Univerzitet u Banjoj Luci

Prirodno-matematički fakultet

Informatika

Uvod u vještačku inteligenciju

The minimum weight vertex cover problem *Greedy randomized adaptive search procedure i Tabu search*

Studenti Mentor

Savo Glavan i Alekandar Obradović Marko Đukanović

Banja Luka, 2023.

# 1.0 Uvod

U teoriji grafova, pokrivač čvorova (vertex cover) grafa je skup čvorova koji

uključuje najmanje jednu krajnju tačku svake ivice grafa. The minimum weight vertex cover (MWVC) je NP težak grafovski poblem. Ako imamo neusmjereni graf G=(V,E) gdje čvorovi imaju date težine, cilj problema je pronaći među svim podskupovima S ⊂ V pokrivač čvorova S∗ pri čemu je zbir težina čvorova minimalan.

Ovaj problem se može primjeniti u mnogim važnim oblastima iz stvarnog života:

* bežičnoj komunikaciji
* dizajnu kola i
* mrežnim tokovima

Zbog svoje složenosti računanja, većina razvijenih tehnika je zasnovana na

heuristici za pružanje približnih rješenja u razumnom vremenu izvršavanja

Prva ideja za rješavanje ovog problema bila je da se počinje sa praznim

djelimičnim rješenjem i da se dodaje po jedan čvor. U svakom koraku se bira čvor sa minimalnim odnosom težine čvora i trenutnog stepena. Trenutni stepen čvora je određen brojem grana koje nisu još uvijek pokrivene.

Kasnije je predložen 2-aproksimacioni algoritam koji uzastopno bira jednu krajnu tačku(čvor) proizvoljno izabrane ivice sa vjerovatnoćom obrnuto proporcionalnoj težini.

2-aproksimacioni algoritam je algoritam koji daje rješenja koja nikada ne koštaju više od duplo više od optimalnih ako je problem minimizacije, ili nikada ne koštaju manje od polovine optimalne vrednosti ako je problem maksimizacije. Takođe je osmišljeno nekoliko metaheursitka za rješavanje ovog problema Khuri i Back su na MWVC gledali kao na ograničeni kombinatorni optimizacioni problem. Uveli su genetski algoritam koji koristi stepenovanu kaznu ugrađenu u fitnes funkciju da se penalizuju nedopustiva rješenja.

Skoro je predložen i mravlji algoritam koji je po empirijskim rezultatima pokazao bolje performanse od ostalih metaheuristika kao što su tabu pretraga i simulirano žarenje.

Ovaj algoritam je dalje poboljšan ugradnjom strategije korekcije feromona u nadi da će izbjeći stagnacija pretrage i konvergencija do lokalnog optimuma.

Međutim predložene metaheuristike se nisu pokazale najbolje. Najčeće im je bilo potrebno dosta vremena za velike instance

# 2.0 Matematički model

Instanca problema MWVC je n-torka (G, ω) koja se sastoji od:

* neusmjerenog grafa *G*(*V*, *E*), gdje je je *V* skup čvorova, a *E* je skup ivica i
* funkcije *ω* : *V* → *R+*koja pridružuje pozitivnu vrijednost težine *ω*(*v*) svakom čvoru *v* ∈ *V*

MWVC problem može biti formalno definisan na sljedeći način:

Minimizovati

Tako da ili ,

* Kandidat za rješenje je označen sa *S* i to je podskup od *V*
* Kandidat za rješenje  *S* je validno rješenje ako je pokrivač čvorova, tj. ako svaka ivica u *G* ima bar jednu krajnu tačku u *S*
* Funkcija cilja *ω*(*S*) je definisana kao suma težina čvorova u *S*
* Cilj ovog problema je naći validnog kandidata koji minimizuje funkciju cilja
* Sa *n = |V =* {*v1, v2 ,. . ., vn*}*|*  označavamo broj čvorova u *G*
* Sa *m* = |*E*| označavamo broj grana u *G*

Podrazumijeva se da je *G* dato kao matrica susjeda

*A* = {*aij* | i = *1*, . . ., *n* ;*j* = *1*, . . ., *n*}

    gdje je:

Označavamo sa *d*(*v*) stepen čvora *v* ∈ *V*, tj. broj grana incidentih sa ovim čvorom i to definišemo na sljedeći način:

# 3.0 Tabu search

## 3.1 Opis algoritma

Tabu pretraga unaprijeđuje učinkovitost lokalnog pretraživanja modifikujući njegovu osnovnu postavku. U svakoj fazi može se prihvatiti lošiji potez ako nema boljih opcija na raspolaganju.

Uvode se i zabrane (tabu liste) kako se pretraga ne bi vraćala na već posjećena rješenja.

Implementacija tabu pretrage koristi memorijske mehanizme koji bilježe posjećena rješenja ili skupove pravila definisane od strane korisnika.

Ako je potencijalno rješenje prethodno posjećeno u određenom kratkoročnom periodu ili ako je prekršilo pravilo, ono se označava kao "tabu" (zabranjeno) tako da algoritam tu mogućnost više puta ne razmatra.

Tabu pretraga koristi lokalnu ili pretragu po okolini kako bi iterativno prešla s jednog mogućeg rješenja x na unaprijeđeno rešenje x' blizu x, sve dok se ne dostigne određeni uslov za prekid.

Tabu pretraga radi tako što pretražuje susjede svakog novog rješenja. Rješenja koja se uzimaju u novom susjedstvu određena su na osnovu struktura memorije. Uz pomoć ovih struktura memorije, pretraga napreduje tako da se u iteracijama prelazi sa trenutnog rješenja x na poboljšano rješenje x' u novom susjedstvu

## 3.2 Opis primjene algoritma

„MAX\_TABU\_TENURE“ predstavlja maksimalan broj iteracija koliko jedan potez ostaje zabranjen.

Aspiracioni kriterijum predstavlja kriterijum po kojem čak i ako je potez tabu, ako vodi do rješenja boljeg od bilo kojeg koje smo do sada pronašli, i dalje možemo da ga napravimo.

### 3.2.1 Pohlepna inicijalizacija

Cilj je da se nađe početni pokrivajući skup čvorova koristeći pohlepni algoritam. Ovaj pristup uzima u obzir odnos težine čvora i broja njegovih susjeda:

* Algoritam počinje sa praznim pokrivajućim skupom čvorova
* Računa se odnos težine čvora i broja susjeda za svaki čvor
* Iterativno se bira čvor s najmanjim odnosom, dodaje se u pokrivajući skup čvorova, a zatim se sve grane povezane s tim čvorom uklanjaju iz liste nepokrivenih grana
* Ovaj proces se ponavlja dok sve grane nisu pokrivene.

### 3.2.2 Izračunavanje ukupne težine

Računa se zbir težina svim čvorova u datom rješenju.

### 3.2.3 Pronalaženje najboljeg susjeda:

* Metoda traži najbolje susjedno rješenje dodavanjem ili uklanjanjem čvora iz trenutnog rješenja.
* Za svaki čvor u listi:
  + Ako se nalazi u trenutnom rješenju, susjed se kreira uklanjanjem istog
  + U suprotnom, susjed se kreira dodavanjem
* Zatim provjeravamo da li ovaj susjed pokriva sve grane.
* Takođe pratimo težinu ovog susjednog rješenja.
* Ako je najbolja koju smo do sada pronašli, a nije tabu (ili čak ako jeste tabu, ali je bolja od našeg najboljeg rešenja do sada), ažuriramo našeg najboljeg susjeda.

### 3.2.4 Pomoćna klasa NeighborResult

Ovo je pomoćna klasa koja služi da čuva dvije informacije:

* neighbor: Jedno rješenje koje je susjed trenutnom rješenju.
* modifiedNode: Čvor koji je dodat ili uklonjen da bi se dobio susjed.

### 3.2.5 Ažuriranje Tabu liste

Lista se ažurira tako što se:

* Dodaje modifiedNode u nju.
* Proverava da li veličina tabu liste premašuje njen limit (MAX\_TABU\_TENURE). Ako premašuje, najstariji čvor se uklanja kako bi se održala veličina tabu liste

### 3.2.6 Glavna metoda tabu pretrage

* Prvo, inicijalizujemo rješenje pomoću pohlepne metode.
* Pratimo najbolje rješenje do sada (bestSolution).
* Takođe inicijalizujemo tabuList koji će čuvati čvorove koje smo nedavno mijenjali.
* While petlja se izvršava dok nismo imali 100 iteracija bez poboljšanja.
* U svakoj iteraciji tražimo najboljeg susjeda (koristeći findBestNeighbor) i upoređujemo ga s trenutnim najboljim rešenjem.
* Ako je sused bolji, ažuriramo najbolje rešenje i resetujemo brojač bez poboljšanja.
* Zatim ažuriramo našu tabu listu i postavljamo trenutno rješenje na susjedno rješenje.