UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Min-Graph Equipartition Problem with Simulated Annealing

Ana Marija Kravanja, Urška Jeranko, Oskar Kregar

Ljubljana, 2019

Kazalo

1. OPIS PROBLEMA

V naši projektni nalogi smo reševali problem deljenja grafa z uporabo dveh predpisanih metod. Poljuben graf smo morali razdeliti na dva skoraj enaka dela tako, da je bilo med tema nastalima deloma čim manj povezav. Projekta smo se lotili tako, da smo napisali algoritma požrešne metode (Greedy Method) in metode simuliranega ohlajanja (Simulated Annealing Heuristic), napisali pa smo tudi funkcijo, ki že razdeljen graf nariše in z barvami prikaže optimalno delitev vozlišč. Algoritma smo preizkusili na več različnih grafih in analizirali, kako se metodi obneseta na gostih, redkih ter poljubnih grafih. Pri metodi simuliranega ohlajanja pa smo spreminjali tudi začetno temperaturo in nato primerjali dobljene rešitve med seboj.

Grafe, ki smo jih preučevali, smo v oba algoritma vstavljali v obliki matrike sosednosti, zato smo na začetku definirali: Naj bo $G = (V, E)$ enostaven graf, pri čemer je V množica vozlišč in E množica povezav. Naj bo število vozlišč enako n. Definirali smo delitev grafa na dve množici X in Y, pri čemer je $|X| = \lceil \frac{n}{2} \rceil$. Iskali smo minimalno širino bisekcije, to je najmanjše število povezav med X in Y med vsemi možnimi delitvami.

Definicija 1.1. Naj bo $G = (V, E)$ enostaven graf in $X, Y \subseteq V$, tako da je $X \cap Y = \emptyset$ in $X \cup Y = V$.

- Za $x \in X$ označimo z $I(x)$ notranjo vrednost, to je število povezav $(x, z) \in E; z \in X \backslash \{x\}$. Analogno definiramo $I(y)$ za $y \in Y$.
- Za $x \in X$ označimo z $O(x)$ zunanjo vrednost, to je število povezav $(x, z) \in E; z \in Y$. Analogno definiramo $O(y)$ za $y \in Y$.
- Za $x \in X, y \in Y$ naj bo $\omega(x, y) := \begin{cases} 1, & \text{if}(x, y) \in E \\ 0, & \text{sicer} \end{cases}$.
- Za $x \in X, y \in Y$ naj bo $S(x, y) := O(x) - I(x) + O(y) - I(y) - 2\omega(x, y)$.

2. POŽREŠNA METODA

Požrešna metoda je strategija, ki na vsakem posameznem koraku izbere optimalno rešitev s ciljem, da nas to privede do globalno optimalne rešitve. To pomeni, da algoritem izbere rešitev, ki je trenutno najboljša, vendar se pri tem ne ozira na posledice - ne gleda celotne slike. Težava te metode je, da na vsakem koraku izbere le lokalno najboljšo rešitev, samo upamo pa lahko, da je to tudi prava pot do globalnega optimuma. Pogosto torej sploh ne najde najboljše rešitve, povsem mogoče je celo, da nas pripelje do najslabše možne.

2.1. **Algoritem požrešne metode.** Začnemo z naključno delitvijo vozlišč grafa na dve skoraj enako veliki množici X in Y. Algoritem zamenja vozlišči na različnih straneh (pri čemer se vse povezave ohranijo), če s tem dobimo boljšo bisekcijo (manjše število povezav med stranema) in se ustavi, ko to ni več možno. Pri menjavi notranje vrednosti postanejo zunanje in obratno, zato dobimo izboljšano bisekcijo le, če je zgoraj definiran $S(x, y) > 0$.

Vhodni podatki: Graf $G = (V, E), |V| = n$.

(1) Izberemo nekljčno delitev (X, Y).
(2) Izberemo $x \in X, y \in Y$, tako da je $S(x, y) > 0$.
(3) Zamenjamo vozlišči x in y.
(4) Ponavljamo 2. in 3. korak, dokler ne obstajata več $x \in X, y \in Y$, da je $S(x, y) > 0$.

Izhodni podatki: delitev (X, Y).

Algoritma smo se lotili tako, da smo najprej definirali število vozlišč n kot dolžino vhodnega grafa G, ki ga v funkcijo vstavimo kot seznam seznamov, da dobimo matriko sosednosti. Nato smo morebitne enke na diagonali matrike G spremenili v ničle in s tem izbrisali povezave vozlišč s samimi seboj, ki na rezultat tako nimajo vpliva, s tem pa smo preprečili morebitne komplikacije. V naslednjem koraku smo izbrali delitvi X in Y tako, da je v množici X prva polovica vozlišč, v množici Y pa druga polovica, kar je tudi ena od možnih naključnih delitev. Ker to skoraj zagotovo ni optimalna delitev, smo napisali pomožno funkcijo, ki preveri vse možne delitve na način, ki smo ga opisali zgoraj. Uporabili smo for zanki, ki tečeta po vseh elementih množice X in množice Y, da preverimo vse možne delitve. Notranjo in zunanjo vrednost teh dveh množic smo nastavili na 0. Nato smo uporabili še for zanki, ki štejeta povezave znotraj množic X in Y ter med njima in tako računata notranje in zunanje vrednosti. Na ta način dobimo vrednost $S(x, y)$, ki nam pove, ali vozlišči $x \in X$ in $y \in Y$ zamenjamo. Zanka se izvaja, dokler ne obstaja več pozitiven $S(x, y)$, takrat algoritem vrne množici vozlišč X in Y, ki predstavljata optimalno delitev vozlišč.

3. METODA SIMULIRANEGA OHLAJANJA

Algoritem se je v osnovi razvil zaradi procesa toplotne obdelave kovin. Različne materiale segrevamo oz. ohlajamo, da bi spremenili fizikalne lastnosti njihove notranje strukture. Ko se kovina enkrat ohladi, postane njena nova struktura fiksna in kovina tako obdrži pridobljene lastnosti. Pri simuliranem ohlajanju je temperatura naša spremenljivka, ki ponazarja toplotni proces. Na začetku temperaturo nastavimo na visoko vrednost in jo nato sčasoma nižamo, ko se algoritem izvaja. Na začetku, ko je temperatura nastavljena na visoko vrednost, algoritem pogosteje sprejema rešitve, ki so slabše od naše trenutne rešitve zato, da se izogne lokalnim optimumom, ki nas ne bodo pripeljali do globalnega optimuma. Z nižanjem temperature pa se niža tudi verjetnost, da bo algoritem izbiral slabše rešitve; postopoma se torej osredotoči na iskalno območje, na katerem upamo, da je mogoče najti rešitev blizu optimalne. Algoritem je zelo učinkovit pri iskanju rešitev za probleme velikih velikosti, ki vsebujejo številne lokalne optimume.

3.1. **Algoritem metode simuliranega ohlajanja.** Graf definiramo analogno kot pri požrešni metodi. Vozlišči pa zdaj zamenjamo z verjetnostjo

$$P(x, y, t) = e^{\frac{S(x,y)}{t}}; t \geq 0$$

S $Q(S)$ je označena funkcija, ki množici S priredi neko kvaliteto, torej če je $Q(S) > Q(R)$, pomeni, da je množica S(???) bolj ugodna za nadaljnjo obravnavo. Za funkcijo Q smo si v našem primeru izbrali število povezav med množicama. Torej

če je $Q(R)$ funkcija, kjer smo v množici R zamenjali vozlišča, in $Q(S)$ množica S, kjer ju nismo zamenjali, in velja, da je $Q(R) < Q(S)$, potem z določeno verjetnostjo zamenjamo R z S. V množici R je v tem primeru med podmnožicama namreč manj povezav kot v S.

(1) t = temperatura, ki jo nastavimo na visoko vrednost
(2) S = začetna delitev grafa na množici X in Y
(3) N = S (t.j. trenutna najboljša rešitev)
(4) Ponavljaj, dokler je N najboljša rešitev, nam je zmanjkalo časa ali pa je $t \leq 0$:
(5) R = delitev, kjer zamenjamo $x \in X$ in $y \in Y$
(6) če je $Q(R) < Q(S)$ ali naključno število med 0 in 1, ki je manjše od $P(x, y, t)$, potem je $S = R$
(7) zmanjšamo t
(8) če je $Q(S) > Q(N)$:
(9) $N = S$
(10) vrni N

Literatura

[1] L. A. Wolsey, Integer Programming, Wiley Interscience,1998.

[2] C. Blum, A. Roli, Metaheuristics in Combinatorial Optimization: Overview and Conceptual Compariosn. [online]

[3] S. Luke, Essentials of Metaheuristics: a set of undergraduate lscture notes, online.

[4] S. Clinton, Generic algorithms with Python.

[5] K.L. Du, M.N.S. Swamy, Search and optimization by metaheuristics: tehniques and algorithms inspired by nature.

[6] El. G. Talbi, Metaheuristics: from design to implementation.