69	3679.5	4360.41	2125.48	2175.73	**1631.86**
r107_21	100	21	1622.3	1667.12	4367.68	3171.71	3939.92	1960.04	1974.88	**1464.48**
r108_21	100	21	1659.99	1567.41	3777.63	3148.2	3842.15	1803.75	1896.34	**1333.39**
r109_21	100	21	1633.74	1686.55	4785.61	3265.71	4165.72	2017.84	2125.52	**1524.24**
r110_21	100	21	1547.18	1570.44	3930.24	3068.24	3763.82	1887.01	1971.54	**1395.67**
r111_21	100	21	1579.15	1591.72	4002.72	3263.61	3794.43	1905.17	1960.6	**1358.38**
r112_21	100	21	1566.66	1517.13	3430	2996.42	3609.79	1801.97	1876.42	**1337.33**
r201_21	100	21	**1354.63**	1460.73	9109.1	3618.1	3945.04	1514.42	1617.57	1476.82
r202_21	100	21	**1197.12**	1368.33	7293.41	3451.05	3691.79	1403.09	1386.42	1644.51
r203_21	100	21	**1141.68**	1250.58	5644.23	3099.28	3486.49	1324.14	1358.04	1179.42
r204_21	100	21	**977.72**	1142.21	4344.18	2577.75	3091.49	1251.32	1049.85	1168.16
r205_21	100	21	**1251.36**	1341.63	7204.62	3406.71	3704.34	1373.46	1416.86	1321.24
r206_21	100	21	**1158.78**	1284.37	6530.5	3175.31	3556.3	1363.17	1472.6	1283.62
r207_21	100	21	**995.39**	1219.77	5426.24	2916.46	3279.66	1255.59	1137.23	1099.87
r208_21	100	21	**969.62**	1149.34	3906.2	2528.71	2932.52	1230.26	1097.95	1003.38
r209_21	100	21	**1155.3**	1275.85	6196.48	3003.66	3500.81	1321.15	1370.6	1248.08
r210_21	100	21	**1074.74**	1257.83	6044.9	3036.6	3448.61	1307.52	1276.63	1292.63
r211_21	100	21	**1016.09**	1215.59	4728.86	2821.15	2989.32	1252.71	1188.34	1026.7
rc101_21	100	21	**2129.64**	2398.53	7120.9	5032.09	6183.92	2757.4	2780.47	2192.01
rc102_21	100	21	**2013.97**	2215.32	6041.88	4618.89	5834.21	2573.19	2464.05	2059.87
rc103_21	100	21	1874.88	2079.65	5449.87	4240.15	5610.2	2316.86	2430.5	**1761.22**
rc104_21	100	21	1827.82	1984.4	4667.72	3994.76	5247.86	2158.96	2009.18	**1609.19**
rc105_21	100	21	2108.15	2246.12	6004.72	4561.53	5877.65	2530.66	2594.22	**1909.58**
rc106_21	100	21	2014.35	2104.15	5789.62	4462.33	5785.44	2360.43	2577.03	**1789.5**
rc107_21	100	21	1725.66	2022.34	4728.82	3917.8	5211.24	2356.66	2187.47	**1617.22**
rc108_21	100	21	1795.84	1967.27	4442.52	3859.3	5095.93	2185.75	2204.76	**1634.98**
rc201_21	100	21	**1620.68**	1741.42	11604.5	5043.25	5233.7	1846.1	2051.55	1879.23
rc202_21	100	21	**1462.91**	1629.89	9207.9	4429.39	4745.96	1662.07	1718.93	1666.89
rc203_21	100	21	**1305.03**	1472.08	7290.6	3806.97	4421.27	1581.25	1588.93	1462.24
rc204_21	100	21	1373.49	1394.24	4686.12	3514.07	3915.11	1473.11	1234.75	**1362.74**
rc205_21	100	21	**1490.25**	1618.61	9054.15	4332.52	4785.62	1678.2	1859.9	1618.51
rc206_21	100	21	**1515.85**	1578.51	8771.81	4360.65	4754.36	1640.36	1707.7	1557.2
rc207_21	100	21	**1383.18**	1472.28	6873.72	3714.88	4305.31	1556.92	1586.6	1391.57
rc208_21	100	21	1360.65	1450.69	4510.55	3502.27	3711.72	1439.74	1452.4	**1169.12**
Count best			19	0	0	0	0	0	1	36

the algorithms. To pinpoint the specific algorithms that differ from one another, we conducted pairwise Mann-Whitney U tests with FDR correction for each algorithm pair. The outcomes of these tests are illustrated in Figure 5.2.

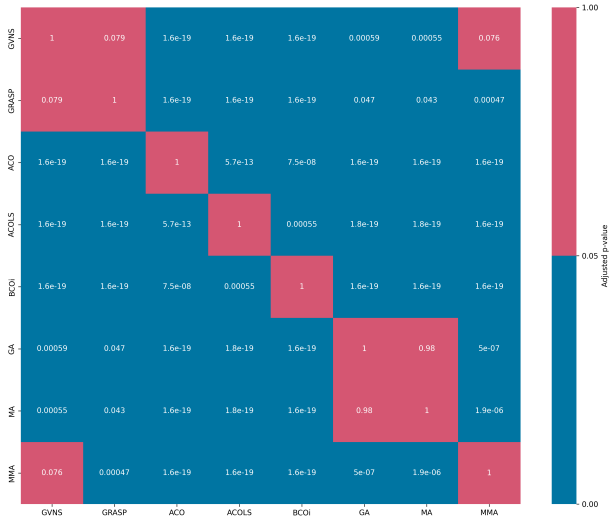

FIGURE 5.2: Heatmap of pairwise comparisons of TDPR-EVRP-STW algorithms for I_2 dataset (Mann-Whitney U test with FDR correction)

As observed from the figure, nearly all algorithms exhibit statistically significant performance differences when compared to their counterparts. The notable exception is the pair of GA and MA, which demonstrated similar levels of performance, setting them apart from the general trend identified among the other algorithms.

In Figure 5.3, we showcase a boxplot that illustrates the average solution quality performance of each algorithm over all instances from I_2. Complementing this, Figure 5.4 features a line plot where each algorithm is represented by a distinct line, tracing the average solution quality achieved in each instance. From the boxplot in Figure 5.3, it is evident that ACO, ACOLS, and BCOi underperformed relative to their counterparts. To facilitate a more nuanced comparison among the higher-performing algorithms, we have excluded these three from our analysis, as demonstrated in Figure 5.5.

The refined comparison in Figure 5.5 reveals that MMA exhibits the lowest median value, suggesting its superiority in tackling the TDPR-EVRP-STW. Consequently, MMA emerges as a highly recommended choice for this specific application.

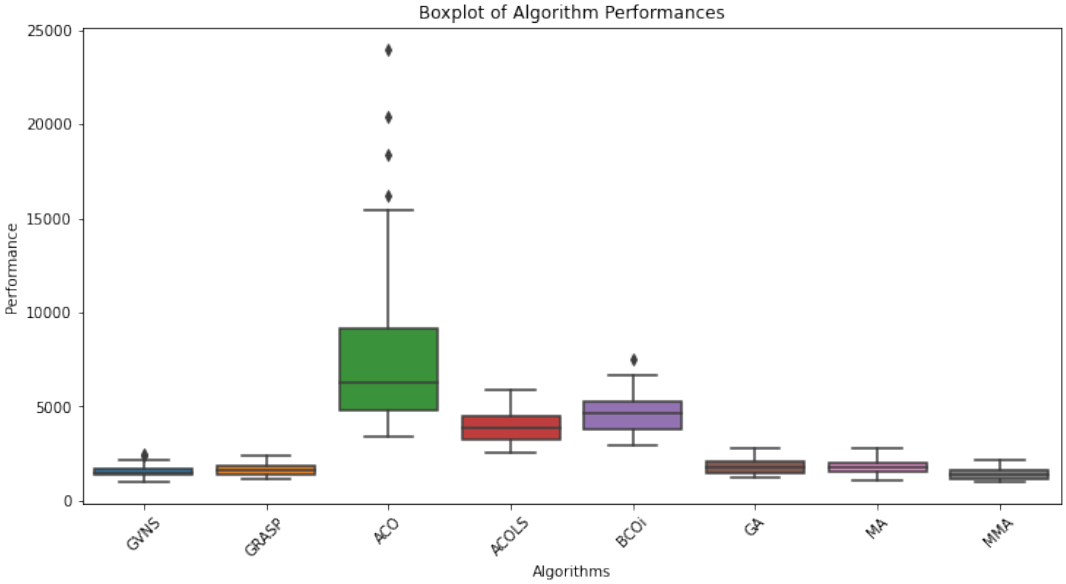

FIGURE 5.3: Boxplot of Algorithm Performances

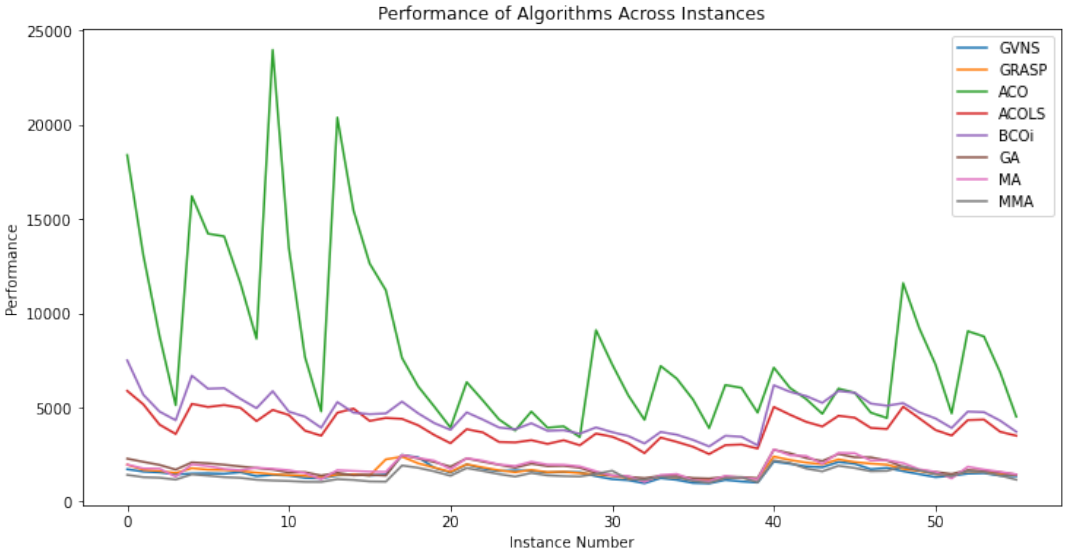

FIGURE 5.4: Performance of Algorithms Across Instances

TABLE 5.4: Performance comparison of TDPR-EVRP-STW metaheuristics across various metrics (Dataset I_2).

Metric	Method							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality	19	0	0	0	0	0	1	**36**
Best-found solution	9	0	0	0	0	0	1	**46**
Worst-found solution	18	6	0	0	0	1	1	**30**
Penalty	6	17	0	0	0	9	**34**	11
Average route duration	0	0	0	1	**31**	0	24	0
Maximum route duration	**18**	9	0	0	1	3	16	11
Number of routes	8	2	0	0	0	0	0	**47**

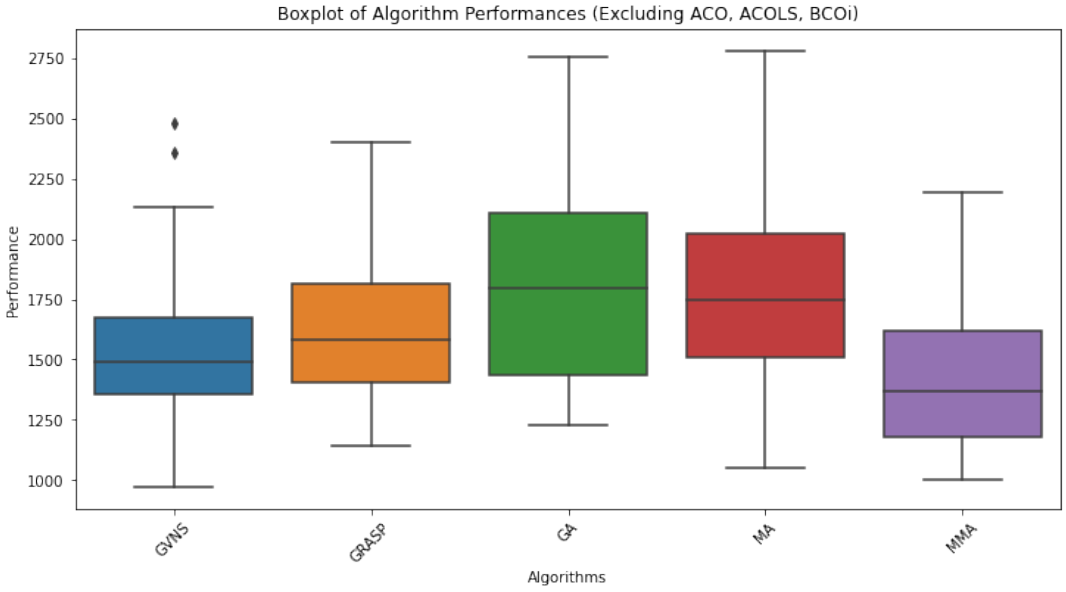

FIGURE 5.5: Boxplot of Algorithm Performances (Excluding ACO, ACOLS, BCOi)

In Table 5.4, we present an analysis of the performance of our algorithms across a range of key metrics. For each metric, the table enumerates the instances where a particular algorithm outperformed its counterparts, thereby highlighting its relative effectiveness. This comprehensive overview provides valuable insights into the strengths of each algorithm within different performance parameters.

Similar to the previous version of the problem, here we focus on examining the convergence speed of various algorithms. Consistent with our earlier observations, the most significant advancements occur within the first 200 seconds. To effectively present our findings for this critical time frame, we have chosen to limit the y-axis to 10000. This approach allows for a clearer visual distinction between the methods. Our analysis is based on the results from the *r101_21* instance, as depicted in Figure 5.6.

From this figure, we observe that GVNS achieves rapid convergence. However, in this scenario, MMA demonstrates even better performance, matching the solution quality of GVNS in approximately 100 seconds and subsequently surpassing it. This aligns with our previous findings where MMA generally outperformed other methods, reinforcing its efficiency and effectiveness in solving the problem at hand.

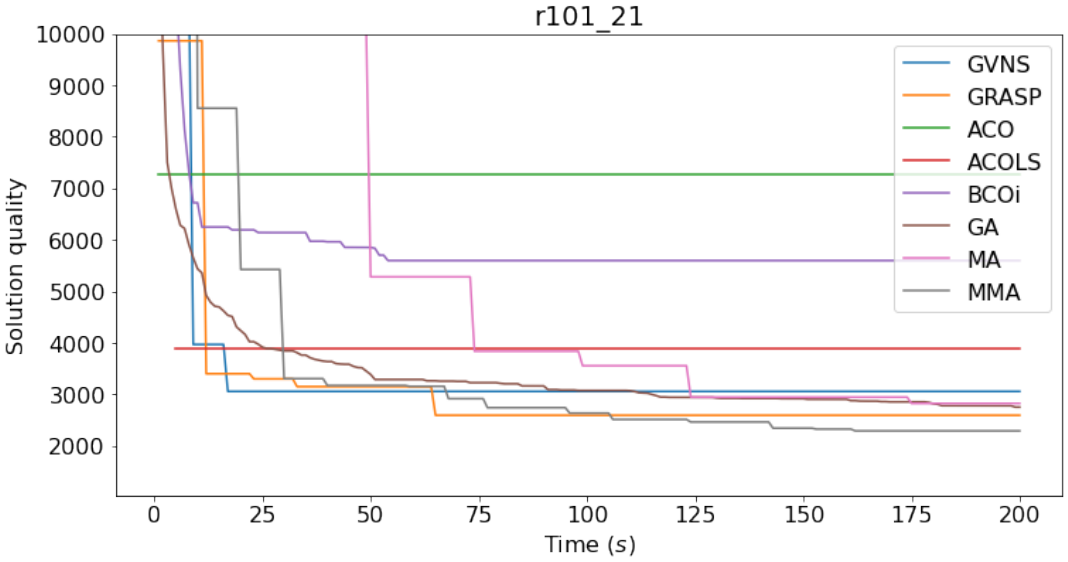

FIGURE 5.6: An illustration of TDPR-EVRP-STW algorithms convergence over 200s (Instance *r101_21*)

In conclusion, our assessment of algorithms for the TDPR-EVRP-STW yields significant insights. Upon evaluation on the smaller instance set I_1, no algorithm distinctly outperformed the others, as most demonstrated comparable efficacy, with the notable exception of ACO, which lagged in performance. However, when applied to the larger instance set I_2, MMA distinguished itself as the most effective algorithm for this specific problem, underscoring its suitability for more complex scenarios. As with the previous problem, we now examine ACO and ACOLS in greater detail, as well as compare MA and MMA.

5.5.3 ACO and ACOLS comparison

Mirroring the approach taken for TD-EVRP-STW, we now proceed to compare the performance of the ACO and ACOLS algorithms.

In the comparison using dataset I_1, ACOLS surpassed ACO in 31 instances, with the differences in solution quality averaging at 54.85%, and ranging from a minimum of 3.4% to a maximum of 135.72%. Conversely, ACO excelled over ACOLS in 5 instances, where the differences in solution quality averaged 69.39%, with a minimum of 0.5% and a maximum of 226.37%. Figure 5.7 graphically represents the average solution quality achieved for each instance (indicated by dots), where dots below the line signify instances where ACOLS was superior to ACO, and those above the line indicate the opposite. The results of the Wilcoxon signed-rank test reveal a statistically significant difference in average solution quality between the two algorithms, evidenced by a p-value < 0.0001.

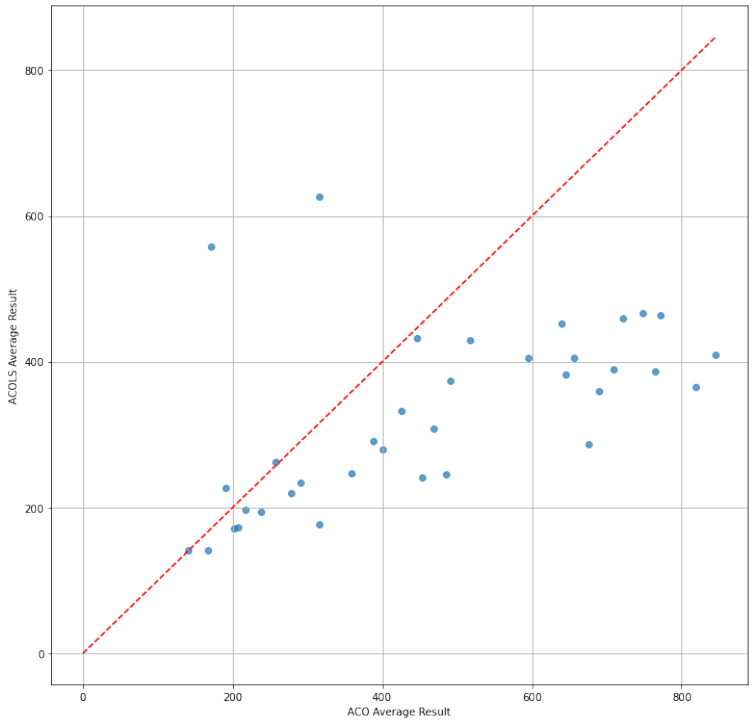

FIGURE 5.7: ACO vs ACOLS performance in terms of average solution quality on instances from I_1

Regarding the best-found solutions, ACOLS outshined ACO in 34 instances (with average, minimum, and maximum differences in solution quality at 44.81%, 0.87%, and 116.99%, respectively) and matched ACO's performance in one instance. In the sole instance where ACO had a better best-found solution than ACOLS, the difference was marginal, at approximately 0.85%. Additionally, ACOLS was superior in terms of the worst-found solution in 28 instances, while it lagged behind in 8 instances.

As with the TD-EVRP-STW version of the problem, penalties, average and maximum route duration were lower in solutions produced by ACO. However, ACOLS tended to yield solutions with fewer routes. The detailed results for each instance, covering best-found solutions, worst-found solutions, average route duration, and route counts, are showcased in Figure 5.8.

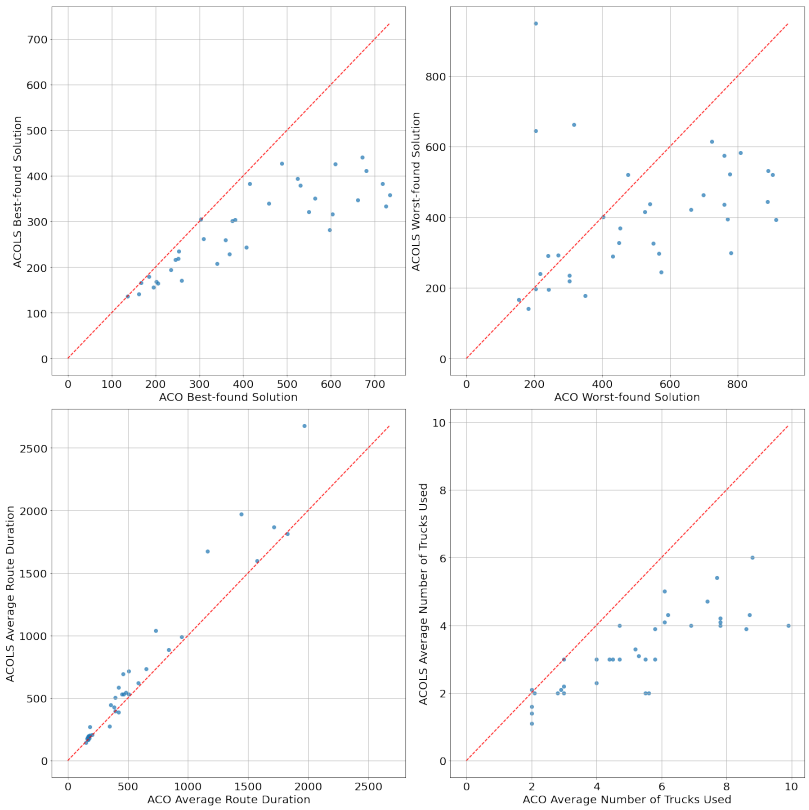

FIGURE 5.8: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from I_1

In the analysis conducted on dataset I_2, ACOLS consistently outperformed ACO in all 56 instances. The average improvement in solution quality was a remarkable 95.82%, with the minimum and maximum differences being 14.47% and 392.19%, respectively. Figure 5.9 visually details the average solution quality for each instance.

Moreover, ACOLS demonstrated a significant edge over ACO across all 56 instances in terms of best-found solutions, worst-found solutions, average penalties, and route count. While there was no statistically significant difference in average route duration between the two algorithms, ACOLS surpassed ACO in terms of maximum route duration in 49 out of the 56 cases. Figure 5.10 illustrates the results for these various metrics, providing a comprehensive comparison between ACOLS and ACO.

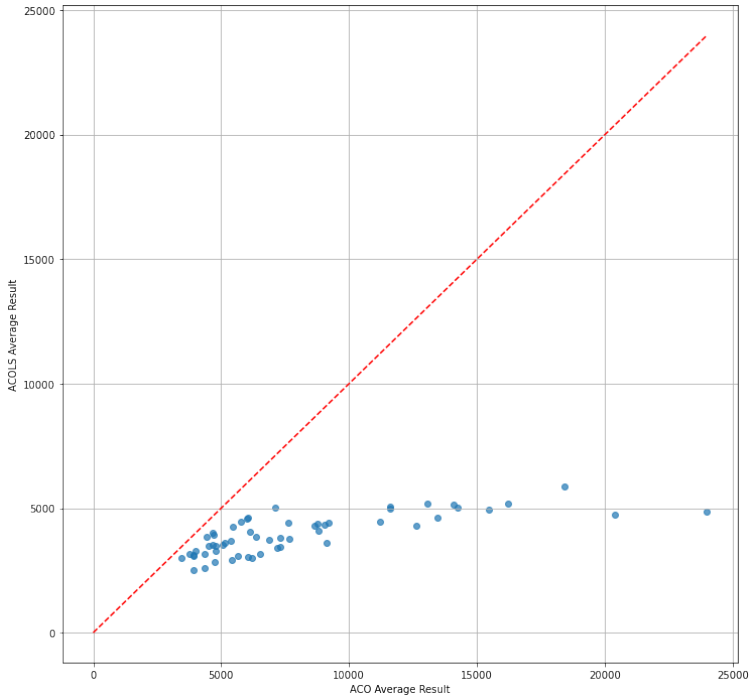

FIGURE 5.9: ACO vs ACOLS performance in terms of average solution quality on instances from I_2

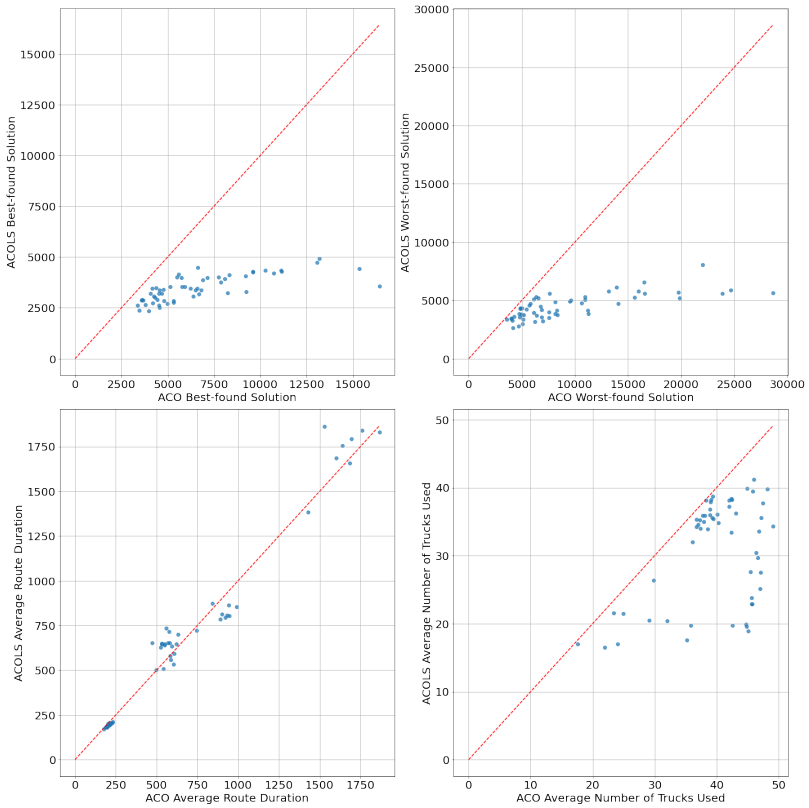

FIGURE 5.10: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from I_2

5.5.4 MA vs MMA comparison

In this section, we conduct a comparative analysis of our MA and MMA algorithms across both datasets.

In the assessment conducted using dataset I_1, our analysis revealed that MMA outperformed MA in 23 instances. The differences in solution quality for these cases averaged 45.8%, with a range spanning from a minimum of 0.15% to a maximum of 520.35%. On the other hand, MMA delivered inferior solutions compared to MA in 13 instances, with an average difference in solution quality of 5.59%, and the variations ranging between 0.56% and 27.75%. Employing the Wilcoxon signed-rank test at a significance level of 0.05, we determined that the performance disparity between these two algorithms is statistically significant, as indicated by a p-value $p < 0.01$. The specific results for each instance, showcasing this comparative performance, are detailed in Figure 5.11.

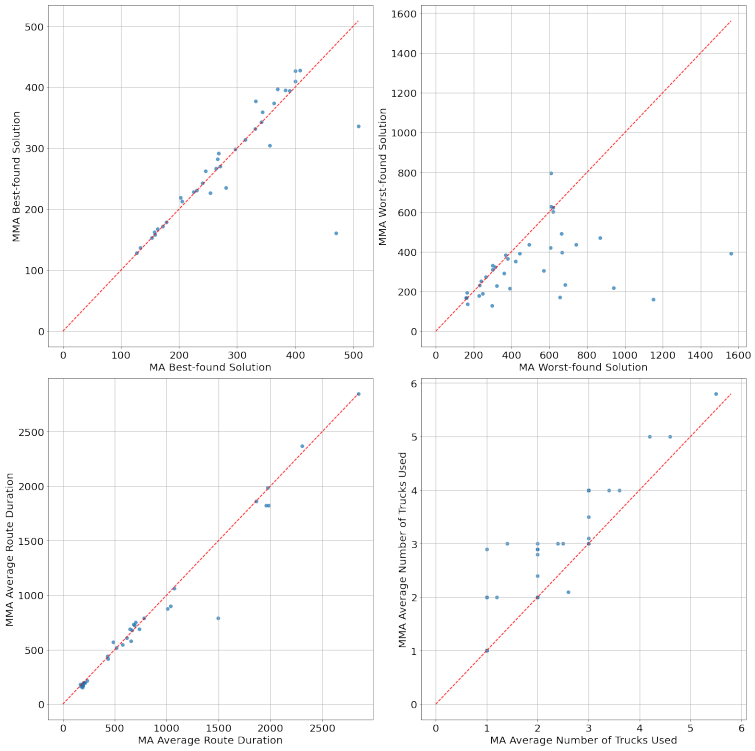

FIGURE 5.11: MA vs MMA performance in terms of average solution quality on instances from I_1

In the context of the best-found solutions, MA actually demonstrated greater success than MMA. Specifically, MA yielded better best-found solutions in 27 instances, equal quality solutions in 4 instances, and inferior solutions in 5 instances. However, it is important to note the disparity in the magnitude of improvements: where MA surpassed MMA, the average improvement in solution quality was 3.56%, whereas in instances where MMA excelled, the average improvement was a significant 58.31%. This suggests that while MA frequently outperformed MMA, the degree of improvement by MMA in its favorable cases was considerably more substantial.

Regarding the worst-found solutions, MMA showcased superior performance, producing better solutions in 23 instances and falling short in 13. The average difference

in solution quality was notably higher at 109.36% in cases where MMA surpassed MA, compared to a 7.05% improvement when MA outperformed MMA.

A similar pattern was observed in terms of average penalties, with MMA achieving solutions with lower penalties in 27 instances, the same in 5 instances, and higher in 4 instances. Intriguingly, MA typically found solutions with a lower route count, indicating a different strategic approach.

When evaluating the average and maximum route duration, we found no statistically significant differences between the performance of these two methods. The comparative performance of MA and MMA across these metrics, including best-found solutions, worst-found solutions, average penalties, and route counts, is depicted in Figure 5.12.

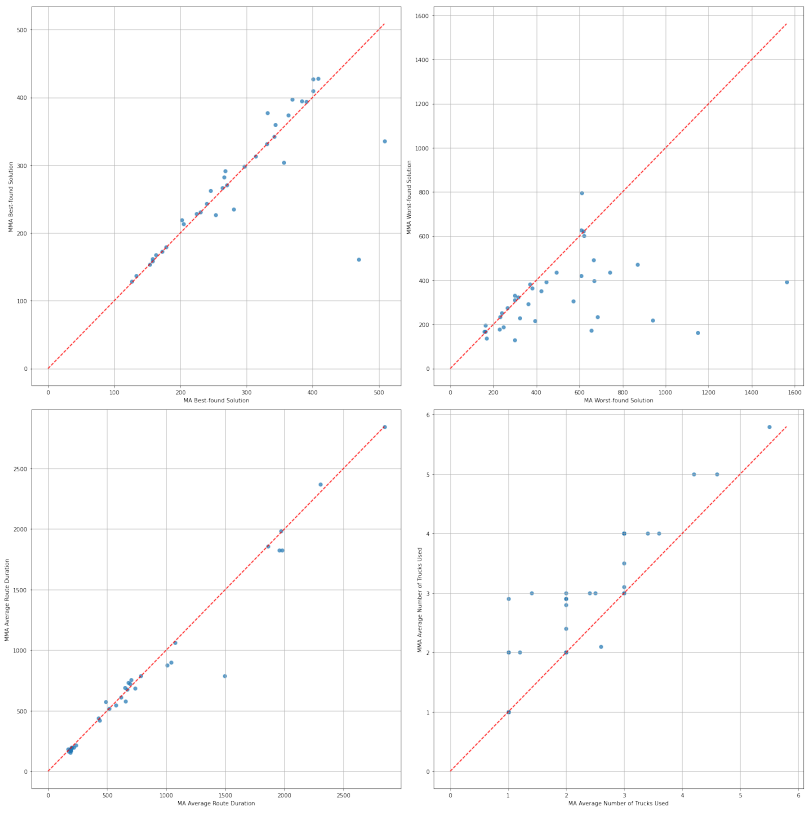

FIGURE 5.12: Comparative scatter plots showcasing various performance metrics for MA and MMA on instances from I_1

In the evaluation using instances from dataset I_2, MMA demonstrated superior performance to MA in 52 instances, while it lagged behind in 4 instances. The average difference in solution quality also favored MMA. Specifically, in scenarios where MMA outperformed MA, the average improvement was 29.11%, with the minimum and maximum differences at 3.12% and 55.41%, respectively. In contrast, where MA surpassed MMA, the average improvement was 10.38%, with a range from 1.25% to 18.62%.

Further reinforcing these findings, statistical tests were conducted, which confirmed a statistically significant difference in the performance of these two algorithms in terms of average solution quality, evidenced by a p-value $p < 0.0001$. The detailed performance comparison of these two algorithms for each instance is illustrated in Figure 5.13.

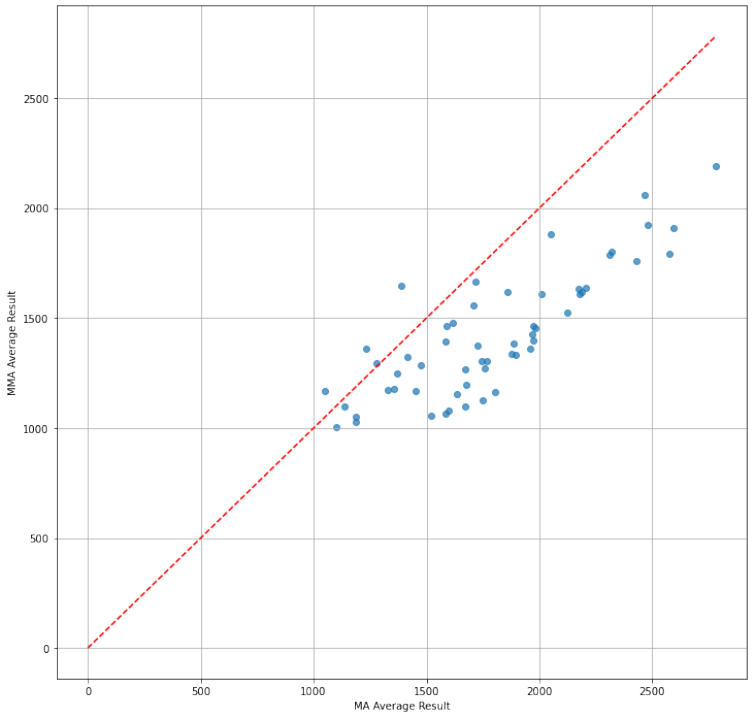

FIGURE 5.13: MA vs MMA performance in terms of average solution quality on instances from I_2

MMA also excelled over MA in terms of best-found solutions, worst-found solutions, and route count. However, MA outshone MMA when it came to average route duration, maximum route duration, and penalties. The comprehensive comparison of MA and MMA across these various metrics is depicted in Figure 5.14.

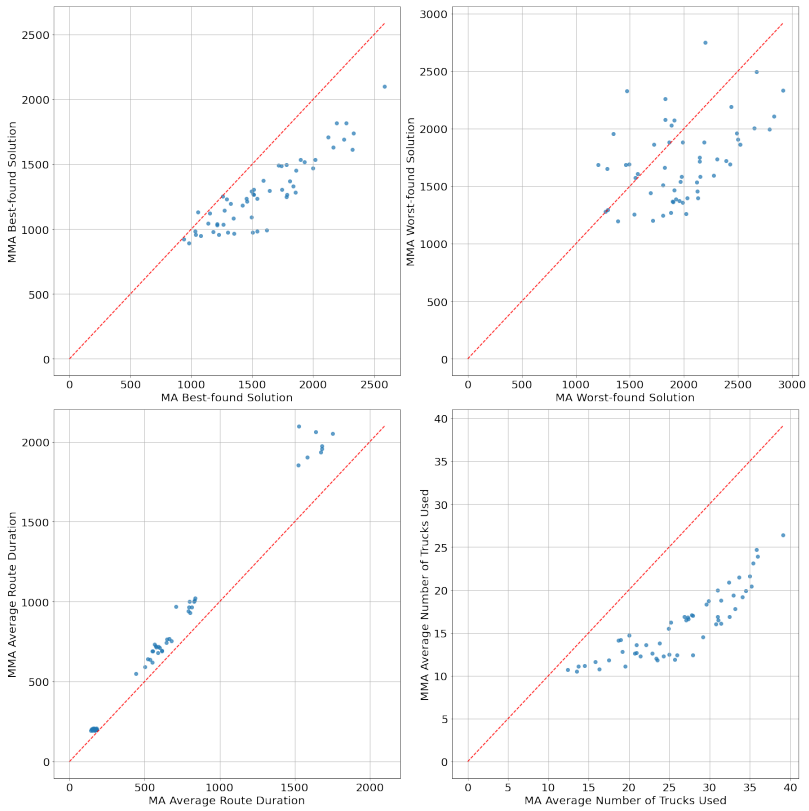

FIGURE 5.14: Comparative scatter plots showcasing various performance metrics for MA and MMA on instances from I_2

5.6 Chapter conclusion

In this chapter, we build upon the problem introduced in Chapter 4 by incorporating partial vehicle recharging. We evaluated eight metaheuristics and conducted a statistical analysis to identify the top performers based on various metrics. The results are clearly presented, accompanied by visual aids to help the reader interpret the data effectively.

CHAPTER
SIX

MULTI-OBJECTIVE GREEN VEHICLE ROUTING PROBLEM

6.1 Problem definition

In our third problem iteration, we delve deeper into the complexities of VRP, incorporating more attributes as detailed in Section 3.3. This version not only retains all constraints from the problem outlined in Section 5.1 — including features like partial recharge, soft time windows, and time-variable speeds — but also introduces new dimensions. We refer to this version of the problem as *Multi-Objective Heterogeneous Fleet Vehicle Routing Problem with Soft Time Windows* (**MOHF-VRP-STW**)

One significant addition is the consideration of asymmetrical distances. This means the distance from location i to location j does not necessarily mirror the distance from j to i. Such a feature is particularly crucial in urban settings riddled with one-way streets, adding a layer of realism to the problem-solving.

Moreover, we now contemplate a diverse fleet, comprising of both ICVs and AFVs. It is a more grounded perspective as delivery companies are likelier to operate with mixed fleets, having a combination of ICVs and AFVs, rather than relying solely on AFVs. Such a blend not only extends our reach, serving customers that might be beyond the tether of battery-restricted AFVs, but it also optimizes the potential of serving more customers within their favored time slots.

However, the integration of a mixed fleet also ushers in added complexity. It necessitates discerning the optimal vehicle for each route, a task that demands meticulous strategizing.

To truly capture the intricacies of this scenario, a multi-objective optimization is warranted. A singular focus on distance or penalties would inadvertently sideline the inherent disparities between ICVs and AFVs. Consequently, our approach will simultaneously juggle two objectives: one that emphasizes the weighted sum of distances and penalties, akin to the previous problem, and another that aims to curtail the total distance covered by ICVs. The essence behind this dual-focus is understanding that reducing ICV distances might inadvertently stretch the overall distance due to a heightened reliance on AFVs and subsequent increased AFS visits. Striking the right balance between these objectives becomes paramount.

6.2 Objective functions

As mentioned earlier, for this version of the problem, two objectives were evaluated independently:

- $\mathcal{O}_1$ - This objective, previously applied to EVRP and EVRP-PR, comprises a weighted sum of the aggregate distance traversed by all vehicles, along with penalties incurred for any time windows that were not met. The formula for this objective is presented in Equation (4.1).
- $\mathcal{O}_2$ - This objective seeks to reduce the overall distance traveled by ICVs, which consequently lowers the emissions generated by the vehicles during customer service.

Although the two objectives are assessed independently, it is crucial to establish a primary metric to steer the metaheuristic algorithms effectively. For instance, in the GVNS algorithm, we must identify a criterion to decide when a new solution surpasses the current one. Opting for either of the individual objectives as this primary metric might skew the metaheuristic's search, resulting in a suboptimal pareto frontier. To circumvent this, a more balanced strategy is to amalgamate the two objectives, such as through a weighted sum, an approach we have adopted in our research.

Nonetheless, it is important to note that these two objective functions inherently yield values within vastly different ranges. This discrepancy primarily arises due to the penalty component of $\mathcal{O}_1$. Consequently, it becomes imperative to normalize these objectives, ensuring that the influence of $\mathcal{O}_1$ does not overshadow that of $\mathcal{O}_2$. To achieve this normalization, we adopted the *Min-Max Scaling* technique, which is outlined by the formula presented in Equation (6.1).

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{6.1}$$

In our scenario, we established the minimum value (X_{min}) as 1 for both objectives, and the maximum value (X_{max}) corresponds to the objective value of the initial solution for each respective objective. Consequently, the scaled value may exceed 1 if the initial solution outperforms the value undergoing scaling. However, this does not pose an issue within the context of our study.

However, it is not sufficient to merely scale the objectives because their rates of growth are disparate, primarily due to the penalty component of $\mathcal{O}_1$. Take, for instance, an initial penalty of 50,000, which is plausible when the routes have not been refined. Under these circumstances, a reduction of 100 in the penalty during subsequent iterations might be negligible, causing the algorithm to bias towards objective $\mathcal{O}_2$. This happens because smaller fluctuations in $\mathcal{O}_2$ would exert a more pronounced influence given its initially lower value. To mitigate this imbalance, we first apply a logarithmic transformation to each objective, which normalizes their rates of growth. After this transformation, we implement Min-Max Scaling. Finally, we aggregate the outcomes using a weighted sum, as delineated in Equation (6.2), to derive a composite metric for evaluation.

$$\mathcal{F}(\mathcal{O}_1, \mathcal{O}_2) = \alpha \left(\frac{\log \mathcal{O}_1 - \log \mathcal{O}_1^{min}}{\log \mathcal{O}_1^{max} - \log \mathcal{O}_1^{min}} \right) + \beta \left(\frac{\log \mathcal{O}_2 - \log \mathcal{O}_2^{min}}{\log \mathcal{O}_2^{max} - \log \mathcal{O}_2^{min}} \right) \tag{6.2}$$

6.3 Test instances

To effectively evaluate the MOHF-VRP-STW, it was necessary to develop new sets of instances, building upon those outlined in Section 4.3. We crafted two distinct

sets: $\mathcal{I}_1$, derived from the I_1 set, and $\mathcal{I}_2$, modeled on the I_2 set. These newly created instances are available for download at `www.mi.sanu.ac.rs/~luka/resources/Instances.zip`.

Listing 6.1 provides a sample instance to illustrate its format and structure. The initial section details node information, beginning with a unique string ID. The type of each node is categorized as 'd' for depot, 'f' for AFSs, and 'c' for customers. Following this, we specify the node's demand, which indicates the quantity of goods required, a detail pertinent only to customer nodes.

The next set of data relates to time windows: *'ReadyTime'* marks the start, and *'DueTime'* the end of this window. We then include *'ServiceTime'*, the duration a vehicle is expected to spend at a node. This is applicable only to customer nodes; for AFS nodes, the service time is set to zero since the duration spent at these nodes dynamically depends on the charging time.

Finally, we provide information on AFS capacity, defining the number of vehicles that can be charged simultaneously at an AFS. Although this detail does not play a crucial role in our current study, it has been incorporated for the benefit of future research.

In the subsequent section, we detail the distances between each node pair. Notably, these distances are asymmetric, meaning the distance from node A to node B may differ from the distance from node B to node A. We determined these distances through a two-step process. Firstly, we calculated the distances based on the coordinates given in the original instance, using the Euclidean distance formula. Following this, we introduced an element of randomness: a number is generated between 0 and 1 using a uniform distribution. If this number is less than 0.33, we then adjust the calculated distance by multiplying it with a randomly generated multiplier. This multiplier falls within the range of 0.8 to 1.2 and is also determined using a uniform distribution.

The following section of our instance provides comprehensive information about the available vehicles. This includes several key parameters for each vehicle: ID, type of engine, battery capacity, load capacity, consumption rate, recharging rate, average velocity, and the total number of available vehicles of that type. It is important to note that fields marked with 'N/A' indicate that certain information is not applicable to a specific vehicle type. For instance, battery capacity is a parameter that does not apply to ICVs.

The final section of focuses on time-dependent speed data. Here, we outline each time interval, specifying its start and end times, along with an associated multiplier for that interval. This multiplier is employed to adjust the average speed designated for each specific vehicle type. The methodology for creating these intervals aligns with the approach detailed in Section 4.3, ensuring consistency with previous versions of the problem.

```
1  NODE SECTION
2  ID Type demand ReadyTime DueTime ServiceTime AfsCapacity
3  D0 d 0.0 0.0 1236.0 0.0 N/A
4  S0 f 0.0 0.0 1236.0 0.0 3
5  S5 f 0.0 0.0 1236.0 0.0 4
6  S15 f 0.0 0.0 1236.0 0.0 3
7  C30 c 10.0 355.0 407.0 90.0 N/A
8  C12 c 20.0 176.0 228.0 90.0 N/A
9  C100 c 20.0 744.0 798.0 90.0 N/A
10 C85 c 30.0 737.0 809.0 90.0 N/A
11 C64 c 10.0 263.0 325.0 90.0 N/A
12 NODE SECTION END
```

```
13
14 DISTANCE SECTION
15 Nodes D0 S0 S5 S15 C30 C12 C100 C85 C64
16 D0 0.0 0.0 41.13 24.02 20.61 31.98 38.07 29.73 24.30
17 S0 0.0 0.0 35.17 24.02 21.34 36.11 38.07 29.73 21.54
18 S5 35.17 35.17 0.0 8.54 31.01 6.08 24.02 49.42 56.61
19 S15 20.83 24.02 58.54 0.0 32.68 70.06 65.79 37.14 9.41
20 C30 20.01 20.61 31.01 34.66 0.0 30.41 48.80 44.71 33.08
21 C12 38.07 38.07 7.20 70.82 30.41 0.0 32.02 49.73 59.61
22 C100 38.07 41.78 25.01 61.13 44.75 30.0 0.0 28.17 55.44
23 C85 32.42 29.73 51.34 48.97 54.05 57.89 30.38 0.0 31.54
24 C64 21.54 21.54 56.61 10.95 31.84 59.61 46.11 36.05 0.0
25 DISTANCE SECTION END
26
27 VEHICLE SECTION
28 ID Type Battery_capacity Load_capacity Consumption_rate
    ↪ Recharging_rate Velocity Number
29 V1 AFV 77.75 200.0 1.0 3.47 1.0 1
30 V2 ICV N/A 200 N/A N/A 1 4
31 VEHICLE SECTION END
32
33 SPEED INTERVAL SECTION
34 Start End Multiplicator
35 0 206.0 0.75
36 206.0 1030.0 1
37 1030.0 1236.0 0.8
38 1236.0 1236.0 1
39 SPEED INTERVAL SECTION END
```

Listing 6.1: Example MOHF-VRP-STW Instance

6.4 Solution representation

The solution structure employed for this version of the problem mirrors that of the TDPR-EVRP-STW outlined in Section 5.2, with a notable distinction being that each route is tagged with specific vehicle information that serves the customers within. Specifically, routes are not merely structured as doubly linked lists comprising nodes that match customers, AFS, and depot. Instead, a route is encapsulated as an ordered pair (H_a, S), where H_a denotes the vehicle index assigned to the $a-th$ route, and S is a doubly linked list detailing the nodes. Each node within the list is defined by a five-part sequence: the index, arrival and departure times, node type, and waiting time. This added layer of vehicle information allows for clear differentiation between EVs and ICVs, enabling us to apply the appropriate treatment for each route type.

6.5 Solution feasibility

In contrast to the previous two versions of the problem, in this version, we only need to include visits to AFSs on certain routes to address the battery depletion-related feasibility issues. As a result, the FIX_SOLUTION approach is slightly different and is outlined in Algorithm 27. In addressing this version of the problem, which involves partial recharging, the procedure invoked at line 4 aligns with the one outlined in Algorithm 25.

Algorithm 27 Procedure for adding visits to AFSs into solution when dealing with HF scenario

```
1: procedure FIX_SOLUTION_HF(Solution S)
2:     for ∀route ∈ S do
3:         if vehicle assigned to route is AFV then
4:             FIX_ROUTE(route)
5:         else if vehicle assigned to route is ICV then
6:             Remove all AFS visits from route
7:         end if
8:     end for
9:     return S
10: end procedure
```

6.6 Neighborhood structures

We introduce an additional neighborhood structure in this version of the problem, in addition to those described in Section 4.8. The purpose of the additional neighborhood structure is to enhance the search space exploration by allowing the algorithm to consider various permutations of vehicle-to-route assignments. This becomes important in instances where a fleet consists of diverse vehicle types, each with potentially unique characteristics such as capacity, range, refueling or recharging times, costs associated with their operation, or in our case engine types.

- $\mathcal{N}_{11}(S)$: Swaps the vehicle types for two different routes.

Incorporating the newly defined neighborhood structure, we can refine our VND for this particular problem to systematically explore neighborhoods $\mathcal{N}_2$ through $\mathcal{N}_8$, in addition to $\mathcal{N}_{11}$. This expanded search pattern allows for a more comprehensive examination of potential solutions within our optimization framework.

As a reminder, we have excluded neighborhood structure $\mathcal{N}_1$ from our consideration due to its suboptimal performance. Moreover, it is worth noting that neighborhoods $\mathcal{N}_9$ and $\mathcal{N}_{10}$ are exclusively utilized during the shaking process of the GVNS method.

6.7 Pareto-front representation

In the realm of multi-objective optimization, maintaining a record of Pareto-optimal solutions is essential. These are solutions that are not dominated by any other solution in the solution space. Determining whether one solution is dominated by another involves comparing multiple objective values. Let us consider two solutions, A and B, each with N objectives $f_1, f_2, \cdots, f_N$. Solution A dominates solution B if and only if both of the following conditions are met:

- **No worse in all objectives:** For each objective f_i, A is no worse than B. Formally, $f_i(A) \leq f_i(B),\ \forall i \in \{1, 2, \cdots, N\}$
- **Better in at least one objective:** There exists at least one objective f_j where A is strictly better than B. Formally, $\exists j \in \{1, 2, \cdots, N\},\ f_j(A) < f_j(B)$

Thus, the concept of dominance can be mathematically formulated as follows:

$$A \text{ dominates } B \iff (\forall i \in \{1,2,\ldots,n\}, f_i(A) \leq f_i(B)) \text{ and } (\exists j \in \{1,2,\ldots,n\}, f_j(A) < f_j(B)) \tag{6.3}$$

To facilitate this, we maintained an archive of all non-dominated solutions, essentially an unordered set of such solutions. In each algorithm we tested, whenever a new solution is generated and evaluated at any stage of the algorithm, it is compared against the solutions in the archive. This comparison is conducted using the method outlined in the pseudocode presented in Algorithm 28. In this algorithm, the initial step involves verifying whether the archive of Pareto-optimal solutions is empty. If it is, the new solution is immediately added as Pareto-optimal (Line 3). Conversely, if the archive contains existing solutions, we systematically examine each one. Should any of these archived solutions dominate the new one, the procedure terminates without any modifications to the archive (Line 9). However, if an existing solution is found to be dominated by the new one, it is placed into a temporary collection, which aggregates all such dominated solutions from the archive (Line 12). Once all solutions in the archive have been assessed, those in the temporary collection (i.e., those dominated by the new solution) are removed from the archive (Line 15). Subsequently, the new solution is added to the archive (Line 16).

Algorithm 28 Procedure for updating the archive of pareto-optimal solutions

```
1:  procedure UPDATE_ARCHIVE(Archive A, Solution x)
2:      if A = ∅ then
3:          A ← {x}
4:          return
5:      end if
6:      dominatedByNew ← ∅
7:      for ∀solution ∈ A do
8:          if solution dominates x then
9:              return
10:         end if
11:         if x dominates solution then
12:             dominatedByNew ← dominatedByNew ∪ {solution }
13:         end if
14:     end for
15:     A ← A \ dominatedByNew
16:     A ← A ∪ {x}
17:     return
18: end procedure
```

6.8 Metaheuristics

The metaheuristic approaches employed for the MOHF-VRP-STW are consistent with the methods described in Section 4. This adherence ensures a foundational continuity in the methodologies applied across different problem versions. MOHF-VRP-STW, much like TDPR-EVRP-STW discussed earlier, considers the critical aspect of partial recharge. This consideration is crucial in addressing the range limitations and

charging dynamics unique to electric vehicles. Consequently, all the nuanced adaptations and specific strategies that were detailed in Section 5 for handling the partial recharge attribute remain relevant and are integrated into this problem version.

Moreover, the problem's complexity is augmented by the introduction of a heterogeneous fleet. This addition implies a diversity in the capabilities and characteristics of the vehicles used, mainly in the context of vehicle engine type, but can also include their range, charging speed, or cargo capacity. Such diversity necessitates a more intricate approach in the algorithmic design to effectively manage the varying constraints and potentials of each vehicle type. To accommodate these considerations, we have adopted a modified the solution structure, elaborated upon in Section 6.4. Additionally, to effectively manage the diversity of vehicle types, we introduced a new neighborhood structure, detailed in Section 6.6, and integrated it into the VND process.

The transition to a multi-objective optimization framework adds a significant layer of intricacy to the problem. In contrast to single-objective optimization, which focuses on maximizing or minimizing a single metric, multi-objective optimization within the MOHF-VRP-STW context demands the simultaneous consideration of two objectives that may be in conflict. To strike an optimal balance between these goals, it is essential to construct a Pareto-front comprising all non-dominated solutions.

Consequently, each metaheuristic is equipped with a specialized structure, as detailed in Section 6.7, to meticulously track these solutions. This approach is exhaustive: every new solution generated, whether during the VND procedure, the construction phase ACO, or following the creation of new offspring in GA, MA, and MMA, is evaluated against the solutions in the Pareto-optimal archive. It is crucial to emphasize that this evaluation is not limited to solutions deemed as the current best but extends to all new solutions.

While this might appear to be a substantial computational burden, the process is relatively efficient, typically operating in $\mathcal{O}(n)$ time complexity, where n represents the number of solutions in the archive, which is usually not particulary high. This efficiency is vital for assembling the most comprehensive and optimal Pareto-front possible. It is important to recognize that not all Pareto-front members would necessarily be identified as the best new solution at any given step. Hence, this thorough approach is indispensable for identifying the truly optimal solutions across the multi-objective landscape.

The modifications outlined above are applicable to all the metaheuristics under consideration. However, there are additional adjustments that are specific to some particular algorithms.

In particular, for the ACO and ACOLS algorithms, we had to reconsider the method of selecting the next customer to be incorporated into the partial solution. Initially, to address the challenge posed by different vehicle types, we introduced a secondary set of pheromone values, aimed at capturing the associations between each customer and vehicle type. However, this strategy was re-evaluated following preliminary testing with instances from both $\mathcal{I}_1$ and $\mathcal{I}_2$ datasets. We eventually shifted away from this method, as we observed significantly improved results by simply implementing a local search within the $\mathcal{N}_{11}(S)$ neighborhood structure post-construction phase. It is noteworthy that this additional local search step was integrated only in ACO, as including it in ACOLS would result in redundancy. Nonetheless, we had to modify the selection process for potential neighbors in both the ACO and ACOLS algorithms. The details of this revised procedure are outlined in Algorithm 29. The key distinction between Algorithm 22 and Algorithm 29 lies in Lines 8 and 16. When the position under scrutiny is on a route served by an ICV, it becomes unnecessary

to verify the vehicle's ability to reach the next station. Likewise, incorporating visits to AFSs as potential moves is redundant for routes specifically handled by ICVs.

Algorithm 29 Procedure for finding a set of potential moves based on a set of available positions in the solution, while considering different vehicle types

```
1: procedure GET_MOVES(positions, available, S)
2:     moves ← empty set o pairs
3:     for ∀pos ∈ positions do
4:         for ∀a ∈ available do
5:             if adding a to S as position pos violates capacity constraint then
6:                 continue
7:             end if
8:             if pos is a part of the route served by EV then
9:                 if no station can be reached after visiting a then
10:                    continue
11:                end if
12:            end if
13:            moves ← moves ∪ {(pos, a)}
14:        end for
15:        if no customers have been added to moves for pos then
16:            if pos is a part of the route served by EV then
17:                stations ← find stations that can be visited from pos
18:                for s ∈ stations do
19:                    moves ← moves ∪ {(pos, s)}
20:                end for
21:            end if
22:        end if
23:    end for
24: end procedure
```

The version of BCO adapted for this problem closely resembles the approach utilized for TDPR-EVRP-STW. However, it integrates an additional type of move into its *TRANSFORM_SOLUTION* procedure:

- $M_6(S)$ - Randomly selects two vehicles of different types and swaps their respective types.

For GA, MA, and MMA, the primary deviation from their previous iterations lies in the chromosome representation and the crossover operator.

The chromosome structure for MOHF-VRP-STW closely aligns with the one described in Section 5.4, where it is encoded as a sequence of pairs. In each pair, the first element represents the node index, while the second indicates the duration of the stop at that node. However, this problem introduces an additional layer of complexity with vehicle types. Consequently, the chromosome for this version includes a second list that specifies the vehicle types. In essence, at any given index i, this list identifies the vehicle type assigned to service the i-th route. This arrangement facilitates the application of the crossover operator not only to the sequence of customers but also to the vehicle types, thereby accommodating the unique requirements of this problem.

In terms of the crossover mechanism, we continue to employ the OX operator on the sequence of customers, as detailed in Section 4.14. However, for assigning vehicle types to the offspring, we have adapted the OX operator in the following manner.

Initially, we randomly select two points within the vehicle type list of the first chromosome. The segment delineated by these points is then directly transferred to the offspring, retaining its original position from the parent chromosome. Subsequently, we populate the remaining positions in the offspring with values from the second chromosome, deliberately omitting those that appear in the transferred segment. To account for any repetition of vehicle types within this segment, we omit the first n occurrences of a vehicle type in the second chromosome if it appears n times in the interval. For instance, if AFVs are assigned to two routes within the chosen segment, we skip the first two occurrences of AFV routes from the second chromosome. This process is visually depicted in Figure 6.1. All other aspects of the GA, MA, and MMA algorithms are in line with the methodologies outlined in Sections 4.14 and 5.4.

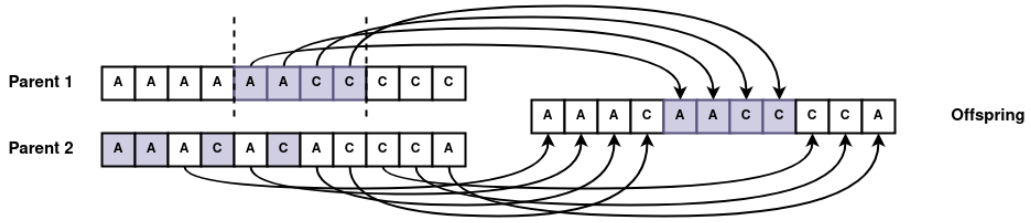

FIGURE 6.1: Modified OX operator applied to the vehicle type part of the chromosome

6.9 Experimental evaluation

6.9.1 Experimental setup

The experimental arrangement for this iteration of the problem mirrored the setup described in Section 4.15.2. Our empirical analysis indicated that the hyperparameter values identified in Section 4.15.3 were equally effective for this version of the problem, leading us to adopt these values for all subsequent tests.

To maintain an unbiased approach, we set both the α and β parameters to 0.5. This ensured that no specific objective was given undue precedence over another. A thorough examination and analysis of the α and β values, exploring their broader implications and optimal configurations, will be presented in the upcoming section.

6.9.2 Objective function constants

In this section, we explore the impact of varying α and β values within our composite evaluation metric (Equation (6.2)) on the resulting Pareto-front. Our hypothesis is that increasing α will yield Pareto-fronts more adept at uncovering superior solutions for objective $\mathcal{O}_1$. Conversely, elevating β should lead to Pareto-fronts that preferentially identify solutions excelling in objective $\mathcal{O}_2$.

To validate our hypothesis, we conducted experiments with five distinct α and β ratios, employing the GVNS as our preferred method. It is important to acknowledge that the choice of method can potentially sway our findings. Our observations revealed that certain methods typically generate Pareto-fronts with a higher quantity of solutions compared to others. However, this does not necessarily imply superior Pareto-frontiers, as some of these solutions may be very similar, diminishing the practical value of having additional solutions.

We selected the following ratios for detailed examination:

- **3:7** ($\alpha = 0.3$, $\beta = 0.7$): This configuration significantly prioritizes objective $\mathcal{O}_2$.
- **4:6** ($\alpha = 0.4$, $\beta = 0.6$): Here, the metric shows a mild preference for objective $\mathcal{O}_2$.
- **5:5** ($\alpha = 0.5$, $\beta = 0.5$): This balanced ratio equally weights both objectives, showing no preference.
- **6:4** ($\alpha = 0.6$, $\beta = 0.4$): In this case, the metric is slightly inclined towards objective $\mathcal{O}_1$.
- **7:3** ($\alpha = 0.7$, $\beta = 0.3$): This ratio strongly emphasizes objective $\mathcal{O}_1$.

In most scenarios, our observations aligned with the anticipated outcomes, as depicted in Figure 6.2 using the *r101_21* example instance. Nevertheless, certain instances revealed unexpected results, with some ratios performing notably better or worse than anticipated. For instance, Figure 6.3 illustrates the Pareto-frontiers for various ratios on the *r105_21* instance. Here, the *7:3* ratio notably underperformed, consistently yielding solutions that were dominated by those discovered using other ratios. While not as stark, the *4:6* ratio was also notably outclassed by the *5:5* ratio. Interestingly, we often found the *5:5* ratio to be the most effective, potentially because its unbiased approach towards both objectives allows it to uncover a broader range of high-quality solutions. When a method favors one objective, it may converge towards a segment of the search space where it becomes challenging to find viable solutions for the other objective.

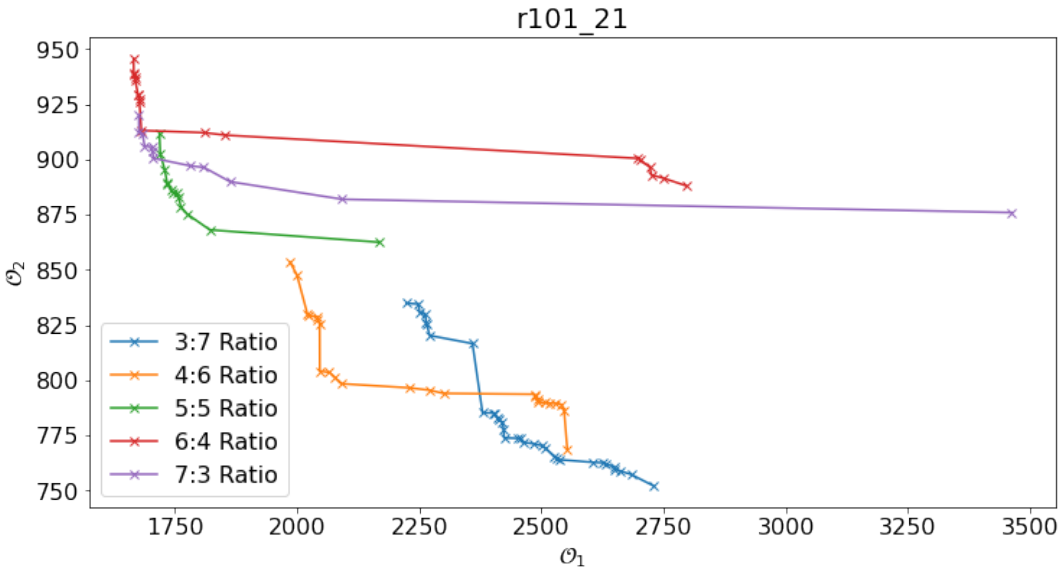

FIGURE 6.2: An illustration of Pareto-fronts: comparative analysis using varied ratios of importance for objectives in instance *r101_21*.

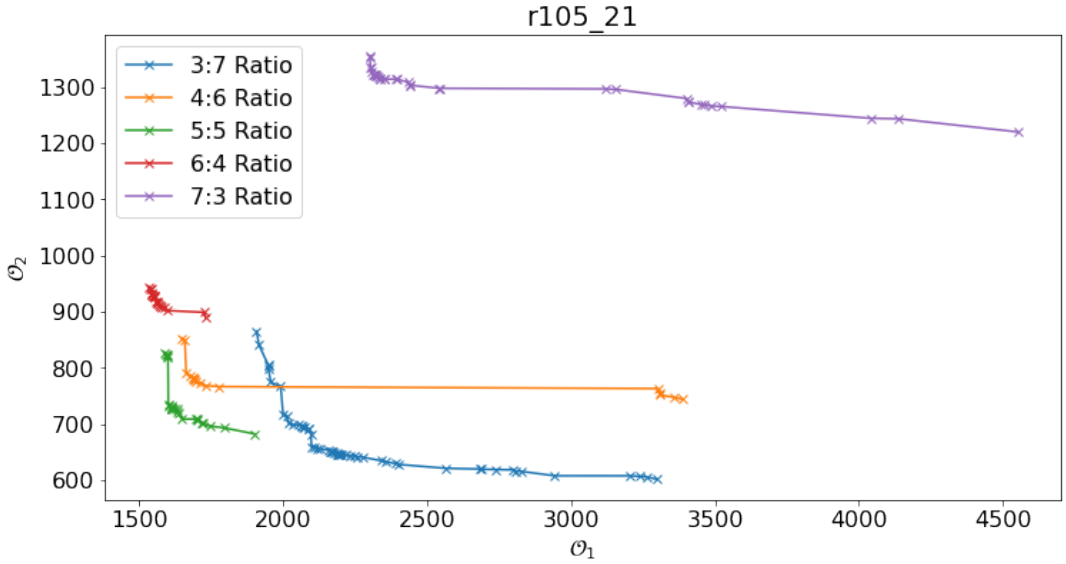

FIGURE 6.3: An illustration of Pareto-fronts: comparative analysis using varied ratios of importance for objectives in instance *r105_21*.

6.9.3 Overall comparison

In Table 6.1, we present a comparative analysis focused on the average solution quality for objectives $\mathcal{O}_1$ and $\mathcal{O}_2$. This analysis is based on thirty iterations of each algorithm applied to instances from the dataset $\mathcal{I}_1$. Values emboldened within the table signify the highest average solution quality achieved among all evaluated algorithms, specifically in relation to each respective objective. The comprehensive results for all algorithms, encompassing all metrics, are detailed in the document provided at `https://www.mi.sanu.ac.rs/~luka/resources/phd/AppendixB.pdf`.

TABLE 6.1: Results for all MOHF-VRP-STW metaheuristics on the instance set $\mathcal{I}_1$.

Instance	**Values**															
	GVNS		**GRASP**		**ACO**		**ACOLS**		**BCOi**		**GA**		**MA**		**MMA**	
	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$
c101C10	345.01	123.68	446.83	**90.11**	397.78	273.82	340.21	95.05	448.29	194.6	380.71	249.03	**318.36**	**90.11**	332.42	122.04
c101C5	231.24	121.72	822.84	40.63	247.46	138.52	233.85	116.9	2260.23	**27.76**	**231.04**	134.51	1469.94	57.27	1095.2	100.58
c103C15	**350.06**	196.5	506.01	**164.02**	472.34	327.84	359.89	194.38	448.44	271.68	378.4	250.31	351.27	178.8	362.34	208.41
c103C5	215.5	21.35	176.1	20	177.41	74.9	**162.58**	57.25	1944.89	**2**	1173.43	42.45	2020.17	**2**	1325.42	7
c104C10	268.38	76.06	286.35	86.05	370.13	130.78	**267.67**	56.4	396.28	53.46	303.23	94.32	272.2	**48.55**	269.66	71.58
c106C15	269.21	124.74	297.8	**79.01**	371.98	262.62	271.09	98.23	377.28	199.74	305.05	187.71	269.97	98.44	**263.72**	129.48
c202C10	**235.87**	100.74	449.79	**39.1**	307.54	122.19	251.18	106.84	367.02	83.55	273.63	111.83	246.63	81.92	250.04	95.66
c202C15	364.93	179.16	376.8	**149.25**	509.53	326.28	364.05	193.15	476.13	276.96	**360.02**	227.4	364.77	179.46	367.78	192.55
c205C10	**231.48**	96.61	276.99	**48.13**	337.69	135.47	236.27	105.33	323.44	62.15	274.95	131.53	240.48	89.06	246.94	135.18
c206C5	**218.46**	81.36	298.6	34.13	249.99	89.72	224.95	83.08	1334.31	**13.65**	435.06	59.72	1324.64	21.44	964.37	48.91
c208C15	291.98	**105.92**	291.93	**105.92**	454.35	251.82	**291.65**	**105.92**	426.27	178.23	327.72	131.71	291.98	**105.92**	292.31	**105.92**
c208C5	**172.86**	9.12	187.9	45.61	227.74	74.76	185.71	82.24	1567.26	**0**	630.23	35.82	1494.2	11.02	1017.43	29.52
r102C10	260.38	98.44	335.47	**87.46**	259.58	163.26	264.27	145.19	307.51	139.22	**248.99**	148.9	257.48	87.94	255.5	166.24
r102C15	**409.72**	258.66	548.51	**180.45**	493.53	316.14	435.27	227.62	427.43	274.15	428.17	248.02	441.82	215.68	422.03	233.28
r103C10	184.01	61.28	284.66	38.83	212.36	91.08	**148.56**	61.9	612.96	**10.25**	213.79	103.24	197	62.34	186.39	58.63
r104C5	137.47	59.84	172.8	36.77	137.47	59.84	**117.59**	50.48	931.74	**3.68**	410.29	33.47	932.38	7.1	759.06	22.91
r105C15	370.51	208.56	753.58	**192.1**	545.28	310.36	**327.31**	198.81	377.14	241.83	333.2	220.37	353.6	198.41	350.9	209.57
r105C5	**154.33**	87.04	317.28	12.64	199.5	72.59	178.01	38.11	1338.91	**0**	661.34	35.36	1338.91	**0**	628.8	25.56
r201C10	208.95	79.82	225.6	**40.36**	283.71	124.33	214.39	86.46	263.34	53.28	227.15	122.13	**203.55**	72.26	211.33	90.15
r202C15	350.84	200.73	359.35	**162.51**	498.25	309.87	356.28	190.84	466.04	274.93	370.92	205.45	**350.26**	169.09	355.01	211.92
r202C5	**138.13**	69.83	152.07	29.07	141	69.86	143.59	49.54	295.65	**11.63**	218.54	37.14	285.09	14.73	216.08	25.51
r203C10	238.25	71.22	258.83	**51.86**	306.94	124.45	243.56	72.1	275.39	67.57	265.62	113.87	243.5	61.47	**237.28**	64.41
r203C5	**191.26**	98.81	194.6	30.53	245.65	97.92	202.32	43.48	266.87	15.61	233.64	54.75	251.54	**12.21**	222.48	21.37
r209C15	282.39	106.98	274.92	99.68	425.94	273.95	**270.39**	**99.39**	406.56	208.47	329.26	149.84	278.91	116.78	287.49	114.83
rc102C10	482.08	202.89	464.4	206.4	557.43	320.04	**420.41**	233.85	500.02	260.58	441.65	298.07	530.48	**199.48**	488.7	255.21
rc103C15	394.35	261.58	774.53	**218.94**	511.77	364.98	455.88	286.76	410.5	282.42	**378.36**	266.71	464.88	266.41	495.39	285.66
rc105C5	**226.14**	112.72	524.55	45.24	372.08	97.9	476.82	83.44	1261.52	**5.66**	704.42	53.7	1258.62	7.45	911.3	32.66
rc108C10	476.87	**155.4**	694.77	176.47	383.22	257.21	413.16	185.28	376.23	203.94	**355.37**	205.68	477.55	211.42	407.52	213.87
rc108C15	**369.06**	223.07	463.44	**212.42**	469.09	369.01	381.56	245.33	468.38	276.92	383.3	253.87	373.82	220.69	379.53	231.36
rc108C5	**248.47**	143.64	438.89	91.35	**248.47**	143.64	267.94	152.59	2024.37	**5.19**	868.57	105.65	2002.41	11.13	870.16	118.93
rc201C10	293.03	99.03	557.61	75.55	383.55	156.5	297.52	**73.34**	433.63	91.96	334.64	197.52	**288.62**	**73.34**	289.3	**73.34**
rc202C15	374.89	195.01	385.84	165.51	563.54	319.1	375.16	168.48	506.69	261.54	403.65	217.23	379.98	**160.56**	**368.85**	185.77
rc204C15	305.72	**137.1**	325.94	145.14	473.71	273.72	305.01	140.37	430.3	187.51	337.13	182.18	**304.56**	140.03	305.65	150.81
rc204C5	**166.15**	26.53	174.53	24.08	232.22	70.15	167.56	24.08	285.89	**0**	260.44	47.37	372.62	**0**	253.72	18.08
rc205C10	349.58	117.8	355.02	**74.63**	449.58	182.52	361.7	120.45	408.06	80.06	383.25	114.48	**348.28**	147.01	356.15	157.24
rc208C5	164.41	91.3	**164.25**	26.08	200.8	22.9	168.91	26.08	254.89	11.38	192.14	23.47	229.63	**2.61**	173.96	10.43
Count best	13	3	1	16	1	0	8	3	0	11	5	0	6	11	3	2

To ascertain the presence of statistically significant differences in the performance of our algorithms, we conducted the Kruskal-Wallis test on the average solution quality relative to objectives $\mathcal{O}_1$ and $\mathcal{O}_2$. This decision was prompted by the failure to meet the necessary conditions for conducting ANOVA on these objectives. The Kruskal-Wallis test yielded a p-value $p < 0.0001$ for objective $\mathcal{O}_1$ and $p < 0.001$ for objective $\mathcal{O}_2$. Based on these results, we can confidently conclude that there are statistically significant differences in the performances of our algorithms with respect to both objectives. Consequently, we executed a series of Mann-Whitney U tests, applying FDR correction to each pair of algorithms, while distinctly analyzing each objective. The outcomes of these comprehensive tests are systematically illustrated in Figure 6.4. The heatmap on the left displays the results corresponding to objective $\mathcal{O}_1$, whereas the heatmap on the right is dedicated to showcasing the results for objective $\mathcal{O}_2$. From the insights obtained from this figure, it is evident that in relation to objective $\mathcal{O}_1$, the algorithms GVNS and ACOLS displayed the most pronounced differences when compared to the others, yet their performances were notably similar to each other. In the context of objective $\mathcal{O}_2$, the disparities among the algorithms were generally less marked, with ACO being the notable exception that deviated from this trend.

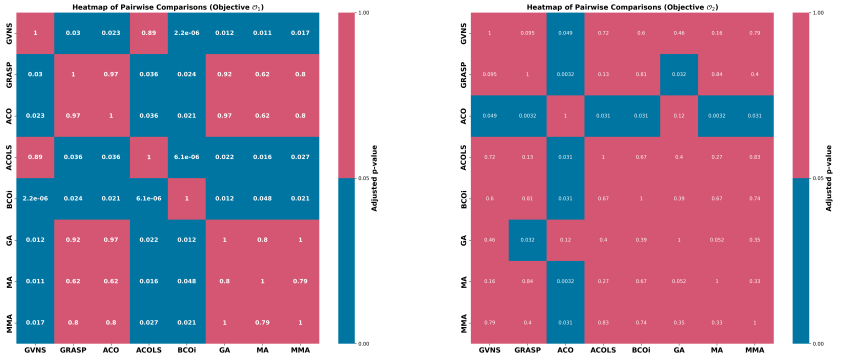

FIGURE 6.4: Heatmap of pairwise comparisons of MOHF-VRP-STW algorithms for $\mathcal{I}_1$ dataset (Mann-Whitney U test with FDR correction)

Figure 6.5, which illustrates the performance of our methods with respect to objective $\mathcal{O}_1$, indicates that GVNS and ACOLS have superior performance compared to other methods. It is also notable that while the median values of GRASP, BCOi, GA, MA, and MMA are somewhat on par with those of GVNS and ACOLS, these methods exhibited a greater number of outliers where the average solution quality significantly deteriorated. Intriguingly, for this particular problem, ACO and ACOLS demonstrated commendable performance, a notable deviation from their typically less impressive results in previous versions of the problem. Regarding objective $\mathcal{O}_2$, no method stood out as significantly superior, although ACO did yield somewhat lower-quality solutions for this specific objective.

TABLE 6.2: Performance comparison of MOHF-VRP-STW meta-heuristics across various metrics (Dataset $\mathcal{I}_1$).

Metric	Method							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality ($\mathcal{O}_1$)	**13**	1	1	8	0	5	6	3
Best-found solution ($\mathcal{O}_1$)	**20**	0	1	15	0	1	12	12
Worst-found solution ($\mathcal{O}_1$)	**16**	3	2	9	0	5	7	5
Average solution quality ($\mathcal{O}_2$)	3	**16**	0	3	11	0	11	2
Best-found solution ($\mathcal{O}_2$)	11	13	4	8	18	13	**21**	17
Worst-found solution ($\mathcal{O}_2$)	2	**25**	0	4	12	1	11	5
Penalty	**20**	12	11	14	3	15	8	9
Average route duration	2	6	7	3	**15**	2	2	0
Maximum route duration	**14**	**14**	7	11	5	11	8	6
Number of routes	8	15	0	10	12	1	**20**	6

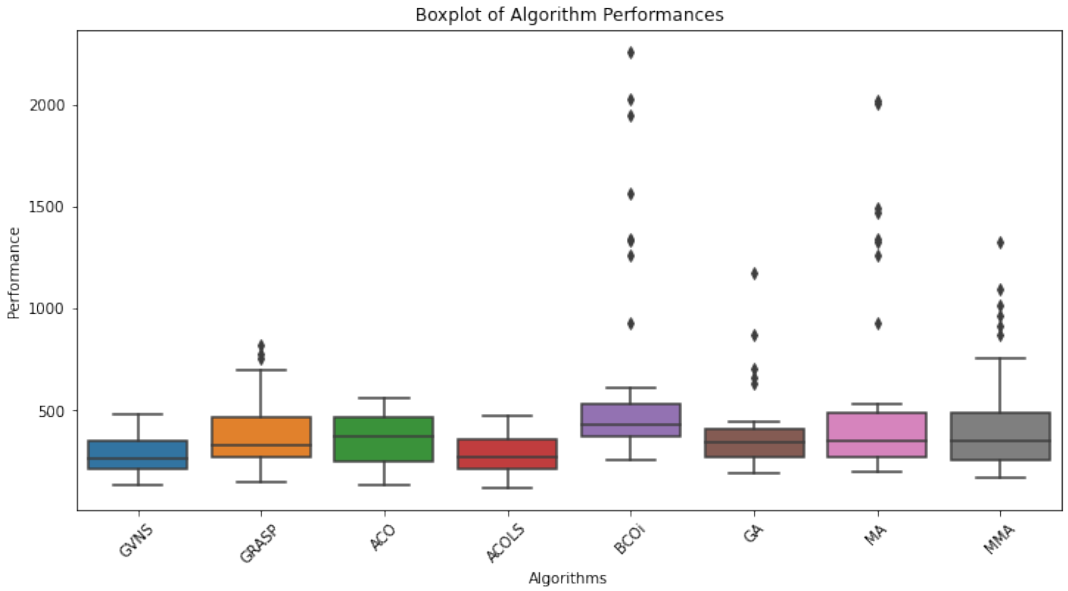

FIGURE 6.5: Boxplot of algorithm performances for the objective $\mathcal{O}_1$

In Table 6.2, we provide an analysis of our algorithms' performance across a spectrum of metrics. The table enumerates the instances where each algorithm excelled beyond its counterparts in terms of effectiveness for each specified metric.

In Table 6.3, we conduct a detailed comparative analysis of the average solution quality pertaining to objectives $\mathcal{O}_1$ and $\mathcal{O}_2$. This examination is grounded in thirty iterations of each algorithm, as applied to instances from the dataset $\mathcal{I}_2$. Within the table, values highlighted in bold represent the pinnacle of average solution quality achieved by any of the evaluated algorithms, corresponding to each objective in focus. This approach provides a clear and concise example for algorithmic performance relative to the specified objectives.

TABLE 6.3: Results for MOHF-VRP-STW obtained on the instance set $\mathcal{I}_2$.

Instance	Values															
	GVNS		GRASP		ACO		ACOLS		BCOi		GA		MA		MMA	
	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$
c101_21	1423.92	897.21	1526.89	1203.32	27662.82	2706.5	6619.16	1890.28	5517.38	3317.88	3270.44	1589.31	1632.82	1145.25	**1212.72**	**558.73**
c102_21	1317.93	831.91	1417.87	1052	19083.57	2713.34	4500.17	1782.39	4501.08	3267.46	3168.69	1423.14	1652.13	1035.49	**1232.49**	**528.7**
c103_21	1362.25	1087	1366.99	1078.03	11353	2695.92	3556.74	1730.87	4059.35	2937.54	3031.87	1172.69	1527.46	1011.05	**1209.96**	**538.34**
c104_21	1170.16	667.32	1262.45	985.94	5939.3	2486.64	2849.12	1291.24	3738.11	2461.74	2963.42	871.33	1326.72	799.17	**1068.26**	**390.52**
c105_21	1402.29	833.57	1469.65	1063.29	22864.03	2824.11	5247.13	1789.4	4812.88	3326.57	3042.68	1441.21	1559.45	1066.32	**1206.85**	**387.49**
c106_21	1361.92	945.95	1395.33	1135.39	20381.56	2816.46	4117.17	1658.01	4595.46	3204.48	2999.9	1326.37	1550.69	1051.7	**1157.62**	**463.5**
c107_21	1375.17	822.33	1396.5	1130.56	21654.74	2732.95	4492.35	1770.28	4676.51	3341.5	2982.04	1325.18	1528.06	965.95	**1157.23**	**443.95**
c108_21	1315.23	749.78	1346.46	1099.97	16396.62	2903.95	4013.41	1666.4	4373.59	3213.16	2848.94	1223.02	1500.15	897.55	**1141.68**	**405.4**
c109_21	1284.09	819.19	1286.32	1016.68	12837.5	2617.06	3187.89	1626.7	4057.83	3021.06	2910.8	1218.5	1406.29	851.94	**1120.88**	**322.81**
c201_21	1290.45	590.2	1388.84	1169.97	50660.83	2114.04	4815.76	1608.08	4981.67	3432.01	2900.67	837.11	1458.24	914.08	**1048.84**	**100.49**
c202_21	1238.72	558.61	1385	1089.85	27598.16	2309.45	3816.43	1395.42	4510.28	3302.05	2834	727.95	1543.59	840.31	**1050.87**	**144.19**
c203_21	1287.03	565.99	1307.51	1023.4	13375.12	2269.34	3359	1406.94	4324.71	2931.7	2940.05	675.49	1442.5	719.75	**1008.23**	**63.52**
c204_21	1194.54	639.25	1265.49	1013.4	4449.3	2266.22	2691.01	1107.79	4257.17	2076.5	3015.94	565.19	1298.97	560.2	**1004.08**	**54.67**
c205_21	1279.12	619.03	1371.71	1059.78	40214.52	2262.32	3817.72	1603.83	4657.01	3475.5	2840.06	686.41	1470.86	721.58	**1022.44**	**65.34**
c206_21	1204.99	580.84	1361.71	1010.75	32020.12	2302.89	3515.77	1470.05	4358.35	3225.31	2844.67	709.38	1457.73	776.28	**1052.87**	**90.02**
c207_21	1292.89	629.58	1368.26	1033.06	24551.43	2278.12	3474.44	1351.59	4351.29	3172.26	2769.71	596.07	1453.9	802.87	**1017.72**	**76.62**
c208_21	1278.29	616.49	1344.18	1040.46	26489.61	2165.29	3397.36	1351.89	4351.89	3199.1	2914.96	695.42	1498.58	805.58	**999.28**	**32.77**
r101_21	**1797.13**	**906.06**	1859.58	1336.35	13351.75	2288.64	6038.41	1437.31	4135.03	2687.33	3157.65	1310.36	1933.98	1254.36	1839.31	916.35
r102_21	**1552.77**	867.39	1647.77	1225.2	10288.72	2532.66	4801.43	1401.7	3660.11	2553.63	2821.6	1174.91	1713.85	1157.59	1647.62	**861.78**
r103_21	**1444.02**	1067.05	1460.06	1177.04	8208.09	2503.18	4167.05	1289.52	3467	2483.15	2598.5	1039.71	1625.24	1062.75	1524.31	**738.93**
r104_21	1486.56	941.26	1314.51	1124.91	6292.75	2460	3171.62	1063.58	3229.67	2345.6	2388.91	922.76	1476.62	862.95	**1299.48**	**552.17**
r105_21	**1558.69**	842.84	1633.74	1271.47	12029.61	2371.49	5125.77	1398.6	3808.58	2611.11	2846.11	1140.12	1753.3	1055.23	1671.19	**736.65**
r106_21	1507.92	837.22	1516.4	1200.69	9526.39	2381	4257.31	1360.73	3557.1	2514.09	2789.91	1118.11	1702.77	1068.95	**1479.04**	**748.94**
r107_21	**1345.79**	730.99	1387.93	1109.82	7833.17	2417.2	3331.8	1260.71	3387.32	2444.55	2566.81	992.42	1561.88	1047.97	1386.36	**624.61**
r108_21	1288.54	688.51	1329.72	1126.06	6483.64	2607.23	3151.98	1307.5	3147.43	2271.82	2299.09	880.09	1463.39	876.23	**1218.19**	**540.35**
r109_21	**1426.07**	789.98	1460.15	1095.65	8822.79	2502.3	3890.91	1229.02	3435.2	2419.36	2668.85	1008.87	1659.16	1074.7	1433.54	**696.45**
r110_21	**1289.83**	709.36	1348.34	1058.44	7729.71	2282.58	3413.7	1126.26	3222.7	2270.5	2345.45	924.37	1564.57	976.7	1334.34	**581.93**
r111_21	**1308.87**	742.95	1350.9	1044.6	7803.01	2313.93	3489.35	1331.66	3246.06	2333.95	2596.02	1118.67	1598.15	950.57	1318.05	**576.93**
r112_21	**1280.31**	950.89	1284.57	1064.57	6292.33	2324.28	3012.81	1221.25	3109.97	2217.76	2439.02	907.59	1552.82	935.76	1336.05	**343.39**
r201_21	1483.06	639.15	1488.52	1114.98	16549.89	1227.13	3277.45	1033.71	3792.24	2531.16	2748.13	594.26	1696.23	708.75	**1357.52**	**93.1**
r202_21	1368.5	444.22	1404.49	1024.97	11626.5	862.09	2774.16	835.15	3680.64	2145.24	2687.55	460.06	1463.49	558.88	**1210.75**	**97.05**
r203_21	1251.53	425.78	1299.49	901.47	5837.42	1275.83	2609.72	858.11	3502.52	2074	2676.55	439.49	1341.46	509.97	**1142.94**	**89.55**
r204_21	1079.7	358.4	1200.97	899.28	3772.08	708.5	2315.67	620.66	3308.91	1530.99	2667.84	335.94	1160.55	370.35	**1000.76**	**38.94**
r205_21	1311.03	562.24	1417.05	1071.77	10796.36	1077.39	2809.8	937.85	3699.6	2264.08	2694.22	520.8	1436.41	617.28	**1188.16**	**91.72**
r206_21	1358.58	474.86	1371.64	940.74	7198.07	1247.41	2624.62	973.93	3726.94	1958.09	2594.16	346.39	1460.4	619.32	**1130.2**	**35.37**
r207_21	1158.72	481.39	1258.03	943.08	4960.47	617.45	2334.26	635.52	3336.36	1756.51	2703.85	340.52	1273.34	449.77	**1043.28**	**112.4**
r208_21	1068.47	276.71	1181.06	842.78	3468.1	888.81	2055.96	709.91	3242.26	1476.39	2592.61	287.88	1160.41	508.6	**1002.42**	**27.81**

Table 6.3 continued from previous page

Instance	Values															
	GVNS		**GRASP**		**ACO**		**ACOLS**		**BCOi**		**GA**		**MA**		**MMA**	
	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathcal{O}_1$	$\mathcal{O}_2$
r209_21	1260.76	464.27	1317.47	973.83	6073.43	1216.02	2571.76	900.95	3609.9	1973.02	2617.83	453.24	1401.15	641.83	**1098.26**	**38.5**
r210_21	1245.51	396.39	1287.97	973.13	5493.84	1250.57	2598.91	859.36	3498.65	1936.72	2615.14	432.04	1341.34	538.99	**1086.26**	**81.61**
r211_21	1139.74	275.17	1250.74	935.07	4330.62	588.19	2453.11	746.21	3361.23	1543.4	2860.22	370.4	1243.91	530.6	**1032.23**	**57.92**
rc101_21	1987.73	1362.77	2003.71	1527.11	13780.62	3176.24	6434.81	1778.85	5033.77	3275.47	3579.31	1318.13	2231.72	1457.54	**1887.85**	**924.3**
rc102_21	1788.25	953.27	1842.67	1425.67	12188.36	3247.56	5795.88	1790.89	4502.71	3244.3	3341.35	1336.11	1917.78	1276.68	**1711.65**	**700.59**
rc103_21	1813.52	1215.75	1656.85	1321.48	10468.83	3196.61	4596.6	1539.1	4220.03	3097.69	3144.56	1135.99	1815.88	1233.68	**1581.57**	**758.54**
rc104_21	1681.17	1251.87	1594.32	1321.84	8892.25	3259.02	3943.57	1344.53	4074.76	2962.78	2987.2	1337.51	1757.15	1138	**1369.59**	**649.6**
rc105_21	1650.4	907.77	1777.36	1447.19	12336.81	3036.16	5709.67	1911.96	4528.59	3066.46	3237.95	1300.23	2016.05	1358.72	**1598.8**	**703.26**
rc106_21	1800.11	1306.8	1804.2	1452.85	12192.12	3053.35	5552.03	1867.85	4461.12	3081.97	3219.55	1265.86	1970.34	1252.95	**1616.86**	**755.34**
rc107_21	1670.61	1296.55	1675.1	1293.31	10449.32	3150.17	4635.12	1607.69	4180.29	2974.63	3060.24	1356.7	1854.48	1169.62	**1488.78**	**610.99**
rc108_21	1573.79	980.74	1585.5	1242.42	9355.64	3247.24	4171.89	1527.13	4025.44	2935.76	2893.49	1071.57	1698.66	1015.21	**1427.67**	**498.77**
rc201_21	1705.38	706.14	1800.88	1423.44	17609.94	1765.14	4640.17	1403.52	4908.8	3327.35	3316.94	813.67	1853.49	943.03	**1524.87**	**123.71**
rc202_21	1590.62	629.92	1692.9	1262.62	12619.08	1376.51	3889.13	1153.73	4626.61	2934.78	3430.95	622.97	1749.6	888.08	**1434.63**	**176.33**
rc203_21	1500.78	416.67	1515.2	1189.94	7327.74	1702.94	3364.21	1231.09	4359.6	2573.55	3215.81	529.98	1628.12	784.68	**1244.22**	**141.14**
rc204_21	1338.01	543.49	1457.03	1091.3	4738.89	2008.24	3117.68	1108.3	4272.07	2090.51	3406.12	629.39	1458.93	684.09	**1116.35**	**13.5**
rc205_21	1510.19	632.86	1684.59	1292.02	10514.14	1925.6	3895.98	1385.66	4663.29	2916.28	3506.86	648.16	1770.73	978.63	**1326.67**	**103.28**
rc206_21	1469.72	539.22	1655.06	1255.3	11239.72	1610.74	3675.4	1321.2	4617.85	2904.32	3319.85	546.34	1753.35	947.55	**1342.84**	**87.15**
rc207_21	1418.66	560.39	1541.83	1229.71	6731.15	1710.77	3493.85	1324.22	4313.23	2449.37	3184.17	439.56	1676.87	861.42	**1217.58**	**45.7**
rc208_21	1367.95	572.7	1451.4	1170.31	4633.86	1828.47	3325.44	1389.22	4140.46	1994.55	3315.95	520.4	1589.65	849.65	**1135.38**	**59.15**
Count best	9	1	0	0	0	0	0	0	0	0	0	0	0	0	47	55

To assess the existence of statistically significant performance differences among our methods on instances from dataset $\mathcal{I}_2$ concerning both objectives, we initially evaluated the suitability of conducting ANOVA. However, due to the failure to meet ANOVA's requisite conditions, we instead opted for the Kruskal-Wallis test, focusing on the average solution quality for both objectives. Regarding objective $\mathcal{O}_1$, the Kruskal-Wallis test yielded a p-value lesser than 0.0001, providing compelling evidence of a statistically significant disparity in the performance of our methods. A similar pattern emerged for objective $\mathcal{O}_2$, where the p-value was an equally significant $p < 0.0001$. Subsequently, we conducted a series of Mann-Whitney U tests, incorporating FDR correction for each algorithmic pairing. The outcomes of these analyses are displayed in Figure 6.6. Here, the heatmap on the left delineates the results pertaining to objective $\mathcal{O}_1$, while the heatmap on the right elucidates the findings related to objective $\mathcal{O}_2$. This figure reveals that for objective $\mathcal{O}_1$, each algorithm exhibited statistically significant differences in performance compared to all others. When considering objective $\mathcal{O}_2$, a similar pattern emerged, with most methods demonstrating distinct statistical variations in their performance. The only exception was observed between MA and GA, which showed comparable levels of performance to each other.

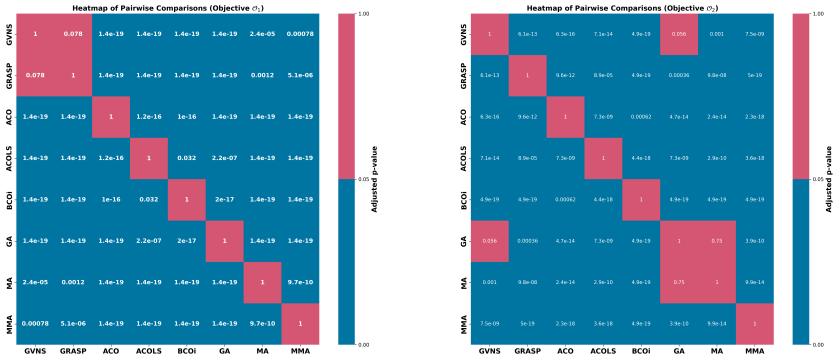

FIGURE 6.6: Heatmap of pairwise comparisons of MOHF-VRP-STW algorithms for $\mathcal{I}_2$ dataset (Mann-Whitney U test with FDR correction)

To ascertain the top-performing algorithm, we closely examined Figures 6.7 and 6.8. Figure 6.7 focuses on algorithm performance with respect to objective $\mathcal{O}_1$. Here, we excluded ACO, ACOLS, and BCOi due to their significantly lower solution quality, enhancing the visibility of the distinctions among the more effective algorithms. The insights from Figure 6.7 indicate that for objective $\mathcal{O}_1$, MMA was the standout performer, surpassing other algorithms, with GVNS, GRASP, and MA also demonstrating strong performances.

Turning to objective $\mathcal{O}_2$, the performance evaluations of the algorithms are showcased in Figure 6.8. Similar to the findings for objective $\mathcal{O}_1$, the results here also underscore MMA's superior performance, consistently outshining the other algorithms.

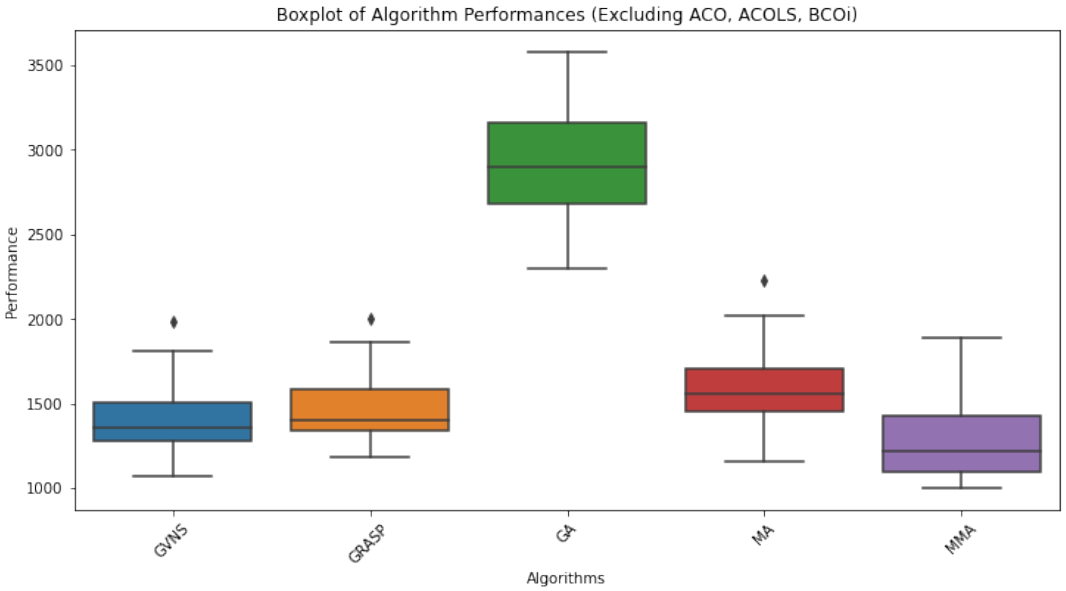

FIGURE 6.7: Boxplot of algorithm performances for the objective $\mathcal{O}_1$, with ACO, ACOLS, and BCOi excluded

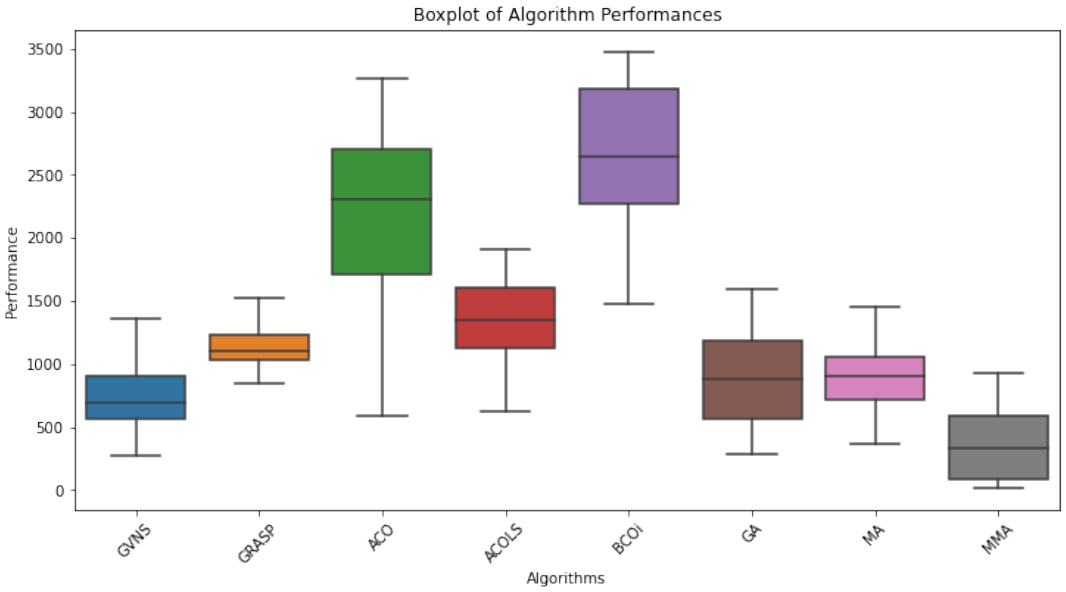

FIGURE 6.8: Boxplot of algorithm performances for the objective $\mathcal{O}_2$

Table 6.4 presents a detailed analysis of the performance of our algorithms across various metrics. This table systematically details the number of instances in which each algorithm surpassed its counterparts in effectiveness, specific to each metric. From this comprehensive overview, it is evident that MMA surpassed other meta-heuristics in 7 out of the 10 evaluated metrics.

Figures 6.9, 6.10, and 6.11 showcase illustrations of Pareto-fronts across three distinct instances, using various methods. These visual representations bolster our initial finding that the MMA method surpasses others in performance, consistently exhibiting superior Pareto-fronts.

TABLE 6.4: Performance comparison of MOHF-VRP-STW metaheuristics across various metrics (Dataset $\mathcal{I}_2$).

Metric	Method							
	GVNS	**GRASP**	**ACO**	**ACOLS**	**BCOi**	**GA**	**MA**	**MMA**
Average solution quality ($\mathcal{O}_1$)	9	0	0	0	0	0	0	**47**
Best-found solution ($\mathcal{O}_1$)	6	2	0	0	0	0	0	**50**
Worst-found solution ($\mathcal{O}_1$)	7	7	0	0	0	0	0	**42**
Average solution quality ($\mathcal{O}_2$)	1	0	0	0	0	0	0	**55**
Best-found solution ($\mathcal{O}_2$)	0	0	0	0	0	0	0	**56**
Worst-found solution ($\mathcal{O}_2$)	7	0	0	0	0	1	0	**48**
Penalty	**23**	13	0	0	0	0	21	17
Average route duration	0	0	0	0	**56**	0	0	0
Maximum route duration	15	**22**	0	0	3	0	10	6
Number of routes	9	5	0	0	0	0	0	**43**

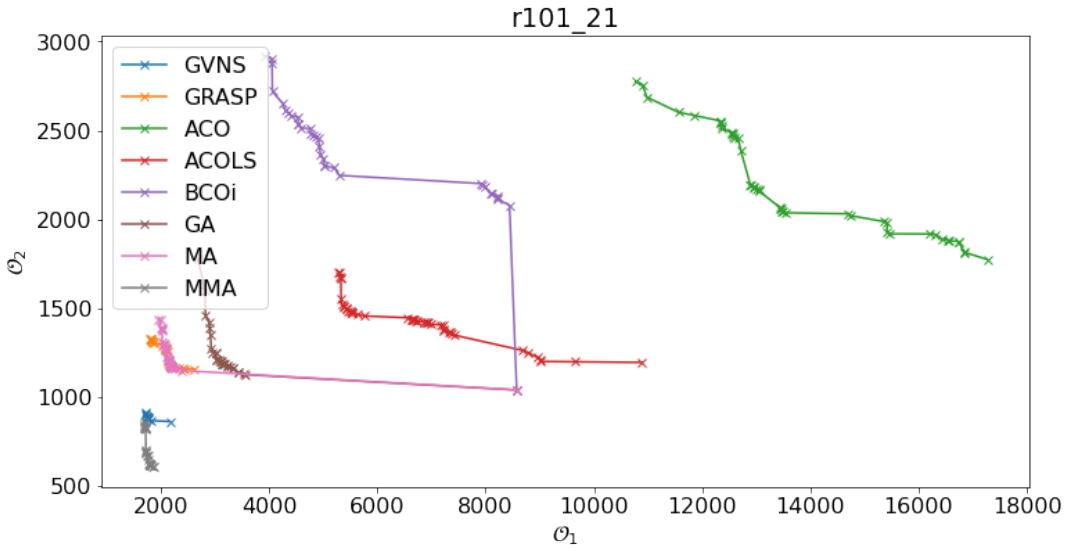

FIGURE 6.9: An illustration of Pareto-fronts: comparative analysis using different metaheuristics in instance *r101_21*.

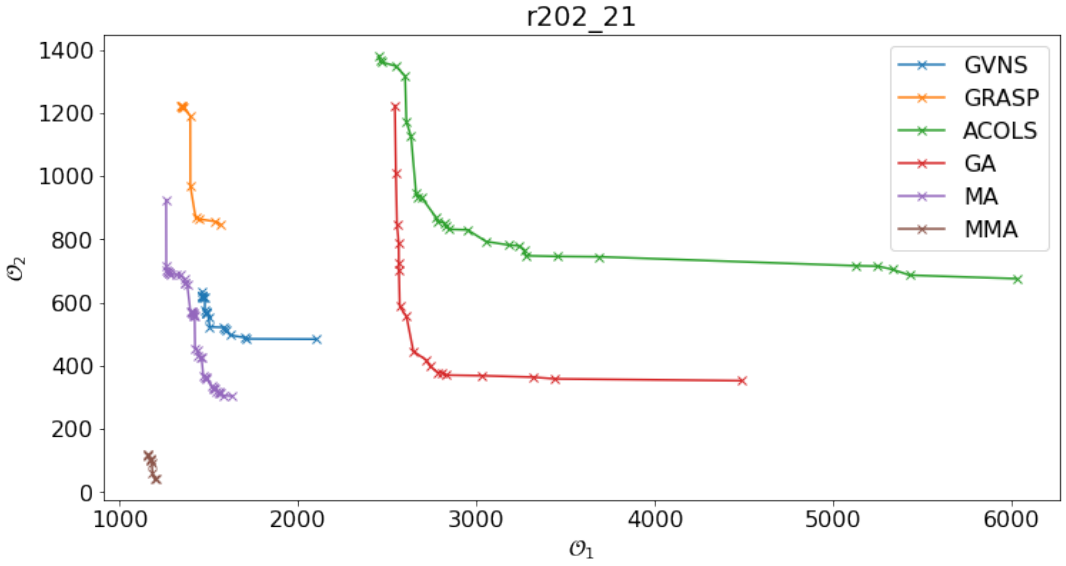

FIGURE 6.10: An illustration of Pareto-fronts: comparative analysis using different metaheuristics in instance *r202_21*.

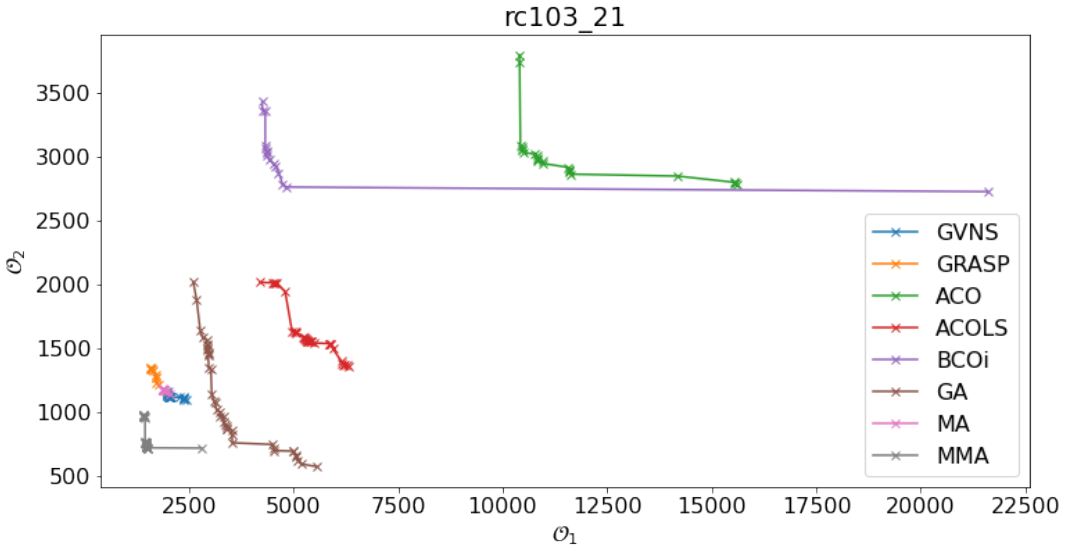

FIGURE 6.11: An illustration of Pareto-fronts: comparative analysis using different metaheuristics in instance *rc103_21*.

In conclusion, when applied to smaller instances, GVNS and ACOLS demonstrated superior performance over other methods with respect to objective $\mathcal{O}_1$. However, for objective $\mathcal{O}_2$, there was no distinctly dominant performer in these smaller instances. In contrast, on larger instances, MMA consistently excelled, outshining other metaheuristics across both objectives.

6.9.4 ACO and ACOLS comparison

In the dataset $\mathcal{I}_1$ comparison, ACOLS demonstrated superior performance over ACO in achieving both objectives. Specifically, for objective $\mathcal{O}_1$, ACOLS achieved a more

favorable average solution in 31 instances, exhibiting an average improvement of 30.53%, a minimum of 5.82%, and a maximum of 66.59%. Conversely, in 5 instances, ACOLS lagged behind, with the solution quality differing by an average of 9.49%, a minimum of 1.8%, and a maximum of 28.15%. These findings are illustrated in Figure 6.12. Regarding objective $\mathcal{O}_2$, ACOLS surpassed ACO in 33 cases, marking an average enhancement of 71.47%, with the minimum and maximum improvements being 7.99% and 191.32%, respectively. In contrast, ACOLS underperformed in 3 cases, with solution quality differences of 10.04% on average, and ranging from 6.23% to 13.89%. Additionally, ACOLS outshone ACO in terms of the best and worst solutions found, as well as in route count efficiency. However, in terms of penalties, no statistically significant differences were observed. The comparative performance of MA and MMA across various metrics is detailed in Figure 6.13.

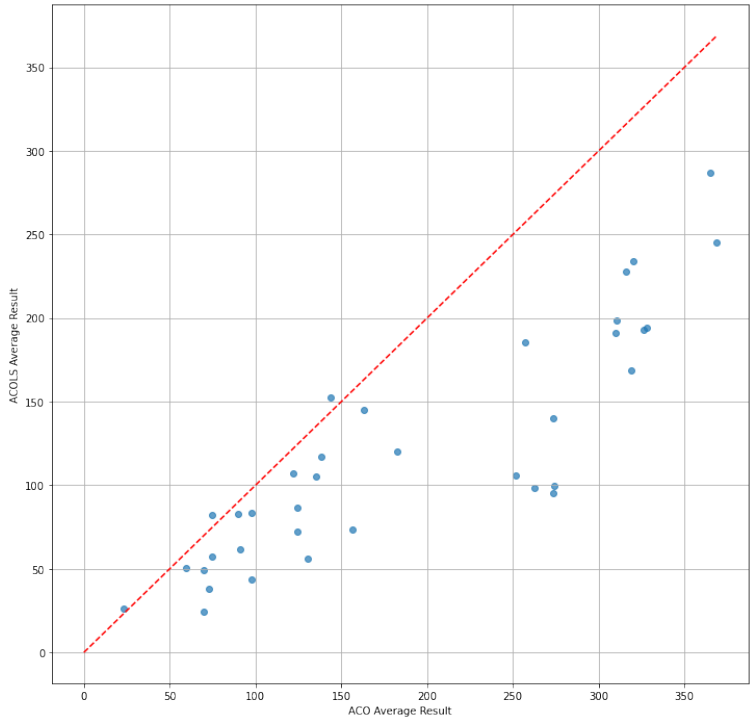

FIGURE 6.12: ACO vs ACOLS performance in terms of average solution quality on instances from $\mathcal{I}_1$

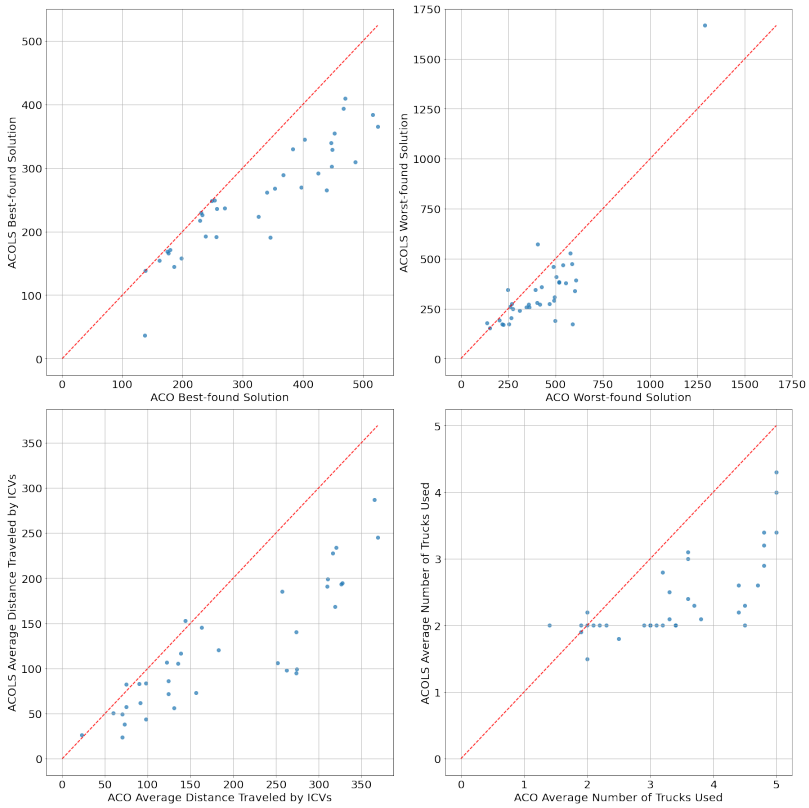

FIGURE 6.13: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from $\mathcal{I}_1$

In the dataset $\mathcal{I}_2$ evaluation, ACOLS significantly exceeded ACO's performance with respect to objective $\mathcal{O}_1$, achieving consistent superiority across all instances. The data revealed an impressive average difference of 234.24%, with the variances ranging from a minimum of 39.35% to a maximum of 953.36%. These results are clearly depicted in Figure 6.14. Regarding objective $\mathcal{O}_2$, ACOLS once again outperformed ACO, though the margin was more moderate compared to $\mathcal{O}_1$. Specifically, in 54 instances, ACOLS's performance lead was marked by an average difference of 62.63%, a minimum of 3.23%, and a maximum of 142.39%. Notably, for both objectives, the differences were not only apparent but also statistically significant, as confirmed by the Wilcoxon signed-rank test with a significance level of 0.05. Additionally, ACOLS demonstrated superior results over ACO in all other assessed metrics, as elaborated in Figure 6.15.

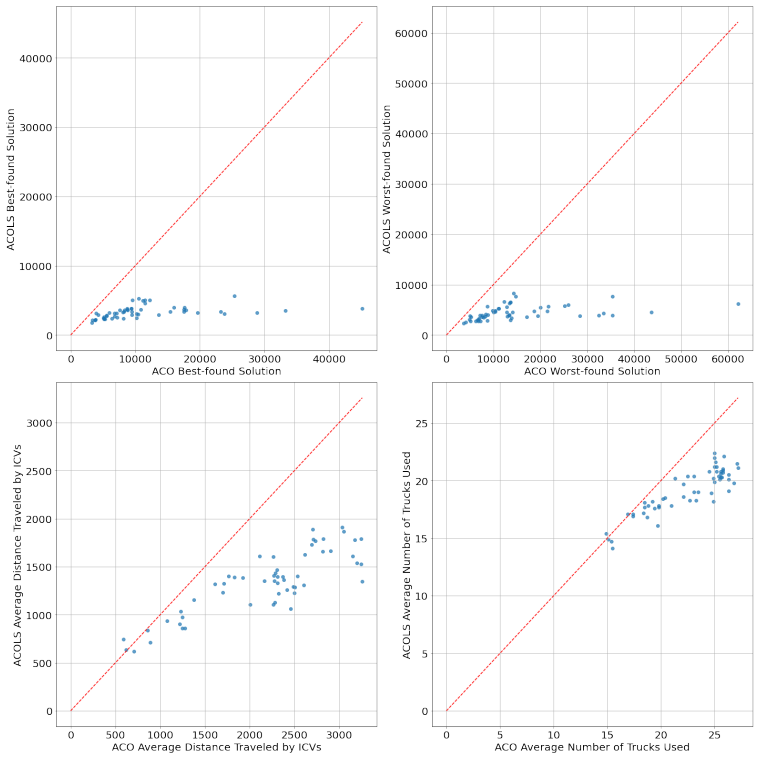

FIGURE 6.14: ACO vs ACOLS performance in terms of average solution quality on instances from $\mathcal{I}_2$

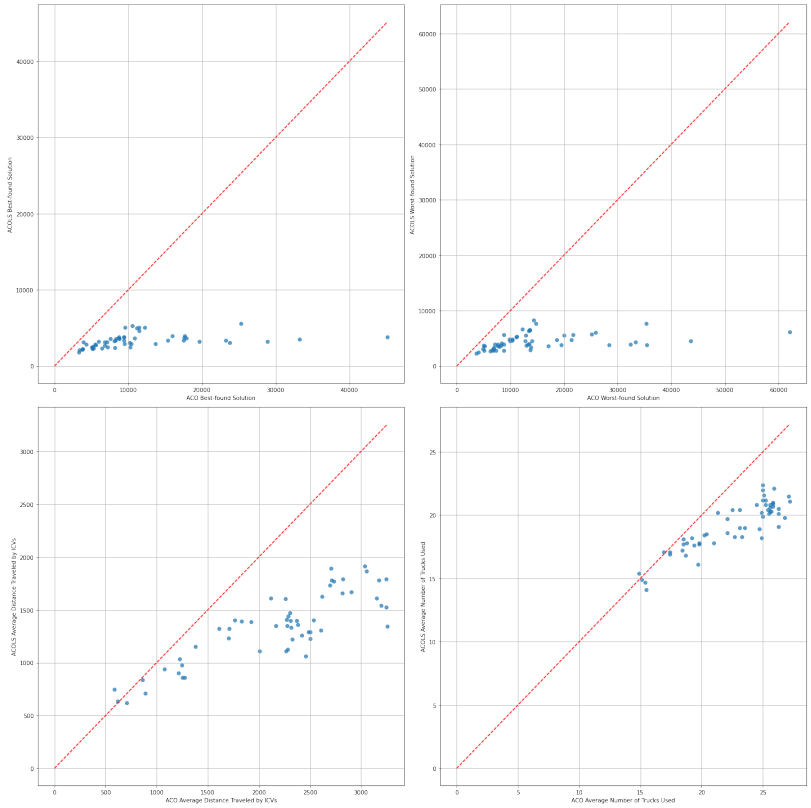

FIGURE 6.15: Comparative scatter plots showcasing various performance metrics for ACO and ACOLS on instances from $\mathcal{I}_2$

6.9.5 MA vs MMA comparison

In this section, we delve into an extensive comparative analysis between the MA and the MMA. Our focus will be to rigorously evaluate and contrast the performances of these two algorithms across the both datasets.

In the analysis of instances from dataset $\mathcal{I}_1$, MMA demonstrated superior performance to MA in 22 instances, with the differences in average solution quality across 30 runs being quite notable: an average improvement of 29.33%, a minimum of 0.77%, and a maximum of 130.12%. Conversely, in the 14 instances where MA outperformed MMA, the improvements were more modest, with an average of 2.27%, and ranging from a minimum of 0.11% to a maximum of 6.56%. This indicates that not only did MMA achieve better average solutions in more instances than MA, but also the extent of improvement was significantly larger when MMA outperformed MA compared to the reverse. A Wilcoxon signed-rank test, conducted at a significance level of 0.05, further substantiated these findings, revealing a statistically significant difference in performance between the two algorithms in terms of average solution quality, as indicated by a p-value < 0.01. The comparative average solution qualities obtained by both algorithms are illustrated in Figure 6.16.

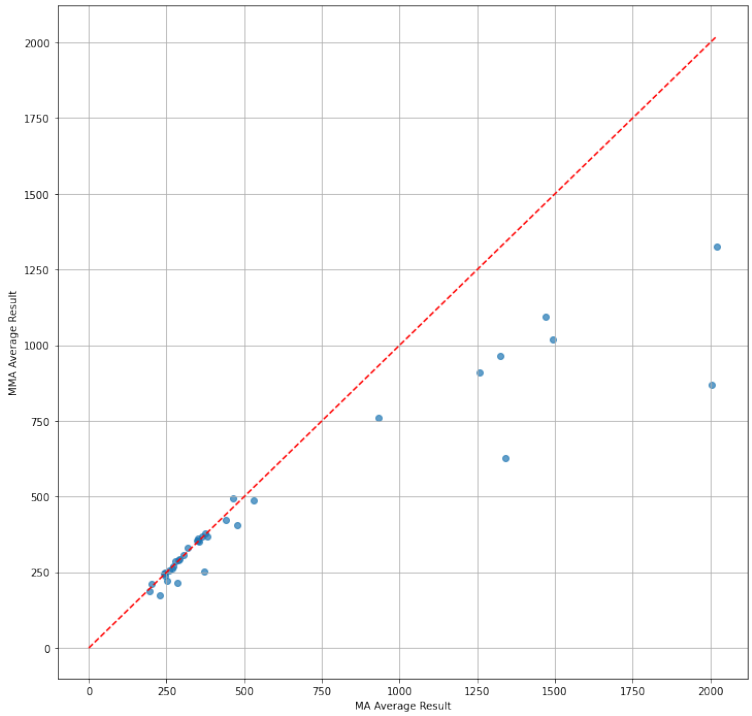

FIGURE 6.16: MA vs MMA performance in terms of average solution quality on instances from $\mathcal{I}_1$

In the comparative analysis of MA and MMA focusing on the best-found and worst-found solutions, no statistically significant differences were observed. However, a notable variance emerged in the penalties of the solutions obtained. MMA surpassed MA in 23 instances, fell short in 6, and matched the penalty value in 7. Conversely, MA typically generated solutions with a lower route count, averaging around a 12.7% reduction. A particularly interesting finding was in the context of objective $\mathcal{O}_2$, where MA outperformed MMA in 32 instances, with average, minimum, and maximum differences of 101.17%, 1.16%, and 968.55%, respectively. This observation is intriguing, considering both algorithms were assigned the same objective weights ($\alpha = 0.5$ and $\beta = 0.5$ in Equation (6.2)), yet it appears that MMA favored objective $\mathcal{O}_1$, while MA showed a preference for objective $\mathcal{O}_2$. The performance of these two algorithms across various metrics, including the distance traveled by ICVs, which relates to objective $\mathcal{O}_2$, is detailed in Figure 6.17.

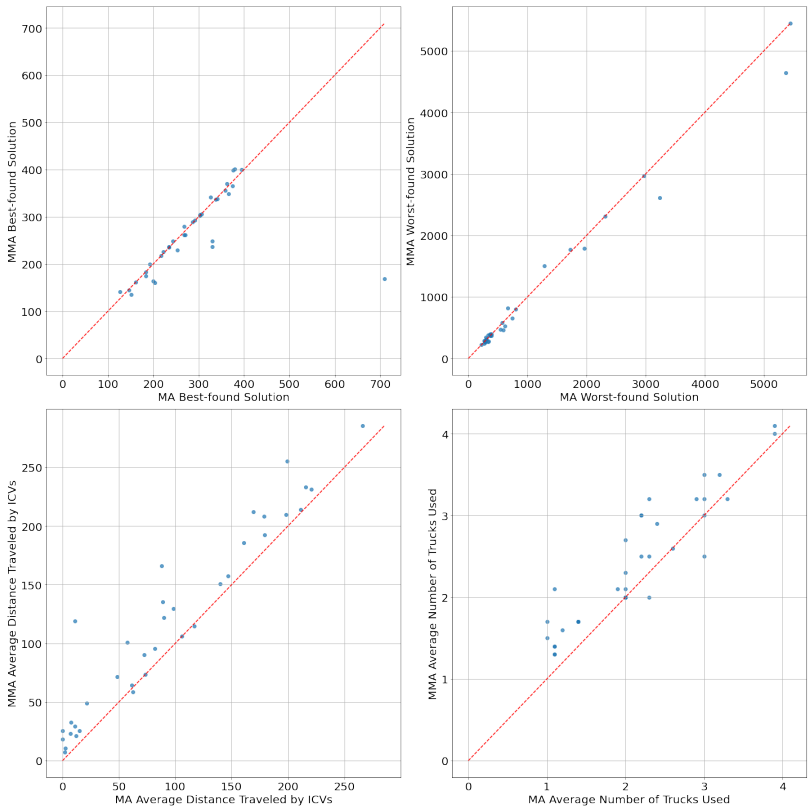

FIGURE 6.17: Comparative scatter plots showcasing various performance metrics for MA and MMA on instances from $\mathcal{I}_1$

In the evaluation using dataset $\mathcal{I}_2$, MMA consistently outperformed MA in relation to both objectives $\mathcal{O}_1$ and $\mathcal{O}_2$, across all instances. Specifically, for objective $\mathcal{O}_1$, MMA's performance advantage was marked, with an average improvement of 25.14%, a minimum of 4.02%, and a maximum of 49.97%. For objective $\mathcal{O}_2$, the difference was even more pronounced, with average, minimum, and maximum improvements of 569.63%, 34.33%, and 4967.33%, respectively. Additionally, MMA excelled over MA in terms of best-found and worst-found solutions, as well as in route count. Conversely, MA managed to outperform MMA in terms of penalties and average route duration. Figure 6.18 illustrates the comparative performance of both algorithms across all instances with respect to objective $\mathcal{O}_1$, while Figure 6.19 showcases their performance across other metrics.

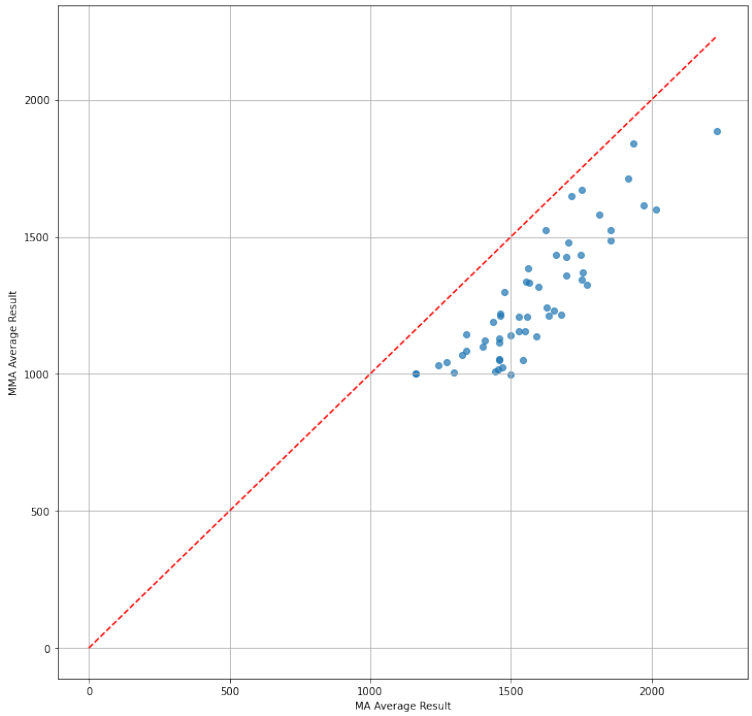

FIGURE 6.18: MA vs MMA performance in terms of average solution quality on instances from $\mathcal{O}_2$

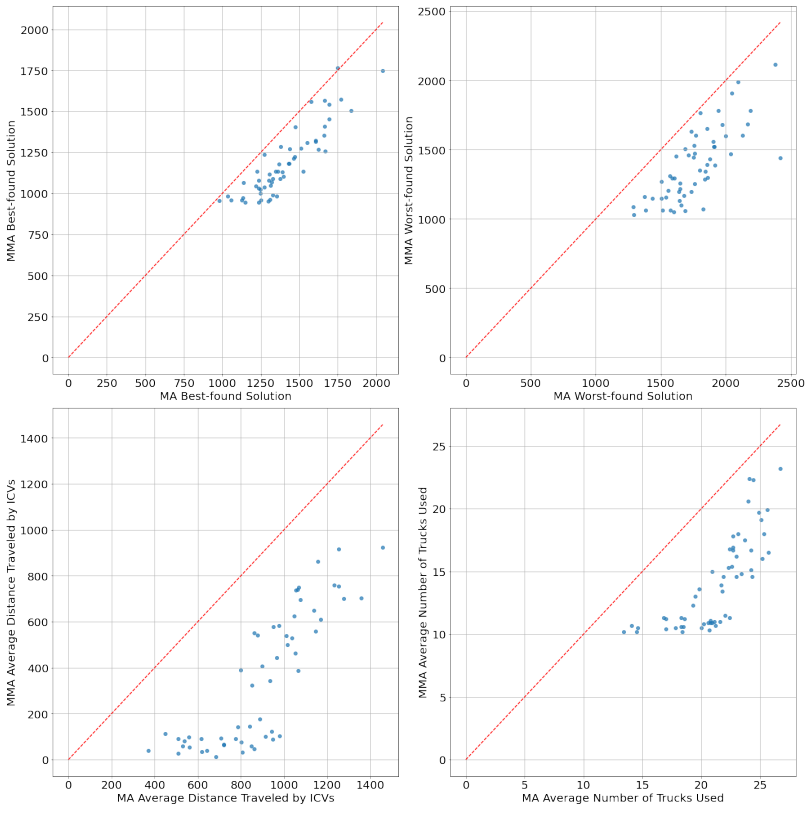

FIGURE 6.19: Comparative scatter plots showcasing various performance metrics for MA and MMA on instances from $\mathcal{O}_2$

6.10 Chapter conclusion

In this chapter, we present the final stage of our research, focusing on an extension of the problem addressed in Chapter 5. Specifically, this version considers a mixed fleet of both EVs and ICVs, along with asymmetric distances. The study adopts a multi-objective optimization approach, taking into account two simultaneous objectives.

As in previous cases, eight metaheuristics were considered. To evaluate their performance, two new sets of instances were created. The results were statistically analyzed, with MMA and GVNS yielding the best outcomes.

CHAPTER
SEVEN

CONCLUSION

7.1 Summary

Inspired by the increasing prevalence of electric vehicles globally, we delved into the vehicle routing problem tailored for this category of vehicles. Our belief is that this problem will gain greater relevance in the coming years. To ensure our research has practical industry applications, we focused on common delivery scenarios, particularly last-mile delivery in urban settings. Given the complexity of this problem – it falls into the category of NP-hard problems and is too challenging to solve exactly for larger real-world instances with numerous customers – we turned to metaheuristic algorithms. These algorithms are adept at finding high-quality solutions within a feasible timeframe, although they do not provide a guarantee of optimality.

The main contributions of this thesis include (Also illustrated in Figure 7.1):

- We have identified and defined three industry-relevant problems that, until now, have not been addressed in existing literature.
- We developed a MILP model specifically tailored for the electric vehicle routing problem, incorporating time-dependent speeds and soft time windows.
- We have compiled an exhaustive list of existing metaheuristics, surpassing the comprehensiveness of any similar compilations currently available in the literature.
- We have developed two novel sets of instances that feature asymmetric distances, a two different vehicle types (ICVs and EVs), time-dependent speeds, and varying capacities of AFS. These innovative instances are designed to serve as comprehensive benchmarks for future assessments of diverse methods applied to VRP scenarios that take these specific attributes into account.
- We have developed eight metaheuristic methods designed to address these three variations of the problem. These methods were rigorously tested: for the first two versions, we utilized benchmark instance sets available in the literature, and for the third version, we employed our uniquely created set of instances. A thorough statistical analysis was conducted to evaluate the performance of these methods, with a focus on determining the most effective approaches relative to the size of the instance and the corresponding complexity of the problem.

- We conducted a detailed analysis of the hyperparameters in metaheuristic algorithms, successfully identifying those with the most significant influence in the context of these problems.
- We developed an effective heuristic for generating initial solutions, which demonstrated considerable success. Additionally, we crafted heuristics specifically designed to optimize vehicle schedules, minimizing penalties associated with missing designated time windows.

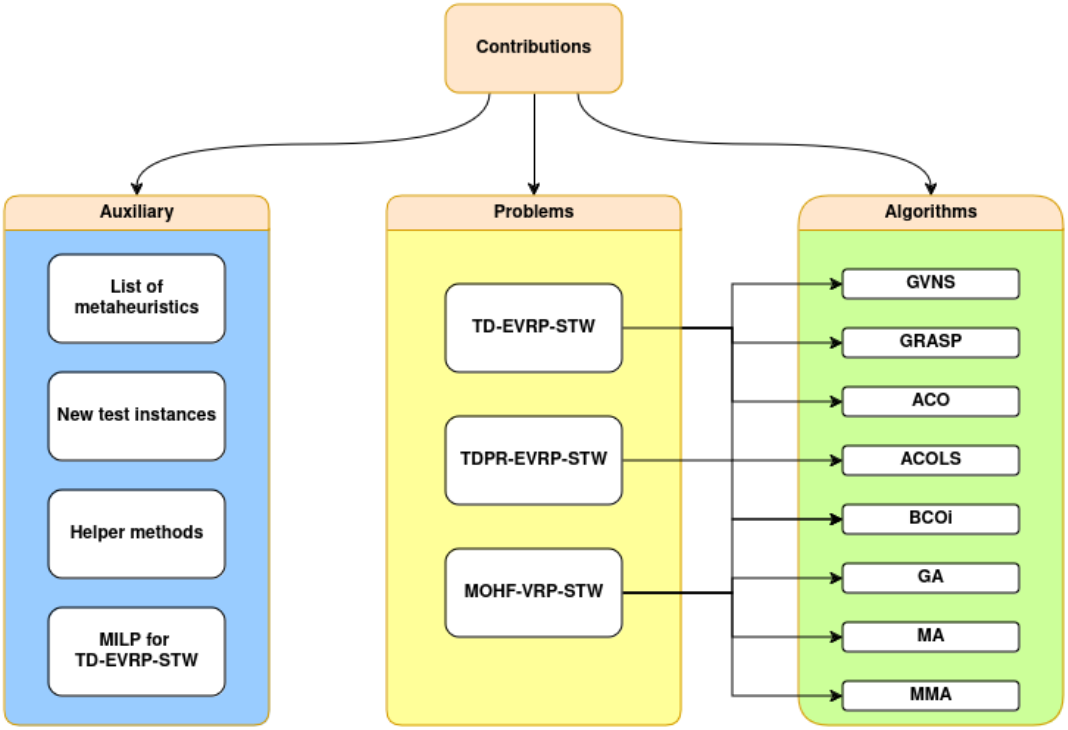

FIGURE 7.1: The contributions of this thesis

From our comprehensive testing, we concluded that for smaller instances, the performance disparity among algorithms was generally narrower compared to larger instances. For instance, in the cases of TD-EVRP-STW and TDPR-EVRP-STW, no single algorithm consistently outperformed the others. It is possible that the results might have differed with a shorter time limit, however, we determined that a 5-minute time limit was a reasonable duration for running our algorithms, within which no significant performance differences were observed.

In the MO-EVRP-STW scenario, GVNS and ACOLS showed slightly superior performance in achieving one objective (minimizing the sum of the total distance traveled and the penalty from missed time windows). Yet, when it came to the other objective (minimizing the distance traveled by ICVs), we again noticed no notable differences between the algorithms.

However, these findings were not particularly surprising as most of the instances were small enough to be solved optimality using the CPLEX commercial solver. This underlines that while our methods are effective, their advantages become more apparent in handling larger, more complex instances where exact solutions are not feasible within a limited time.

The most significant differences between the algorithms emerged during tests on larger instances, specifically those with 100 customers and 21 AFSs. In these scenarios, our modified Memetic Algorithm generally yielded the best results, closely followed by the GVNS. For the TD-EVRP-STW, MMA and GVNS demonstrated similar levels of performance, but GVNS reached the optimal solution more rapidly. Interestingly, we noted that GVNS often favored solutions with fewer routes, an advantage in situations where fleet size is a critical constraint.

In the TDPR-EVRP-STW scenario, MMA once again stood out for its superior performance, clearly distinguishing itself from other algorithms. In the multi-objective version of the problem, MMA continued to outperform other algorithms, with GVNS coming in as a close second. The dominance of MMA was even more pronounced in achieving objective $\mathcal{O}_2$, where it found the best solution in all but one instance, underlining its efficacy in handling the complexities of larger-scale problems.

7.2 Future work

There are numerous exciting prospects for future research, each offering a valuable chance to either broaden the scope of application for the problem at hand or discover more effective methods for solving it. These potential areas include, but are not limited to:

- While our study aimed to include a comprehensive range of VRP attributes relevant to last-mile delivery, it is important to recognize that different scenarios might prioritize various other attributes. For instance, the Multi-Compartment attribute is particularly crucial in industries like food delivery, where maintaining distinct temperatures for various products is essential. Similarly, the Multi-Trip attribute gains significance in scenarios involving dynamic requests, necessitating mid-route returns to the depot. Although there is an allure in attempting to amalgamate all these attributes into a single, rich VRP to encapsulate a wide array of realistic situations, caution is advised. Such a generalized version of the problem can become exceedingly complex to optimize. In many cases, solutions tailored to specific problems may yield more efficient and effective outcomes.

- There remains a vast array of unexplored metaheuristics in the field (along with those in forests, oceans, and the air). While it is impractical to test every metaheuristic approach for each specific problem, experimenting with some renowned methods, such as the Gray Wolf Optimizer, Bat Algorithm, or Spotted Hyena Optimizer, could potentially lead to satisfactory outcomes.

- As previously mentioned, the charging of an electric vehicle's battery in real-world scenarios is not linear. Although incorporating this non-linearity into our methods is straightforward, it would be interesting to compare the quality of solutions derived under these realistic conditions with those from our idealized scenario.

- Much like the previous point, the energy consumption model of electric vehicles is not a straightforward linear process. It varies based on multiple factors, including the vehicle's load, road inclination, speed, etc. Implementing this non-linear aspect into our methods, as in the previous case, is feasible. The use of different consumption functions could lead to varied outcomes, such as the need for an EV to stop for charging at an AFS either earlier or later than predicted

in our idealized model. By examining several functions and comparing these results with real-world data, we can gain valuable insights into which model most accurately predicts energy consumption. This, in turn, would lead to the development of more practical and reliable methods for managing EV energy use.

- As outlined in Section 3.4, a variety of objectives can be taken into account while optimizing the routes of EVs and ICVs. By strategically guiding our metaheuristics with a single objective while simultaneously assessing multiple objectives, we could potentially uncover the correlations among these objectives. This approach might provide a deeper understanding of how optimizing for one aspect influences the others, offering a more holistic perspective in route optimization.
- In our idealized model, we assume that each AFS is equipped with an unlimited number of chargers, all employing identical charging technology. However, this scenario is quite far from reality. In actuality, each AFS has a finite number of charging stations, potentially featuring varying charging technologies and speeds. Consequently, EVs need to consider the availability of these chargers. When AFSs are owned by the delivery company, this information is readily accessible to all vehicles. However, the situation becomes more complex with public AFSs, where predicting charger availability is challenging. While we have not yet implemented and tested this more realistic scenario, in the creation of our new instance sets $\mathcal{I}_1$ and $\mathcal{I}_2$, we incorporated data regarding the number of chargers at each AFS. It is vital for future research to delve deeper into this scenario, exploring the intricacies and potential strategies for effectively managing EV charging in a setting with limited and varied charging resources.
- In recent years, the application of *Artificial Intelligence* (**AI**) across diverse problem domains has been a growing trend. Consequently, it is worthwhile to explore leveraging the capabilities of AI in two potential ways:
  - We can enhance metaheuristic methods by integrating *Machine Learning* (**ML**) principles, creating methods known as *Learnheuristics*. These hybrid methods are gaining popularity, yet many metaheuristics have yet to fully harness the capabilities of ML. For instance, ML can be applied to dynamically adjust hyperparameters, or reinforcement learning could be employed to select the most effective neighborhoods for local search in each iteration. Such integration has the potential not only to boost the efficiency of our methods but also to reduce the necessity for meticulous tuning, thereby streamlining the optimization process.Learnheuristics have demonstrated promising results, as evidenced in studies [28, 56, 155].
  - AI can be leveraged to forecast information that is typically challenging to ascertain. For example, AI could potentially predict the availability of charging stations, a key factor previously mentioned. Similarly, it could be used to estimate energy consumption, average vehicle speed at any given time, or even anticipate the volume of additional requests in dynamic scenarios. Incorporating AI in this manner allows metaheuristics to utilize more realistic and accurate data, resulting in solutions that are more applicable in real-world contexts.
- Our observations from testing various methods revealed that those incorporating local search generally outperformed those without it. Specifically, ACOLS

showed superior results compared to ACO, and both MA and MMA were more effective than GA. This pattern strongly suggests that local search is a critical component of our methodologies. Given its apparent importance, it would be beneficial to explore different local search techniques in this context. One such promising method to consider is the Path Relinking algorithm [109], which could potentially enhance the performance of our approaches even further.

- The concept of time-dependent speeds enables us to simulate varying traffic conditions at different times of the day. However, there is potential to expand this concept further. The average speed achievable by a vehicle depends not just on the time of day but also on its location. For instance, traffic patterns in tourist areas and business districts of a city may differ significantly. Consequently, the TD attribute could be evolved into a more comprehensive model: *Time and Place Dependent Speeds*. This approach would incorporate both spatial and temporal factors, offering a more nuanced and accurate representation of speed variations based on time and specific urban locations.

BIBLIOGRAPHY

[1] Abdel-Basset, Mohamed, Abdel-Fatah, Laila, and Sangaiah, Arun Kumar. "Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review". In: *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Ed. by Sangaiah, Arun Kumar, Sheng, Michael, and Zhang, Zhiyong. Intelligent Data-Centric Systems. Academic Press, 2018, pp. 185–231. ISBN: 978-0-12-813314-9. DOI: https://doi.org/10.1016/B978-0-12-813314-9.00010-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128133149000104.

[2] Abreu, Levi R, Cunha, Jesus O, Prata, Bruno A, and Framinan, Jose M. "A genetic algorithm for scheduling open shops with sequence-dependent setup times". In: *Computers & Operations Research* 113 (2020), p. 104793. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.104793. URL: https://www.sciencedirect.com/science/article/pii/S0305054819302357.

[3] Adiba, ElBouzekri Elidrissi, Aahmed, ElHilali Alaoui, and Youssef, Benadada. "The green capacitated vehicle routing problem: Optimizing of emissions of greenhouse gas". In: *2014 International Conference on Logistics Operations Management*. 2014, pp. 161–167. DOI: 10.1109/GOL.2014.6887434.

[4] Affenzeller, Michael, Wagner, Stefan, Winkler, Stephan, and Beham, Andreas. *Genetic algorithms and genetic programming: modern concepts and practical applications*. Crc Press, 2009.

[5] Affi, Mannoubia, Derbel, Houda, and Jarboui, Bassem. "Variable neighborhood search algorithm for the green vehicle routing problem". In: *International Journal of Industrial Engineering Computations* 9.2 (2018), pp. 195–204.

[6] Akobeng, Anthony K. "Understanding type I and type II errors, statistical power and sample size". In: *Acta Paediatrica* 105.6 (2016), pp. 605–609. DOI: https://doi.org/10.1111/apa.13384. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/apa.13384.

[7] Alam, Tanweer, Qamar, Shamimul, Dixit, Amit, and Benaida, Mohamed. "Genetic algorithm: Reviews, implementations, and applications". In: *arXiv preprint arXiv:2007.12673* (2020).

[8] Almoustafa, Samira, Hanafi, Said, and Mladenović, Nenad. "New exact method for large asymmetric distance-constrained vehicle routing problem". In: *European Journal of Operational Research* 226.3 (2013), pp. 386–394. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.11.040. URL: https://www.sciencedirect.com/science/article/pii/S0377221712008922.

[9] Amjad, Shaik, Neelakrishnan, S, and Rudramoorthy, R. "Review of design considerations and technological challenges for successful development and deployment of plug-in hybrid electric vehicles". In: *Renewable and Sustainable Energy Reviews* 14.3 (2010), pp. 1104–1110. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2009.11.001. URL: https://www.sciencedirect.com/science/article/pii/S1364032109002639.

[10] Andelmin, Juho and Bartolini, Enrico. "A multi-start local search heuristic for the green vehicle routing problem based on a multigraph reformulation". In: *Computers & Operations Research* 109 (2019), pp. 43–63.

[11] Andelmin, Juho and Bartolini, Enrico. "An exact algorithm for the green vehicle routing problem". In: *Transportation Science* 51.4 (2017), pp. 1288–1303.

[12] Anderson, Larry G. "Effects of biodiesel fuels use on vehicle emissions". In: *Journal of Sustainable Energy & Environment* 3 (2012), pp. 35–47.

[13] Anokić, Ana, Stanimirović, Zorica, Davidović, Tatjana, and Stakić, Đorđe. "Variable neighborhood search based approaches to a vehicle scheduling problem in agriculture". In: *International Transactions in Operational Research* 27.1 (2020), pp. 26–56. DOI: https://doi.org/10.1111/itor.12480. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12480.

[14] Aranha, Claus, Camacho Villalón, Christian L, Campelo, Felipe, Dorigo, Marco, Ruiz, Rubén, Sevaux, Marc, Sörensen, Kenneth, and Stützle, Thomas. "Metaphor-based metaheuristics, a call for action: the elephant in the room". In: *Swarm Intelligence* 16.1 (2022), pp. 1–6. DOI: https://doi.org/10.1007/s11721-021-00202-9.

[15] Archetti, Claudia, Savelsbergh, Martin WP, and Speranza, M Grazia. "Worst-case analysis for split delivery vehicle routing problems". In: *Transportation science* 40.2 (2006), pp. 226–234. DOI: https://doi.org/10.1287/trsc.1050.0117.

[16] Archetti, Claudia and Speranza, Maria Grazia. "Vehicle routing problems with split deliveries". In: *International transactions in operational research* 19.1-2 (2012), pp. 3–22. DOI: https://doi.org/10.1111/j.1475-3995.2011.00811.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2011.00811.x.

[17] Arora, Sankalap and Singh, Satvir. "Butterfly optimization algorithm: a novel approach for global optimization". In: *Soft Computing* 23.3 (2019), pp. 715–734.

[18] Asghari, Mohammad, Fathollahi-Fard, Amir M., hashem, S. M. J. Mirzapour Al-e, and Dulebenets, Maxim A. "Transformation and Linearization Techniques in Optimization: A State-of-the-Art Survey". In: *Mathematics* 10.2 (2022). ISSN: 2227-7390. DOI: 10.3390/math10020283. URL: https://www.mdpi.com/2227-7390/10/2/283.

[19] Asghari, Mohammad and Mirzapour Al-e-hashem, S. Mohammad J. "Green vehicle routing problem: A state-of-the-art review". In: *International Journal of Production Economics* 231 (2021), p. 107899.

[20] Attanasio, Andrea, Cordeau, Jean-François, Ghiani, Gianpaolo, and Laporte, Gilbert. "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem". In: *Parallel Computing* 30.3 (2004), pp. 377–387. ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2003.12.001. URL: https://www.sciencedirect.com/science/article/pii/S0167819104000043.

[21] Augerat, Philippe. *VRP problem instances set A-B-P*. Last accessed 21 May 2023. 1995. URL: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.

[22] Avci, Mustafa and Topaloglu, Seyda. "A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery". In: *Expert Systems with Applications* 53 (2016), pp. 160–171. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.01.038. URL: https://www.sciencedirect.com/science/article/pii/S0957417416000610.

[23] Ayadi, Rajaà, ElIdrissi, Adiba ELBouzekri, Benadada, Youssef, and El Hilali Alaoui, Ahmed. "Evolutionary algorithm for a Green vehicle routing problem with multiple trips". In: *2014 International Conference on Logistics Operations Management*. 2014, pp. 148–154. DOI: 10.1109/GOL.2014.6887432.

[24] Azhaganathan, Gurusamy and Bragadeshwaran, Ashok. "Critical review on recent progress of ethanol fuelled flex-fuel engine characteristics". In: *International Journal of Energy Research* 46.5 (2022), pp. 5646–5677. DOI: https://doi.org/10.1002/er.7610. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.7610.

[25] Baldacci, Roberto, Mingozzi, Aristide, and Roberti, Roberto. "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints". In: *European Journal of Operational Research* 218.1 (2012), pp. 1–6. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2011.07.037. URL: https://www.sciencedirect.com/science/article/pii/S0377221711006692.

[26] Barth, Matthew, Younglove, Theodore, and Scora, George. *Development of a Heavy-Duty Diesel Modal Emissions and Fuel Consumption Model*. Tech. rep. UC Berkeley: California Partners for Advanced Transportation Technology, 2005. URL: https://escholarship.org/uc/item/67f0v3zf.

[27] Baykasoğlu, Adil, Yunusoglu, Mualla Gonca, and Özsoydan, F Burcin. "A GRASP based solution approach to solve cardinality constrained portfolio optimization problems". In: *Computers & Industrial Engineering* 90 (2015), pp. 339–351. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2015.10.009. URL: https://www.sciencedirect.com/science/article/pii/S0360835215004027.

[28] Bayliss, Christopher, Juan, Angel A., Currie, Christine S.M., and Panadero, Javier. "A learnheuristic approach for the team orienteering problem with aerial drone motion constraints". In: *Applied Soft Computing* 92 (2020), p. 106280. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.106280. URL: https://www.sciencedirect.com/science/article/pii/S1568494620302209.

[29] Bazgan, Cristina, Hassin, Refael, and Monnot, Jérôme. "Approximation algorithms for some vehicle routing problems". In: *Discrete Applied Mathematics* 146.1 (2005), pp. 27–42. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2004.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0166218X04002860.

[30] Beaudry, Alexandre, Laporte, Gilbert, Melo, Teresa, and Nickel, Stefan. "Dynamic transportation of patients in hospitals". In: *OR spectrum* 32 (2010), pp. 77–107. DOI: https://doi.org/10.1007/s00291-008-0135-6.

[31] Bektaş, Tolga and Laporte, Gilbert. "The pollution-routing problem". In: *Transportation Research Part B: Methodological* 45.8 (2011), pp. 1232–1250.

[32] Benjamini, Yoav and Hochberg, Yosef. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1 (1995), pp. 289–300.

[33] Bhattacharjee, Prattusha, Ahmed, Nafi, Akbar, MD Shaiful, and Habib, MD Saimum. "An Efficient Ant Colony Algorithm for Multi-Depot Heterogeneous Fleet Green Vehicle Routing Problem". In: *5th NA International Conference on Industrial Engineering and Operations Management, Detroit, Michigan, USA*. IEOM Society International. 2020, pp. 1337–1348.

[34] Birattari, Mauro, Paquete, Luis, Stützle, Thomas, and Varrentrapp, Klaus. "Classification of metaheuristics and design of experiments for the analysis of components". In: *Teknik Rapor, AIDA-01-05* (2001).

[35] Birattari, Mauro, Stützle, Thomas, Paquete, Luis, Varrentrapp, Klaus, et al. "A Racing Algorithm for Configuring Metaheuristics." In: *Gecco*. Vol. 2. 2002. Citeseer. 2002.

[36] Birattari, Mauro, Yuan, Zhi, Balaprakash, Prasanna, and Stützle, Thomas. "F-Race and Iterated F-Race: An Overview". In: (2010). Ed. by Bartz-Beielstein, Thomas, Chiarandini, Marco, Paquete, Luís, and Preuss, Mike, pp. 311–336. DOI: 10.1007/978-3-642-02538-9_13. URL: https://doi.org/10.1007/978-3-642-02538-9_13.

[37] Bonferroni, Carlo Emilio. "Teoria statistica delle classi e calcolo delle probabilità". In: *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* (1936), pp. 3–62.

[38] Börjesson, Pål and Mattiasson, Bo. "Biogas as a resource-efficient vehicle fuel". In: *Trends in biotechnology* 26.1 (2008), pp. 7–13. DOI: https://doi.org/10.1016/j.tibtech.2007.09.007.

[39] Boyd, Stephen P and Vandenberghe, Lieven. *Convex optimization*. Cambridge university press, 2004.

[40] Bräysy, Olli and Gendreau, Michel. "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms". In: *Transportation science* 39.1 (2005), pp. 104–118. DOI: https://doi.org/10.1287/trsc.1030.0056.

[41] Breiman, Leo. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[42] Brimberg, Jack, Mladenović, Nenad, Todosijević, Raca, and Urošević, Dragan. "Solving the capacitated clustering problem with variable neighborhood search". In: *Annals of Operations Research* 272 (2019), pp. 289–321. DOI: https://doi.org/10.1007/s10479-017-2601-5.

[43] Brimberg, Jack, Salhi, Said, Todosijević, Raca, and Urošević, Dragan. "Variable neighborhood search: The power of change and simplicity". In: *Computers & Operations Research* 155 (2023), p. 106221.

[44] Bruglieri, Maurizio, Pezzella, Ferdinando, Pisacane, Ornella, and Suraci, Stefano. "A variable neighborhood search branching for the electric vehicle routing problem with time windows". In: *Electronic Notes in Discrete Mathematics* 47 (2015), pp. 221–228.

[45] Caprara, Alberto, Toth, Paolo, and Fischetti, Matteo. "Algorithms for the set covering problem". In: *Annals of Operations Research* 98.1-4 (2000), pp. 353–371. DOI: https://doi.org/10.1023/A:1019225027893.

[46] Cardelino, Carlos. "Daily variability of motor vehicle emissions derived from traffic counter data". In: *Journal of the Air & Waste Management Association* 48.7 (1998), pp. 637–645. DOI: https://doi.org/10.1080/10473289.1998.10463709.

[47] Carić, Tonči and Gold, Hrvoje. *Vehicle Routing Problem*. Rijeka: IntechOpen, 2008. DOI: 10.5772/64. URL: https://doi.org/10.5772/64.

[48] Cheng, Tai C Edwin, Peng, Bo, and Lü, Zhipeng. "A hybrid evolutionary algorithm to solve the job shop scheduling problem". In: *Annals of Operations Research* 242.2 (2016), pp. 223–237.

[49] Cheung, BK-S, Langevin, André, and Villeneuve, B. "High performing evolutionary techniques for solving complex location problems in industrial system design". In: *Journal of Intelligent Manufacturing* 12.5 (2001), pp. 455–466.

[50] Chopard, Bastien and Tomassini, Marco. *An introduction to metaheuristics for optimization*. Springer, 2018.

[51] Christofides, Nicos, Mingozzi, Aristide, and Toth, Paolo. "The vehicle routing problem." In: *Combinatorial Optimization* (1979), pp. 315–338.

[52] Clarke, Geoff and Wright, John W. "Scheduling of vehicles from a central depot to a number of delivery points". In: *Operations research* 12.4 (1964), pp. 568–581.

[53] Contardo, Claudio and Martinelli, Rafael. "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". In: *Discrete Optimization* 12 (2014), pp. 129–146. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2014.03.001. URL: https://www.sciencedirect.com/science/article/pii/S1572528614000097.

[54] Cooray, PLNU and Rupasinghe, Thashika D. "Machine learning-based parameter tuned genetic algorithm for energy minimizing vehicle routing problem". In: *Journal of Industrial Engineering* 2017 (2017), p. 3019523. DOI: 10.1155/2017/3019523.

[55] Costa, Paulo Roberto de Oliveira da, Mauceri, Stefano, Carroll, Paula, and Pallonetto, Fabiano. "A genetic algorithm for a green vehicle routing problem". In: *Electronic notes in discrete mathematics* 64 (2018). 8th International Network Optimization Conference - INOC 2017, pp. 65–74.

[56] Crawford, Broderick, Soto, Ricardo, Lemus-Romani, José, Becerra-Rozas, Marcelo, Lanza-Gutiérrez, José M., Caballé, Nuria, Castillo, Mauricio, Tapia, Diego, Cisternas-Caneo, Felipe, García, José, Astorga, Gino, Castro, Carlos, and Rubio, José-Miguel. "Q-Learnheuristics: Towards Data-Driven Balanced Metaheuristics". In: *Mathematics* 9.16 (2021). ISSN: 2227-7390. DOI: 10.3390/math9161839. URL: https://www.mdpi.com/2227-7390/9/16/1839.

[57] Curtin, Richard, Shrago, Yevgeny, and Mikkelsen, Jamie. *Plug-in hybrid electric vehicles*. Tech. rep. Reuters/University of Michigan, Surveys of Consumers, 2009. URL: http://www.emic-bg.org/files/files/Plug_In_Hybrid_Electric_Vehicles.pdf.

[58] Daggett, RB. "The electric vehicle as a corrector of poor load-factor". In: *American Institute of Electrical Engineers* 28.11 (1909), pp. 12–13.

[59] Daham, Hajem Ati and Mohammed, Husam Jasim. "An evolutionary algorithm approach for vehicle routing problems with backhauls". In: *Mater. Today Proc* (2021). DOI: https://doi.org/10.1016/j.matpr.2020.12.1028.

[60] Dai, Qin and Liu, Jianbo. "Application of ant colony optimization (ACO) algorithm to remote sensing image classification". In: *MIPPR 2007: Pattern Recognition and Computer Vision*. Ed. by Maybank, S. J., Ding, Mingyue, Wahl, F., and Zhu, Yaoting. Vol. 6788. International Society for Optics and Photonics. SPIE, 2007, 67881A. DOI: 10.1117/12.749344. URL: https://doi.org/10.1117/12.749344.

[61] Dantzig, George B and Ramser, John H. "The truck dispatching problem". In: *Management science* 6.1 (1959), pp. 80–91.

[62] Dardiotis, Christos, Fontaras, Georgios, Marotta, Alessandro, Martini, Giorgio, and Manfredi, Urbano. "Emissions of modern light duty ethanol flex-fuel vehicles over different operating and environmental conditions". In: *Fuel* 140 (2015), pp. 531–540. ISSN: 0016-2361. DOI: https://doi.org/10.1016/j.fuel.2014.09.085. URL: https://www.sciencedirect.com/science/article/pii/S0016236114009545.

[63] Davidović, Tatjana, Ramljak, Dušan, Šelmić, Milica, and Teodorović, Dušan. "Bee colony optimization for the p-center problem". In: *Computers & Operations Research* 38.10 (2011), pp. 1367–1376. DOI: https://doi.org/10.1016/j.cor.2010.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0305054810002923.

[64] Davidović, Tatjana, Šelmić, Milica, Teodorović, Dušan, and Ramljak, Dušan. "Bee colony optimization for scheduling independent tasks to identical processors". In: *Journal of heuristics* 18 (2012), pp. 549–569. DOI: https://doi.org/10.1007/s10732-012-9197-3.

[65] Davidović, Tatjana, Teodorović, Dušan, and Šelmić, Milica. "Bee colony optimization Part I: The algorithm overview". In: *Yugoslav Journal of Operations Research* 25.1 (2015), pp. 33–56.

[66] Davidović, Tatjana. "Scheduling Tasks to Multiprocessor Systems by Applying Metaheuristics". In Serbian. PhD thesis. University of Belgrade, Faculty of Mathematics, 2006.

[67] Deb, Kalyanmoy, Agrawal, Samir, Pratap, Amrit, and Meyarivan, T. "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II". In: *Parallel Problem Solving from Nature PPSN VI*. Ed. by Schoenauer, Marc, Deb, Kalyanmoy, Rudolph, Günther, Yao, Xin, Lutton, Evelyne, Merelo, Juan Julian, and Schwefel, Hans-Paul. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 849–858. ISBN: 978-3-540-45356-7.

[68] Deb, Kalyanmoy, Pratap, Amrit, Agarwal, Sameer, and Meyarivan, TAMT. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[69] Decerle, Jérémy, Grunder, Olivier, El Hassani, Amir Hajjam, and Barakat, Oussama. "A memetic algorithm for a home health care routing and scheduling problem". In: *Operations research for health care* 16 (2018), pp. 59–71. ISSN: 2211-6923. DOI: https://doi.org/10.1016/j.orhc.2018.01.004. URL: https://www.sciencedirect.com/science/article/pii/S2211692317300735.

[70] Delorme, Xavier, Gandibleux, Xavier, and Rodriguez, Joaquin. "GRASP for set packing problems". In: *European Journal of Operational Research* 153.3 (2004). EURO Young Scientists, pp. 564–580. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(03)00263-7. URL: https://www.sciencedirect.com/science/article/pii/S0377221703002637.

[71] Dewi, Shanty Kusuma and Utama, Dana Marsetiya. "A new hybrid whale optimization algorithm for green vehicle routing problem". In: *Systems Science & Control Engineering* 9.1 (2021), pp. 61–72.

[72] Di Martino, Andrea, Miraftabzadeh, Seyed Mahdi, and Longo, Michela. "Strategies for the Modelisation of Electric Vehicle Energy Consumption: A Review". In: *Energies* 15.21 (2022). ISSN: 1996-1073. DOI: 10.3390/en15218115. URL: https://www.mdpi.com/1996-1073/15/21/8115.

[73] Dietz, Audrey, Adams, Warren, and Yang, Boshi. "A conditional-logic interpretation for Miller–Tucker–Zemlin inequalities and extensions". In: *Optimization Letters* 17.2 (2023), pp. 245–264. DOI: https://doi.org/10.1007/s11590-022-01947-w.

[74] Dorigo, Marco. "Optimization, learning and natural algorithms". PhD thesis. Politecnico di Milano, 1992.

[75] Dorigo, Marco and Blum, Christian. "Ant colony optimization theory: A survey". In: *Theoretical Computer Science* 344.2 (2005), pp. 243–278. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2005.05.020. URL: https://www.sciencedirect.com/science/article/pii/S0304397505003798.

[76] Dorigo, Marco, Maniezzo, Vittorio, and Colorni, Alberto. "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41. DOI: 10.1109/3477.484436.

[77] Dorigo, Marco, Maniezzo, Vittorio, and Colorni, Alberto. *The ant system: An autocatalytic optimizing process*. Tech. rep. Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1991.

[78] Dorigo, Marco and Stützle, Thomas. "Ant Colony Optimization: Overview and Recent Advances". In: *Handbook of Metaheuristics*. Ed. by Gendreau, Michel and Potvin, Jean-Yves. Cham: Springer International Publishing, 2019, pp. 311–351. ISBN: 978-3-319-91086-4. DOI: 10.1007/978-3-319-91086-4_10. URL: https://doi.org/10.1007/978-3-319-91086-4_10.

[79] Dunn, Olive Jean. "Multiple Comparisons Using Rank Sums". In: *Technometrics* 6.3 (1964), pp. 241–252. DOI: 10.1080/00401706.1964.10490181. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1964.10490181.

[80] Eglese, Richard and Bektaş, Tolga. "Chapter 15: Green Vehicle Routing". In: *Vehicle Routing*. SIAM, pp. 437–458. DOI: 10.1137/1.9781611973594.ch15. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611973594.ch15. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973594.ch15.

[81] Ehsani, Mehrdad, Gao, Yimin, and Miller, John M. "Hybrid Electric Vehicles: Architecture and Motor Drives". In: *Proceedings of the IEEE* 95.4 (2007), pp. 719–728. DOI: 10.1109/JPROC.2007.892492.

[82] Ehsani, Mehrdad, Singh, Krishna Veer, Bansal, Hari Om, and Mehrjardi, Ramin Tafazzoli. "State of the Art and Trends in Electric and Hybrid Electric Vehicles". In: *Proceedings of the IEEE* 109.6 (2021), pp. 967–984. DOI: 10.1109/JPROC.2021.3072788.

[83] Elbouzekri, A, Elhassania, Messaoud, and Alaoui, A El Hilali. "A hybrid ant colony system for green capacitated vehicle routing problem in sustainable transport". In: *J. Theor. Appl. Inf. Technol* 54.2 (2013).

[84] Ene, Seval, Küçükoğlu, İlker, Aksoy, Aslı, and Öztürk, Nursel. "A hybrid metaheuristic algorithm for the green vehicle routing problem with a heterogeneous fleet". In: *International Journal of Vehicle Design* 71.1-4 (2016), pp. 75–102.

[85] Erdelić, Tomislav and Carić, Tonči. "A survey on the electric vehicle routing problem: variants and solution approaches". In: *Journal of Advanced Transportation* 2019 (2019).

[86] Erdoğan, Sevgi and Miller-Hooks, Elise. "A green vehicle routing problem". In: *Transportation research part E: logistics and transportation review* 48.1 (2012), pp. 100–114.

[87] Eshelman, Larry J. and Schaffer, J. David. "Genetic algorithms for the traveling salesman problem". In: *Proceedings of the 1st International Conference on Genetic Algorithms*. 1985, pp. 160–168.

[88] European Environment Agency. *Annual European Union greenhouse gas inventory 1990–2018 and inventory report 2020*. 2020. URL: https://www.eea.europa.eu/publications/european-union-greenhouse-gas-inventory-2020.

[89] Fang, Yidong, Lu, Yiji, Roskilly, Anthony Paul, and Yu, Xiaoli. "A review of compressed air energy systems in vehicle transport". In: *Energy Strategy Reviews* 33 (2021), p. 100583. ISSN: 2211-467X. DOI: https://doi.org/10.1016/j.esr.2020.100583. URL: https://www.sciencedirect.com/science/article/pii/S2211467X2030136X.

[90] Farahani, Maryam, Zegordi, Seyed Hessameddin, and Kashan, Ali Husseinzadeh. "A Tailored Meta-Heuristic for the Autonomous Electric Vehicle Routing Problem Considering the Mixed Fleet". In: *IEEE Access* 11 (2023), pp. 8207–8222. DOI: 10.1109/ACCESS.2023.3237481.

[91] Fausto, Fernando, Reyna-Orta, Adolfo, Cuevas, Erik, Andrade, Ángel G, and Perez-Cisneros, Marco. "From ants to whales: metaheuristics for all tastes". In: *Artificial Intelligence Review* 53 (2020), pp. 753–810.

[92] Felipe, Ángel, Ortuño, M Teresa, Righini, Giovanni, and Tirado, Gregorio. "A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges". In: *Transportation Research Part E: Logistics and Transportation Review* 71 (2014), pp. 111–128.

[93] Feo, Thomas A and Resende, Mauricio GC. "Greedy randomized adaptive search procedures". In: *Journal of global optimization* 6 (1995), pp. 109–133.

[94] Ferreira, Júlio César and Steiner, Maria Teresinha Arns. "A Bi-objective green vehicle routing problem: a new hybrid optimization algorithm applied to a newspaper distribution". In: *Journal of Geographic Information System* 13.4 (2021), pp. 410–433.

[95] Ferreira, Júlio César, Steiner, Maria Teresinha Arns, and Canciglieri Junior, Osíris. "Multi-objective optimization for the green vehicle routing problem: A systematic literature review and future directions". In: *Cogent Engineering* 7.1 (2020), p. 1807082.

[96] Ferrer, José M, Ortuño, M Teresa, and Tirado, Gregorio. "A GRASP metaheuristic for humanitarian aid distribution". In: *Journal of Heuristics* 22 (2016), pp. 55–87. DOI: https://doi.org/10.1007/s10732-015-9302-5.

[97] Firoz, Shafaque. "A review: advantages and disadvantages of biodiesel". In: *International Research Journal of Engineering and Technology* 4.11 (2017), pp. 530–533.

[98] Friedman, Milton. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.1937.10503522.

[99] Fu, Xiuwen, Pace, Pasquale, Aloi, Gianluca, Yang, Lin, and Fortino, Giancarlo. "Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm". In: *Computer Networks* 177 (2020), p. 107327. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107327. URL: https://www.sciencedirect.com/science/article/pii/S1389128619317888.

[100] Funke, Birger, Grünert, Tore, and Irnich, Stefan. "Local search for vehicle routing and scheduling problems: Review and conceptual integration". In: *Journal of heuristics* 11 (2005), pp. 267–306.

[101] Garcia-Najera, Abel and Bullinaria, John A. "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows". In: *Computers & Operations Research* 38.1 (2011). Project Management and Scheduling, pp. 287–300. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2010.05.004. URL: https://www.sciencedirect.com/science/article/pii/S0305054810001176.

[102] García-Nájera, Abel, Bullinaria, John A, and Gutiérrez-Andrade, Miguel A. "An evolutionary approach for multi-objective vehicle routing problems with backhauls". In: *Computers & Industrial Engineering* 81 (2015), pp. 90–108. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2014.12.029. URL: https://www.sciencedirect.com/science/article/pii/S0360835214004586.

[103] Gaviano, Marco and Lera, Daniela. "Complexity of general continuous minimization problems: a survey". In: *Optimization Methods and Software* 20.4-5 (2005), pp. 525–544. DOI: 10.1080/10556780500139872.

[104] Ghadikolaei, Meisam Ahmadi, Wong, Pak Kin, Cheung, Chun Shun, Zhao, Jing, Ning, Zhi, Yung, Ka-Fu, Wong, Hang Cheong, and Gali, Nirmal Kumar. "Why is the world not yet ready to use alternative fuel vehicles?" In: *Heliyon* 7.7 (2021). URL: https://www.cell.com/heliyon/pdf/S2405-8440(21)01630-3.pdf.

[105] Ghoseiri, Keivan and Ghannadpour, Seyed Farid. "Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm". In: *Applied Soft Computing* 10.4 (2010). Ed. by Bhattacharya, Arijit, Susanto, Sani, Geraghty, John, and Young, Paul. Special issue: Optimisation Methods & Applications in Decision-Making Processes, pp. 1096–

1107. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2010.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1568494610000773.

[106] Giallanza, Antonio and Puma, Gabriella Li. "Fuzzy green vehicle routing problem for designing a three echelons supply chain". In: *Journal of Cleaner Production* 259 (2020), p. 120774. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2020.120774. URL: https://www.sciencedirect.com/science/article/pii/S0959652620308210.

[107] Glabowski, Mariusz, Musznicki, Bartosz, Nowak, Przemyslaw, and Zwierzykowski, Piotr. "Shortest path problem solving based on ant colony optimization metaheuristic". In: *Image Processing & Communications* 17.1-2 (2012), p. 7. DOI: 0.2478/v10248-012-0011-5.

[108] Glover, Fred and Laguna, Manuel. "Tabu search". In: *Modern Heuristic Techniques for Combinatirial Problems*. Blackwell Scientific Publications, 1993, pp. 70–150.

[109] Glover, Fred, Laguna, Manuel, and Martí, Rafael. "Fundamentals of scatter search and path relinking". In: *Control and cybernetics* 29.3 (2000), pp. 653–684.

[110] Goeke, Dominik. "The Pickup and Delivery Problem with Time Windows and Electric Vehicles". In: *Preprint* (Dec. 2017). URL: https://www.researchgate.net/publication/327111412_The_Pickup_and_Delivery_Problem_with_Time_Windows_and_Electric_Vehicles.

[111] Goeke, Dominik and Schneider, Michael. "Routing a mixed fleet of electric and conventional vehicles". In: *European Journal of Operational Research* 245.1 (2015), pp. 81–99.

[112] Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[113] Goldberg, David E and Lingle, Robert. "Alleles, loci, and the traveling salesman problem". In: *Proceedings of the first international conference on genetic algorithms and their applications*. L. Erlbaum Associates Inc., 1985, pp. 154–159.

[114] Golden, B., Bodin, L., Doyle, T., and Stewart, W. "Approximate Traveling Salesman Algorithms". In: *Operations Research* 28.3-part-ii (1980), pp. 694–711. DOI: 10.1287/opre.28.3.694.

[115] Golden, Bruce L, Raghavan, Subramanian, Wasil, Edward A, et al. *The vehicle routing problem: latest advances and new challenges*. Part of the book series: Operations Research/Computer Science Interfaces Series (ORCS, volume 43). Springer, 2008.

[116] Gomory, Ralph E. "Outline of an Algorithm for Integer Solutions to Linear Programs". In: *Bulletin of the American Mathematical Society* 64.5 (1958), pp. 275–278.

[117] Gong, Guiliang, Deng, Qianwang, Chiong, Raymond, Gong, Xuran, and Huang, Hezhiyuan. "An effective memetic algorithm for multi-objective job-shop scheduling". In: *Knowledge-Based Systems* 182 (2019), p. 104840. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0950705119303156.

[118] Gong, Minglun and Yang, Yee-Hong. "Multi-resolution stereo matching using genetic algorithm". In: *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*. IEEE. 2001, pp. 21–29. DOI: 10.1109/SMBV.2001.988759.

[119] González-Benito, Javier and González-Benito, Óscar. "A review of determinant factors of environmental proactivity". In: *Business Strategy and the Environment* 15.2 (2006), pp. 87–102.

[120] González-Neira, Eliana M and Montoya-Torres, Jairo R. "A GRASP metaheuristic for the hybrid flowshop scheduling problem". In: *Journal of Decision Systems* 26.3 (2017), pp. 294–306. DOI: 10.1080/12460125.2017.1351863. URL: https://doi.org/10.1080/12460125.2017.1351863.

[121] Hänggi, Severin, Elbert, Philipp, Bütler, Thomas, Cabalzar, Urs, Teske, Sinan, Bach, Christian, and Onder, Christopher. "A review of synthetic fuels for passenger vehicles". In: *Energy Reports* 5 (2019), pp. 555–569. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2019.04.007. URL: https://www.sciencedirect.com/science/article/pii/S235248471830266X.

[122] Hannan, Mahammad A, Azidin, FA, and Mohamed, Azah. "Hybrid electric vehicles and their challenges: A review". In: *Renewable and Sustainable Energy Reviews* 29 (2014), pp. 135–150. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.08.097. URL: https://www.sciencedirect.com/science/article/pii/S1364032113006370.

[123] Hansen, Pierre and Mladenović, Nenad. "Developments of Variable Neighborhood Search". In: *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002, pp. 415–439. ISBN: 978-1-4615-1507-4. DOI: 10.1007/978-1-4615-1507-4_19. URL: https://doi.org/10.1007/978-1-4615-1507-4_19.

[124] Hansen, Pierre and Mladenović, Nenad. "Variable neighborhood search for the p-median". In: *Location Science* 5.4 (1997), pp. 207–226. ISSN: 0966-8349. DOI: https://doi.org/10.1016/S0966-8349(98)00030-8. URL: https://www.sciencedirect.com/science/article/pii/S0966834998000308.

[125] Hansen, Pierre, Mladenović, Nenad, Brimberg, Jack, and Pérez, José A Moreno. *Variable neighborhood search*. Springer, 2019.

[126] Hansen, Pierre, Mladenović, Nenad, Todosijević, Raca, and Hanafi, Saïd. "Variable neighborhood search: basics and variants". In: *EURO Journal on Computational Optimization* 5.3 (2017), pp. 423–454. ISSN: 2192-4406. DOI: https://doi.org/10.1007/s13675-016-0075-x. URL: https://www.sciencedirect.com/science/article/pii/S2192440621000873.

[127] Hansen, Pierre, Mladenović, Nenad, and Urošević, Dragan. "Variable Neighbourhood Search and Local Branching". In: *Computers & Operations Research* 33.10 (2006), pp. 3034–3045.

[128] Health Effects Institute. *State of Global Air 2020*. Special Report. Boston, MA, 2020.

[129] Hemert, Jano I. van and La Poutré, J. A. "Dynamic Routing Problems with Fruitful Regions: Models and Evolutionary Computation". In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Yao, Xin, Burke, Edmund K., Lozano, José A., Smith, Jim, Merelo-Guervós, Juan Julián, Bullinaria, John A., Rowe, Jonathan E., Tiňo, Peter, Kabán, Ata, and Schwefel, Hans-Paul. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 692–701. ISBN: 978-3-540-30217-9.

[130] Henke, Tino, Speranza, M Grazia, and Wäscher, Gerhard. "The multi-compartment vehicle routing problem with flexible compartment sizes". In: *European Journal of Operational Research* 246.3 (2015), pp. 730–743. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2015.05.020. URL: https://www.sciencedirect.com/science/article/pii/S0377221715004105.

[131] Hickman, John, Hassel, Dieter, Joumard, Robert, Samaras, Zissis, and Sorenson, S. *Methodology for calculating transport emissions and energy consumption*. European Commission, DG VII. Technical report. 1999. URL: https://trimis.ec.europa.eu/sites/default/files/project/documents/meet.pdf.

[132] Hien, Vu Quoc, Dao, Tran Cong, and Binh, Huynh Thi Thanh. "A greedy search based evolutionary algorithm for electric vehicle routing problem". In: *Applied Intelligence* 53.3 (2023), pp. 2908–2922. DOI: https://doi.org/10.1007/s10489-022-03555-8.

[133] Hiermann, Gerhard, Hartl, Richard F, Puchinger, Jakob, and Vidal, Thibaut. "Routing a mix of conventional, plug-in hybrid, and electric vehicles". In: *European Journal of Operational Research* 272.1 (2019), pp. 235–248.

[134] Hiermann, Gerhard, Puchinger, Jakob, Ropke, Stefan, and Hartl, Richard F. "The electric fleet size and mix vehicle routing problem with time windows and recharging stations". In: *European Journal of Operational Research* 252.3 (2016), pp. 995–1018.

[135] Hoekstra, Auke. "The underestimated potential of battery electric vehicles to reduce emissions". In: *Joule* 3.6 (2019), pp. 1412–1414.

[136] Hõimoja, Hardi, Rufer, Alfred, Dziechciaruk, Grzegorz, and Vezzini, Andrea. "An ultrafast EV charging station demonstrator". In: *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*. IEEE. 2012, pp. 1390–1395.

[137] Holland, John H. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975. ISBN: 0–262-581116.

[138] Holm, Sture. "A simple sequentially rejective multiple test procedure". In: *Scandinavian journal of statistics* (1979), pp. 65–70.

[139] Hsueh, Che-Fu. "The green vehicle routing problem with stochastic travel speeds". In: *CICTP 2016*. 2016, pp. 1–12.

[140] Huang, Nan, Li, Jiliu, Zhu, Wenbin, and Qin, Hu. "The multi-trip vehicle routing problem with time windows and unloading queue at depot". In: *Transportation Research Part E: Logistics and Transportation Review* 152 (2021), p. 102370. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2021.102370. URL: https://www.sciencedirect.com/science/article/pii/S1366554521001381.

[141] Hvattum, Lars M, Løkketangen, Arne, and Laporte, Gilbert. "Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic". In: *Transportation Science* 40.4 (2006), pp. 421–438. DOI: 10.1287/trsc.1060.0166.

[142] IEA. *Global Energy & CO2 Status Report 2019*. 2019. URL: https://www.iea.org/reports/global-energy-co2-status-report-2019.

[143] Ilin, Vladimir, Matijević, Luka, Davidović, Tatjana, and Pardalos, Panos. "Asymmetric capacitated vehicle routing problem with time window". In: *Proceedings of the XLV Symposium on Operational Research (SYM-OP-IS 2018)*. 2018, pp. 16–18.

[144] Imran, Arif, Salhi, Said, and Wassan, Niaz A. "A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem". In: *European Journal of Operational Research* 197.2 (2009), pp. 509–518.

[145] Ismkhan, Hassan and Zamanifar, Kamran. "Study of some recent crossovers effects on speed and accuracy of genetic algorithm, using symmetric travelling salesman problem". In: *International Journal of Computer Applications* 80.6 (2015), pp. 1–6.

[146] Jabir, E., Panicker, Vinay V., and Sridharan, R. "Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem". In: *Transportation Research Part D: Transport and Environment* 57 (2017), pp. 422–457. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2017.09.003. URL: https://www.sciencedirect.com/science/article/pii/S1361920917301864.

[147] Jabir, E., Panicker, Vinay V., and Sridharan, R. "Multi-objective Optimization Model for a Green Vehicle Routing Problem". In: *Procedia - Social and Behavioral Sciences* 189 (2015). Operations Management in Digital Economy, pp. 33–39. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2015.03.189. URL: https://www.sciencedirect.com/science/article/pii/S1877042815019825.

[148] Jakšić-Krüger, Tatjana. "Development, Implementation and Theoretical Analysis of the Bee Colony Optimization Meta-heuristic Method". PhD thesis. Novi Sad: University of Novi Sad, Faculty of Technical Sciences, 2017.

[149] Janković, Olivera and Stanimirović, Zorica. "A general variable neighborhood search for solving the uncapacitated r-allocation p-hub maximal covering problem". In: *Electronic Notes in Discrete Mathematics* 58 (2017). 4th International Conference on Variable Neighborhood Search, pp. 23–30. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2017.03.004. URL: https://www.sciencedirect.com/science/article/pii/S1571065317300409.

[150] Jiang, Jun, Ng, Kien Ming, Poh, Kim Leng, and Teo, Kwong Meng. "Vehicle routing problem with a heterogeneous fleet and time windows". In: *Expert Systems with Applications* 41.8 (2014), pp. 3748–3760. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.11.029. URL: https://www.sciencedirect.com/science/article/pii/S0957417413009494.

[151] Jovanovic, Raka and Ashhab, Sahel. "A GRASP approach for Symbolic Regression". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2019, pp. 1723–1728.

[152] Jovanovic, Raka, Bayram, Islam Safak, Bayhan, Sertac, and Voß, Stefan. "A grasp approach for solving large-scale electric bus scheduling problems". In: *Energies* 14.20 (2021), p. 6610.

[153] Joy, Geethu, Huyck, Christian, and Yang, Xin-She. "Review of Parameter Tuning Methods for Nature-Inspired Algorithms". In: *Benchmarks and Hybrid Algorithms in Optimization and Applications*. Ed. by Yang, Xin-She. Springer Nature Singapore, 2023, pp. 33–47. ISBN: 978-981-99-3970-1. DOI: 10.1007/

978-981-99-3970-1_3. URL: https://doi.org/10.1007/978-981-99-3970-1_3.

[154] Kaabachi, Islem, Jriji, Dorra, and Krichen, Saoussen. "An improved ant colony optimization for green multi-depot vehicle routing problem with time windows". In: *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE. 2017, pp. 339–344. DOI: 10.1109/SNPD.2017.8022743.

[155] Kalatzantonakis, Panagiotis, Sifaleras, Angelo, and Samaras, Nikolaos. "A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem". In: *Expert Systems with Applications* 213 (2023), p. 118812. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118812. URL: https://www.sciencedirect.com/science/article/pii/S0957417422018309.

[156] Karaboga, Dervis and Basturk, Bahriye. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm". In: *Journal of global optimization* 39.3 (2007), pp. 459–471.

[157] Karaboga, Dervis et al. *An idea based on honey bee swarm for numerical optimization*. Tech. rep. Technical report-tr06, Erciyes university, 2005.

[158] Karagul, Kenan, Sahin, Yusuf, Aydemir, Erdal, and Oral, Aykut. "A simulated annealing algorithm based solution method for a green vehicle routing problem with fuel consumption". In: *Lean and green supply chain management*. Ed. by Paksoy, Turan, Weber, Gerhard-Wilhelm, and Huber, Sandra. Springer, 2019, pp. 161–187.

[159] Karakostas, Panagiotis, Sifaleras, Angelo, and Georgiadis, Michael C. "A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing". In: *Computers & Chemical Engineering* 126 (2019), pp. 263–279. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2019.04.015. URL: https://www.sciencedirect.com/science/article/pii/S0098135419302315.

[160] Keskin, Merve and Çatay, Bülent. "Partial recharge strategies for the electric vehicle routing problem with time windows". In: *Transportation research part C: emerging technologies* 65 (2016), pp. 111–127.

[161] Kirkpatrick, Scott and Toulouse, Gérard. "Configuration space analysis of travelling salesman problems". In: *Journal de Physique* 46.8 (1985), pp. 1277–1292. DOI: 10.1051/jphys:019850046080127700.

[162] Knowlen, Carl, Mattick, AT, Bruckner, Adam P, and Hertzberg, A. "High efficiency energy conversion systems for liquid nitrogen automobiles". In: *SAE transactions* (1998), pp. 1837–1842.

[163] Koç, Çağrı, Bektaş, Tolga, Jabali, Ola, and Laporte, Gilbert. "Thirty years of heterogeneous vehicle routing". In: *European Journal of Operational Research* 249.1 (2016), pp. 1–21. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2015.07.020. URL: https://www.sciencedirect.com/science/article/pii/S0377221715006530.

[164] Koç, Çağrı and Karaoglan, Ismail. "The green vehicle routing problem: A heuristic based exact solution approach". In: *Applied Soft Computing* 39 (2016), pp. 154–164.

[165] König, Adrian, Nicoletti, Lorenzo, Schröder, Daniel, Wolff, Sebastian, Waclaw, Adam, and Lienkamp, Markus. "An overview of parameter and cost for battery electric vehicles". In: *World Electric Vehicle Journal* 12.1 (2021), p. 21. DOI: 10.3390/wevj12010021.

[166] Konstantakopoulos, Grigorios D, Gayialis, Sotiris P, and Kechagias, Evripidis P. "Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification". In: *Operational research* (2020), pp. 1–30.

[167] Kovač, Nataša, Davidović, Tatjana, and Stanimirović, Zorica. "Variable neighborhood search methods for the dynamic minimum cost hybrid berth allocation problem". In: *Information Technology and Control* 47.3 (2018), pp. 471–488.

[168] Kramer, Oliver. *Genetic Algorithm Essentials*. Springer Cham, 2017. DOI: https://doi.org/10.1007/978-3-319-52156-5.

[169] Kramer, Oliver. "K-Nearest Neighbors". In: *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 13–23. ISBN: 978-3-642-38652-7. DOI: 10.1007/978-3-642-38652-7_2. URL: https://doi.org/10.1007/978-3-642-38652-7_2.

[170] Kramer, Raphael, Maculan, Nelson, Subramanian, Anand, and Vidal, Thibaut. "A speed and departure time optimization algorithm for the pollution-routing problem". In: *European Journal of Operational Research* 247.3 (2015), pp. 782–787.

[171] Krumke, Sven O, Saliba, Sleman, Vredeveld, Tjark, and Westphal, Stephan. "Approximation algorithms for a vehicle routing problem". In: *Mathematical methods of operations research* 68.2 (2008), pp. 333–359. DOI: https://doi.org/10.1007/s00186-008-0224-y.

[172] Kruskal, William H and Wallis, W. Allen. "Use of ranks in one-criterion variance analysis". In: *Journal of the American Statistical Association* 47.260 (1952), pp. 583–621. DOI: 10.1080/01621459.1952.10483441.

[173] Küçükoğlu, İlker, Ene, Seval, Aksoy, Aslı, and Öztürk, Nursel. "A memory structure adapted simulated annealing algorithm for a green vehicle routing problem". In: *Environmental Science and Pollution Research* 22.5 (2015), pp. 3279–3297.

[174] Kumar, Ravi Shankar, Kondapaneni, Karthik, Dixit, Vijaya, Goswami, Adrijit, Thakur, Lakshman S, and Tiwari, MK. "Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach". In: *Computers & Industrial Engineering* 99 (2016), pp. 29–40.

[175] Kuo, R.J. and Zulvia, Ferani E. "The gradient evolution algorithm: A new metaheuristic". In: *Information Sciences* 316 (2015). Ed. by Li, Xiaodong, Tang, Ke, Suganthan, P.N, and Yang, Zhenyu. Special issue: Nature-Inspired Algorithms for Large Scale Global Optimization, pp. 246–265. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2015.04.031. URL: https://www.sciencedirect.com/science/article/pii/S0020025515002996.

[176] Kwon, Yong-Ju, Choi, Young-Jae, and Lee, Dong-Ho. "Heterogeneous fixed fleet vehicle routing considering carbon emission". In: *Transportation Research Part D: Transport and Environment* 23 (2013), pp. 81–89.

[177] Laherrère, Jean, Hall, Charles AS, and Bentley, Roger. "How much oil remains for the world to produce? Comparing assessment methods, and separating fact from fiction". In: *Current Research in Environmental Sustainability* 4 (2022), p. 100174. ISSN: 2666-0490. DOI: https://doi.org/10.1016/j.crsust.2022.100174. URL: https://www.sciencedirect.com/science/article/pii/S2666049022000524.

[178] Lai, Young-Jou, Liu, Ting-Yun, and Hwang, Ching-Lai. "TOPSIS for MODM". In: *European Journal of Operational Research* 76.3 (1994). Ed. by Aikens, C.H. Special issue: Facility Location Models for Distribution Planning, pp. 486–500. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(94)90282-8. URL: https://www.sciencedirect.com/science/article/pii/0377221794902828.

[179] Lamini, Chaymaa, Benhlima, Said, and Elbekri, Ali. "Genetic algorithm based approach for autonomous mobile robot path planning". In: *Procedia Computer Science* 127 (2018). PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2017, pp. 180–189. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.01.113. URL: https://www.sciencedirect.com/science/article/pii/S187705091830125X.

[180] Lan, Shaowen, Fan, Wenjuan, Yang, Shanlin, Pardalos, Panos M, and Mladenović, Nenad. "A survey on the applications of variable neighborhood search algorithm in healthcare management". In: *Annals of Mathematics and Artificial Intelligence* 89 (2021), pp. 1–35. DOI: https://doi.org/10.1007/s10472-021-09727-5.

[181] Land, Ailsa H and Doig, Alison G. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (1960), pp. 497–520.

[182] Lau, Hoong Chuin, Sim, Melvyn, and Teo, Kwong Meng. "Vehicle routing problem with time windows and a limited number of vehicles". In: *European Journal of Operational Research* 148.3 (2003), pp. 559–569. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(02)00363-6. URL: https://www.sciencedirect.com/science/article/pii/S0377221702003636.

[183] Lee, Su-Yol and Klassen, Robert D. "Drivers and enablers that foster environmental management capabilities in small-and medium-sized suppliers in supply chains". In: *Production and Operations management* 17.6 (2008), pp. 573–586.

[184] Leggieri, Valeria and Haouari, Mohamed. "A matheuristic for the asymmetric capacitated vehicle routing problem". In: *Discrete Applied Mathematics* 234 (2018). Special Issue on the Ninth International Colloquium on Graphs and Optimization (GO IX), 2014, pp. 139–150. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2016.03.019. URL: https://www.sciencedirect.com/science/article/pii/S0166218X16301573.

[185] Lehmann, E. L. and Romano, Joseph P. "Generalizations of the familywise error rate". In: *The Annals of Statistics* 33.3 (2005), pp. 1138 –1154. DOI: 10.1214/009053605000000084. URL: https://doi.org/10.1214/009053605000000084.

[186] Lenstra, Jan Karel and Kan, AHG Rinnooy. "Complexity of vehicle routing and scheduling problems". In: *Networks* 11.2 (1981), pp. 221–227.

[187] Levene, Howard. "Robust Tests for Equality of Variances". In: *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Ed. by Olkin, Ingram et al. Palo Alto, CA: Stanford University Press, 1960, pp. 278–292.

[188] Li, Changhe, Yang, Shengxiang, and Nguyen, Trung Thanh. "A self-learning particle swarm optimizer for global optimization problems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.3 (2011), pp. 627–646.

[189] Li, Gen Bao. "Dimethyl ether (DME): a new alternative fuel for diesel vehicle". In: *Advanced Materials Research* 156 (2011), pp. 1014–1018. DOI: https://doi.org/10.4028/www.scientific.net/AMR.156-157.1014.

[190] Li, Hui and Landa-Silva, Dario. "An Elitist GRASP Metaheuristic for the Multi-objective Quadratic Assignment Problem". In: *Evolutionary Multi-Criterion Optimization*. Ed. by Ehrgott, Matthias, Fonseca, Carlos M., Gandibleux, Xavier, Hao, Jin-Kao, and Sevaux, Marc. Springer Berlin Heidelberg, 2009, pp. 481–494. ISBN: 978-3-642-01020-0.

[191] Li, Jin, Wang, Feng, and He, Yu. "Electric vehicle routing problem with battery swapping considering energy consumption and carbon emissions". In: *Sustainability* 12.24 (2020), p. 10537.

[192] Li, Xiangyong, Baki, Md Fazle, and Aneja, Yash P. "An ant colony optimization metaheuristic for machine–part cell formation problems". In: *Computers & Operations Research* 37.12 (2010), pp. 2071–2081. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2010.02.007. URL: https://www.sciencedirect.com/science/article/pii/S0305054810000511.

[193] Li, Xiaohui, Shi, Xuemin, Zhao, Yi, Liang, Huagang, and Dong, Yuan. "SVND enhanced metaheuristic for plug-in hybrid electric vehicle routing problem". In: *Applied Sciences* 10.2 (2020). DOI: 10.3390/app10020441.

[194] Li, Yan, Lim, Ming K, Tan, Yingshuang, Lee, Yee, and Tseng, Ming-Lang. "Sharing economy to improve routing for urban logistics distribution using electric vehicles". In: *Resources, Conservation and Recycling* 153 (2020), p. 104585.

[195] Li, Yan, Lim, Ming K, and Tseng, Ming-Lang. "A green vehicle routing model based on modified particle swarm optimization for cold chain logistics". In: *Industrial Management & Data Systems* (2019).

[196] Li, Yang, Qian, Bin, Hu, Rong, Wu, Li-Ping, and Liu, Bo. "Two-Stage Algorithm for Solving Multi-depot Green Vehicle Routing Problem with Time Window". In: *Intelligent Computing Theories and Application*. Ed. by Huang, De-Shuang, Bevilacqua, Vitoantonio, and Premaratne, Prashan. Cham: Springer International Publishing, 2019, pp. 665–675. ISBN: 978-3-030-26763-6.

[197] Li, Yongbo, Soleimani, Hamed, and Zohal, Mostafa. "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives". In: *Journal of cleaner production* 227 (2019), pp. 1161–1172.

[198] Lin, Canhong, Choy, K.L., Ho, G.T.S., Chung, S.H., and Lam, H.Y. "Survey of Green Vehicle Routing Problem: Past and future trends". In: *Expert Systems with Applications* 41.4, Part 1 (2014), pp. 1118–1138. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.07.107. URL: https://www.sciencedirect.com/science/article/pii/S095741741300609X.

[199] Lin, Shen and Kernighan, Brian W. "An effective heuristic algorithm for the traveling-salesman problem". In: *Operations research* 21.2 (1973), pp. 498–516.

[200] Liu, Changshi, Kou, Gang, Zhou, Xiancheng, Peng, Yi, Sheng, Huyi, and Alsaadi, Fawaz E. "Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach". In: *Knowledge-Based Systems* 188 (2020), p. 104813. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.06.021. URL: https://www.sciencedirect.com/science/article/pii/S0950705119302886.

[201] Liu, Ran, Xie, Xiaolan, Augusto, Vincent, and Rodriguez, Carlos. "Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care". In: *European Journal of Operational Research* 230.3 (2013), pp. 475–486. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2013.04.044. URL: https://www.sciencedirect.com/science/article/pii/S0377221713003585.

[202] Lucic, Panta and Teodorovic, Dusan. "Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence". In: *Preprints of the TRISTAN IV triennial symposium on transportation analysis*. 2001, pp. 441–445.

[203] Lučić, Panta and Teodorović, Dušan. "Computing with bees: attacking complex transportation engineering problems". In: *International Journal on Artificial Intelligence Tools* 12.03 (2003), pp. 375–394.

[204] Lucic, Panta and Teodorovic, Dusan. "Transportation modeling: an artificial life approach". In: *14th IEEE International Conference on Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings.* IEEE. 2002, pp. 216–223.

[205] Lučić, Panta and Teodorović, Dušan. "Vehicle Routing Problem With Uncertain Demand at Nodes: The Bee System and Fuzzy Logic Approach". In: *Fuzzy Sets Based Heuristics for Optimization*. Ed. by Verdegay, José-Luis. Springer Berlin Heidelberg, 2003, pp. 67–82. ISBN: 978-3-540-36461-0. DOI: 10.1007/978-3-540-36461-0_5. URL: https://doi.org/10.1007/978-3-540-36461-0_5.

[206] MacQueen, James et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[207] Macrina, Giusy, Laporte, Gilbert, Guerriero, Francesca, and Pugliese, Luigi Di Puglia. "An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows". In: *European Journal of Operational Research* 276.3 (2019), pp. 971–982.

[208] Macrina, Giusy, Pugliese, Luigi Di Puglia, Guerriero, Francesca, and Laporte, Gilbert. "The green mixed fleet vehicle routing problem with partial battery recharging and time windows". In: *Computers & Operations Research* 101 (2019), pp. 183–199.

[209] Madureira, Ana, Ramos, Carlos, and Carmo Silva, SD do. "A coordination mechanism for real world scheduling problems using genetic algorithms". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 1. IEEE. 2002, pp. 175–180. DOI: 10.1109/CEC.2002.1006229.

[210] Maksimović, P. and Davidović, T. "Parameter Calibration in the Bee Colony Optimization Algorithm". In: *XI Balcan Conference on Operational Research*. BALCOR 2013, Beograd-Zlatibor, Serbia, 2013, pp. 263–272.

[211] Manivannan, S and Kaleeswaran, E. "Solar powered electric vehicle". In: *2016 First International Conference on Sustainable Green Buildings and Communities (SGBC)*. IEEE. 2016, pp. 1–4. DOI: 10.1109/SGBC.2016.7936074.

[212] Mann, Henry B. and Whitney, Donald R. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. DOI: 10.1214/aoms/1177730491.

[213] Marcinko, Tomáš. "Consequences of assumption violations regarding one-way ANOVA". In: *Proceedings of The 8th International Days of Statistics and Economics*. Vol. 116. 47. 2014, pp. 974–985.

[214] Marinelli, Mario, Caggiani, Leonardo, Alnajajreh, Abedelkareem, and Binetti, Mario. "A two-stage Metaheuristic approach for solving the Vehicle Routing Problem with Simultaneous Pickup/Delivery and Door-to-Door service". In: *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2019, pp. 1–9. DOI: 10.1109/MTITS.2019.8883340.

[215] Maron, Oded and Moore, Andrew. "Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation". In: *Advances in Neural Information Processing Systems*. Ed. by Cowan, J., Tesauro, G., and Alspector, J. Vol. 6. Morgan-Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper_files/paper/1993/file/02a32ad2669e6fe298e607fe7cc0e1a0-Paper.pdf.

[216] Marques, Alexandra, Soares, Ricardo, Santos, Maria João, and Amorim, Pedro. "Integrated planning of inbound and outbound logistics with a Rich Vehicle Routing Problem with backhauls". In: *Omega* 92 (2020), p. 102172. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2019.102172. URL: https://www.sciencedirect.com/science/article/pii/S0305048319300350.

[217] Marrekchi, Emna, Besbes, Walid, and Dhouib, Diala. "An Overview of the Recent Solution Approaches in the Green Vehicle Routing Problem". In: *Solving Transport Problems: Towards Green Logistics*. Ed. by Besbes, Walid, Dhouib, Diala, Wassan, Niaz, and Marrekchi, Emna. Wiley Online Library, 2019, pp. 115–133.

[218] Martinez, Clara Marina, Hu, Xiaosong, Cao, Dongpu, Velenis, Efstathios, Gao, Bo, and Wellers, Matthias. "Energy Management in Plug-in Hybrid Electric Vehicles: Recent Progress and a Connected Vehicles Perspective". In: *IEEE Transactions on Vehicular Technology* 66.6 (2017), pp. 4534–4549. DOI: 10.1109/TVT.2016.2582721.

[219] Matai, Rajesh, Singh, Surya Prakash, and Mittal, Murari Lal. "Traveling salesman problem: an overview of applications, formulations, and solution approaches". In: *Traveling salesman problem, theory and applications*. Ed. by Davendra, Donald. Vol. 1. 1. InTech, Croatia, 2010, pp. 1–25.

[220] Matijević, Luka. "General Variable Neighbourhood Search for Electric Vehicle Routing Problem with Time-dependent Speeds and Soft Time Windows". In: *International Journal of Industrial Engineering Computations* 14 (2023), pp. 275–292. DOI: 10.5267/j.ijiec.2023.2.001.

[221] Matijević, Luka. "Metaheuristic Approaches for the Green Vehicle Routing Problem". In: *Yugoslav Journal of Operations Research* 33.2 (2022), pp. 153–198. DOI: http://dx.doi.org/10.2298/YJOR211120016M.

[222] Matijević, Luka, Jelić, Slobodan, and Davidović, Tatjana. "General variable neighborhood search approach to group steiner tree problem". In: *Optimization Letters* 17.9 (2023), pp. 2087–2111. DOI: https://doi.org/10.1007/s11590-022-01904-7.

[223] Matijević, Luka, Davidović, Tatjana, Ilin, Vladimir, and Pardalos, Panos M. "General Variable Neighborhood Search for Asymmetric Vehicle Routing Problem". In: *Proc. XLVI Symposium on Operational Research, SYMOPIS 2019*. Kladovo, 2019, pp. 185–190.

[224] Matijević, Luka, Ilin, Vladimir, Davidović, Tatjana, Jakšić-Krüger, Tatjana, and Pardalos, Panos M. "General VNS for asymmetric vehicle routing problem with time and capacity constraints". In: *Computers & Operations Research* 167 (2024), p. 106630. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2024.106630. URL: https://www.sciencedirect.com/science/article/pii/S0305054824001023.

[225] Mavrovouniotis, Michalis, Ellinas, Georgios, and Polycarpou, Marios. "Ant Colony optimization for the Electric Vehicle Routing Problem". In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2018, pp. 1234–1241. DOI: 10.1109/SSCI.2018.8628831.

[226] Mavrovouniotis, Michalis, Li, Changhe, Ellinas, Georgios, and Polycarpou, Marios. "Parallel ant colony optimization for the electric vehicle routing problem". In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2019, pp. 1660–1667.

[227] Mavrovouniotis, Michalis, Menelaou, Charalambos, Timotheou, Stelios, Ellinas, Georgios, Panayiotou, Christos, and Polycarpou, Marios. "A Benchmark Test Suite for the Electric Capacitated Vehicle Routing Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185753.

[228] Metawa, Noura, Hassan, M Kabir, and Elhoseny, Mohamed. "Genetic algorithm based model for optimizing bank lending decisions". In: *Expert Systems with Applications* 80 (2017), pp. 75–82. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.03.021. URL: https://www.sciencedirect.com/science/article/pii/S0957417417301677.

[229] Mi, Chris and Masrur, M Abul. *Hybrid electric vehicles: principles and applications with practical perspectives*. John Wiley & Sons, 2017.

[230] Micale, R., Marannano, G., Giallanza, A., Miglietta, P.P., Agnusdei, G.P., and La Scalia, G. "Sustainable vehicle routing based on firefly algorithm and TOPSIS methodology". In: *Sustainable Futures* 1 (2019), p. 100001. ISSN: 2666-1888. DOI: https://doi.org/10.1016/j.sftr.2019.100001. URL: https://www.sciencedirect.com/science/article/pii/S2666188819300012.

[231] Mikić, Marija, Todosijević, Raca, and Urošević, Dragan. "Less is more: General variable neighborhood search for the capacitated modular hub location problem". In: *Computers & Operations Research* 110 (2019), pp. 101–115. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.05.020. URL: https://www.sciencedirect.com/science/article/pii/S0305054819301339.

[232] Miller, Clair E, Tucker, Albert W, and Zemlin, Richard A. "Integer programming formulation of traveling salesman problems". In: *Journal of the ACM (JACM)* 7.4 (1960), pp. 326–329. DOI: https://doi.org/10.1145/321043.321046.

[233] MirHassani, SA and Abolghasemi, N. "A particle swarm optimization algorithm for open vehicle routing problem". In: *Expert Systems with Applications* 38.9 (2011), pp. 11547–11551. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.03.032. URL: https://www.sciencedirect.com/science/article/pii/S0957417411004404.

[234] MirHassani, SA and Jalaeian Bashirzadeh, A. "A GRASP meta-heuristic for two-dimensional irregular cutting stock problem". In: *The International Journal of Advanced Manufacturing Technology* 81 (2015), pp. 455–464. DOI: https://doi.org/10.1007/s00170-015-7107-1.

[235] Mirjalili, Seyedali and Lewis, Andrew. "The whale optimization algorithm". In: *Advances in engineering software* 95 (2016), pp. 51–67.

[236] Mladenović, Nenad and Hansen, Pierre. "Variable neighborhood search". In: *Computers & operations research* 24.11 (1997), pp. 1097–1100.

[237] Mladenović, Nenad, Todosijević, Raca, and Urošević, Dragan. "An efficient GVNS for solving traveling salesman problem with time windows". In: *Electronic Notes in Discrete Mathematics* 39 (2012). EURO Mini Conference, pp. 83–90. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2012.10.012. URL: https://www.sciencedirect.com/science/article/pii/S1571065312000133.

[238] Mladenović, Nenad. "A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization". In: vol. 12. Abstracts of papers presented at Optimization days (p. 112). Montréal. 1995.

[239] Moalic, Laurent and Gondran, Alexandre. "Variations on memetic algorithms for graph coloring problems". In: *Journal of Heuristics* 24 (2018), pp. 1–24. DOI: https://doi.org/10.1007/s10732-017-9354-9.

[240] Mockus, Jonas, Eddy, William, and Reklaitis, Gintaras. *Bayesian Heuristic approach to discrete and global optimization: Algorithms, visualization, software, and applications*. Part of the book series: Nonconvex Optimization and Its Applications (NOIA, volume 17). Springer Science & Business Media, 2013. URL: https://link.springer.com/book/10.1007/978-1-4757-2627-5.

[241] Moghdani, Reza, Salimifard, Khodakaram, Demir, Emrah, and Benyettou, Abdelkader. "The green vehicle routing problem: A systematic literature review". In: *Journal of Cleaner Production* 279 (2021), p. 123691. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2020.123691. URL: https://www.sciencedirect.com/science/article/pii/S0959652620337367.

[242] Molina, Jose Carlos, Eguia, Ignacio, Racero, Jesus, and Guerrero, Fernando. "Multi-objective Vehicle Routing Problem with Cost and Emission Functions". In: *Procedia - Social and Behavioral Sciences* 160 (2014). XI Congreso de Ingenieria del Transporte (CIT 2014), pp. 254–263. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2014.12.137. URL: https://www.sciencedirect.com/science/article/pii/S1877042814062387.

[243] Mongomery, Douglas C. *Design and analysis of experiments*. John Willy & Sons, 2017.

[244] Montoya, Alejandro, Guéret, Christelle, Mendoza, Jorge E, and Villegas, Juan G. "A multi-space sampling heuristic for the green vehicle routing problem". In: *Transportation Research Part C: Emerging Technologies* 70 (2016), pp. 113–128.

[245] Montoya, Alejandro, Guéret, Christelle, Mendoza, Jorge E, and Villegas, Juan G. "The electric vehicle routing problem with nonlinear charging function". In: *Transportation Research Part B: Methodological* 103 (2017), pp. 87–110.

[246] Montoya-Torres, Jairo R, Aponte, Andres, Rosas, Paula, and Caballero-Villalobos, Juan Pablo. "Applying GRASP meta-heuristic to solve the single-item two-echelon uncapacitated facility location problem". In: *International Journal of Applied Decision Sciences* 3.4 (2010), pp. 297–310. DOI: https://doi.org/10.1504/IJADS.2010.036849.

[247] Moscato, Pablo. "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms". In: *Caltech concurrent computation program, C3P Report* (1989).

[248] Muštović, Faruk. *Autogas Propulsion Systems for Motor Vehicles*. IBC Sarajevo, 2011.

[249] Muthukumar, M, Rengarajan, N, Velliyangiri, B, Omprakas, MA, Rohit, CB, and Raja, U Kartheek. "The development of fuel cell electric vehicles–A review". In: *Materials Today: Proceedings*. Ed. by G, Kumaresan. Vol. 45. International Conference on Advances in Materials Research - 2019. Elsevier, 2021, pp. 1181–1187. DOI: https://doi.org/10.1016/j.matpr.2020.03.679. URL: https://www.sciencedirect.com/science/article/pii/S2214785320324615.

[250] Nedeljković, Ranko, Mitrović, Slobodan, and Drenovac, Dragana. "Bee colony optimization meta-heuristic for backup allocation problem". In: *Proc. PosTel XXVII* (2009), pp. 115–122.

[251] Neri, Ferrante, Cotta, Carlos, and Moscato, Pablo. *Handbook of memetic algorithms*. Part of the book series: Studies in Computational Intelligence (SCI, volume 379). Springer, 2011.

[252] Nikolić, Miloš and Teodorović, Dušan. "Transit network design by bee colony optimization". In: *Expert Systems with Applications* 40.15 (2013), pp. 5945–5955. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0957417413002881.

[253] Nikolić, Miloš, Teodorović, Dušan, and Šelmić, Milica. "Solving the vehicle routing problem with time windows by bee colony optimization metaheuristic". In: *Proceedings of the 1st Logistics International Conference*. 2013, pp. 44–48.

[254] Niu, Yunyun, Yang, Zehua, Chen, Ping, and Xiao, Jianhua. "Optimizing the green open vehicle routing problem with time windows by minimizing comprehensive routing cost". In: *Journal of cleaner production* 171 (2018), pp. 962–971.

[255] Niu, Yunyun, Zhang, Yongpeng, Cao, Zhiguang, Gao, Kaizhou, Xiao, Jianhua, Song, Wen, and Zhang, Fangwei. "MIMOA: A membrane-inspired multi-objective algorithm for green vehicle routing problem with stochastic demands". In: *Swarm and Evolutionary Computation* 60 (2021), p. 100767. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2020.100767. URL: https://www.sciencedirect.com/science/article/pii/S221065022030420X.

[256] Normasari, Nur Mayke Eka, Yu, Vincent F, Bachtiyar, Candra, and Sukoyo. "A simulated annealing heuristic for the capacitated green vehicle routing problem". In: *Mathematical Problems in Engineering* 2019 (2019), p. 2358258. DOI: https://doi.org/10.1155/2019/2358258.

[257] Norouzi, Narges, Sadegh-Amalnick, Mohsen, and Tavakkoli-Moghaddam, Reza. "Modified particle swarm optimization in a time-dependent vehicle routing problem: minimizing fuel consumption". In: *Optimization Letters* 11.1 (2017), pp. 121–134.

[258] Olgun, Büşra, Koç, Çağrı, and Altıparmak, Fulya. "A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery". In: *Computers & Industrial Engineering* 153 (2021), p. 107010.

[259] Oliver, IM, Smith, DJd, and Holland, John RC. "A study of permutation crossover operators on the traveling salesman problem". In: *Proceedings of the second international conference on genetic algorithms and their applications*. L. Erlbaum Associates Inc., 1987, pp. 224–230.

[260] Paczuski, Maciej, Marchwiany, Marcin, Puławski, Ryszard, Pankowski, Andrzej, Kurpiel, Kamil, and Przedlacki, Marcin. "Liquefied petroleum gas (LPG) as a fuel for internal combustion engines". In: *Alternative fuels, technical and environmental conditions*. Ed. by Biernat, Krzysztof. Vol. 13. IntechOpen, 2016.

[261] Padberg, Manfred W and Rinaldi, Giovanni. "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems". In: *SIAM Review* 33.1 (1991), pp. 60–100.

[262] Pan, Binbin, Zhang, Zhenzhen, and Lim, Andrew. "A hybrid algorithm for time-dependent vehicle routing problem with time windows". In: *Computers & Operations Research* 128 (2021), p. 105193. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2020.105193. URL: https://www.sciencedirect.com/science/article/pii/S0305054820303105.

[263] Pan, Binbin, Zhang, Zhenzhen, and Lim, Andrew. "Multi-trip time-dependent vehicle routing problem with time windows". In: *European Journal of Operational Research* 291.1 (2021), pp. 218–231. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.09.022. URL: https://www.sciencedirect.com/science/article/pii/S0377221720308262.

[264] Panicker, Vinay V, Vanga, Ratnaji, and Sridharan, R. "Ant colony optimisation algorithm for distribution-allocation problem in a two-stage supply chain with a fixed transportation charge". In: *International journal of production research* 51.3 (2013), pp. 698–717.

[265] Papacz, Wladyslaw. "Biogas as vehicle fuel". In: *Journal of KONES Powertrain and Transport,* 18.1 (2011), pp. 403–410.

[266] Papadimitriou, Christos H and Steiglitz, Kenneth. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[267] Park, Su Han and Lee, Chang Sik. "Applicability of dimethyl ether (DME) in a compression ignition engine as an alternative fuel". In: *Energy Conversion and Management* 86 (2014), pp. 848–863. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2014.06.051. URL: https://www.sciencedirect.com/science/article/pii/S0196890414005767.

[268] Park, Younghwan and Chae, Junjae. "A review of the solution approaches used in recent G-VRP (Green Vehicle Routing Problem)". In: *International Journal of Advanced Logistics* 3.1-2 (2014), pp. 27–37.

[269] Pathak, Saurabh, Swetha, Kontham, Vulloju, Sreedhar, and Prabhakar, V. "Compressed air vehicle: A review". In: *Int. J. Mech. Prod. Eng* 24 (2014), pp. 9–13.

[270] Pedemonte, Martín, Nesmachnow, Sergio, and Cancela, Héctor. "A survey on parallel ant colony optimization". In: *Applied Soft Computing* 11.8 (2011), pp. 5181–5197. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2011.05.042. URL: https://www.sciencedirect.com/science/article/pii/S156849461100202X.

[271] Peng, Bo, Wu, Lifan, Yi, Yuxin, and Chen, Xiding. "Solving the multi-depot green vehicle routing problem by a hybrid evolutionary algorithm". In: *Sustainability* 12.5 (2020), p. 2127.

[272] Peng, Bo, Zhang, Yuan, Gajpal, Yuvraj, and Chen, Xiding. "A Memetic Algorithm for the Green Vehicle Routing Problem". In: *Sustainability* 11.21 (2019). ISSN: 2071-1050. DOI: 10.3390/su11216055. URL: https://www.mdpi.com/2071-1050/11/21/6055.

[273] Plummer, Mitty C, Ordonez, Carlos A, and Reidy, Richard F. *Liquid nitrogen as a non-polluting vehicle fuel*. Tech. rep. SAE Technical Paper, 1999.

[274] Poonthalir, Ganesan and Nadarajan, Rethnaswamy. "A fuel efficient green vehicle routing problem with varying speed constraint (F-GVRP)". In: *Expert Systems with Applications* 100 (2018), pp. 131–144.

[275] Poonthalir, Ganesan, Nadarajan, Rethnaswamy, and Geetha, Shanmugam. "Vehicle routing problem with limited refueling halts using particle swarm optimization with greedy mutation operator". In: *RAIRO-Operations Research* 49.4 (2015), pp. 689–716.

[276] Pramuanjaroenkij, Anchasa and Kakaç, Sadık. "The fuel cell electric vehicles: The highlight review". In: *International Journal of Hydrogen Energy* 48.25 (2023), pp. 9401–9425. ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2022.11.103. URL: https://www.sciencedirect.com/science/article/pii/S0360319922053368.

[277] Pregger, Thomas, Schiller, Günter, Cebulla, Felix, Dietrich, Ralph-Uwe, Maier, Simon, Thess, André, Lischke, Andreas, Monnerie, Nathalie, Sattler, Christian, Clercq, Patrick Le, Rauch, Bastian, Köhler, Markus, Severin, Michael, Kutne, Peter, Voigt, Christiane, Schlager, Hans, Ehrenberger, Simone, Feinauer, Mario, Werling, Lukas, Zhukov, Victor P., Kirchberger, Christoph, Ciezki, Helmut K., Linke, Florian, Methling, Torsten, Riedel, Uwe, and Aigner, Manfred. "Future Fuels—Analyses of the Future Prospects of Renewable Synthetic Fuels". In: *Energies* 13.1 (2020). ISSN: 1996-1073. DOI: 10.3390/en13010138. URL: https://www.mdpi.com/1996-1073/13/1/138.

[278] Probstein, Ronald F and Hicks, R Edwin. *Synthetic fuels*. Courier Corporation, 2013.

[279] Puma-Benavides, David Sebastian, Izquierdo-Reyes, Javier, Calderon-Najera, Juan de Dios, and Ramirez-Mendoza, Ricardo A. "A systematic review of technologies, control methods, and optimization for extended-range electric vehicles". In: *Applied Sciences* 11.15 (2021), p. 7095. DOI: 10.3390/app11157095.

[280] Queiroz, Thiago Alves de, Iori, Manuel, Kramer, Arthur, and Kuo, Yong-Hong. "Scheduling of patients in emergency departments with a variable neighborhood search". In: *Variable Neighborhood Search*. Ed. by Mladenovic, Nenad, Sleptchenko, Andrei, Sifaleras, Angelo, and Omar, Mohammed. ICVNS 2021. Lecture Notes in Computer Science(), vol 12559. Cham: Springer International Publishing, 2021, pp. 138–151. ISBN: 978-3-030-69625-2. DOI: https://doi.org/10.1007/978-3-030-69625-2_11.

[281] Raeesi, Ramin and Zografos, Konstantinos G. "Coordinated routing of electric commercial vehicles with intra-route recharging and en-route battery swapping". In: *European Journal of Operational Research* 301.1 (2022), pp. 82–109. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.09.037. URL: https://www.sciencedirect.com/science/article/pii/S0377221721008122.

[282] Ralphs, Ted K, Kopman, Leonid, Pulleyblank, William R, and Trotter, Leslie E. "On the capacitated vehicle routing problem". In: *Mathematical programming* 94 (2003), pp. 343–359. DOI: https://doi.org/10.1007/s10107-002-0323-0.

[283] Ramasubramanian, S, Chandrasekaran, M, Baskar, S, Dowhithamaran, A, et al. "Design and development of pneumatic compressed air vehicle". In: *Materials Today: Proceedings* 37 (2021). International Conference on Newer Trends and Innovation in Mechanical Engineering: Materials Science, pp. 690–693. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2020.05.637. URL: https://www.sciencedirect.com/science/article/pii/S2214785320342401.

[284] Ramos, Tânia Rodrigues Pereira, Gomes, Maria Isabel, and Póvoa, Ana Paula Barbosa. "Multi-depot vehicle routing problem: a comparative study of alternative formulations". In: *International Journal of Logistics Research and Applications* 23.2 (2020), pp. 103–120. DOI: 10.1080/13675567.2019.1630374.

[285] Reed, Martin, Yiannakou, Aliki, and Evering, Roxanne. "An ant colony algorithm for the multi-compartment vehicle routing problem". In: *Applied Soft Computing* 15 (2014), pp. 169–176. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2013.10.017. URL: https://www.sciencedirect.com/science/article/pii/S1568494613003517.

[286] Ren, Xuan, Huang, Hui, Feng, Shuo, and Liang, Gongqian. "An improved variable neighborhood search for bi-objective mixed-energy fleet vehicle routing problem". In: *Journal of Cleaner Production* 275 (2020), p. 124155.

[287] Repoussis, Panagiotis P, Tarantilis, Christos D, and Ioannou, George. "The open vehicle routing problem with time windows". In: *Journal of the Operational Research Society* 58.3 (2007), pp. 355–367. DOI: 10.1057/palgrave.jors.2602143.

[288] Resende, Mauricio G.C. and Ribeiro, Celso C. "Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications". In: *Handbook of Metaheuristics*. Ed. by Gendreau, Michel and Potvin, Jean-Yves. Boston, MA: Springer US, 2010, pp. 283–319. ISBN: 978-1-4419-1665-5. DOI: 10.1007/978-1-4419-1665-5_10. URL: https://doi.org/10.1007/978-1-4419-1665-5_10.

[289] Ritchie, Hannah, Roser, Max, and Rosado, Pablo. "Energy". In: *Our World in Data* (2022). https://ourworldindata.org/energy.

[290] Robins, G. and Zelikovsky, A. "Minimum Steiner Tree Construction". In: *The Handbook of Algorithms for VLSI Physical Design Automation*. Ed. by Alpert, C.J., Mehta, D.P., and Sapatnekar, S.S. CRC Press, 2009.

[291] Rodrigue, Jean-Paul, Slack, Brian, and Comtois, Claude. "The paradoxes of green logistics". In: *Proceedings of the 9th World Conference on Transport Research*. Citeseer. Seoul, 2001.

[292] Rubio, José-Miguel, Palma, Wenceslao, Rodriguez, Nibaldo, Soto, Ricardo, Crawford, Broderick, Paredes, Fernando, and Cabrera, Guillermo. "Solving the balanced academic curriculum problem using the ACO metaheuristic". In: *Mathematical Problems in Engineering* 2013 (2013). Ed. by Zhang, Yudong, Balochian, Saeed, Agarwal, Praveen, Bhatnagar, Vishal, and Housheya, Orwa Jaber. Special Issue: Artificial Intelligence and Its Applications, p. 793671. DOI: https://doi.org/10.1155/2013/793671.

[293] Ruiz, Efrain, Soto-Mendoza, Valeria, Barbosa, Alvaro Ernesto Ruiz, and Reyes, Ricardo. "Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm". In: *Computers & Industrial Engineering* 133 (2019), pp. 207–219. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0360835219302694.

[294] Russell, Stuart J and Norvig, Peter. *Artificial intelligence: a modern approach*. Pearson, 2010.

[295] Saadaoui, Achraf, Ouassaid, Mohammed, and Maaroufi, Mohamed. "Overview of Integration of Power Electronic Topologies and Advanced Control Techniques of Ultra-Fast EV Charging Stations in Standalone Microgrids". In: *Energies* 16.3 (2023), p. 1031.

[296] Sabar, Nasser R., Kieu, Le Minh, Chung, Edward, Tsubota, Takahiro, and Maciel de Almeida, Paulo Eduardo. "A memetic algorithm for real world multi-intersection traffic signal optimisation problems". In: *Engineering Applications of Artificial Intelligence* 63 (2017), pp. 45–53. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2017.04.021. URL: https://www.sciencedirect.com/science/article/pii/S0952197617300854.

[297] Sadati, Mir Ehsan Hesam, Akbari, Vahid, and Çatay, Bülent. "Electric vehicle routing problem with flexible deliveries". In: *International Journal of Production Research* 60.13 (2022), pp. 4268–4294. DOI: 10.1080/00207543.2022.2032451.

[298] Sadati, Mir Ehsan Hesam and Çatay, Bülent. "A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem". In: *Transportation Research Part E: Logistics and Transportation Review* 149 (2021), p. 102293. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2021.102293. URL: https://www.sciencedirect.com/science/article/pii/S1366554521000673.

[299] Safari, Momo. "Battery electric vehicles: Looking behind to move forward". In: *Energy Policy* 115 (2018), pp. 54–65. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2017.12.053. URL: https://www.sciencedirect.com/science/article/pii/S0301421517308777.

[300] Sandstrom, Kristian and Norstrom, Christer. "Managing complex temporal requirements in real-time control systems". In: *Proceedings Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*. IEEE. 2002, pp. 103–109. DOI: 10.1109/ECBS.2002.999828.

[301] Sbihi, Abdelkader and Eglese, Richard W. "Combinatorial optimization and green logistics". In: *Annals of Operations Research* 175 (2010), pp. 159–175.

[302] Schneider, Michael, Stenger, Andreas, and Goeke, Dominik. "The electric vehicle-routing problem with time windows and recharging stations". In: *Transportation science* 48.4 (2014), pp. 500–520.

[303] Schneider, Michael, Stenger, Andreas, and Hof, Julian. "An adaptive VNS algorithm for vehicle routing problems with intermediate stops". In: *Or Spectrum* 37.2 (2015), pp. 353–387.

[304] Sehgal, Adarsh, La, Hung, Louis, Sushil, and Nguyen, Hai. "Deep reinforcement learning using genetic algorithm for parameter optimization". In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE. 2019, pp. 596–601. DOI: 10.1109/IRC.2019.00121.

[305] Semelsberger, Troy A, Borup, Rodney L, and Greene, Howard L. "Dimethyl ether (DME) as an alternative fuel". In: *Journal of power sources* 156.2 (2006), pp. 497–511. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2005.05.082. URL: https://www.sciencedirect.com/science/article/pii/S0378775305007846.

[306] Shabeera, TP, Kumar, SD Madhu, Salam, Sameera M, and Krishnan, K Murali. "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm". In: *Engineering Science and Technology, an International Journal* 20.2 (2017), pp. 616–628. ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2016.11.006. URL: https://www.sciencedirect.com/science/article/pii/S2215098616304232.

[307] Shao, Sai, Guan, Wei, and Bi, Jun. "Electric vehicle-routing problem with charging demands and energy consumption". In: *IET Intelligent Transport Systems* 12.3 (2018), pp. 202–212.

[308] Shapiro, Samuel S and Wilk, Martin B. "An analysis of variance test for normality (complete samples)". In: *Biometrika* 52.3 and 4 (1965), pp. 591–611. DOI: 10.1093/biomet/52.3-4.591.

[309] Shaw, Paul. "Using constraint programming and local search methods to solve vehicle routing problems". In: *Principles and Practice of Constraint Programming — CP98*. Ed. by Maher, Michael and Puget, Jean-Francois. Springer Berlin Heidelberg, 1998, pp. 417–431. ISBN: 978-3-540-49481-2.

[310] Shi, Yu, Li, Wei, Raman, Aaswath, and Fan, Shanhui. "Optimization of multilayer optical films with a memetic algorithm and mixed integer programming". In: *Acs Photonics* 5.3 (2018), pp. 684–691. DOI: 10.1021/acsphotonics.7b01136.

[311] Sifaleras, Angelo and Konstantaras, Ioannis. "General variable neighborhood search for the multi-product dynamic lot sizing problem in closed-loop supply chain". In: *Electronic Notes in Discrete Mathematics* 47 (2015). The 3rd International Conference on Variable Neighborhood Search (VNS'14), pp. 69–76. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2014.11.010. URL: https://www.sciencedirect.com/science/article/pii/S1571065314000523.

[312] Silva, Marcos Melo, Subramanian, Anand, and Ochi, Luiz Satoru. "An iterated local search heuristic for the split delivery vehicle routing problem". In: *Computers & Operations Research* 53 (2015), pp. 234–249. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2014.08.005. URL: https://www.sciencedirect.com/science/article/pii/S0305054814002159.

[313] Silvestrin, Paulo Vitor and Ritt, Marcus. "An iterated tabu search for the multi-compartment vehicle routing problem". In: *Computers & Operations Research* 81 (2017), pp. 192–202. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.12.023. URL: https://www.sciencedirect.com/science/article/pii/S0305054816303306.

[314] Sivanandam, SN and Deepa, SN. *Introduction to Genetic Algorithms*. Springer Berlin, Heidelberg, 2008. DOI: https://doi.org/10.1007/978-3-540-73190-0.

[315] Solomon, Marius M. "Algorithms for the vehicle routing and scheduling problems with time window constraints". In: *Operations research* 35.2 (1987), pp. 254–265.

[316] Sorlei, Ioan-Sorin, Bizon, Nicu, Thounthong, Phatiphat, Varlam, Mihai, Carcadea, Elena, Culcer, Mihai, Iliescu, Mariana, and Raceanu, Mircea. "Fuel Cell Electric Vehicles—A Brief Review of Current Topologies and Energy Management Strategies". In: *Energies* 14.1 (2021), p. 252. ISSN: 1996-1073. DOI: 10.3390/en14010252. URL: https://www.mdpi.com/1996-1073/14/1/252.

[317] Speight, James. *Synthetic fuels handbook: properties, process and performance*. McGraw-Hill Companies Inc., New York, NY (United States), 2008.

[318] Stamadianos, Themistoklis, Kyriakakis, Nikolaos A, Marinaki, Magdalene, and Marinakis, Yannis. "A hybrid simulated annealing and variable neighborhood search algorithm for the close-open electric vehicle routing problem". In: *Annals of Mathematics and Artificial Intelligence* (2023), pp. 1–24. DOI: https://doi.org/10.1007/s10472-023-09858-x.

[319] Stegherr, Helena, Heider, Michael, and Hähner, Jörg. "Classifying Metaheuristics: Towards a unified multi-level classification system". In: *Natural Computing* 21 (2022), pp. 155–171. DOI: https://doi.org/10.1007/s11047-020-09824-0.

[320] Stojanović, Tatjana, Davidović, Tatjana, and Ognjanović, Zoran. "Bee colony optimization for the satisfiability problem in probabilistic logic". In: *Applied Soft Computing* 31 (2015), pp. 339–347. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.03.017. URL: https://www.sciencedirect.com/science/article/pii/S1568494615001738.

[321] Stützle, Thomas, Dorigo, Marco, et al. "ACO algorithms for the traveling salesman problem". In: *Evolutionary algorithms in engineering and computer science* 4 (1999), pp. 163–183.

[322] Stützle, Thomas and Hoos, Holger H. "MAX–MIN ant system". In: *Future generation computer systems* 16.8 (2000), pp. 889–914.

[323] Subramanian, Anand, Drummond, Lúcia MA, Bentes, Cristiana, Ochi, Luiz Satoru, and Farias, Ricardo. "A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery". In: *Computers & Operations Research* 37.11 (2010). Ed. by Prins, Christian, Labadi, Nacima, Prodhon, Caroline, and Calvo, Roberto Wolfler. Special issue: Metaheuristics for Logistics

and Vehicle Routing, pp. 1899–1911. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2009.10.011. URL: https://www.sciencedirect.com/science/article/pii/S0305054809002779.

[324] Subramanian, Anand, Penna, Puca Huachi Vaz, Uchoa, Eduardo, and Ochi, Luiz Satoru. "A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem". In: *European Journal of Operational Research* 221.2 (2012), pp. 285–295. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.03.016. URL: https://www.sciencedirect.com/science/article/pii/S0377221712002093.

[325] Sun, Yanan, Xue, Bing, Zhang, Mengjie, Yen, Gary G, and Lv, Jiancheng. "Automatically designing CNN architectures using the genetic algorithm for image classification". In: *IEEE transactions on cybernetics* 50.9 (2020), pp. 3840–3854. DOI: 10.1109/TCYB.2020.2983860.

[326] Suzuki, Yoshinori. "A dual-objective metaheuristic approach to solve practical pollution routing problem". In: *International Journal of Production Economics* 176 (2016), pp. 143–153.

[327] Syswerda, Gilbert. "Uniform crossover in genetic algorithms." In: *Proc. 3rd Intl Conference on Genetic Algorithms 1989*. 1989.

[328] Talbi, El-Ghazali. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[329] Tarantilis, Christos D, Kiranoudis, Chris T, and Vassiliadis, Vassilios S. "A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem". In: *European Journal of Operational Research* 152.1 (2004), pp. 148–158.

[330] Teodorović, Dušan, Davidović, Tatjana, Šelmić, Milica, and Nikolić, Miloš. "Bee Colony Optimization and its Applications". In: *Handbook of AI-based Metaheuristics*. Ed. by Kulkarni, A. and Siarry, P. Boca Raton: CRC Press, 2021, pp. 301–322.

[331] Teodorović, Dušan, Šelmić, Milica, and Davidović, Tatjana. "Bee colony optimization-part II: The application survey". In: *Yugoslav Journal of Operations Research* 25.2 (2015), pp. 185–219.

[332] Thipse, SS. "Compressed air car". In: *Tech Monitor* 1.2 (2008), pp. 33–37.

[333] Thompson, Paul Michael, Orlin, James B, et al. *The theory of cyclic transfers*. Tech. rep. Massachusetts Institute of Technology, Operations Research Center, 1989.

[334] T'kindt, Vincent and Billaut, Jean-Charles. "Introduction to scheduling". In: *Multicriteria Scheduling: Theory, Models and Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 5–27. ISBN: 978-3-662-04986-0. DOI: 10.1007/978-3-662-04986-0_2. URL: https://doi.org/10.1007/978-3-662-04986-0_2.

[335] Todorović, Milan and Matijević, Luka. "Integrating Blockchain into Supply Chain Management". In: *Proc. XLIX Symposium on Operational Research, SYMOPIS 2022*. Vrnjacka Banja, 2022, pp. 395–400.

[336] Toth, Paolo and Vigo, Daniele. *The vehicle routing problem*. SIAM, 2002.

[337] Toth, Paolo and Vigo, Daniele. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[338] Tukey, John W. "The Philosophy of Multiple Comparisons". In: *Statistical Science* 6.1 (1991), pp. 100 –116. DOI: 10.1214/ss/1177011945. URL: https://doi.org/10.1214/ss/1177011945.

[339] Tunga, Harinandan, Bhaumik, Arup Kumar, and Kar, Samarjit. "A method for solving bi-objective green vehicle routing problem (G-VRP) through genetic algorithm". In: *Journal of the Association of Engineers (India)* 87.2 (2017), pp. 33–48.

[340] Tzanetos, Alexandros, Fister, Iztok, and Dounias, Georgios. "A comprehensive database of Nature-Inspired Algorithms". In: *Data in Brief* 31 (2020), p. 105792. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.105792. URL: https://www.sciencedirect.com/science/article/pii/S2352340920306867.

[341] Úbeda, Sergio, Faulin, Javier, Serrano, Adrián, and Arcelus, Francisco J. "Solving the green capacitated vehicle routing problem using a tabu search algorithm". In: *Lecture Notes in Management Science* 6.1 (2014), pp. 141–149.

[342] Uhrig, Martin, Weiß, Lennart, Suriyah, Michael, and Leibfried, Thomas. "E-mobility in car parks–guidelines for charging infrastructure expansion planning and operation based on stochastic simulations". In: *EVS28 International Electric Vehicle Symposium and Exhibition*. 2015, pp. 1–12.

[343] Urošević, Dragan. "Variable neighborhood search for maximum diverse grouping problem". In: *Yugoslav Journal of Operations Research* 24.1 (2014), pp. 21–33. DOI: 10.2298/YJOR121223003U.

[344] Urošević, D. "Rešavanje nekih problema teorije grafova metodom promenljivih okolina". PhD thesis. Matematički fakultet Univerziteta u Beogradu, 2004.

[345] Utama, Dana Marsetiya, Fitria, Triani Aulya, and Garside, Annisa Kesy. "Artificial Bee Colony Algorithm for Solving Green Vehicle Routing Problems with Time Windows". In: *Journal of Physics: Conference Series*. Vol. 1933. 1. IOP Publishing. 2021, p. 012043.

[346] Utama, Dana Marsetiya, Widodo, Dian Setiya, Ibrahim, Muhammad Faisal, and Dewi, Shanty Kusuma. "A new hybrid butterfly optimization algorithm for green vehicle routing problem". In: *Journal of Advanced Transportation* 2020 (2020), p. 8834502. URL: https://doi.org/10.1155/2020/8834502.

[347] Vazirani, Vijay V. *Approximation algorithms*. Springer, 2003.

[348] Vepsäläinen, Jari, Kivekäs, Klaus, Otto, Kevin, Lajunen, Antti, and Tammi, Kari. "Development and validation of energy demand uncertainty model for electric city buses". In: *Transportation Research Part D: Transport and Environment* 63 (2018), pp. 347–361.

[349] Vidal, Thibaut, Crainic, Teodor Gabriel, Gendreau, Michel, Lahrichi, Nadia, and Rei, Walter. "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems". In: *Operations Research* 60.3 (2012), pp. 611–624.

[350] Vidal, Thibaut, Crainic, Teodor Gabriel, Gendreau, Michel, and Prins, Christian. "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows". In: *Computers & operations research* 40.1 (2013), pp. 475–489.

[351] Wamborikar, Yogesh Sunil and Sinha, Abhay. "Solar powered vehicle". In: *Proceedings of the World Congress on Engineering and Computer Science 2010*. Vol. 2. Citeseer. 2010, pp. 20–22.

[352] Wang, Ling and Lu, Jiawen. "A memetic algorithm with competition for the capacitated green vehicle routing problem". In: *IEEE/CAA Journal of Automatica Sinica* 6.2 (2019), pp. 516–526.

[353] Wang, Yong, Assogba, Kevin, Fan, Jianxin, Xu, Maozeng, Liu, Yong, and Wang, Haizhong. "Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost". In: *Journal of Cleaner Production* 232 (2019), pp. 12–29. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.05.344. URL: https://www.sciencedirect.com/science/article/pii/S0959652619318864.

[354] Wang, Yong, Li, Qin, Guan, Xiangyang, Fan, Jianxin, Xu, Maozeng, and Wang, Haizhong. "Collaborative multi-depot pickup and delivery vehicle routing problem with split loads and time windows". In: *Knowledge-Based Systems* 231 (2021), p. 107412. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107412. URL: https://www.sciencedirect.com/science/article/pii/S0950705121006742.

[355] Wang, Yuan, Wang, Ling, Chen, Guangcai, Cai, Zhaoquan, Zhou, Yongquan, and Xing, Lining. "An Improved Ant Colony Optimization algorithm to the Periodic Vehicle Routing Problem with Time Window and Service Choice". In: *Swarm and Evolutionary Computation* 55 (2020), p. 100675. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2020.100675. URL: https://www.sciencedirect.com/science/article/pii/S221065021830943X.

[356] Werpy, Marcy Rood, Burnham, Andrew, and Bertram, Kenneth. *Propane vehicles: status, challenges, and opportunities.* Tech. rep. Center for Transportation Research, Energy Systems Division, Argonne National Laboratory, 2010. DOI: https://doi.org/10.2172/982693.

[357] Whitley, Darrell, Starkweather, Timothy, and Shaner, Dan. *The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination.* Citeseer, 1991.

[358] Williamson, David P and Shmoys, David B. *The design of approximation algorithms.* Cambridge university press, 2011.

[359] Winkler, SL, Anderson, JE, Garza, L, Ruona, WC, Vogt, R, and Wallington, TJ. "Vehicle criteria pollutant (PM, NO x, CO, HCs) emissions: how low should we go?" In: *Npj Climate and atmospheric science* 1.1 (2018), p. 26.

[360] Winston, Wayne L. *Operations research: applications and algorithm.* Thomson Learning, Inc., 2004.

[361] Woller, David, Kozák, Viktor, and Kulich, Miroslav. "The GRASP Metaheuristic for the Electric Vehicle Routing Problem". In: *Modelling and Simulation for Autonomous Systems.* Ed. by Mazal, Jan, Fagiolini, Adriano, Vasik, Petr, and Turi, Michele. Cham: Springer International Publishing, 2021, pp. 189–205. ISBN: 978-3-030-70740-8.

[362] Wolpert, David H and Macready, William G. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893.

[363] Wu, Xinkai, Freese, David, Cabrera, Alfredo, and Kitch, William A. "Electric vehicles' energy consumption measurement and estimation". In: *Transportation Research Part D: Transport and Environment* 34 (2015), pp. 52–67. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2014.10.007. URL: https://www.sciencedirect.com/science/article/pii/S1361920914001485.

[364] Xiao, B, Ruan, J, Yang, W, Walker, PD, and Zhang, N. "A review of pivotal energy management strategies for extended range electric vehicles". In: *Renewable and Sustainable Energy Reviews* 149 (2021), p. 111194. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2021.111194. URL: https://www.sciencedirect.com/science/article/pii/S1364032121004822.

[365] Xiao, Jianhua, Du, Jingguo, Cao, Zhiguang, Zhang, Xingyi, and Niu, Yunyun. "A diversity-enhanced memetic algorithm for solving electric vehicle routing problems with time windows and mixed backhauls". In: *Applied Soft Computing* 134 (2023), p. 110025. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2023.110025. URL: https://www.sciencedirect.com/science/article/pii/S1568494623000431.

[366] Xiao, Yiyong and Konak, Abdullah. "A genetic algorithm with exact dynamic programming for the green vehicle routing & scheduling problem". In: *Journal of Cleaner Production* 167 (2017), pp. 1450–1463.

[367] Xiao, Yiyong and Konak, Abdullah. "A simulating annealing algorithm to solve the green vehicle routing & scheduling problem with hierarchical objectives and weighted tardiness". In: *Applied Soft Computing* 34 (2015), pp. 372–388.

[368] Xiao, Yiyong, Zhang, Yue, Kaku, Ikou, Kang, Rui, and Pan, Xing. "Electric vehicle routing problem: A systematic review and a new comprehensive model with nonlinear energy recharging and consumption". In: *Renewable and Sustainable Energy Reviews* 151 (2021), p. 111567.

[369] Xiao, Yiyong, Zhao, Qiuhong, Kaku, Ikou, and Xu, Yuchun. "Development of a fuel consumption optimization model for the capacitated vehicle routing problem". In: *Computers & operations research* 39.7 (2012), pp. 1419–1431.

[370] Xu, Wei, Zhang, Chenghao, Cheng, Ming, and Huang, Yucheng. "Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery: Mathematical Modeling and Adaptive Large Neighborhood Search Heuristic Method". In: *Energies* 15.23 (2022), p. 9222. DOI: 10.3390/en15239222.

[371] Yang, Xin-She. "Firefly Algorithms for Multimodal Optimization". In: *Stochastic Algorithms: Foundations and Applications*. Ed. by Watanabe, Osamu and Zeugmann, Thomas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 169–178. ISBN: 978-3-642-04944-6.

[372] Yang, Xin-She. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[373] Yavuz, Mesut and Çapar, Ismail. "Alternative-fuel vehicle adoption in service fleets: Impact evaluation through optimization modeling". In: *Transportation Science* 51.2 (2016), pp. 480–493.

[374] Yilmaz, Yusuf and Kalayci, Can B. "Variable Neighborhood Search Algorithms to Solve the Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery". In: *Mathematics* 10.17 (2022). ISSN: 2227-7390. DOI: 10.3390/math10173108. URL: https://www.mdpi.com/2227-7390/10/17/3108.

[375] Yogesh, KJ. "Cryogenic Liquid Nitrogen Vehicles (ZEV'S)". In: *International Journal of Scientific and Research Publications* 6.9 (2016), p. 562.

[376] Yu, Vincent F., Redi, A.A.N. Perwira, Hidayat, Yosi Agustina, and Wibowo, Oktaviyanto Jimat. "A simulated annealing heuristic for the hybrid vehicle routing problem". In: *Applied Soft Computing* 53 (2017), pp. 119–132.

[377] Yu, Zixuan, Zhang, Ping, Yu, Yang, Sun, Wei, and Huang, Min. "An Adaptive Large Neighborhood Search for the Larger-Scale Instances of Green Vehicle Routing Problem with Time Windows". In: *Complexity* 2020 (2020), p. 8210630. URL: https://doi.org/10.1155/2020/8210630.

[378] Yuan, Xinmei, Zhang, Chuanpu, Hong, Guokai, Huang, Xueqi, and Li, Lili. "Method for evaluating the real-world driving energy consumptions of electric vehicles". In: *Energy* 141 (2017), pp. 1955–1968.

[379] Zhang, Ruiyou, Guo, Jingmei, and Wang, Junwei. "A Time-Dependent Electric Vehicle Routing Problem With Congestion Tolls". In: *IEEE Transactions on Engineering Management* 69.4 (2022), pp. 861–873. DOI: 10.1109/TEM.2019.2959701.

[380] Zhang, Shuai, Gajpal, Yuvraj, and Appadoo, SS. "A meta-heuristic for capacitated green vehicle routing problem". In: *Annals of Operations Research* 269.1 (2018), pp. 753–771.

[381] Zhang, Shuai, Gajpal, Yuvraj, Appadoo, S.S., and Abdulkader, M.M.S. "Electric vehicle routing problem with recharging stations for minimizing energy consumption". In: *International Journal of Production Economics* 203 (2018), pp. 404–413. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2018.07.016. URL: https://www.sciencedirect.com/science/article/pii/S0925527318302810.

[382] Zhang, Shuai, Zhang, Weiheng, Gajpal, Yuvraj, and Appadoo, S. S. "Ant Colony Algorithm for Routing Alternate Fuel Vehicles in Multi-depot Vehicle Routing Problem". In: *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation*. Ed. by Deep, Kusum, Jain, Madhu, and Salhi, Said. Singapore: Springer Singapore, 2019, pp. 251–260.

[383] Zhang, Shuzhu, Lee, C.K.M., Chan, H.K., Choy, K.L., and Wu, Zhang. "Swarm intelligence applied in green logistics: A literature review". In: *Engineering Applications of Artificial Intelligence* 37 (2015), pp. 154–169. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2014.09.007. URL: https://www.sciencedirect.com/science/article/pii/S0952197614002218.

[384] Zhang, Weiheng, Gajpal, Yuvraj, Appadoo, Srimantoorao. S., and Wei, Qi. "Multi-depot green vehicle routing problem to minimize carbon emissions". In: *Sustainability* 12.8 (2020), p. 3500. DOI: https://doi.org/10.3390/su12083500.

[385] Zhang, Xingyi, Li, Jun, and Zhang, Lei. "A multi-objective membrane algorithm guided by the skin membrane". In: *Natural Computing* 15.4 (2016), pp. 597–610.

[386] Zhang, Zhongheng. "Introduction to machine learning: k-nearest neighbors". In: *Annals of translational medicine* 4.11 (2016), p. 218. DOI: 10.21037/atm.2016.03.37.

[387] Zhao, PX, Luo, WH, and Han, X. "Time-dependent and bi-objective vehicle routing problem with time windows". In: *Advances in Production Engineering & Management* 14.2 (2019), pp. 201–212. DOI: https://doi.org/10.14743/apem2019.2.322.

[388] Zhen, Lu, Ma, Chengle, Wang, Kai, Xiao, Liyang, and Zhang, Wei. "Multi-depot multi-trip vehicle routing problem with time windows and release dates". In: *Transportation Research Part E: Logistics and Transportation Review* 135 (2020), p. 101866. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2020.101866. URL: https://www.sciencedirect.com/science/article/pii/S1366554519301486.

[389] Zhen, Lu, Xu, Ziheng, Ma, Chengle, and Xiao, Liyang. "Hybrid electric vehicle routing problem with mode selection". In: *International Journal of Production Research* 58.2 (2020), pp. 562–576.

[390] Zhu, Yanfei, Li, Chunhui, and Lee, Kwang Y. "The NR-EGA for the EVRP Problem with the Electric Energy Consumption Model". In: *Energies* 15.10 (2022), p. 3681.

[391] Zulvia, Ferani E, Kuo, RJ, and Nugroho, Dwiyanti Y. "A many-objective gradient evolution algorithm for solving a green vehicle routing problem with time windows and time dependency for perishable products". In: *Journal of Cleaner Production* 242 (2020), p. 118428.

[392] Zündorf, Tobias. "Electric Vehicle Routing with Realistic Recharging Models". Unpublished Master's thesis. MA thesis. Karlsruhe, Germany: Karlsruhe Institute of Technology, 2014.

План третмана података

Назив пројекта/истраживања
Metaheuristic approaches to the green vehicle routing problem including alternative fuel vehicles (срп. Метахеуристичке методе за проблем еколошког рутирања возила укључујући возила на алтернативни погон)
Назив институције/институција у оквиру којих се спроводи истраживање
а) **Универзитет у Новом Саду, Факултет техничких наука, Департман за опште дисциплине у технициа**   б**) Математички институт САНУ**
Назив програма у оквиру ког се реализује истраживање
Истраживање је реализовано у оквиру израде докторске дисертације на студијском програму Математика у техници, у оквиру докторских академских студија на Факултету техничких наука у Новом Саду, Универзитета у Новом Саду.
1. Опис података
1.1 Врста студије   *Укратко описати тип студије у оквиру које се подаци прикупљају*   **Докторска дисертација**   1.2 Врсте података   а) **квантитативни**   б) **квалитативни**   1.3. Начин прикупљања података   а) анкете, упитници, тестови   б) клиничке процене, медицински записи, електронски здравствени записи   в) генотипови: навести врсту _____   г) административни подаци: навести врсту _____   д) узорци ткива: навести врсту_____   ђ) снимци, фотографије: навести врсту_____   е) текст, навести врсту _____   ж) мапа, навести врсту _____

з) остало: описати **Рачунарски експерименти**

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:

а) Excel фајл, датотека _____

b) SPSS фајл, датотека _____

c) **PDF фајл, датотека** _____

d) Текст фајл, датотека _____

e) JPG фајл, датотека _____

f) Остало, датотека _____

1.3.2. Број записа (код квантитативних података)

а) број варијабли **велики број**

б) број мерења (испитаника, процена, снимака и сл.) **велики број**

1.3.3. Поновљена мерења

а) **да**

б) не

Уколико је одговор да, одговорити на следећа питања:

а) временски размак између поновљених мера је **променљив**

б) варијабле које се више пута мере односе се на **време извршења и квалитет решења**

в) нове верзије фајлова који садрже поновљена мерења су именоване као _____

Напомене: _____

Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?

а) **Да**

б) *Не*

Ако је одговор не, образложити _____

_____

2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?

а) експеримент, навести тип **рачунарски експеримент**

б) корелационо истраживање, навести тип _____

ц) анализа текста, навести тип **анализа доступне литературе**

д) остало, навести шта _____

2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).

_____

_____

2.2 Квалитет података и стандарди

2.2.1. Третман недостајућих података

а) Да ли матрица садржи недостајуће податке? Да **Не**

Ако је одговор да, одговорити на следећа питања:

а) Колики је број недостајућих података? _____

б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не

в) Ако је одговор да, навести сугестије за третман замене недостајућих података

_____

2.2.2. На који начин је контролисан квалитет података? Описати

Квалитет података је контролисан поређењем са егзактним методама, поновљеним мерењима и статистичким тестовима.

_____

_____

2.2.3. На који начин је извршена контрола уноса података у матрицу?

Контрола уноса података је изведена на бази експертног знања

3. Третман података и пратећа документација

3.1. Третман и чување података

3.1.1. Подаци ће бити депоновани у ***Универзитет у Новом Саду*** *репозиторијум.*

3.1.2. URL адреса ***https://www.mi.sanu.ac.rs/~luka/resources/phd/***

3.1.3. DOI _____

3.1.4. Да ли ће подаци бити у отвореном приступу?

а) ***Да***

б) *Да, али после ембарга који ће трајати до* _____

в) *Не*

Ако је одговор не, навести разлог _____

3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.

Образложење

_____

_____

3.2 Метаподаци и документација података

3.2.1. Који стандард за метаподатке ће бити примењен? _____

Стандард који примењује Репозиторијум докторских дисертација Универзитета у Новом Саду

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

_____

_____

Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.

_____

_____

_____

_____

3.3 Стратегија и стандарди за чување података

3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? _____

3.3.2. Да ли ће подаци бити депоновани под шифром? Да **Не**

3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? Да Не

3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?

Да **Не**

Образложити

_____

_____

4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.

4.1 Формални стандарди за сигурност информација/података

Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о заштити података о личности *(https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html)* и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да Не

Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да Не

Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:

а) Подаци нису у отвореном приступу

б) Подаци су анонимизирани

ц) Остало, навести шта

_____

_____

5. Доступност података

5.1. Подаци ће бити

а) ***јавно доступни***

б) доступни само уском кругу истраживача у одређеној научној области

ц) затворени

Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:

_____

_____

Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима:

_____

_____

5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.

Ауторство - некомерцијално

6. Улоге и одговорност

6.1. Навести име и презиме и мејл адресу власника (аутора) података

Лука Матијевић, luka@mi.sanu.ac.rs

6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима

Лука Матијевић, luka@mi.sanu.ac.rs

6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима

Лука Матијевић, luka@mi.sanu.ac.rs