Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/16BE06Hodzic

RAZVOJ APLIKACIJE ZA PODELU GRAFA PRIMENOM UČENJA SA PODSTICAJEM

DEVELOPMENT OF AN APPLICATION FOR GRAPH PARITIONING USING REINFORCEMENT LEARNING

Maja Hodžić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Ovaj rad obrađuje podelu grafa korišćenjem jednog od algoritama učenja sa podsticajem. Grafovi se često koriste kao apstrakcije prilikom modeliranja problema. Podela grafa na manje delove jedna je od osnovnih algoritamskih operacija. Cilj ovog rada je prikaz implementacije algoritma učenja sa podsticajem za podelu grafa na dve particije.*

Ključne reči: *Podela grafa, Učenje sa podsticajem*

Abstract – *This paper deals with the division of graphs using one of the incentive learning algorithms. Graphs are often used as abstractions when modeling problems. Cutting graphs into smaller pieces is one of the basic algorithmic operations. The aim of this paper is to present the implementation of a learning algorithm with an incentive to divide a graph into two partitions.*

Keywords: *Graph partitioning, Reinforcement learning*

1. UVOD

Teorija grafova je oblast matematike veoma zastupljena u informatici. Česta je upotreba grafova za opis modela i struktura podataka, tj. njima se mogu predstaviti i vizualizovati mnogi problemi. Grafovi su sposobni da predstave složene podatke prisutne u različitim domenima. Za lakšu i bržu analizu grafova koristimo particioniranje (podelu). Postoje mnogobrojni algoritmi za ovu podelu a poslednjih godina je sve veće interesovanje za korišćenje tehnika mašinskog učenja za rešenje ovog problema.

Automatsko rezonovanje i mašinsko učenje imaju važne uloge u veštačkoj inteligenciji. Metode zasnovane na logici, koje se razvijaju u okviru automatskog rezonovanja pogodne su u slučajevima u kojima je problem moguće precizno matematički definisati.

Obično se radi o problemima koje čovek može relativno lako da formuliše, ali ih vrlo teško rešava i u kojima nisu prihvatljiva pogrešna rešenja. Sa druge strane, mašinsko učenje je posebno pogodno upravo za suprotnu vrstu problema - probleme koje čovek ne može lako ni da definiše, iako neke od njih čak vrlo lako rešava i u kojima je prihvatljiva povremena greška. Primer takvog problema može da bude prepoznavanje lica, vrste, oblika na slikama... Za ljude je ovo neobično nazivati problemom i govoriti o njegovom rešavanju, ali ako pokušamo da taj problem precizno definišemo naletećemo na mnoštvo problema.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, vanr. prof.

Zato se ovakvim problemima ne pristupa automatskim rezonovanjem već mašinskim učenjem [2], koje poput ljudi može vrlo uspešno da se nosi sa ovakvim problemom.

Obično se identifikuju tri grupe problema mašinskog učenja: to su problemi *nadgledanog učenja* (*eng. supervised learning*), problemi *nenadgledanog učenja* (*eng. unsupervised learning*) i problemi *učenja sa podsticajem* (*eng. reinforcement learning*). Ovaj rad odnosiće se upravo na *učenje sa podsticajem*, odnosno na podelu grafa korišćenjem jednog od algoritama *učenja sa podsticajem*. Grafovi se često koriste kao apstrakcije prilikom modeliranja problema. Rezanje grafa na manje komade jedna je od osnovnih algoritamskih operacija. Pojavom sve većih primera u aplikacijama kao što su naučna simulacija, društvene mreže ili putne mreže, particioniranje grafova (*graph partitioning*) stoga postaje sve važnije, mnogostranije i izazovnije.

Cilj ovog rada je prikaz implementacije algoritma *učenja sa podsticajem* za podelu grafa na dve particije približno jednake veličine.

2. UČENJE SA PODSTICAJEM

Postoje tri tipa mašinskog učenja, u odnosu na prirodu problema koji se rešava pomoću učenja:

- Nadgledano učenje
- Nenadgledano učenje
- Učenje sa podsticajem

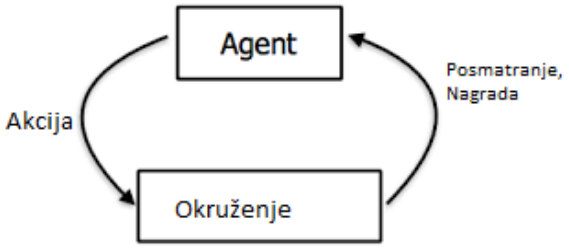
Pretpostavlja se da postoji agent koji prati tekuće stanje okruženja i ima mogućnost da izvršava određene akcije. Na osnovu izvršene akcije dobija određene numeričke vrednosti (nagrade), a cilj je da se kao ishod učenja prođe kroz aktivnosti koje dovode do maksimalne (ili dovoljno visoke) ukupne nagrade. Ključna pretpostavka kod ovog vida učenja je da nije poznato koja od preduzetih akcija je bila prava u datom kontekstu. Postoje pozitivne i negativne nagrade, a agent koristi signale za korekciju svog ponašanja. Ovaj vid učenja ima primenu u robotici, industrijskoj automatizaciji, sistemima za trening, obradu podataka, itd.

2.1. Teorijske osnove učenja sa podsticajem

Cilj algoritama učenja sa podsticajem je pronaći najbolju moguću akciju koju treba preduzeti u određenoj situaciji. Ova vrsta mašinskog učenja može naučiti da postigne cilj u neizvesnim i složenim okruženjima.

2.1.1 Osnovni pojmovi

Teorijski okvir učenja sa podsticajem se opisuje Markov-ovim procesima odlučivanja (*eng. Markov decision processes*), ili skraćeno MDP [4].



Slika 1. *Agent – Okruženje* [4]

Gradivni elementi učenja sa podsticajem su agent koji se kreće, akcija (radnja) koju agent preduzme, nagrada koju stiče, okruženje u kom je agent i stanje u kom se agent nalazi (na tačno određenom mestu u okruženju). U svakom koraku agent izvrši akciju At primi stanje okruženja (sistema) Ot i primi nagradu/kaznu Rt. U isto vreme okruženje primi akciju At, šalje stanje okruženja Ot i šalje nagradu Rt. Agent izvršava akciju na osnovu posmatranja okruženja i nagrade. Agent je algoritam. Stanje je informacija koja se koristi za određivanje sledećeg koraka. Stanje može da gleda samo poslednje u istoriji, da ga prethodno ne zanima.

Istorija je skup posmatranja, akcija i nagrada prikazana u izrazu 1:

$$H_t = A_1 O_1 R_1, \dots, A_t O_t R_t \quad (1)$$

Stanje St je funkcija istorije H_t:

$$S_t = f(H_t) \quad (2)$$

Stanje na osnovu prethodnog stanja je stanje Markov-a ako bi takvo bilo i na osnovu svih prethodnih. Stanje okruženja je stanje Markov-a [4].

$$P[S_{t+1}|S_t] = P\,[S_{t+1}|S_1, \dots, S_t] \quad (3)$$

Iz izraza 3 vidimo da bi stanje S_{t+1} na osnovu prethodnog stanja S_t bilo isto i na osnovu svih prethodnih. Agent i okruženje interaguju u diskretnim trenucima $t = 0, 1, \dots$ U svakom trenutku t, agent opaža stanje okruženja S_t iz konačnog skupa stanja S i preduzima akciju A_t iz konačnog skupa dopustivih akcija $A(s)$ u konkretnom stanju s. Pritom, dobija nagradu R_{t+1} iz konačnog skupa nagrada R i prelazi u novo stanje S_{t+1}. MDP i agent zajedno proizvode putanju: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

Osnovno svojstvo MDP-a je Markov-o svojstvo, da novo stanje i nagrada zavise samo od prethodnog stanja i preduzete akcije, a ne od cele istorije procesa.

Uloga nagrada je da na implicitan način definišu cilj agenta. Treba da budu tako izabrane da maksimizujući nagradu, što je agentov cilj, agent istovremeno obavlja posao koji želimo da obavi. Nagrada je način da agentu saopštimo šta treba da uradi, a ne kako to treba da uradi. Cilj agenta je da maksimizuje dugoročnu nagradu. Dugoročna nagrada G_t u trenutku t u odnosu na neku putanju definišemo na sledeći način:

$$G_t = R_{t+1} + R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (4)$$

pri čemu je ɣ metaparametar umanjenja (*eng. discount*) za koji važi $0 \leq \gamma \leq 1$.

3. PODELA GRAFA

Particioniranje tj. podela grafa je smanjenje grafa na manji graf podelom njegovog skupa čvorova u međusobno isključujuće grupe. Rubovi originalnog grafa koji se ukrštaju između grupa proizvešće grane u particioniranom grafu. Ako je broj rezultirajućih grana mali u poređenju sa originalnim grafom, tada će particionirani graf možda biti pogodniji za analizu i rešavanje problema od originala. Pronalaženje particije koja pojednostavljuje analizu grafova je težak problem. Iizmeđu ostalog ima primenu u naučnom računanju, raspoređivanju zadataka u višeprocesorskim računarima...

Dva uobičajena primera podele grafova su problemi sa minimalnim rezanjem i maksimalnim rezanjem. Analiza grafova je ključ obrade velikih grafova: podela velikog grafa na podgrafove i distribucija podgrafa.

Dat je graf $G = (V, E)$ gde je V skup čvorova, a E skup grana, k uravnotežena podela grafa deli ga na k podgrafova

$$\frac{|E|}{\mathrm{k}\,(1+\mathrm{e})} \quad (5)$$

gde je u izrazu 5 k broj particija, a $e > 0$ je neuravnoteženi odnos.

Nasumični i Hash algoritmi za particioniranje izuzetno su loši u pozicioniranju i odsecanju. Učenje sa podsticajem je klasa algoritama mašinskog učenja inspirisanog principom stimulus-odgovor.

3.2. Revolver algoritam

Revolver [3] je aplikacija učenja sa podsticajem (*Reinforcement Learning)* za podelu (particioniranje) grafa. Algoritam koristi *Learning Automata* i *Label Propagation* za particioniranje grafova.

Pri obuci *Learning Automata* akcije se biraju pomoću vektora verovatnoće P. Za svaku particiju čvora v daje se rezultat ($\psi(v)$). Čvorovi migriraju na svoju kandidatsku (novu) particiju sa verovatnoćom migracije.

Signal nagrade R izračunava se na sledeći način: signal nagrade ako $\psi(v)$ ima najveći rezultat ili ima pozitivnu verovatnoću migracije, kazneni signal suprotno. Vektor verovatnoće se ažurira uzimajući u obzir izračunati signal.

LA pomaže *Revolver*-u da proizvede lokalizovane particije (lokalitet) i uravnotežene particije (skalabilnost). *Revolver* je paralelni algoritam za particioniranje sposoban za particioniranje velikih grafova na jednoj mašini sa zajedničkom memorijom. *K*oristi asinhroni okvir za obradu, koji koristi *Reinforcement learning* i *Label propagation* za prilagodljivo razdvajanje grafa. Pored toga, usvaja vertikalno centrični prikaz grafa gde svakom čvoru je dodeljen autonomni agent odgovoran za odabir odgovarajuće particije, distribuirajući time računanje u svim čvorovima.

Otud normalizovani *LP*, koji se sastoji od normalizovanog ponderisanja τ i kaznenog π definisanog na sledeći način [3]:

$$Score\,(v,l) = \tau\,(v,l) + \frac{\pi(l)}{2} \quad (6)$$

$$\tau\,(v,l) = \sum_{u \in N_{(v)}} \frac{\varpi\,(u,v) \cdot \delta(\psi b(u),l)}{\sum_{u \in N_{(v)}} \varpi\,(u,v)} \quad (7)$$

$$\pi(l) \frac{1-(\frac{b(l)}{C})}{\sum_{i=1}^{k} 1-(\frac{b(l_i)}{C})} \quad (8)$$

U *Revolveru,* novi normalizovani algoritam *LP* ekstrapoliran je da bi formirao ciljnu funkciju koja stvara pondere koji izražavaju adekvatnost dodeljenih particija pomoću *LA*. Štaviše, *Revolver* deli graf na vertekscentrični način. Konkretno, svaki čvor izvlači informacije iz svojih susednih čvorova da bi izračunao rezultat za svaku particiju, pre nego što vrati obračunate ocene (kao težine). Potom se nagrade izračunavaju u svakom čvoru na osnovu akumuliranih težina prikupljenih od svih suseda čvorova. Konačno, verovatnoće akcija povezanih sa particijama se ažuriraju koristeći težine i nagrade u skladu s tim.

Learning Automata (LA) je potklasa učenja sa podsticajem algoritma koji bira svoje nove radnje koristeći prethodno iskustvo sa određenim sredinama.

Label Propagation (*LP*) je polunadgledano mašinsko učenje, algoritam koji dodeljuje oznake velikom skupu neobeleženih podataka koristeci malu količinu označenih podataka.

Suštinska ideja je podela grafa *G = (V, E)* sa *LA = (A, P, R)* na *k* particija

- Mreža *LA*-a analogna *G*-u je stvorena gde je |V| = |LA|
- *LA* je dodeljen svakom čvoru *v*
- Susedni *LA* mogu se ispitati pomoću skupa *E*.
- Skup akcija *A* je isti kao skup dostupnih particija, *|A| = k*
- Vektor verovatnoće *P* je inicijalizovan sa 1 / *k*
- Skup nagrada *R* izračunava se pomoću *LP*

Pri obuci *Learning Automata* akcije se biraju pomoću vektora verovatnoće *P*. Za svaku particiju čvora *v* daje se rezultat *(ψ(v))*. Čvorovi migriraju na svoju kandidatsku (novu) particiju sa verovatnoćom migracije.
Signal nagrade *R* izračunava se na sledeći način: signal nagrade ako *ψ(v)* ima najveći rezultat ili ima pozitivnu verovatnoću migracije, kazneni signal u suprotnom. Vektor verovatnoće se ažurira uzimajući u obzir izračunati signal. Ukoliko je *R* signal nagrada verovatnoće se azuriraju pomocu izraza:

$$P_j\,(n\ +\ 1) = \begin{cases} P_j(n) + \alpha \cdot w_j(n)\left(1 - P_j(n)\right) & j\ = i \\ P_j(n)\left(1 - \alpha \cdot w_j(n)\right) & j \neq i \end{cases} \quad (9)$$

U slucaju da je *R* kazna verovatnoće se ažuriraju sa kaznom:

$$P_j\,(n\ +\ 1) = \begin{cases} P_j(n)(1 - \beta \cdot w_j(n))) & j = i \\ P_j(n)\big(1 - (\beta \cdot w_j(n))\big) + \frac{\beta}{m-1} & j \neq i \end{cases} \quad (10)$$

4. IMPLEMENTACIJA REVOLVER ALGORITMA

Kod implementiran u radu može se predstaviti pseudo kodom na sledeći način (11):
Input: graf predstavljen matricom susedstva
Output: podeljen graf na dve particije P_1 i P_2

1 Inicijalizacija α i β parametara nagrade i kazne
2 Random odabir početnog čvora od kog se deli
3 **repeat**
4 Biranje sledećeg kandidata na osnovu liste verovatnoće izbora (RoulleteWheel)
5 Za izabrani čvor izračunati Score funkciju (7)
6 **if** Score > 0 i verovatnoća > 0
7 Postaviti čvor u posmatranu particiju, ažurirati verovatnoće sa nagradom (10)
8 **else** Čvor ne seli u particiju, verovatnoće se ažuriraju sa kaznom (11)
9 **until** veličina particije <= polovina grafa
10 **repeat**
11 **for each** Čvor in P_1 do
12 P = veze sa čvorovima iz druge particije - veze sa čvorovima iz trenutne particije
13 **end for each**
14 **for each** Čvor in P_2 do
15 D = veze sa čvorovima iz druge particije - veze sa čvorovima iz trenutne particije
16 **end for each**
17 Za čvorove $V_i \in P_1$ & $V_j \in P_2$ izračunati J = P+D, J_{ij} = max(J) □ V_i i V_j menjaju particije
18 **until** (max(J) <=0)

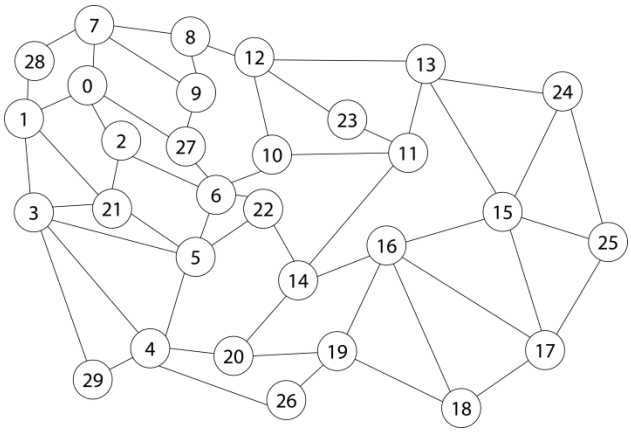
U prvom delu koda (linija 2 do 9) koristi se princip *Revolver* algoritma za početnu (inicijalnu) podelu grafa. Na slučajan način se izabere početni čvor, zatim se za početni čvor i sve naredne lista verovatnoće izbora njegovih suseda prosleđuje na Roulete-selekcija (Roulete Wheel funkciju), gde čvor sa većom verovatnoćom ima veću šansu da bude izabran. Za izabrani čvor se računa vrednost Score funkcije po izrazu 6. Prvi član izraza računa vrednost *τ (v, l)*, po izrazu 7, koja daje odnos koliko je susednih čvorova čvora *v* u particiji za koju računamo u odnosu na broj u drugoj particiji.

Kronerova konstanta *δ* ima vrednost *1* ukoliko je čvor u razmatranoj particiji. Treba naglasiti da se rad odnosi na bestežinske grafove tj. težine svih grana imaju vredost *1*. Dakle, rezultat računanja *τ (v, l)* je razlomak gde je u brojiocu broj suseda koji se nalaze u particiji za koju računamo dok je u imeniocu ukupan broj suseda čvora *v*. Iz izraza 8, gde je *b(l)* trenutno opterećenje particije a *C* ukupan kapacitet, dobijamo drugi član *π(l)* koji proizvodi kazne za particije i normalizuje se na osnovu ukupnog opterećenja sistema. Ukoliko je vrednost Score-a veća od nule vrednost nagrade je *1*, čvor se seli u razmatranu particiju i verovatnoće se ažuriraju pomoću izraza 9. U suprotnom čvor ne seli, a verovatnoće se ažuriraju prema izrazu 10.

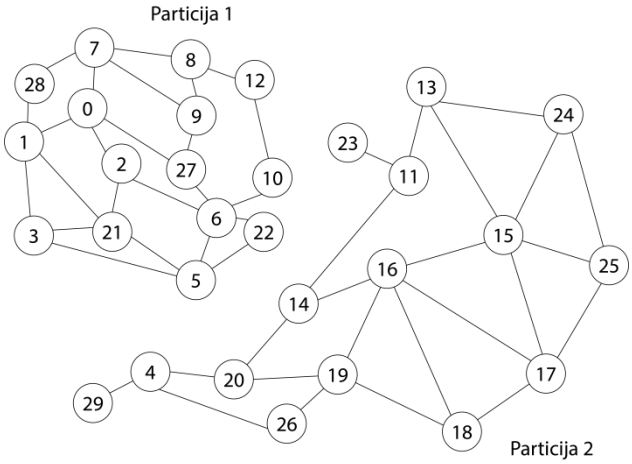
Nakon inicijalne podele naveden je iterativni postupak za unapređenje podele. Razmenjuju se čvorovi između particija kako bi se smanjio broj grana između čvorova koji se nalaze u različitim particijama. Implementiran je po uzoru na KL algoritam (Kernighan i Lin) [5]. Za svaki čvor proveravaju se njegovi susedi tj. pronalaze se čvorovi čijom se razmenom između particija smanjuje broj presečenih grana uz zadovoljavanje ograničenja o veličini particija. Uvode se pojmovi spoljašnjih i unutrašnjih veza - odnos između njih definiše da li čvor već pripada odgovarajućoj particiji ili ga treba prebaciti. Ukoliko čvor ima više spoljašnjih veza nego unutrašnjih to jest suseda u drugoj particiji on postaje kandidat za prebacivanje. Kada se nađe čvor u particiji *1* i drugi u particiji 2 (pogodni za prebacivanje) praktično im se zamene mesta. Ažurira se stanje u particijama i traže se drugi kandidati za razmenu. Ukoliko ih više ne nalazimo postupak se prekida.

5. REZULTATI TESTIRANJA

Navedeni kod testiran je na više grafova. Svi grafovi su neusmereni i bestežinski generisani na slučajan način [6]. Rezultati podele prilično zavise od inicijalne podele, samim tim i početnog čvora od kog se deli. Za inicijalnu podelu se koristi Revolver algoritam, te se inicijalna podela vrlo malo ili nikako razlikuje od finalne podele. Što govori u prilog korišćenju samog Revolver algoritma. Na slikama 2 i 3 prikazan je primer podele grafa.



Slika 2: *Graf za podelu*



Slika 3: *Podeljen graf*

6. ZAKLJUČAK

U ovom radu prikazana je teorijska analiza, razvoj aplikacije (implementacija algoritma) i dobijeni rezultati pri pokretanju aplikacije. Algoritam je kodiran u c# programskom jeziku i pri pokretanju daje vrlo dobre rezultate na različitim grafovima koji su korišteni za testiranje. Konkretno, Revolver algoritam se vrlo dobro pokazao pri inicijalnoj podeli koju u nekim slučajevima nije potrebno korigovati. Revolver čuva lokalnu podelu bez žrtvovanja ravnoteže opterećenja.

7. LITERATURA

[1] Alan Turing, Computing Machinery and Intelligence, (1950.)

[2] Mladen Nikolić, Anđelka Zečević, Mašinsko učenje, Beograd (2019.)

[3] Mohammad Hasanzadeh Mofrad, Rami Melhem and Mohammad Hammoud, Partitioning Graphs for the Cloud using - Reinforcement Learning, Pittsburgh USA https://arxiv.org/pdf/1907.06768.pdf (17.07.2019.)

[4] Introduction to Reinforcement Learning with David Silver (https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver), (08.2021.)

[5] Darko Čapko, Optimalna podela velikih modelapodataka u okviru nadzorno-upravljačkih elektroenergetskih sistema, Novi Sad (2012.)

[6] Graph online (https://graphonline.ru/en/) (09.2021.)

Kratka biografija:

Maja Hodžić, rođena 29.05.1987. godine u Dubrovniku, Republika Hrvatska. Diplomski rad (osnovne akademske studije) na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika - usmerenje Automatika i upravljanje sistemima – odbranila je 2018. godine