SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 000

Primjena konstruktivnog optimizacijskog algoritma na problem rasporeda studenata

Martin Čekada

Zagreb, svibanj 2019.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

Uvod rada. Nakon uvoda dolaze poglavlja u kojima se obrađuje tema.

2. Pregled područja

Problemi raspoređivanja su specijalizacija transportnih problema te su jedni od temeljnih optimizacijskih problema. Općenito gledano, problemi raspoređivanja bi se mogli definirati na sljedeći način:

> *Problem se sastoji od agenata i zadataka. Svakom agentu može biti dodijeljen bilo koji od zadataka uz određenu cijenu, a cijena može varirati ovisno o uparivanju agenta i zadatka. Potrebno je svakom agentu dodijeliti zadatak tako da ukupna cijena dodjeljivanja bude minimalna.*

Ovakvi problemi bi se mogli pokušati riješiti tako da se generira svaka kombinacija dodjeljivanja agenata i zadataka te da se odredi dodjeljivanje s najmanjom cijenom. Ukoliko bi se dodjeljivanje vršilo za n agenata i n zadataka, složenost dodjeljivanja bi bila $n!$. Porastom broja n vrijeme potrebno da se na ovaj način odredi dodjeljivanje s najmanjom cijenom vrlo brzo postaje preveliko da bi se izračunalo u realnom vremenu. Zbog toga kod problema raspoređivanja (i općenito optimizacijskih problema) ne tražimo iscrpno optimalno rješenje, nego različitim pristupima pokušavamo pronaći dovoljno dobro rješenje. U nastavku slijedi opis nekih specifičnih problema raspoređivanja.

Primjer 1.[1] Zadana je funkcija $g(x, y, z)$ nad domenom $[-300, 500] \times [-300, 500] \times [-300, 500] \subset \mathbb{R} \times \mathbb{R} \times \mathbb{R}$. Pronaći točku (x, y, z) za koju funkcija g poprima maksimalnu vrijednost.

Primjer 2. Satničaru su na raspolaganju popis kolegija koji se predaju, popis studenata i njihov izbor kolegija, popis slobodnih dvorana i termina, željeni tjedni broj predavanja za svaki od kolegija te popis nastavnika koji predaju određene kolegije. Satničar treba zadovoljiti sljedeće uvjete:

– svi studenti imaju zakazana sva predavanja i mogu ih slušati bez kolizija,

[1] Preuzet iz (**?**)

– niti jedan nastavnik ne drži više predavanja istovremeno,

– niti ujednu dvoranu ne smije biti smješteno više studenata nego što je kapacitet dvorane te

– niti ujednu prostoriju ne smiju biti smještena dva predavanja istovremeno.

Također, bilo bi poželjno kada bi satničar osigurao da:

– student u danu ima barem dva predavanja ili niti jedno,

– student ima minimalan broj rupa u danu,

– nastavnik ima minimalan broj rupa u danu te

– nastavnik ima minimalan broj promjena dvorana u danu.

Primjer 3. Primjer u nastavku je pojednostavljen i prilagođen organizaciji međuispita na Fakultetu elektrotehnike i računarstva. Prilikom izrade ispita dostupni su podaci o predmetima za koje treba održati ispit, podaci o slobodnim terminima te podaci o studentima i ispitima kojima oni mogu pristupiti. Potrebno je izraditi raspored međuispita tako da niti jedan student ne piše istovremeno dva ispita, da svaki student može pristupiti svakom svojem ispitu te tako da se niti u jednoj dvorani ne pišu istovremeno dva ispita. Bilo bi poželjno da student ima što ravnomjernije raspoređen broj slobodnih dana između ispita.

Na prethodnim primjerima se može primijetiti da se neki od uvjeta moraju ispuniti, dok su neka samo poželjna. To su tvrda (engl. *hard*) i meka (engl. *soft*) ograničenja (engl. *constraints*). Tvrda ograničenja moraju biti ispunjena kako bi rješenje bilo prihvatljivo (npr. u primjeru 2 ne može konačno rješenje biti ono u kojem nastavnik istovremeno predaje u dva termina). S druge strane, meka ograničenja nisu obavezna, ali što su ona ispunjenija, to je rješenje bolje.

Kako je u početku poglavlja ustanovljeno da se ovakvi problemi ne mogu rješavati tehnikom grube sile (engl. *brute force*), postavlja se pitanje na koje se načine u realnom vremenu može pronaći dovoljno dobro rješenje. Neki od optimizacijskih algoritama korišteni za rješavanje problema raspoređivanja su navedeni u nastavku.

2.1. Optimizacijski algoritmi

U ovom poglavlju su navedena samo dva primjera algoritama koji se mogu primijeniti na probleme raspoređivanja. Konstruktivni optimizacijski algoritam je detaljno opisan u narednim poglavljima, a postoji još mnoštvo drugih algoritama koji se mogu koristiti (algoritam roja čestica, algoritmi umjetnih imunoloških sustava, genetski algoritmi

itd.). Svaki od algoritama ima svoje prednosti i nedostatke te je prikladniji za određenu specijalizaciju problema, dok za drugu vrstu problema daje slabije rezultate. Ova tvrdnja je poznata kao *no-free-lunch* teorem kojeg su Wolpert i Macready dokazali u svojim radovima, a u originalu glasi:

> *All algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.* [2]

2.1.1. Mađarski algoritam

Algoritam je razvio i objavio Harold Kuhn 1955. godine, a nadjenuo mu je ime *mađarski* jer se algoritam velikom mjerom oslanja na rad dvaju mađarskih matematičara Dénes Kőnig i Jenő Egerváry. Algoritam je namijenjen raspoređivanju n poslova na n radnika pri čemu jedan radnik može obavljati samo jedan posao. Dakako, problem bi se mogao preslikati na prethodno naveden primjer 1. ili na raspoređivanje studenata u kojem je svakom studentu pridružen različit termin.

Vremenska složenost ovog algoritma je $\mathcal{O}(n^3)$, a algoritam koristi sljedeći teorem: *Ako svakom elementu bilo kojeg retka ili stupca matrice kazne dodamo ili oduzmemo neki broj, tada je optimalno raspoređivanje za rezultantnu matricu također optimalno i za prvotnu matricu.* Pritom se matricom kazne smatra matrica čiji redci predstavljaju radnike, stupci predstavljaju poslove, a vrijednost matrice na mjestu $C(i, j)$ predstavlja cijenu dodjeljivanja radniku i posao j. Koraci algoritma slijede.

1. Za svaki redak matrice, pronađi najmanji element i oduzmi ga od svakog elementa u njegovom retku.

2. Ponovi korak 1. za svaki stupac.

3. Prekrij sve nule u matrici koristeći minimalan broj horizontalnih i vertikalnih linija.

4. Test optimalnosti: ako je minimalan broj linija potrebnih da se prekriju sve nule u matrici jednak n, tada je optimalno rješenje moguće i završavamo s izvođenjem algoritma. Ako je broj linija manji on n, tada optimalno rješenje nije pronađeno te nastavljamo na korak 5.

5. Pronađi najmanji element matrice koji nije prekriven niti jednom linijom. Oduzmi taj element od svakog neprekrivenog retka i potom ga dodaj svakom prekrivenom stupcu. Vrati se na korak 3.

Pogledajmo to sada na primjeru raspoređivanja poslova utovara robe, transporta robe te istovara robe među radnicima Petrom, Ivanom i Markom. Neka je cijena pridjeljivanja poslova između radnika određena sljedećom matricom:

$$\begin{bmatrix} 25 & 40 & 35 \\ 40 & 60 & 35 \\ 20 & 40 & 25 \end{bmatrix}.$$

1. Oduzmi najmanju vrijednost svakog retka.

$$\begin{bmatrix} 0 & 15 & 10 \\ 5 & 25 & 0 \\ 0 & 20 & 5 \end{bmatrix}.$$

2. Prekrij sve nule minimalnim brojem linija.

$$\begin{bmatrix} \cancel{0} & 0 & 10 \\ \cancel{5} & 25 & \cancel{0} \\ \cancel{0} & 20 & 5 \end{bmatrix}$$

3. S obzirom na to da je broj linija potrebnih za prekrivanje svih nula jednak 3 optimalno rješenje je pronađeno.

Poslove treba raspodijeliti tako da Petar radi transport robe, Ivan istovar, a Marko utovar. Cijena takvog raspoređivanja je 95.

2.1.2. Algoritam kolonije mrava

Ovaj algoritam je inspiriran procesom kojim mravi pronalaze najkraći put između mravinjaka i hrane. Mravi se prilikom potrage za hranom ne služe svojim osjetilom vida, nego osjetilom feromona. Fermoni su kemijski tragovi koje mravi ostavljaju za sobom krećući se od mravinjaka do hrane i povratno. Intenzitet mirisa fermona slabi s vremenom ako tim putem ne prolaze mravi. Kako mravi na ovaj način pronalaze najkraći put između hrane i mravinjaka, intuitivno je jasno da je ovaj algoritam prikladan za pretraživanje grafova. Osim za pretraživanje grafova, algoritam kolonije mrava je uspješno

primjenjivan na probleme izrade rasporeda [3] te je u velikoj mjeri sličan konstruktivnom optimizacijskom algoritmu koji je tema ovog rada (fus nota: detaljnija usporedba algoritama je dana u narednim poglavljima). Pseudo kod algoritma je: [4]

$ponavljajdoknijekrajponovizasvakogmravastvorirjeenjevrednujrjeenjekrajponoviisparifero$

Rješenje za svakog mrava se gradi tako da se odabire brid po brid kojima će mrav prolaziti (parovi bridova naravno moraju imati jednu zajedničku točku, odnosno mora se moći nakon prvog brida prijeći na drugi brid itd.). Odabir kojim bridom će mrav poći iz trenutnog čvora se određuje slučajnim proporcionalnim pravilom (engl. *random proportional rule*):

$$p_i^k j = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_k^i} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{ako } j \in N_k^i \\ 0 & \text{ako} j \notin N_k^i \end{cases}$$

pri čemu su α i β konstante, τ je intenzitet fermona na bridu, a η je heuristička informacija koliko je povoljno krenuti bridom. N_k^i predstavlja sve bridove kojima je početni čvor trenutni čvor u kojem se mrav nalazi. Prije prve iteracije, sve vrijednosti tao se postavljaju na vrijednost koja je nešto veća od očekivane vrijednosti koju će mravi ostavljati u svakoj iteraciji. Za izračun se koristi formula:

$$\tau_0 = \frac{m}{C^{mn}}$$

pri čemu je m broj mrava, a C^{mn} je procjena najkraćeg puta dobivena nekim jednostavnijim algoritmom.

Isparavanje fermonskih tragova se vrši tako da se trenutna vrijednost pomnoži s konstantnim koeficijentom prema izrazu:

$$\tau_{ij} \leftarrow \tau_{ij} \cdot (1 - \rho)$$

Mravi ažuriraju feromonske tragove proporcionalno dobroti rješenja koje su izgradili prema izrazu:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

pri čemu je:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{C^k} & \text{ako je } i\text{-}j \text{ na stazi mrava} \\ 0 & \text{ako nije na stazi} \end{cases}$$

[3] Na primjer `https://www.researchgate.net/publication/268047044_Evolucijsko_racunanje_i_problem_izrade_rasporeda`

[4] Preuzeto s `https://www.researchgate.net/publication/268047044_Evolucijsko_racunanje_i_problem_izrade_rasporeda`

3. Opis algoritma

Konstruktivni optimizacijski algoritmi su široko korišteni algoritmi za heurističku optimizaciju. Inačica konstruktivnog optimizacijskog algoritma koji je korišten u ovom radu se temelji na radu (**?**).

3.1. Opći matematički model algoritma

Promatramo problem s konačnim skupom izvodljivih rješenja S i funkcijom kazne $f : S \to \mathbb{R}$. Neka je skup S^* skup optimalnih rješenja te isključimo trivijalna rješenja, odnosno pretpostavim da $|S| > 1$ i da $S^* \neq S$. Neka se svako rješenje iz skupa S može zapisati kao konačan niz simbola $s = (s_1, ..., s_L)$ iz skupa abecede $\mathbb{A} := a_1, ..., a_K$. L je fiksna duljina rješenja. Model je generaliziran funkcijom izvodljivosti $C_i(y, a)$ koja dodjeljuje težinu svakom $a \in \mathbb{A}$ za svako moguće parcijalno rješenje y duljine i. Prema tome, ako je $C_i(y, a) = 0$, parcijalno rješenje y se ne može nastaviti tako da se na njega nadoda znak a. Što je vrijednost funkcije $C_i(y, a)$ veća, toliko je poželjnije da se niz nastavi znakom a pohlepno (engl. *greedy*) gledano. Pretpostavlja se da su vrijednosti funkcije normirane. Model gradi rješenja korak po korak dodavajući nove simbole na desni kraj niza sve dok rješenje nije potpuno. Formalnije rečeno, definiramo skup R_i izvodljivih parcijalnih rješenja rekurzivno kako slijedi: neka $\Diamond$ pretstavlja prazan string. Pretpostavljamo da je dano:

$$C_0(\Diamond, \cdot) : \mathbb{A} \to [0, 1], \sum_{a \in \mathbb{A}} C_0(\Diamond, a) = 1$$

što pretstavlja poželjnost i izvodljivost znaka a na prvom mjestu rješenja. Nadalje definiramo:

$$R_1 := \{a \in \mathbb{A} | C_0(\Diamond, a) > 0\}$$

Kao skup izvodljivih rješenja duljine 1. Pretpostavimo da je definiran skup R_i za neki $i \in 0, ..., L - 1$ i da je dano:

$$C_i(y, \cdot) : \mathbb{A} \to [0, 1], \sum_{a \in \mathbb{A}} C_i(y, a) = 1, \text{za svaki} y \in R_i$$

Neka (y, a) označava konkateniranje simabola $a \in \mathbb{A}$ na desni kraj parcijalnog rješenja y. Definiramo:

$$R_{i+1} := \{(y, a) | y \in R_i, a \in \mathbb{A}, C_i(y, a) > 0\}$$

i neka je $S := R_L$. Za svaki $y \in R_i$, $i \in \{0, ..., L-1\}$, neka je

$$C_i(y) := \{a \in \mathbb{A} | C_i(y, a) > 0\}$$

bude potpora $C_i(y, \cdot)$.

Ovakva generalizacija modela ne postavlja gotovo nikakva ograničenja na optimizacijski problem. Odnosno, odgovarajućim odabirom A, S i $C(y, a)$ svaki se problem može prilagoditi na ovaj model, no efikasnost može varirati. Model je razumnije koristiti u slučajevima kada je $|A| \ll |S|$. Ovo dodatno potvrđuje teorem s kraja prethodnog poglavlja koji tvrdi da svaki algoritam ne daje jednako dobre rezultate na svakom problemu.

3.2. Generalizirani algoritam

U suštini, algoritam razvija distribuciju nad skupom $S = R_L$ svih izvodljivih rješenja dajući pritom veliku vjerojatnost optimalnim rješenjima iz skupa S^*. Neka $\mathbb{P}(\mathbb{A})$ označeva skup svih vjerojatnosti nad skupom $\mathbb{A}$. Tada je $p \in \mathbb{P}(\mathbb{A})^L, p = (p(1), ..., p(L))$ vjerojatnost nad skupom $\mathbb{A}^L$ koja opisuje odabiranje rješenja $s = (s_1, ..., s_L) \in \mathbb{A}^L$ pri čemu je L simbola $s_1, ..., s_L$ odabrano neovisno. Pritom je $p(i) = p(a; i)_{a \in \mathbb{A}} \in \mathbb{P}(\mathbb{A})$ je distribucija vjerojatnosti za simbol na i-toj lokaciji. Ulazni podaci za algoritam su:

1. funkcija poželjnosti $C_i(\cdot, \cdot), i \in 0, ..., L-1$,
2. niz koeficijenata izglađivanja $(\varrho_t)_{t \geq 1}$ uz $\varrho_t \in (0, 1)$,
3. veličinu uzorka N i veličinu poduzorka N_b te
4. početnu distribuciju $p_0 in \mathbb{P}(\mathbb{A})^L$

3.2.1. Početak

Za $t = 0$, postavi $p = p_0$. Iteriraj kroz korake $t = 1, 2, ...$ sve dok uvijet zaustavljanja nije zadovoljen.

3.2.2. Uzorkovanje (engl. *sampling*)

Ako je trenutna distribucija $p \in \mathbb{P}(\mathbb{A})^L$, tada je vjerojatnost uzrokovanje rješenje $s = (s1, ..., sL) \in S$ dana izrazom:

$$Q_p(s) := Q_p(s_1; 1, \Diamond) \cdot \prod_{i=2}^{L} Q_p(s_i; i, (s_1, ..., s_{i-1}))$$

pri čemu je:

$$Q_p(a; i, y) := \frac{p(a, i)C_{i-1}(y, a)}{\sum_{a' \in \mathbb{A}} p(a', i)C_{i-1}(y, a')}$$

vjerojatnost da se izvodljivi simbol $a \in \mathbb{A}$ nadoda na poziciju i izvodljivog parcijalnog rješenja $y \in R_{i-1}$. Koristi se konvencija $\frac{0}{0} = 0$. Na ovakav način, algoritam uzorkuje N rješenja $s^{(1)}, ..., s^{(N)}$ neovisno i podjednako distribuirano.

3.2.3. Ocjenjivanje (engl. *evaluation*)

Neka su uzorci $x := (s^{(1)}, ..., s^{(N)})$ poredani prema funkciji kazne f:

$$f(s^{n_1}) \leq f(s^{n_2}) \leq \cdots \leq f(s^{n_N})$$

te neka je odabrano najboljih N_b uzoraka $N_b := \{s^{(n_1)}, s^{(n_2)}, ..., s^{(n_{N_b})}\}$. Nakon toga određujemo relativnu frekvenciju simbola a na poziciji $i \in 1, ..., L$ u odabranom djelu uzorka

$$w(a; i, x) := \frac{1}{N_b} \sum_{s \in N_b} \mathbb{1}_{\{a\}}(s_i)$$

i prikupimo te frekvencije za svaki $a \in \mathbb{A}$ te kreiramo $w(i, x) = w(a; i, x)_{a \in \mathbb{A}}$ i

$$w(x) := (w(1, x), ..., w(L, x)).$$

Tada je $w(x)$ distribucija vjerojatnosti za $\mathbb{P}(\mathbb{A})^L$ koja daje relativne frekvencije simbola iz boljeg dijela uzorka x uzorkovanog s vjerojatnosti Q_p.

3.2.4. Ažuriranje (engl. *update*)

Trenutnu distribuciju p ažuriramo kao kombinaciju p i relativne frekvencije $w(x)$

$$p := (1 - \varrho_{t+1})p + \varrho_{t+1}w(x)$$

U idućem koraku se brojač t uvećava za 1 i korak uzorkovanja se obavlja s novom ditribucijom p.

4. Problem vodećih jedinica

Kako bi se bolje prikazalo ponašanje konstruktivnog optimizacijskog algoritma, u nastavku je razmatrana primjena algoritma na problem vodeće jedinice (engl. *LeadingOne*). Problem se sastoji od generiranja niza nula i jedinica s ciljem maksimiziranja broja početnih jedinica. Optimalno rješenje je ono u kojem se niz sastoji isključivo od jedinica. Prateći notaciju iz prethodnih poglavlja, problem se može formalizirati tako da je abeceda znakova $\mathbb{A} = \{0, 1\}$, skup svih rješenja $S = \{0, 1\}^L$ te funkcija kazne:

$$f(s) := L - \sum_{l=1}^{L} \prod_{i=1}^{l} s_i, zas = (s_1, ..., s_L)$$

Minimiziranjem kazne, broj početnih uzastopnih jedinica se maksimizira. Definirajmo dodatno τ kao prvu iteraciju u kojoj se pronalazi optimalno rješenje:

$$\tau := mint \geq 0 | X_t \cap S^* \neq \emptyset$$

pri čemu je $X_t := \{X_t^{(1)}, ..., X_t^{(N)}\}$ skup svih rješenja uzorkovanih u iteraciji t.

U poglavlju 3, teoremu 2 rada (**?**) se tvrdi da uz odabir konstantnog parametra izglađivanja $\varrho_t = \varrho$, veličine uzorka $N = L^{(2+\epsilon)}$, a $\epsilon > 0$ i $Nb = \lfloor (\beta N) \rfloor$ za $0 < \beta < \frac{1}{3e} \prod_{m=1}^{\infty} (1 - (1 - \varrho)^m)$. Uz početnu distribuciju $\prod_0(1, i) \equiv \frac{1}{2}$, odnosno jednoliku distribuciju, za prethodno definirani problem vodeće jedinice vrijedi $\mathbb{P}(\tau < L) \to 1$ kada $L \to \infty$.

Ovaj teorem je ekspermintalno provjeren sa sljedećim rezultatima:

Tablica 1. Utjecaj duljine uzorka L na broj iteracija (eps = 0.5, β = 0.09)

Iz tablice 1. je vidljivo da je broj iteracija potrebnih da rješenje konvengira u 1 manje od L za svaku testiranu duljinu rješenja. Prema dobivenim rezultatima se naslućuje da bi daljnjim rastom broja L teorem i dalje bio zadovoljen.

U svim navedenim testovima uvjet zaustavljanja je bio da vjerojatnost p da se na poziciji i uzorkuje jedinica bude veća od 0.999: p(1, i) > 0.999, za svaki i element 0,..., L-1

za svaku vrijednost je vršeno 30 testova

5. Zaključak

Zaključak.

Primjena konstruktivnog optimizacijskog algoritma na problem rasporeda studenata

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.