

UNIVERZA V LJUBLJANI

Fakulteta za matematiko in fiziko

Finančni praktikum

Največje neodvisne množice z lokalnim
iskanjem

Avtorja:

Jaka Mrak

Žiga Gartner

Mentorja:

prof. dr. Sergio CABELLO

doc. dr. Janoš VIDALI

Ljubljana, 8. januar 2022

Kazalo

1	Navodilo	2
2	Opis problema	2
3	Opis dela	3
3.1	Generiranje podatkov	3
3.2	Algoritmi	3
3.3	Analiza rezultatov	3
4	Sklep	3
	Literatura	4

1 Navodilo

Naloga je iskanje največje neodvisne množice v grafu $G = (V, E)$ s pomočjo celoštevilkega linearnega programiranja. Velike neodvisne množice v grafu lahko poiščemo s pomočjo metode lokalnega iskanja. Začnemo s poljubno neodvisno množico $U \subseteq V$, kjer k vozlišč nadomestimo s $k + 1$ vozlišči tako, da ohranjamo neodvisnost množice U . Konstanta k je dana na začetku. Primerjali bomo metodi lokalnega iskanja in optimalne rešitve ter primerjali njune rešitve za nekatere preproste grafe.

2 Opis problema

Definicija 1. Naj bo $G = (V, E)$ graf. **Neodvisna množica** U , v grafu G , je taka podmnožica množice vozlišč V , kjer poljubni dve vozlišči iz množice U nista sosednji. **Maksimalna neodvisna množica** v grafu G pa je taka neodvisna množica, kjer ne obstaja vozlišče $v \in V$ in $v \notin U$, ki bi ga lahko dodali množici U in pri tem ohranili neodvisnost množice U . Torej je neodvisna množica U največja taka, če velja ena od naslednjih dveh lastnosti:

1. $v \in U$
2. $S(v) \cap U \neq \emptyset$, kjer je $S(v)$ množica sosedov v .

Največja neodvisna množica je neodvisna množica, največje možne velikosti, za dan graf G . Velikosti največje neodvisne množice, za graf G , pa pravimo **neodvisnostno število** in pogosto označimo $\alpha(G)$.

Definicija 2. Celoštevski linearni program v standardni obliki je dan z matriko $A \in \mathbb{R}^{m \times n}$, vektorjem $b \in \mathbb{R}^m$ in vektorjem $c \in \mathbb{R}^n$. Iščemo

$$\max \langle c, x \rangle,$$

da bodo zadoščeni pogoji

$$Ax \leq b, x \geq 0,$$

kjer je $x \in \mathbb{Z}^n$.

Posledica 1. Problem največje neodvisne množice v grafu $G = (V, E)$ lahko s celoštevilskim linearnim programiranjem modeliramo na sledeč način:

$$\max \sum_{v \in V} x_v,$$

da velja:

$$x_v + x_w \leq 1 \text{ za } \forall vw \in E,$$

$$x_v \in \{0, 1\}$$

$$x_u = \begin{cases} 1, & \text{za } u \in U \\ 0, & \text{za } u \notin U \end{cases}, U \text{ neodvisna množica v grafu } G.$$

Največjo neodvisno množico v množici vseh neodvisnih podmnožic grafa $G = (V, E)$ bomo iskali s pomočjo celoštevilkega linearnega programiranja in lokalnega iskanja. **Lokalno iskanje** temelji na izbiri začetne neodvisne podmnožice vozlišč $U \subset V$ v kateri k vozlišč zamenjamo s $k + 1$ vozlišči in pri tem ohranjamo neodvisnost množice U .

3 Opis dela

V nadaljevanju bomo največjo oz. maksimalno množico iskali s pomočjo *CLP*, v Sage, in s pomočjo implementiranega algoritma *nakljucni_MIS(G)* kasneje izboljšanega z lokalnim iskanjem. Algoritme bomo izvajali na grafih, generiranih s pomočjo Erdős-Rényijevega $G(n, p)$ modela. Primerjali bomo maksimalne (ali pa največje) neodvisne množice, ki jih algoritmi poiščejo, časovno zahtevnost algoritmov, kasneje pa še, kako vpliva spreminjanje števila vozlišč in verjetnosti, v modelu, na velikost maksimalne (ali pa največje) neodvisne množice in časovno zahtevnost algoritmov.

3.1 Generiranje podatkov

Kot omenjeno, bomo grafe generirali s pomočjo Erdős-Rényijevega $G(n, p)$ modela.

Definicija 3. *Erdős-Rényijev model $G(n, p)$ generira graf z n naključno povezanimi vozlišči. Vsaka povezava je v graf vključena neodvisno, z verjetnostjo p .*

Generirali bomo grafe s konstantima (n, p) , z naraščajočim n in konstantnim p ter s konstantim n in naraščajočim p . Za generacijo podatkov v Pythonu bomo uporabili knjižnico *NetworkX*, nato pa objekte grafov, s funkcijo *nx.to_dict_of_lists(graf)*, kjer je *nx* okrajšava za *NetworkX*, pretvorili še v slovarje list, da bomo lahko grafe uporabljali še v okolju *Sage*. Grafe, na katerih bomo izvajali algoritme, bomo shranili še v *JSON* datoteke.

3.2 Algoritmi

Za iskanje maksimalne neodvisne množice smo implementirali algoritem *nakljucni_MIS(G)*, čigar rešitev bomo poizkušali izboljšati še z algoritmom *lokalno_iskanje(G, I)*. Oba algoritma bosta kot argument sprejela graf G , algoritem *lokalno_iskanje(G, I)* pa še neko maksimalno neodvisno množico I , grafa G .

Algorithm 1 *nakljucni_MIS(G)*

```
1:  $I \leftarrow \emptyset$ 
2:  $\forall v \in V$  dobi vrednost  $P(v) \in \text{permutacija}(V)$ 
3: if  $P(v) < P(w)$  za  $\forall w \in \text{sosedi}(v)$  then
4:    $I \leftarrow I \cup v$ 
5:  $V' \leftarrow V \setminus (I \cup \text{sosedi}(I))$ .
6:  $E' \leftarrow E \setminus \text{povezave}(I)$ .
7: return  $I \cup \text{MIS}(G' = (V', E'))$ 
```

3.3 Analiza rezultatov

4 Sklep

Algorithm 2 *lokalno_iskanje*(G, I)

```
1:  $I \leftarrow \emptyset$ 
2:  $\forall v \in V$  dobi vrednost  $P(v) \in \text{permutacija}(V)$ 
3: if  $P(v) < P(w)$  za  $\forall w \in \text{sosedi}(v)$  then
4:    $I \leftarrow I \cup v$ 
5:  $V' \leftarrow V \setminus (I \cup \text{sosedi}(I))$ .
6:  $E' \leftarrow E \setminus \text{povezave}(I)$ .
7: return  $I \cup \text{MIS}(G' = (V', E'))$ 
```

Literatura

- [1] Gary Miller. *Lecture 32: Luby's Algorithm for Maximal Independent Set*, dostopno na <http://www.cs.cmu.edu/afs/cs/academic/class/15750-s18/ScribeNotes/lecture32.pdf>.
- [2] Diogo Andrade, Mauricio G. C. Resende. *Fast Local Search for the Maximum Independent Set Problem*. Conference Paper in Journal of Heuristics, May 2008, dostopno na https://www.researchgate.net/publication/221131653_Fast_Local_Search_for_the_Maximum_Independent_Set_Problem.
- [3] *Maximal independent set*, v: Wikipedia: The Free Encyclopedia,[ogled 6. 1. 2022], dostopno na https://en.wikipedia.org/wiki/Maximal_independent_set.
- [4] *Independent set (graph theory)*, v: Wikipedia: The Free Encyclopedia,[ogled 6. 1. 2022], dostopno na [https://en.wikipedia.org/wiki/Independent_set_\(graph_theory\)](https://en.wikipedia.org/wiki/Independent_set_(graph_theory)).