Razapinjuća stabla: NP-teške varijante i metaheuristike

Primer na rešavanju problema DCkMST

RAF naučna tribina

Milan Tomić

05.12.2025.

Računarski fakultet

Sadržaj

1 Uvodni pojmovi

- Uvodni pojmovi
- Pristup
- Rezultati
- Diskusija
- Reference

Razapinjuće stablo

1 Uvodni pojmovi

- Neka je $G = (V, E)$ prost neusmeren težinski graf, pri čemu je V skup tačaka/čvorova i E skup ivica/grana, pri čemu svaka ivica $e \in E$ ima pridruženu pozitivnu težinu w_e
- Razapinjuće stablo $T = (V_T, E_T)$ grafa G je bilo koje stablo takvo da $V_T = V$ i $E_T \subseteq E$
- Neka je $cost(T) = \sum_{e \in E_T} w_e$ ukupna suma težina svih grana u T
- Neka je $\mathcal{T}_G$ skup svih razapinjućih stabla grafa G
- Minimalno razapinjuće stablo (*MST – minimum spanning tree*) grafa G je:

$$\arg\min_{T \in \mathcal{T}_G} cost(T)$$

tj. ono čiji je zbir težina grana najmanji

Metaheuristika

Svaka *čarobna* metoda za pronalaženje optimalnog rešenja problema čiji su prostori rešenja:

- beskonačni (ili ekstremno veliki)
- nedovoljno poznatih karakteristika
- suviše kompleksni za analitičko rešavanje
- neizračunljivi tradicionalnim metodama pretraživanja zbog nedovoljno resursa

Obično u sebi sadrži jednu ili heurističkih ili lokalnih metoda pretraživanja i kombinuje ih u cilju:

- Identifikacije dobrih podskupova rešenja / okolina (istraživanje)
- Pronalaženja (lokalno) optimalnog rešenja (eksploatacija)

Problemi iz klase P mogu se rešiti na *determinističkoj* Tjuringovoj mašini u polinomijalnom vremenu.
Uprošćeno, ako je veličina instance problema $O(n)$, rešenje ovakvog problema na običnom računaru zahteva $O(P(n))$ instrukcija, gde je $P(n)$ neki polinom po promenljivoj n.
Problemi iz klase NP mogu se rešiti na *nedeterminističkoj* Tjuringovoj mašini u polinomijalnom vremenu.
Uprošćeno, ako običan računar prilikom svakog koraka u izvršavanju programa može odabrati jednu od više opcija i ume unapred da prepozna i bira idealno, onda može da reši takav problem veličine $O(n)$ pomoću $O(P(n))$ instrukcija.

P vs NP

Treba obratiti pažnju da je $P \subset NP$ (ali nije dokazano $P = NP$).
Mogućnost izbora idealne instrukcije koju ima nedeterministička Tjuringova mašina svodi sve probleme za čije rešavanje su poznati algoritmi eksponencijalne složenosti na polinomijalne.
Zbog kompleksnosti pojedinih NP problema, za njihovo rešavanje se često pribegava korišćenju metaheuristika.

Problem MST

- Problem minimalnog razapinjućeg stabla pripada klasi P. Dva poznata algoritma koja ih rešavaju jesu Primov i Kruskalov algoritam. Oba su bazirana na postepenoj izgradnji stabla uz održavanja invarijante da se uvek dodaje grana najmanje težine.
- Primov algoritam održava rezultujuće stablo i uvek dodaje sledeću ivicu najmanje težine koja je incidentna nekom od listova stabla, a čiji je drugi kraj neki od čvorova koji već nisu u stablu.
- Kruskalov algoritam dodaje u rezultat — šumu — najkraću od svih ivica u grafu tako da se ne formira ciklus (tj. da povezuje neka dva stabla iz šume). Rezultujući graf je razapinjuće stablo.

Problem DCMST

1 Uvodni pojmovi

- *Degree-constrained minimum spanning tree (DCMST)* – problem minimalnog razapinjućeg stabla sa ograničenjem stepena čvorova.
- Za razliku od MST, DCMST u postavci sadrži dodatno ograničenje, da stepen svakog čvora u pronađenom rešenju nije veći od zadate granice d.
- Za $d = 2$ ovaj problem se svodi traženje minimalnog Hamiltonovog puta, koji je *NP*-težak.
- Pod određenim okolnostima postoje polinomijalna rešenja za DCMST [1]; za $d > 3$ često je dovoljan Primov algoritam uz male modifikacije.

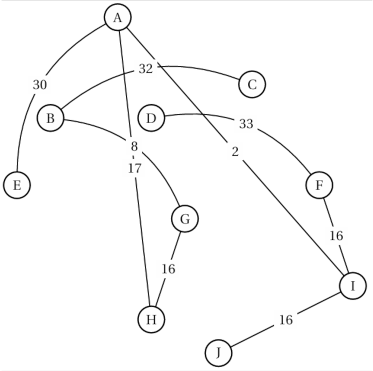
Problem DCkMST

1 Uvodni pojmovi

- *Degree-constrained k-minimum spanning tree (DCkMST)* – problem minimalnog razapinjućeg stabla sa ograničenjem stepena čvorova koje sadrži najmanje k čvorova.
- Pored ograničenja stepena čvora od najviše d, u ovom problemu rešenje treba da zadovoljava i uslov da broj čvorova koji stablo pokriva nije manji od k, pri čemu je $k \leq |V|$ (u daljem tekstu $|V| = n$.
- Dakle, u ovom problemu je potrebno odabrati najmanje k čvorova, tako da rešenje odgovarajućeg DCMST bude minimalno.
- Ako je $w_e > 0$ za svako $e \in E$, lako se može dokazati da je broj čvorova u takvom stablu sigurno k, tj. nije veći.

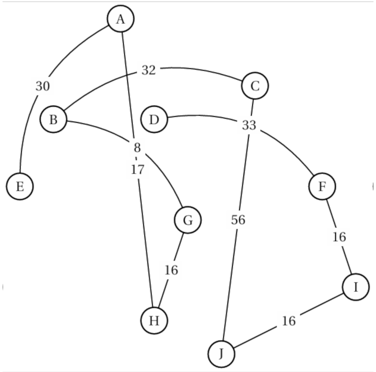
Poređenje rešenja na istom grafu

1 Uvodni pojmovi



Rešenje MST na kompletnom grafu

$n = 10, cost = 170$

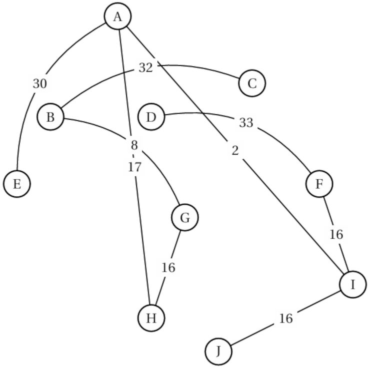


Rešenje DCMST sa $d = 2$ na kompletnom grafu

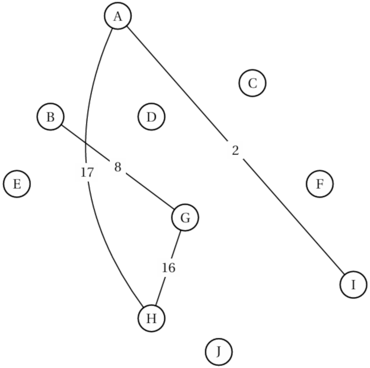
$n = 10, cost = 224$

Poređenje rešenja na istom grafu

1 Uvodni pojmovi



Rešenje MST na kompletnom grafu
$n = 10, cost = 170$



Rešenje DCkMST sa $d = 2$ i $k = 5$ na kompletnom grafu $n = 10, cost = 43$

Sadržaj

2 Pristup

▶ Uvodni pojmovi

▶ Pristup

▶ Rezultati

▶ Diskusija

▶ Reference

Pristup „manje je više” - *LIMA*

Princip rada metaheuristika je da kombinuju neke jednostavne lokalne metode u pretraživanju rešenja.

Primetna je težnja da se u hibridizaciji preteruje, pa su takvi algoritmi veoma komplikovani, a vrlo često ne donose željene rezultate, ili vrlo malo doprinose poboljšanju prethodno poznatih rešenja. Broj korišćenih heuristika ili lokalnih metoda dostiže i nekoliko desetina.

Nasuprot tome, pristup „manje je više” (*”Less is more”approach – LIMA* [2], [3]) u osnovi ima istraživanje svojstava problema i nalaženje tačno onih sastojaka algoritma koji će efikasno dovesti do optimalnog rešenja.

Metoda promenljivih okolina - *VNS* [4], [5]

Osnovna ideja: odrediti nekoliko okolina rešenja koje će se ispitivati. Kreće se od jednog izvodljivog rešenja, koje se može naći proizvoljnom metodom. Do okončanja algoritma:

VND Za svaku unapred zadatu okolinu trenutnog rešenja:

- — Proveriti da li postoji bolje rešenje u toj okolini
- — Ako postoji, prihvatiti ga kao novo i početi ispočetka
- — Ako ne postoji, preći na sledeću okolinu

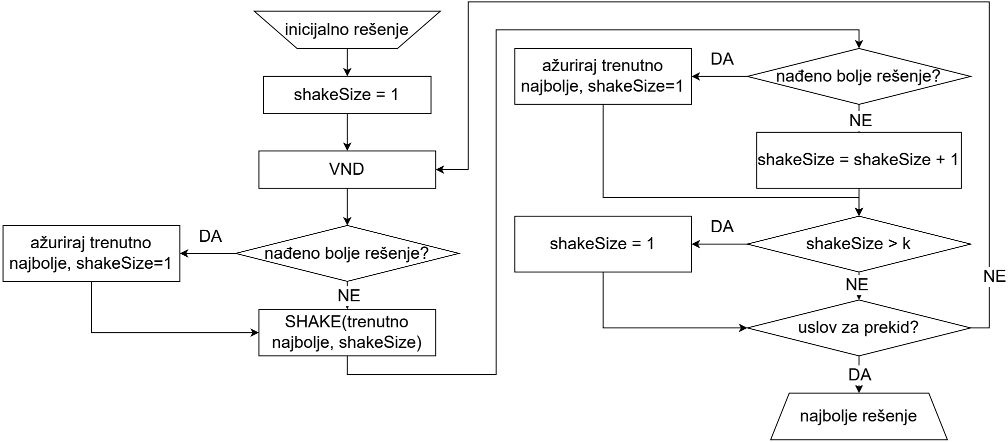
- Ažurirati trenutno najbolje rešenje (ako je primenljivo)

Shake „Protresti” trenutno najbolje rešenje, tako da se dođe do novog izvodljivog rešenja čija će se okolina pretraživati.

- Ažurirati trenutno najbolje rešenje (ako je primenljivo)

VNS - tok algoritma

2 Pristup



VNS za DCkMST – ideje za inicijalno rešenje

- Lako je doći do minimalnog (ili bilo kog) razapinjućeg stabla koje nema nikakva ograničenja (Prim/Kruskal)
- Možemo deformisati takvo stablo tako da dobijemo stablo koje ispunjava ograničenja stepena čvorova zadata preko d
- U varijanti sa $k < n$ možemo odabrati proizvoljnih k čvorova i nad njima postupiti isto
- Takođe, možemo napraviti stablo nad n čvorova i postepeno skidati jedan po jedan čvor dok ne dođemo do k

VNS za DCkMST – inicijalno rešenje

1. Konstruiše se MST pomoću Primovog algoritma
2. Lista grana koje su u MST se promeša (tako da budu u nasumičnom redosledu)
3. Prolazi se kroz listu grana MST i svaka grana čiji krajevi imaju stepen veći od d se uklanja (slučajnim redosledom) – prave se odvojene povezane komponente, pri čemu je svaka stablo sa maksimalnim stepenom čvora d
4. Sve ivice grafa se sortiraju od najmanje do najveće
5. Prolazi se kroz sve ivice (sortirano) i ukoliko neka od njih spaja dve različite komponente, pri čemu njeni krajevi imaju stepen manji od d, onda se ona dodaje u rešenje (spajaju se dve komponente)
6. Čim se nađe komponenta koja ima najmanje k čvorova, prekida se petlja i iz ove komponente se redom uklanjaju listovi čije grane su najduže (kako bi se svela na k čvorova) i ona postaje inicijalno rešenje

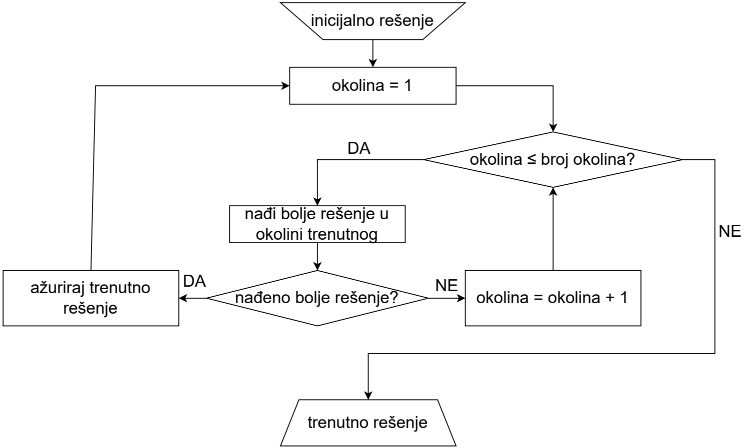
VND za DCkMST – promenljive okoline

Okolina rešenja se najčešće definiše pomoću nekog operatora koji unosi neku malu promenu u trenutno posmatrano rešenje.
Primenom tog operatora na sve moguće načine, mogu se dobiti rešenja slična trenutnom, pri čemu ta mala promena može dovesti do boljeg ili lošijeg rezultata.
Prihvatanje boljeg nađenog rešenja iz posmatrane okoline obično se izvodi na jedan od dva načina:

- Prihvatanje prvog boljeg (*first improvement*)
- Prihvatanje najboljeg (*best improvement*)

VND - tok algoritma

2 Pristup



VND za DCkMST – promenljive okoline

Prilikom rada analizirano je nekoliko mogućih okolina, a u konačan izbor ušle su tri, generisane operatorima:

1. Modifikovani 2-opt (*Modified 2-opt*)
2. Zamena grane uvođenjem ciklusa (*Cycle edge swap*)
3. Zamena jednog lista (*One leaf swap*)

Modifikovani 2-opt

Ideja: ako postoje dve grane (u_1, v_1) i (u_2, v_2) u stablu takve da bi njihova zamena granama (u_1, u_2) i (v_1, v_2) (ili (u_1, v_2) i (u_2, v_1)) smanjilo težinu stabla, pri čemu su svi pomenuti čvorovi različiti, onda takva promena daje bolje izvodljivo rešenje.
Stepeni čvorova se ne menjaju, jer se kod sva četiri čvora jedna incidentna grana menja drugom.
Za proveru koji čvorovi smeju da se povezuju da ne bi došlo do formiranja ciklusa korišćeno je ukorenjeno stablo (tj. primenjen je BFS na stablo sa proizvoljnim korenom).

Modifikovani 2-opt

Analiza složenosti:

- Broj svih rešenja koja se ovako generišu je $O(k^2)$.
- Složenost BFS na stablu je $O(k)$.
- Složenost provere koji čvorovi se povezuju je $O(k)$.
- Ukupna asimptotska složenost jedne pretrage po okolini je $O(k^2)$.

Prilikom ispitivanja prihvatano je prvo bolje rešenje.

Zamena grane uvođenjem ciklusa

Ideja: dodavanjem grane između bilo koja dva čvora u stablu formira se ciklus, a izbacivanjem jedne grane iz ciklusa ponovo stablo. Ako se neka dva čvora povežu i iz tako formiranog ciklusa se izbaci najveća grana, dobija se potencijalno bolje izvodljivo rešenje. Ne smeju se povezivati čvorovi koji imaju stepen d, jer bi po izbacivanju jedne grane najmanje jedan od njih bio stepena $d + 1$.
Metoda takođe koristi BFS ukorenjeno stablo za efikasnije pronalaženje ciklusa, prilikom čega se povezuju čvorovi koji nisu koren stabla. BFS se zato pokreće dva do tri puta po čvoru, tj. ukupno $O(k)$ puta.

Analiza složenosti:

- Broj generisanih rešenja je $O(k^2)$, a ukupan broj pokretanja BFS za sva ta rešenja $O(k)$.
- Složenost BFS na stablu je $O(k)$.
- Složenost traženja najveće grane u ciklusu je $O(k)$.
- Ukupna asimptotska složenost jedne pretrage po okolini je $O(k^3)$.

Prilikom ispitivanja prihvatano je najbolje rešenje.

Zamena jednog lista

2 Pristup

Ideja: izbacivanjem jednog lista (sa njegovom incidentnom granom) iz stabla, dobija se stablo sa $k-1$ čvorova. Bilo koji čvor koji potom nije u stablu može da se doda kao list povezivanjem sa bilo kojim drugim čvorom u stablu koji je stepena manjeg od d. Ako je dodata grana manja od izbačene, dobija se bolje izvodljivo rešenje.
Metoda koristi samo pomoćni niz stepena čvorova za proveru koji čvorovi smeju da se povezuju, što je vrlo efikasno.

Zamena jednog lista

Analiza složenosti:

- Broj generisanih rešenja je $O\left(k(n-k)\right)$.
- Složenost svake provere je $O(1)$.
- Ukupna asimptotska složenost jedne pretrage po okolini je $O\left(k(n-k)\right)$.

Prilikom ispitivanja prihvatano je najbolje rešenje.

VNS za DCkMST – Protresanje

- Prilikom obilaska svake okoline trenutnog rešenja, ako se nađe bolje rešenje, ono se odmah prihvata kao novo trenutno rešenje.
- Ako je trenutno rešenje ažurirano, okoline koje se od njega obilaze kreću od početka (tj. od 2-opt pa do zamene jednog lista), bez obzira na to u kojoj okolini prethodnog rešenja je nađeno bolje.
- Kada nakon obilaska poslednje okoline trenutnog rešenja nema boljeg rešenja, prelazi se na *shaking*.
- Uzima se najbolje poznato rešenje i primenjuje se niz operacija koje od njega prave novo izvodljivo rešenje. Tako dobijeno rešenje postaje novo trenutno.
- Ako je na taj način slučajno nađeno rešenje bolje od najboljeg ikada, ono se ažurira.

VNS za DCkMST – Protresanje

- Jačina protresanja, koja utiče na to koliko daleko može da se ode od protresanog rešenja, kontroliše brojčani parametar *shakeSize*
- Parametar *shakeSize* se menja od 1 do k na sledeći način:
  - Posle svakog shake-a povećava se za 1;
  - Ako prekorači k, vraća se na 1;
  - Ako se posle shake-a dobije novo najbolje rešenje, vraća se na 1.

VNS za DCkMST – Protresanje

1. Kreiraju se komponente: najbolje rešenje je jedna komponenta, svi preostali čvorovi van njega su zasebne komponente.
2. Iz najboljeg rešenja nasumično se uklanja *shakeSize* grana, što dovodi do razbijanja rešenja na *shakeSize+1* komponenata.
3. Dodavanjem nasumično izabranih grana (kao kod inicijalnog rešenja) komponente se spajaju održavajući maksimalan stepen čvorova d dok se ne dobije komponenta od najmanje k čvorova. Za nasumičnost se primenjuje GRASP, tj. prioritet imaju kraće grane, ali se neke od njih nasumično preskaču. Verovatnoća preskakanja grane zavisi od *shakeSize* i empirijski izabranih parametara, a može da zavisi i od k i d.
4. Kao kod inicijalnog rešenja, uklanjaju se listovi sa najdužim incidentnim granama dok se ne dođe do stabla sa k čvorova.

VNS za DCkMST – Protresanje

Analiza složenosti:

- Kreiranje komponenata i razbijanje rešenja na komponente agregirano je u $O(k)$ operacija.
- Spajanje se izvodi tako što se prolazi kroz sve ivice grafa i koriste one čiji su krajevi u različitim komponenatama. Svaka operacija spajanja ima do $O(k)$ operacija.
- Ukupan broj ivica koje se ispituju je $O(n^2)$.
- Ukupna procenjena složenost je $O(kn^2)$.

VNS za DCkMST – Odabir parametara

Parametri za GRASP kod spajanja komponenata su odabirani empirijski, posmatranjem rasta vrednosti funkcije verovatnoće za različite vrednosti k kako raste $shakeSize$ i ispitivanjem uspešnosti procedure na malim instancama.
Konačno odabrana verovatnoća za preskakanje proizvoljne grane prilikom testiranja računa se na sledeći način:

$$p = \min\left(0.85, \frac{3.5 \cdot shakeSize}{100}\right)$$

Sadržaj

3 Rezultati

- ▶ Uvodni pojmovi
- ▶ Pristup
- ▶ Rezultati
- ▶ Diskusija
- ▶ Reference

VNS za DCkMST – Odabir instanci za testiranje

Za testiranje su korišćene poznate DCMST instance iz literature, koje uključuju set posebno generisanih instanci za probleme stabala i neke instance iz TSPLIB.

Testiranje je odrađeno na ukupno 444 različite instance problema, kombinujući grafove sa različitim vrednostima k i d.

Za mnoge instance nađeno je optimalno rešenje korišćenjem LP modela na Gurobi optimizatoru sa maksimalnim ograničenjem vremena 1h, dok za nekoliko instanci nije dokazana optimalnost ili nije uspelo izvršavanje zbog veličine instance.

Rezultati dobijeni VNS-om su upoređeni sa postojećim rezultatima ranije objavljenih metoda iz literature i dokazanim optimalnim vrednostima pomoću Gurobija.

VNS za DCkMST – Proces testiranja

Metoda je implementirana tako da se paralelno izvršava na 10 niti sa maksimalnim zadatim vremenom 300 sekundi (ukupno 50 minuta računarskog vremena po instanci) i obaveznim zaustavljanjem ako dostigne 10^6 iteracija.

Praćeno je vreme do pronalaska poznatog optimalnog rešenja, te se izvršavanje prekidalo ranije ako do toga dođe.

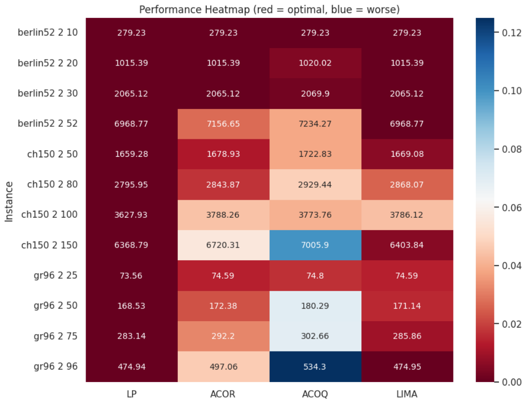
Ispraćena su sva vremena do pronalaženja konačnih najboljih rešenja i izračunati su bitni statistički pokazatelji (minimum, medijana, mod, prosek, standardna devijacija) vremena i troška rešenja za 10 izvršavanja na svakoj instanci, čime su dobijene vrednosti za poređenje sa drugim metodama.

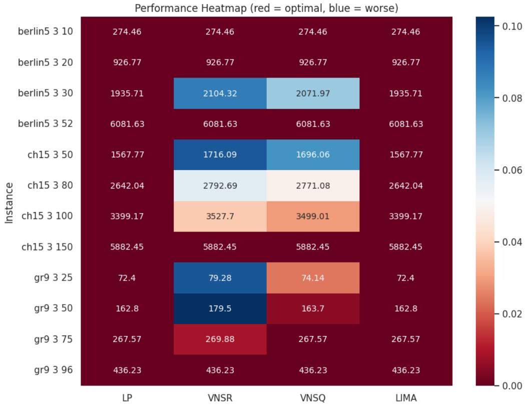
TSPLIB

Na TSPLIB instancama, poređenje je napravljeno sa algoritmima ACOR i ACOQ za $d = 2$ i VNSR i VNSQ za $d = 3$. Ovi pristupi su izloženi u [6]. Od sva četiri algoritma, naš VNS je bio uspešniji u 50% testiranih slučajeva. Pokazao se kao najbolje heurističko rešenje u 10 od 12 instanci i bio optimalan u svim instancama sa $d = 3$.

TSPLIB - poređenje sa [6]

3 Rezultati



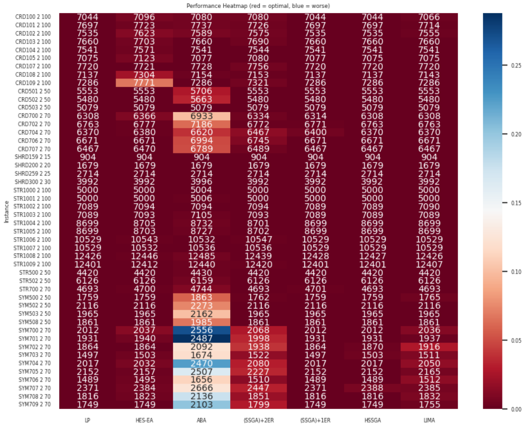


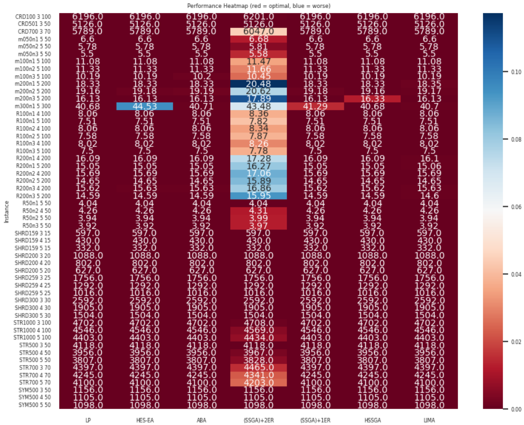
DCMST instance

Na DCMST instancama, poređenje je napravljeno sa ranijim algoritmima konstruisanim za DCMST, odnosno sa varijabilnim d i fiksnim $k = n$. U ovim slučajevima nije se uvek pokazao kao najbolje heurističko rešenje, ali su odstupanja uglavnom reda veličine 0.1%. Bitna razlika je što su ti algoritmi konstruisani specifično za instance $k = n$, pa je naša prednost u tome što je algoritam moćniji, a ne uključuje više raznorodnih komponenata („nema besplatnog ručka”).
Takođe se vide razlike u uspešnosti za instance $d = 2$ u odnosu na $d > 2$, pri čemu se može reći da LIMA daje najkonzistentnije rezultate srazmerno težini problema.

DCMST - poređenje sa [7]

3 Rezultati





Sadržaj

4 Diskusija

- Uvodni pojmovi
- Pristup
- Rezultati
- Diskusija
- Reference

Rezime

4 Diskusija

- Iz prethodnih rezultata vidi se da je LIMA praktično dominantan u odnosu na sve ostale kada su u pitanju instance $d > 2$.
- U poređenju sa LP na svim instancama zajedno (ukupno 444, od čega je 125 ranije izloženo u literaturi), u 71% slučajeva dostiže isto ili bolje[1] rešenje.
- U preko 80% slučajeva dostiže rešenje u intervalu $\leq 1\%$ od optimuma, a u preko 96% slučajeva u $\leq 5\%$.
- Efikasno nalazi rešenja i za instance sa $n \geq 500$, što u ranijoj literaturi nije testirano, a sa LP nije uspešno testirano jer je model preveliki.

[1] ako LP nije dokazao optimalnost u roku od 3600s

Mogućnosti za unapređenja

- Nije u potpunosti jasna efikasnost odabrane metode za *shaking* kod instanci različitih veličina.
- Moguće je da bi zamena ili dodavanje neke okoline poboljšala rezultate za instance $d = 2$ bez gubljenja efikasnosti za ostale.
- Pristup se može primeniti za druge varijante MST problema, npr. *min-degree constrained minimum spanning tree problem* [8] ili *spanning tree with conflicting edge pairs* [9], uz odgovarajuće modifikacije.

Mogućnosti za dalju primenu

- LIMA pristup se mnogo puta pokazao boljim u optimizaciji u odnosu na razne složene hibridizacije.
- Metoda promenljivih okolina ima prednost adaptiranja odnosa istraživanja i eksploatacije u toku rada, u zavisnosti od toga koliko su neke okoline povoljne ili nepovoljne, što je čini robusnom metaheuristikom za razne tipove problema.
- Pažljivom analizom problema i ograničenja koja implicitno važe može se doći do metoda za lokalno pretraživanje koje imaju smisla, a njihovim uklapanjem do upotrebljive metaheuristike.
- Bez te komponente primena bilo koje metode je samo nagađanje i pucanj u prazno.

Sadržaj

5 Reference

▶ Uvodni pojmovi

▶ Pristup

▶ Rezultati

▶ Diskusija

▶ Reference

Reference

6 Reference

[1] S. Arora, "Polynomial time approximation schemes for Euclidean TSP and other geometric problems", u *Proceedings of 37th Conference on Foundations of Computer Science*, IEEE, 1996., str. 2–11.

[2] N. Mladenović, R. Todosijević i D. Urošević, "Less is more: basic variable neighborhood search for minimum differential dispersion problem", *Information Sciences*, sv. 326, str. 160–171, 2016.

[3] N. Mladenović, Z. Drezner, J. Brimberg i D. Urošević, "Less Is More Approach in Heuristic Optimization", u *The Palgrave Handbook of Operations Research*. Springer International Publishing, 2022., str. 469–499.

[4] N. Mladenović i P. Hansen, "Variable neighborhood search", *Computers & operations research*, sv. 24, br. 11, str. 1097–1100, 1997.

Reference

6 Reference

[5] P. Hansen i N. Mladenović, “Variable neighborhood search: Principles and applications”, *European journal of operational research*, sv. 130, br. 3, str. 449–467, 2001.

[6] P. Adasme i A. Dehghan Firoozabadi, “Degree-Constrained k-Minimum Spanning Tree Problem”, *Complexity*, sv. 2020, br. 1, str. 7 628 105, 2020.

[7] K. Singh i S. Sundar, “A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem”, *Soft Computing*, sv. 24, br. 3, str. 2169–2186, 2020.

[8] A. M. de Almeida, P. Martins i M. C. de Souza, “Min-degree constrained minimum spanning tree problem: complexity, properties, and formulations”, *International Transactions in Operational Research*, sv. 19, br. 3, str. 323–352, 2012.

[9] F. Carrabs, R. Cerulli, R. Pentangelo i A. Raiconi, “Minimum spanning tree with conflicting edge pairs: a branch-and-cut approach”, *Annals of Operations Research*, sv. 298, br. 1, str. 65–78, 2021.

Razapinjuća stabla: NP-teške varijante i metaheuristike

Hvala na pažnji!
Pitanja?