Problem Minimalnih K Centara

Jelena Živović, Nikola Janković

Matematički fakultet, Univerzitet u Beogradu

25. april 2020.

Sadržaj

Sadržaj

Sadržaj

Sadržaj

Uvod

Problem koji će biti razmatran je izbor najboljih lokacija za postavljanje objekata na nekom prostoru.

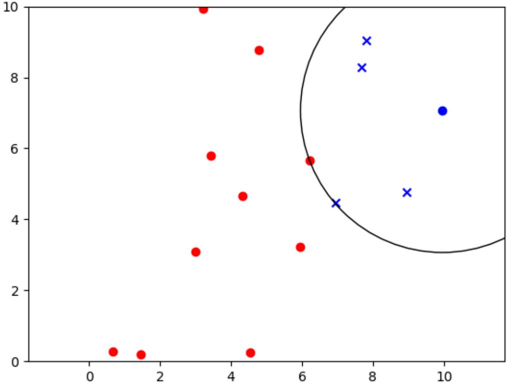
Definicija

Neka je $G = (V, E)$ kompletan, neusmeren graf sa težinama koje zadoljavaju nejednakost trougla. I neka je k pozitivan ceo broj ne veći od $|V|$. Za bilo koji skup $S \subseteq V$ i $v \in V$. Definišimo sad $d(v, S)$ kao težinu najkraće grane od čvora v do bilo kog čvora u S. Potrebno je pronaći takav skup $S \subseteq V$ da važi $|S| \leq k$, koji minimizuje izraz $max_{vinV} d(v, S)$.

2-aproksimativna heuristika

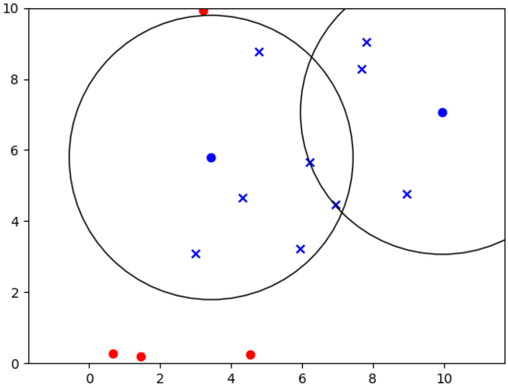
Ovo je pojednostavljena verzija aproksimativne heuristike za koju postoji dokaz da predstavlja 2-aproksimativnu heuristiku. P-aproksimativna heuristika zadovoljava svojstvo da rešenje koje ona nudi je u najgorem slučaju p puta lošije od optimalnog rešenja.

2-aproksimativni algoritam



Slika: Vizuelni prikaz 2-aproksimativne heuristike

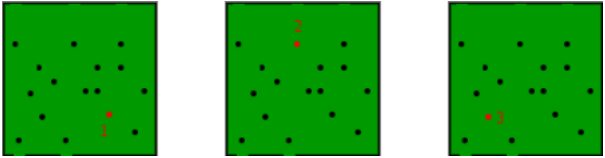
2-aproksimativni algoritam



Slika: Vizuelni prikz 2-aproksimativne heuristike

Gonzalesov algoritam

Drugi pristup, koji je takođe jedan od poznatijih, predstavljen je u radu meksičko-američkog naučnika Teofila Gonzalesa.



Slika: Primer Gonzalesove heuristike

Gonzalesov algoritam

Svodi se na biranje k čvorova iz grafa, tako što se prvi čvor izabere nasumično, a svaki i-ti čvor se bira tako da funkcija razdaljine $d(v_i) = min(w(v_i, v_1), w(v_i, v_2), ..., w(v_i, v_{i-1})$ bude maksimalna.

```
Ulaz: Graf G(V, E), k
Izlaz: Lista čvorova R, koja predstavlja reš
    enje

R[0] = nasumičan čvor u grafu
for i in range(1, k):
    najboljiCvor = -1
    najboljiRez = -1
    for j in range(|V|):
        if j nije u R:
            najbližiCvor = nadjiNajblizi(G,
    i)
            if najbližiCvor > najboljiRez:
                najboljiCvor = j
                najboljiRez = najbliziCvor

    R.append(najboljiCvor)

return R
```

Slika: Pseudokod Gonzalesovog alogritma

Pristup korišćenjem dominirajućeg skupa

Problem minimalnog dominirajućeg skupa podrazumeva nalaženje podskupa D, minimalne kardinalnosti, skupa čvorova grafa takvog da je svaki čvor, koji nije u D, susedan najmanje jednom čvoru iz D.

Pristup korišćenjem dominirajućeg skupa

```
Ulaz: graf G(V, E)
Izlaz: Dominirajući skup D
FOR v in V DO:
    CovCnt[v] := deg(v) + 1;
Score := CovCnt;
D := prazanSkup;

FOR i := 1 to |V| DO:
    x := čvor_sa_najmanjim_brojem_bodova;
    IF postoji y takvo da (x,y) pripada E i
    CovCnt[y]==1 THEN:
        dodaj x u D;
        FOR y : (x, y) pripada E DO:
            CovCnt[y] := 0;
    ELSE:
        FOR y takvo da (x,y) pripada E DO:
            IF CovCnt[y] > 0 THEN:
                CovCnt[y]--;
                Score[y]++;

    Score[x] = inf;
return D;
```

Slika: Pseudokod za nalaženje minimalnog dominirajućeg skupa grafa

Pristup korišćenjem dominirajućeg skupa

Algoritam za rešavanje problema k-centara koristi prethodno opisani algoritam i parametarsko potkresivanje.
Parametarsko potkresivanje podrazumeva eliminisanje onih grana grafa čija je težina veća od neke zadate vrednosti t.

Pristup korišćenjem dominirajućeg skupa

```
Ulaz: Graf G(V, E), k
Izlaz: Skup centara C

sortirati grane grafa prema težini
u neopadajućem poretku;
FOR e in E DO:
    GTmp := ParametricPrunning(G, e);
    C := prazan_skup;
    best_value = inf;
    IF GTmp povezan graf THEN:
        CTmp := DomSetAlgorithm(GTmp);
        IF |CTmp| <= k i evaluate(C) <
    best_value THEN:
            C := CTmp;
            best_value = evaluate(C);

return C;
```

Slika: Pseudokod algoritma za rešavanje problema k-centara koristeći minimalni dominirajući skup grafa

S-metaheuristike

Slovo S u nazivu ovih metaheuristika je skraćenica od reči singl, što znači da ove metaheuristike koriste jedno rešenje koje se popravlja kroz iteracije.

Simulirano kaljenje

Algoritam simuliranog kaljenja je zasnovan na procesu kaljenja čelika u cilju oplemenjivanja metala.
Glavni princip po kom radi algoritam simuliranog kaljenja je poboljšavanje vrednosti jednog rešenja.

Simulirano kaljenje

```
Ulaz: Graf G(V, E), k
Izlaz: suboptimalno rešenje

trenutno_rešenje = generiši_nasumično_reš
    enje()
trenutna_vrednost = odredi_vrednost(
    trenutno_rešenje)
najbolja_vrednost = trenutna_vrednost

for i in range(1, max_br_iter):
    novo_rešenje = izaberi_reš
    enje_u_okolini_trenutnog()
    nova_vrednost = odredi_vrednost(novo_reš
    enje)
    if nova_vrednost < trenutna_vrednost:
        trenutna_vrednost = nova_vrednost
    else:
        p = 1 / i ** 0,5
        q = nasumični_broj(0, 1)
        if p > q:
            trenutna_vrednost =
    nova_vrednost

    if nova_vrednost < najbolja_vrednost:
        najbolja_vrednost = nova_vrednost
return najbolja_vrednost
```

Slika: Pseudokod algoritma simuliranog kaljenja

Tabu pretraga

Glavni princip po kom radi ova metaheuristika zasniva se na pretrazi prostora izvan lokalnog optimuma. Komponenta koja omogućava ovako fleksibilno ponašanje je prilagodljiva memorija. Ova prilagodljivost se ogleda u mogućnosti da se prostor rešenja pregleda ekonomično i efikasno. Pseudokod ukazuje da bi u svakoj iteraciji trebalo izabrati nasumično rešenje iz skupa dozvoljenih.

```
Ulaz: funkcijaCilja
Izlaz: suboptimalno rešenje

s = generisiPocetnoResenje()
f_star = funkcijaCilja(s)
skupZabranjenih = set()
while not stoppingCriteria():
    p = izaberiRandom(nađiOkolinu(s).
     difference(T))
    if funkcijaCilja(p) < funkcijaCilja(s):
        s = p
    if funkcijaCilja(p) < f_star:
        f_star = funkcijaCilja(p)
    update(T)

return s
```

Slika: Pseudokod Tabu pretrage

Metoda promenljivih okolina

Metoda promenljivih okolina predstavlja uopštenje lokalne pretrage, gde su definisane okoline po kojima se vrši pretraživanje.
Postoji više varijanti osnovnog algoritma, među kojima se najčešće javljaju redukovana, osnovna i uopštena.

Metoda promenljivih okolina

```
Ulaz: Graf(G, V), maxEnv
Izlaz: Suboptimalno rešenje
trenutno_rešenje = generiši_nasumično_reš
    enje()
trenutna_vrednost = odredi_vrednost(
    trenutno_rešenje)
najbolja_vrednost = trenutna_vrednost

for i in range(maxIters):
    env = 0
    while env < maxEnv:
        novo_rešenje = izaberi_reš
    enje_u_okolini_trenutnog()
        nova_vrednost = odredi_vrednost(
    novo_rešenje)
        if nova_vrednost < trenutna_vrednost
    :
            trenutna_vrednost =
    nova_vrednost
            env = 0
        else:
            env += 1

        if nova_vrednost < najbolja_vrednost
    :
            najbolja_vrednost =
    nova_vrednost
return najbolja_vrednost
```

Slika: Pseudokod metoda promenljivih okolina

P-metaheuristike

Slovo P u imenu ove grupe metaheuristika označava populaciju. Dakle grupa ovakvih algoritama se u osnovi ne baziraju na jednom rešenju već na populaciji rešenja koja se kroz iteracije menja tako da sva pojedinačna rešenja zajedno konvergiraju ka nekom optimumu.

Genetski algoritam

Modifikacija osnovnog procesa se događa na nekoliko podmetoda. To su konkretno: **proces mutacije i proces ukrštanja**.

Proces **mutacije** se razlikuje jedino u opsegu stope mutacije koju je neophodno povećati u odnosu na standardnih $5 - 10\%$. Glavni razlog za tu promenu je činjenica da je $k \ll |V|$, pa jednostavna mutacija ne bi uspela da izvuče pretragu iz lokalnog minimuma.

Rešenje za problem **ukrštanja** je da pored indikatorske reprezentacije rešenja čuvamo i listu uključenih indikatora za svaku jedinku. Za korektnost procesa neophodno je iz izbora isključiti indekse koji se nalaze u obe jedinke.

```
Ulaz: jedinka1, jedinka2, za_m1, za_m2
Izlaz: ukrštene jedinke
for ind1, ind2 in zip(za_m1, za_m2):
    jedinka1[ind1] = 0
    jedinka2[ind1] = 1

    jedinka2[ind2] = 0
    jedinka1[ind2] = 1
```

Slika: Pseudokod genetskog algoritma

Scatter pretraga

Ova metaheuristika je formalizovana sedamdesetih godina prošlog veka. Prvi radovi datiraju jednu deceniju ranije. Osnovi koraci su:

- Generisanje različitih početnih rešenja (eng. *A Diversification Generation Method*)
- Unapređivanje trenutnih rešenja - Od jednog rešenja je moguće dobiti jedan ili više unapređenih
- Ažuriranje uskog skupa najboljih rešenja
- Kreirati podskupove skupa najboljih rešenja koji će biti korišćen za kreiranje novih rešenja.
- Rekombinovati rešenja iz kreiranih podskupova i kreirati nova.

Scatter pretraga

```
diversificationGenerationMethod()
improveMethod()
while not stoppingCriteria():
    azuriajRefSkup()
    while postoji novo rešenje u refSkupu:
        generisiPodskupove()
        rekombinujRešenja()
        improveMethod()
        azurirajRefSkup()


return najbolji iz refSkupa
```

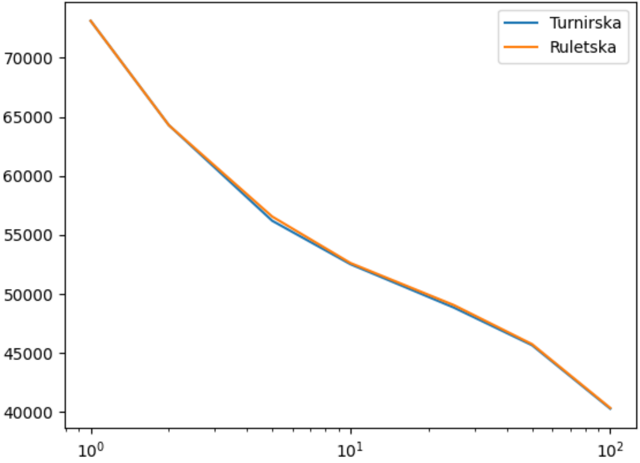
Slika: Pseudokod Scatter pretrage

Rezultati istraživanja

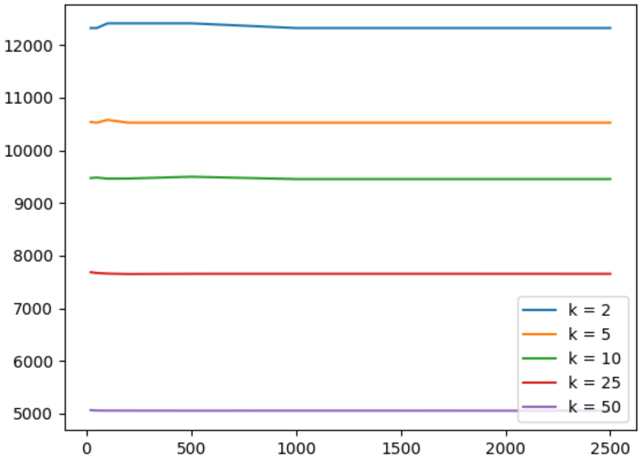
Pri generisanju grafa korišćeno je vremenski efikasno generisanje, složenosti $O(|E|)$, ispred velike varijabilnosti težina grana. Naime, pristup koji je korišćen da sve težine budu uniformno birane iz poluotvorenog intervala $[A, 2A)$ gde je A proizvoljan element skupa R^+. Ovakav pristup daje značajno manju raspršenost vrednosti od metoda generisanja nasumičnih tačaka u $2D$ ravni i računanjem njihovih međusobnih rastojanja, ali je vremenska složenost neuporedivo manja.

pop_vel \ k	2	10	25	50
100	(12474.32, 0.19)	(9465.88, 0.31)	(7643.91, 0.58)	(5043.19, 1.05)
200	(12495.00, 0.40)	(9499.61, 0.64)	(7636.24, 1.19)	(5043.60, 2.16)
500	(12428.25, 1.03)	(9482.85, 1.66)	(7640.65, 3.11)	(5042.89, 5.38)
1000	(12428.25, 2.07)	(9439.88, 3.24)	(7643.19, 6.18)	(5041.06, 10.93)
2500	(12428.25, 5.37)	(9436.71, 8.42)	(7631 15.77)	(5040.10, 27.3)

Tabela: Zavisnost rezultata i potrebnog vremena (rezultat, vreme) od veličine populacije, kod genetskog algoritma

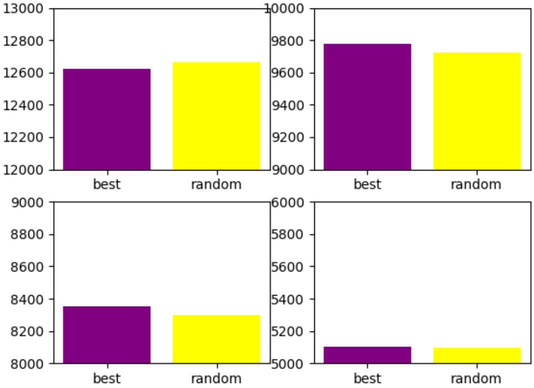


Slika: Kvalitet rezultata od načina selekcije



Slika: Kvalitet rezultata od broja iteracija

Tabu pretraga



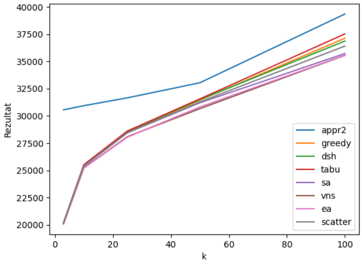
Slika: Analiza Tabu pretrage

Završno istraživanje

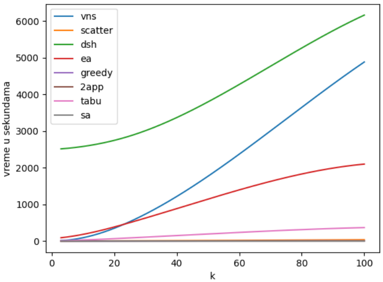
vrsta	rezultat
BruteForce	2191.7802
2-approximation	2727.7366
Greedy	2329.4667
Scatter	2191.7802
Simulated annealing	2192.4801
Dominating set	2401.2494
Variable search neighbourhood	2191.7802
Tabu	2341.0661
EA	2191.7802

Tabela: Pilot testiranje za $n = 20$ i $k = 2$

Završno istraživanje

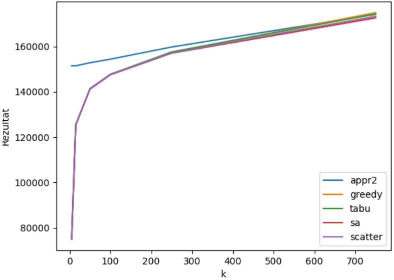


Slika: Kvalitet rezultata za $n = 300$

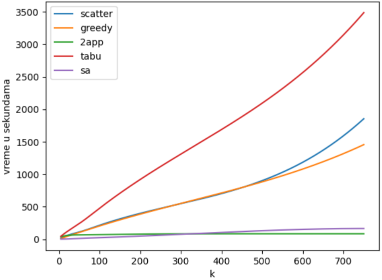


Slika: Vremenska zahtevnost za $n = 300$

Završno istraživanje



Slika: Kvalitet rezultata za $n = 1500$



Slika: Vremenska zahtevnost za $n = 1500$

Zaključak

Ni za jedan algoritam nije dokazano da daje optimalno rešenje, ali neki algoritmi daju prilično dobro rešenje. Za manje ulaze nije preterano bitno koji pristup se koristi svi će dati dovoljno dobro rešenje, za ulaze iz srednjeg raspona neke pristupe bi trebalo izbeći zbog ogromnog vremenskog utroška.
Za najveće ulaze, najbolje je iskoristiti pristup Simuliranog kaljenja ili 2-aproksimativnu heuristiku. Zbog manje vremenske zahtevnosti dalje istraživanje bi trebalo okrenuti ka unapređenju parametara Simuliranog kaljenja kao i istraživanje na sličnim heuristikama iz skupa S.

Pitanja?