Bandwidth minimization problem

Nevena Nikolić 1021/2018
Luka Kalinić 1058/2018

Sadržaj

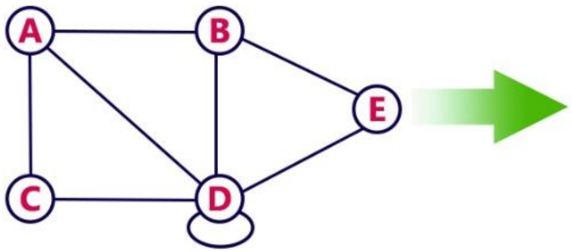
- Uvod
- Formulacija problema
  - *Bandwidth* problem minimizacije za matrice
  - *Bandwidth* problem minimizacije za grafove
- Metode za rešavanje
- VNS pristup rešavanja problema
  - Početno rešenje
  - Razmrdavanje
  - Lokalna pretraga
  - Pomeriti se ili ne?

Uvod

- *Bandwidth problem minimizacije* predstavlja pronalaženje permutacije redova i kolona retke matrice sa ciljem da ne-nula elementi budu sadržani u okviru trake koja je što je moguće bliža glavnoj dijagonali.
- Problem datira još iz pedesetih godina prošlog veka.
- Primene su brojne – najvažnija je u rešavanju velikih sistema linearnih jednačina. Vremenska složenost je O(nb^2) gde je *n* dimenzija matrice a *b* bandwidth. Složenost direktnog pristupa O(n^3).
- Bandwidth problem minimizacije je NP-kompletan.
- Algoritmi za rešavanje *BML-a* mogu da se podele na ***egzaktne i heurističke***. Štaviše, razvijeno je više metaheuristika.

Formulacija problema

- *Bandwidth problem minimizacije* može biti formulisan za matrice ili za grafove pri čemu važi da su obe formulacije problema ekvivalentne.
- Ekvivalencija je zasnovana na činjenici da se dati graf može predstaviti svojom matricom povezanosti i obratno.



$$
\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array}
\begin{array}{c}
\begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{pmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

Bandwidth problem minimizacije za matrice

- Neka je data binarna (svi elementi su 0 ili 1), retka, simetrična matrica $A = \{a_{ij}\}$. Tada se *bandwidth* matrice *A* definiše kao:

$$B(A) = max\{|i - j| : \ a_{ij} \neq 0\}$$

- *Bandwidth* problem minimizacije za matrice predstavlja permutovanje redova i kolona matrice *A* tako da ne-nula elementi budu sadržani u traci što je moguće bliže glavnoj dijagonali.

Bandwidth problem minimizacije za grafove

- Neka je *G* = *(V, E)* konačan neusmereni graf gde je *V* skup čvorova i *E* skup grana. Neka je 1-1 funkcija *f* : *V* → {*1,2,…,n*} funkcija označavanja čvorova grafa. Tada se *bandwidth* čvora v definiše kao:

$$B_f(v) = \ max_{j:(v,j)\in E}\{|f(v) - f(j)|\}$$

- *Bandwidth* grafa *G* za *f* definiše se kao:

$$B_f(v) = \max\{|f(i) - f(j)| : (i,j) \in E\}$$

Bandwidth problem minimizacije za grafove

- *Bandwidth* problem minimizacije predstavlja pronalaženje funkcije f koja minimizuje izraz *bandwidth* grafa, odnosno veličinu $B_f(G)$.
- Za graf sa *n* čvorova broj mogućih označavanja je *n!*.
- Direktan pristup bi bio isprobavanje svih permutacija i izbor najboljeg rešenja. Međutim, vreme izvršavanja ovog metoda je *O(n!)* te ovaj pristup postaje nepraktičan već za grafove koji imaju 10 čvorova.

Metode za rešavanje

- Kao što smo već pomenuli, algoritmi za rešavanje *bandwidth* problema minimizacije mogu da se podele na ***egzaktne*** i ***heurističke*** (odnosno ***metaheurističke***).
- Egzaktni algoritmi se najčešće baziraju na pristupu poznatom kao *grananje i ograničavanje (eng. branch and bound)*. Treba uzeti u obzir i cenu izračunavanja za dobijanje optimalnog rešenja! Stoga ovi metodi mogu rešavati samo probleme čija je veličina dovoljno mala da vreme izvršavanja bude razumno.
- ***Heuristički*** pristupi zasnovani su na iskustvu i dobijeno rešenje ne mora biti optimalno. Ipak, ovakvim pristupom možemo brzo pronaći rešenje koje je dovoljno dobro za dati problem kombinatorne optimizacije
- ***Metaheuristički*** metod koji ćemo koristiti u nastavku je *metod promenljivih okolina (eng. Variable Neighborhood Search, VNS)*.

VNS pristup rešavanju problema

- Metod promenljivih okolina *(VNS)* je metaheuristika predložena 1997. godine koja se pokazala veoma korisnom za dobijanje približnog rešenja optimizacionih problema.
- Predstavlja uopštenje lokalne pretrage gde su definisane okoline po kojima će se sistematski vršiti pretraživanje.
- *VNS* se bazira na sledećim principima:
  - Lokalni minimum za jedan tip okoline ne mora nužno i da bude lokalni minimum za drugi tip okoline.
  - Globalni minimum predstavlja lokalni minimum za sve tipove okolina.
  - Za mnoge probleme u praksi, lokalni minimumi više tipova okolina su relativno blizu jedan drugog.

VNS pristup rešavanju problema

- Osnovni *VNS* algoritam uključuje tri postupka:
  - ***Razmrdavanje*** – pokušaj iskakanja iz tekućeg lokalnog optimuma i pronalaženje novog optimalnog rešenja koje će biti bliže globalno optimalnom rešenju.
  - ***Lokalnu pretragu*** – pronalaženje lokalnog optimuma kako bi se unapredila preciznost pretrage.
  - ***Promenu okoline*** – *„Pomeriti se ili ne?“* označava menjanje okoline, što nam daje iterativni metod i kriterijum zaustavljanja.

VNS pristup rešavanju problema

- Pseudokod osnovnog *VNS* algoritma:
  - Odabrati tipove okolina $U_l(x), 1 \leq l \leq l_{max}$
  - Generisati početno rešenje x i postaviti vrednost najboljeg rešenja $f^* = f(x)$
  - Dok nije zadovoljen kriterijum zaustavljanja, ponavljati sledeće korake:
    - $l = 1$
    - Dok je $l \leq l_{max}$ ponavljati sledeće korake:
      - Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
      - Primenom nekog tipa lokalne pretrage na početno rešenje x' naći rešenje x'' sa najmanjom vrednošću funkcije cilja
      - Ako rešenje x'' ima manju vrednost funkcije cilja od rešenja x, onda dodeliti $x = x''$ i $l = 1$, a inače $l = l + 1$
    - Po potrebi, ažurirati vrednost f^*
  - Ispisati vrednost f^*

Početno rešenje

- Kvalitet početnog rešenja će direktno uticati na performanse algoritma!
- Dobro početno rešenje može se generisati korišćenjem pretrage u širinu. Osnovna ideja je da bi povezani čvorovi trebalo da imaju bliske oznake.
- Struktura nivoa *(eng. Level structure)* grafa, u ozanci *L(G)*, jeste particionisanje njegovih čvorova u nivoe $L_1, L_2, \ldots, L_k$ koji zadovoljavaju sledeće uslove:
  - Čvorovi povezani sa čvorom iz nivoa L_1 su ili u L_1 ili u L_2
  - Čvorovi povezani sa čvorom iz nivoa L_k su ili u L_k ili u L_{k-1}
  - Čvorovi povezani sa čvorom iz nivoa L_i (za *1*<*i*<*k*) su ili u L_{i-1} ili u L_i ili u L_{i+1}
- Različiti izbori početnog čvora vode ka različitim početnim rešenjima.
- Očigledno, *bandwidth* koji se dobija ne može biti gori od maksimalnog *bandwidth-a* grafa. *BFS* daje gornju granicu za kvalitetno rešenje.

Razmrdavanje

- Označavanje f' je u *k*-toj okolini označavanja *f* ako se ta dva označavanja razlikuju u *k+1* oznaci.
- Preciznije, rastojanje ρ između dva rešenja f i f' definiše se kao:

$$\rho(f, f') = \sum_{i=1}^{n} \eta(i) - 1, \qquad \eta(i) = \begin{cases} 1, f(i) \neq f'(i) \\ 0, f(i) = f'(i) \end{cases}$$

- U cilju odabira čvorova za razmenu njihovih oznaka, uvodimo još dve definicije:

$$f_{max}(v) = \max\{f(u), u \in N(v)\}$$
$$f_{min}(v) = \min\{f(u), u \in N(v)\}$$

- $f_{max}(v)$ predstavlja najveću oznaku među čvorovima susednim čvoru v, dok $f_{min}(v)$ predstavlja najmanju.

Razmrdavanje

- Najpre se generiše skup $K \subset V$ takav da mu je kardinalnost veća od datog broja k. Zatim se nasumično bira čvor u iz skupa K i pronađe se njegov kritični čvor.
- Kritični čvor čvora u je čvor v za koji važi:

$$|f(u) - f(v)| = B_f(u)$$

- Dalje, bira se čvor w tako da zadovoljava sledeća dva uslova:
  - vrednost $\max\{f_{max}(w) - f(v), f(v) - f_{min}(w)\}$ je minimalna
  - važi nejednakost $f_{min}(u) \leq f(w) \leq f_{max}(u)$
- Konačno, zamenjuju se oznake čvorovima v i w

Razmrdavanje

- Opisani postupak može se predstaviti narednim pseudokodom:

Algorithm 1 Shaking (k, f)

Initialization:

Let $K = \{v | B_f(v) \geq B'\}$, B' is chosen such that $|K| \geq k$;

Iteration:

1: **for** $i = 1$ to k **do**
2: $u \leftarrow$ RandomInt $(1, |K|)$;
3: $v \leftarrow$ such that $|f(u) - f(v)| = B_f(u)$;
4: **if** $(u, v) \in E$ **then**
5: $w \leftarrow \arg\min_w \{\max\{f_{max}(w) - f(v), f(v) - f_{min}(w)\} | f_{min}(u) \leq f(w) \leq f_{max}(u)\}$;
6: $swap(f(u), f(v))$
7: **end if**
8: **end for**

Lokalna pretraga

- Lokalnu pretragu koristimo da bismo konstruisali skup čvorova pogodnih za zamenu oznake.
- Najbolje označavanje za tekući čvor v definiše se kao:

$$mid(v) = \left\lfloor \frac{\max(v) + \min(v)}{2} \right\rfloor$$

- Tada je skup čvorova pogodnih za zamenu oznake za čvor v definisan kao:

$$N'(v) = \{u : |mid(v) - f(u)| < |mid(v) - f(v)|\}$$

- Kritična grana je ona grana (u, v) takva da važi:

$$B_f(u) = B_f(G) \text{ i } B_f(v) = B_f(G)$$

Lokalna pretraga

- Opisani postupak se može se predstaviti narednim pseudokodom:

Algorithm 2 Local Search (f)

while $CanImprove$ **do**
  $CanImprove = False$;
  for $v = 1$ to n **do**
    if $B_f(v) = B_f(G)$ **then**
      for all u such that $u \in N'(v)$ **do**
        swap $(f(v), f(u))$ and update $(B_f(w), B_f(G)), \forall w \in (N(v) \cup N(u))$;
        if number of critical edges reduced **then**
          $CanImprove = True$;
          $break$;
        end if
        swap $(f(v), f(u))$ and update $(B_f(w), B_f(G)), \forall w \in (N(v) \cup N(u))$;
      end for
    end if
  end for
end while

Pomeriti se ili ne?

- Nakon pronalaska lokalno optimalnog rešenja, moramo doneti odluku da li trenutno rešenje f zamenjuje novim rešenjem f'.
- Razmatraju se sledeća tri slučaja:
  - $B_{f'}(G) < B_f(G)$: ako je *bandwidth* novog rešenja bolji od trenutnog *bandwidth*-a, lako je zaključiti da se treba pomeriti.
  - $|V_c(f')| < |V_c(f)|$: ako se *bandwidth* ne menja, odnosno $B_{f'}(G) = B_f(G)$, upoređujemo broj kritičnih čvorova (onih čiji je *bandwidth* jednak $B_f(G)$) za trenutno i novo rešenje da bismo videli da li je $|V_c(f')|$ smanjen.
  - $\rho(f', f) > \alpha$: Ako prethodna dva uslova nisu ispunjena, onda poredimo rastojanje između trenutnog i novog rešenja sa korisnički zadatim koeficijentom α.

Pomeriti se ili ne?

- Opisani postupak se može prikazati sledećim pseudokodom:

Algorithm 3 Move (f, f', α)

$Move \leftarrow False;$
if $B_{f'}(G) < B_f(G)$ **then**
    $Move \leftarrow True;$
else
    if $B_{f'}(G) = B_f(G)$ **then**
        if $|V_c(f')| < |V_c(f)|$ or $\rho(f', f) > \alpha$ **then**
            $Move \leftarrow True;$
        end if
    end if
end if

VNS pristup rešavanju problema

- Konačno, *VNS* pristup rešavanja *bandwidth* problema minimizacije može se predstaviti narednim pseudokodom:

Algorithm 4 VNS $(A, k_{min}, k_{max}, k_{step}, \alpha)$

Initialization:

$B^* \leftarrow \infty; t \leftarrow 0;$

$i_{max} = Int((k_{max} - kmin)/k_{step});$

$f \leftarrow InitSol(f); f \leftarrow LocalSearch(f);;$

$i \leftarrow 0; k \leftarrow k_{min};$

while $i \leq i_{max}$ **do**

  $f' \leftarrow Shaking(f, k);$

  $f' \leftarrow LocalSearch(f');$

  if $Move(f, f', \alpha)$ **then**

    $f \leftarrow f'; k \leftarrow k_{min}; i \leftarrow 0;$

  else

    $k \leftarrow k + k_{step}; i \leftarrow i + 1;$

  end if

end while

Kraj

Hvala na pažnji!