Algoritmi za optimizaciju stabla sa maksimalnim brojem listova

Veljko Strugar
399/2021

Abstract

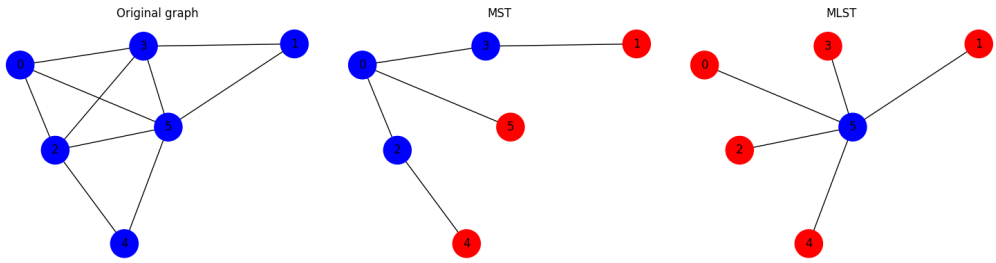
MLST problem je optimizacioni problem u teoriji grafova, gde je za zadati povezan graf, cilj konstruisati razapinjuće stablo sa najvećim brojem listova.

Ovaj rad istražuje računarsku složenost problema, predstavlja različite pristupe optimizaciji uključujući tačne algoritme, aproksimacione algoritme i heuristike, te analizira njihove performanse na različitim instancama grafova (tabela 1).

Contents

1 Uvod

Problem MLST, definisan za povezani graf $G = (V, E)$, traži stablo koje maksimizira broj listova (slika 1). Problem ima primene u dizajnu mreža, rasporedu kola i komunikacionim sistemima. Uprkos praktičnom značaju, problem je **APX-kompletan** i ostaje računski izazovan. Ovaj rad istražuje rešenja od pohlepnih algoritama do metoda metaheuristike.



slika 1: Primer grafa i MST i MLST kreiranih nad njim

1.1 Formulacija Problema

- **Instanca:** Povezani neusmereni graf $G = (V, E)$.
- **Rešenje:** Stablo $T \subseteq G$.
- **Mera:** Broj listova u T, označen sa $|L(T)|$.
- **Složenost:** Problem je APX-kompletan i može se aproksimirati faktorom 3.

  Detaljan opis problema dat je na sajtu [2]

2 Egzaktne metode

Egzaktne metode pružaju optimalna rešenja, ali su računski intenzivne za velike instance.

2.1 Algoritam grube sile

Vremenska složenost: S obzirom na to da po Košijevoj formuli graf može imati i do $n^{(n-2)}$) razapinjućih stabala, ukupna složenost algoritma je $O(n \cdot n^{(n-2)})$, gde je $n = |V|$.

Algorithm 1 Algoritam grube sile za MLSTP

1: Inicijalizuj promenljivu T^* kao prazno stablo.
2: Inicijalizuj promenljivu $L^* = 0$ za broj listova najboljeg stabla.
3: **for** sva moguća stabla T koja spajaju sve čvorove **do**
4: Izračunaj broj listova $L(T)$.
5: **if** $L(T) > L^*$ **then**
6: Ažuriraj $T^* = T$ i $L^* = L(T)$.
7: **end if**
8: **end for**
9: **return** T^*

2.2 Branch and Bound

Metoda istražuje podskupove grana, koristeći ograničenja za eliminaciju suboptimalnih rešenja.

Algorithm 2 Branch and Bound za MLST

1: Inicijalizuj najbolje rešenje T^* sa $|L(T^*)| = 0$.
2: Dodaj početni čvor na stek.
3: **while** stek nije prazan **do**
4: Skini čvor N sa steka.
5: Izračunaj donju $LB(N)$ i gornju $UB(N)$ granicu.
6: **if** $UB(N) > |L(T^*)|$ **then**
7: Proširi N dodavanjem novih čvorova (uključivanje/isključivanje grana).
8: Ažuriraj T^* ako je pronadjeno bolje rešenje.
9: **end if**
10: **end while**
11: **return** T^*

Detaljnije razjašnjenje algoritma u priloženoj implementaciji algoritma [1].

3 Metaheuristički algoritmi

Metaheuristike pružaju okvir za rešavanje MLST kombinovanjem heurističkih pravila sa globalnim strategijama pretrage. Često su inspirisane prirodnim procesima.

3.1 Genetski algoritmi (GA)

GA imitira prirodnu selekciju, evoluirajući populaciju mogućih rešenja kroz ukrštanje, mutaciju i selekciju.

Algorithm 3 Genetski algoritam za MLST

```
Inicijalizuj populaciju stabala P.
for svaka generacija do
    Procena kvaliteta svakog pojedinca u P.
    Izaberi roditelje na osnovu kvaliteta.
    Primeni ukrštanje za generisanje potomaka.
    Primeni mutaciju za povećanje raznovrsnosti.
    Ažuriraj P sa potomcima i elitnim jedinkama.
end for
return Najbolja jedinka u P
```

Inicijalizacija: Kako bi se sprečila preuranjena konvergencija, inicijalna populacija mora biti raznolika. U tu svrhu pri kreiranju inicijalne populacije ubacujemo i težine u igru, koje se pri kreiranju svake nove jedinke menjaju, pa MST algoritam za kreiranje nove jedinke ima veće šanse da kreira jedinku koje za sada nema u populaciji.

Jedinka:

- *Hromozom*: Lista grana.
- *Fitnes*: Broj listova u drvetu.

Selekcija: Turnirska

Ukrštanje: Pokreće se algoritam za pravljenje mst-a nad unijom grana oba roditelja što rezultuje jednim detetom

Mutacija: Kreira se jedinka od svih grana polaznog grafa bez jedne proizvoljne grane iz trenutne jedinke

Vremenska složenost algoritma je $O(I * P * MlogN)$, gde su: I broj iteracija, P veličina populacije, M broj grana i N broj čvorova.

3.2 Simulirano kaljenje

Svaka novokreirana jedinka nakon ukrštanja i mutacije prodje i proces simuliranog kaljenja. Ovaj proces optimizuje strukturu jedinke (stabla) tako što iterativno uklanja ivice i rekonstruiše minimalno razapinjuće stablo. Ako novo stablo poboljša fitnes (broj listova), prihvata se odmah; inače, prihvata se s odreenom verovatnoćom koja opada tokom iteracija. Na kraju, vraća najbolju pronaenu strukturu.

Vremenska složenost algoritma je $O(I_{GA} * P * I_{SA} * MlogN)$, gde su: I_{GA} broj iteracija genetskog algoritma, I_{SA} broj iteracija simuliranog kaljenja, P veličina populacije, M broj grana i N broj čvorova.

3.3 Optimizacija kolonijom mrava (ACO)

ACO simulira ponašanje mrava u potrazi za hranom, gde veštački mravi iterativno grade stabla vodjeni feromonskim tragovima i heurističkim informacijama.

Algorithm 4 ACO algoritam za MLSTP

```
1: Inicijalizuj feromone na svim granama grafa.
2: for svaka iteracija do maksimalnog broja iteracija do
3:    for svaki mrav do maksimalnog broja mrava do
4:       Konstruiši stablo koristeći feromone i heuristiku.
5:       Evaluiraj broj listova stabla.
6:    end for
7:    Ažuriraj feromone na osnovu najboljih rešenja.
8: end for
9: return  najbolje pronadjeno stablo.
```

Vremenska složenost algoritma je $O(I * A * (N^2 + M))$, gde su: I broj iteracija, A broj mrava, M broj grana i N broj čvorova.

tabela 1: Vreme izvrsavanja razlicitih algoritama za razlicite grafove

10	**0.2**	**0.5**	**0.7**
Branch and Bound	1ms	3ms	2ms
Genetski Algoritam (GA)	65ms	81ms	51ms
GA sa simuliranim kaljenjem	1s 781ms	1s 89ms	3s 528ms
Optimizacija kolinijom mrava	120ms	128ms	128ms
20	**0.2**	**0.5**	**0.7**
Branch and Bound	9ms	7ms	8ms
Genetski Algoritam (GA)	112ms	118ms	106ms
GA sa simuliranim kaljenjem	5s 738ms	19s 242ms	11s 574ms
Optimizacija kolinijom mrava	664ms	1s 204ms	1s 551ms
50	**0.2**	**0.5**	**0.7**
Branch and Bound	81ms	36ms	217ms
Genetski Algoritam (GA)	467ms	683ms	578ms
GA sa simuliranim kaljenjem	13s 300ms	30s 113ms	46s 605ms
Optimizacija kolinijom mrava	11s 255ms	49s 679ms	2m 35s
100	**0.2**	**0.5**	**0.7**
Branch and Bound	145ms	139ms	139ms
Genetski Algoritam (GA)	3s 248ms	2s 496ms	3s 143ms
GA sa simuliranim kaljenjem	1m 24s	11m 33s	12m 44s
Optimizacija kolinijom mrava	2m 24s	3m 18s	5m 4s
500	**0.2**	**0.5**	**0.7**
Branch and Bound	5s 657ms	5s 928ms	5s 707ms
Genetski Algoritam (GA)	41s 196ms	1m 25s	1m 53s
GA sa simuliranim kaljenjem	2h 29m	12h 35m	$> 24h$
Optimizacija kolinijom mrava	$> 24h$	$> 24h$	$> 24h$

4 Zaključak

Na osnovu rezultata, algoritam Branch and Bound pokazuje najbolju efikasnost u svim testiranim slučajevima, uključujući i velike grafove. Genetski algoritam je odmah iza njega po brzini i predstavlja dobar izbor za veće grafove kada je potrebno brzo dobiti približno optimalno rešenje. GA sa simuliranim kaljenjem, iako znatno sporiji, može dati preciznija rešenja kod grafova sa malim brojem grana, pa se u tim slučajevima može opravdano koristiti. Optimizacija kolonijom mrava pokazuje lošu skalabilnost i ne preporučuje se za veće instance problema.

References

[1] Hsueh-I Lu i R. Ravi. Approximating Maximum Leaf Spanning Trees in Almost Linear Time. *Journal of Algorithms*, 1998.

[2] https://www.csc.kth.se/ viggo/wwwcompendium/node74.html