UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

# Min-Graph Equipartition Problem with Simulated Annealing
# (Problem delitve grafa z metodo simuliranega ohlajanja)

Ana Marija Kravanja, Urška Jeranko, Oskar Kregar

Ljubljana, 2019

## 1. OSNOVNO O PROJEKTU

V naši projektni nalogi bomo reševali problem deljenja grafa. Naš cilj je graf razdeliti na dva skoraj enaka dela tako, da je med tema nastalima deloma čim manj povezav. Najti tako delitev je težek problem, ki postane še težji, če moramo preveriti zelo veliko rešitev, da najdemo optimalno. Reševali ga bomo s požrešno metodo (Greedy Method) in metodo simuliranega ohlajanja (Simulated Annealing Heuristic) ter nato rešitve metod primerjali med sabo in jih preizkusili na različnih grafih (gosti, redki, naključni) pri različnih parametrih (temperatura, sosednja vozlišča).

## 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.

Naj bo $G = (V, E)$ enostaven graf, pri čemer je $V$ množica vozlišč in $E$ množica povezav. Naj bo število vozlišč enako $n$. Definirajmo delitev grafa na dve množici X in Y, pri čemer je $|X| = \lceil \frac{n}{2} \rceil$. Iskali bomo minimalno širino bisekcije, to je najmanjše število povezav med $X$ in $Y$ med vsemi možnimi delitvami.

**Definicija 2.1.** 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 $S(x, y) := O(x) - I(x) + O(y) - I(y) - 2\omega(x, y)$.

2.1. **Algoritem požrešne metode.** Začnemo z naključno delitvijo vozlišč grafa na dve skoraj enako veliki množici. Algoritem zamenja dve 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 $S(x, y) > 0$.

2.1.1. *Algoritem 1.* Vhodni podatki: Graf $G = (V, E), |V| = n$.

(1) Izberemo neključno 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 $(X, Y)$.

## 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, ki so blizu optimalne, za probleme velikih velikosti, ki vsebujejo številne lokalne optimume.

### 3.1. **Algoritem metode simuliranega ohlajanja.**

Graf si definiramo analogno kot pri požrešni metodi. Vozlišči zamenjamo z verjetnostjo

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

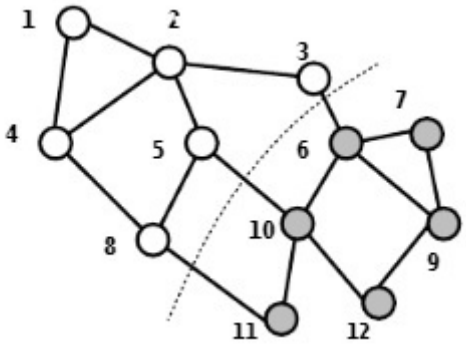
.

3.1.1. *Algoritem 2.* S $Q(S)$ je označena funkcija, ki množici S priredi neko kvaliteto, torej če je $Q(S) > Q(R)$, pomeni, da je množica Z 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 x in y 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.

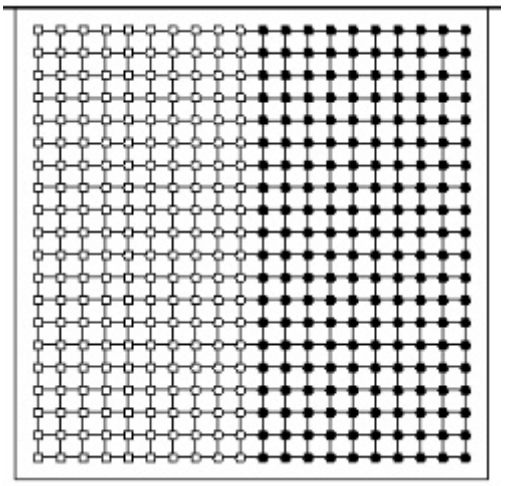
(1) t= temperatura z visoko vrednostjo
(2) S= začetna delitev grafa na množici X in Y
(3) N= S naj predstavlja trenutno najboljšo rešitev
(4) ponovi, dokler je N najboljša rešitev, nam je zmanjkalo časa ali pa je $t \leq 0$
(5) R=delitev, kjer zamenjamo $x$ in $y$
(6) če je $Q(R) < Q(S)$ ali naključno število med 0 in 1 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 $N$

## 4. PRIMERI

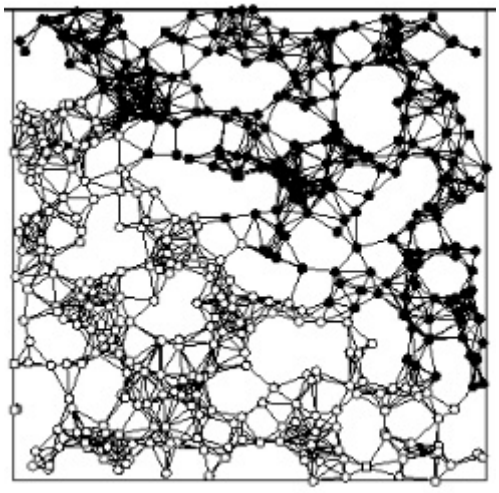
Poglejmo si nekaj primerov delitve grafov.




Slika 1. primer grafa na 12 točkah



Slika 2. primer grafa na 400 točkah, prikazana je optimalna rešitev



Slika 3. Johnsonsov geometrični graf

## 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.