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, 9. januar 2022

Kazalo

1 Navodilo **2**

2 Opis problema **2**

3 Opis dela **3**

3.1 Generiranje podatkov . . . 3

3.2 Algoritmi . . . 3

3.2.1 Algoritem: $naklјucni\_MIS(G)$ . . . 3

3.2.2 Algoritem: $lokalno\_iskanje(G, I)$ . . . 4

3.3 Analiza rezultatov . . . 5

3.3.1 Analiza algoritmov na grafih $G(50, 0.3)$ . . . 5

3.3.2 Analiza izboljšanih algoritmov na grafih $G(50, 0.3)$ . . . 7

3.3.3 Primerjava algoritmov na grafih s konstantno verjetnostjo p in spremenljivim številom vozlišč n . . . 8

3.3.4 Primerjava algoritmov na grafih s konstantim številom vozlišč n in spremenljivo verjetnostjo p . . . 10

4 Zaključek **10**

Literatura **11**

1 Navodilo

Naloga je iskanje največje neodvisne množice v grafu $G = (V, E)$ s pomočjo celoštevilskega 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. *Celoštevilski 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*

$$max < c, x >,$$

da bodo zadoščeni pogoji

$$Ax \leqslant b, x \geqslant 0,$$

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:

$$x_v + x_w \leqslant 1 \ za \ \forall vw \in E,$$

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

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

Največjo neodvisno množico v množici vseh neodvisnih podmnožic grafa $G = (V, E)$ bomo iskali s pomočjo celoštevilskega lineranega programiranja in lokalnega iskanja. **Lokalno iskanje** temelji na izbiri začetne neodvisne podmnožiče 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/maksimalno neodvisno 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 na 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 seznamov, 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.

3.2.1 Algoritem: $nakljucni\_MIS(G)$

Algoritem 1 $nakljucni\_MIS(G)$

1: $I \leftarrow \varnothing$
2: $\forall v \in V$ dobi vrednost $P(v) \in permutacija(V)$
3: **if** $P(v) < P(w)$ za $\forall w \in sosedi(v)$ **then**
4: $\quad I \leftarrow I \cup v$
5: $V' \leftarrow V \backslash (I \cup sosedi(I))$.
6: $E' \leftarrow E \backslash povezave(I)$.
7: return $I \cup MIS(G' = (V', E'))$

3.2.2 Algoritem: *lokalno_iskanje*(G, I)

Najprej definirajmo $\tau(v)$, v algoritmu imenovan *tightness*(v).

Definicija 4. *Naj bo* $G = (V, E)$ *in* $v \in S \subset G$. *Potem je:*

$$\tau(v) = |W|, \text{ kjer je } W = \{w | w \in (sosedi(v) \cap V \backslash S)\}$$

Algoritem *lokalno_iskanje*(G, I) sprejme graf G in neko maksimalno neodvisno množico v grafu G. Nato vsakemu vozliscu $v \in I$ priredi mnozico $L(v) = \{w \in sosedi(v) | \tau(w) = 1\}$. Če je ta množica več kot, ali pa dvo elementna, potem v vseh možnih parih $(x, y) \in L(v) \times L(v)$, ki zadoščajo $x \neq y$, poišče prvega, da velja $y \notin sosedi(x)$, ko najde prvi tak par iz množice I odstrani vozlišče v, in množici I doda vozlišči x in y. Algoritem vrne maksimalno neodvisno množico I, ko ne obstaja $z \in I$, da bi veljalo $|L(z)| \geqslant 2$.

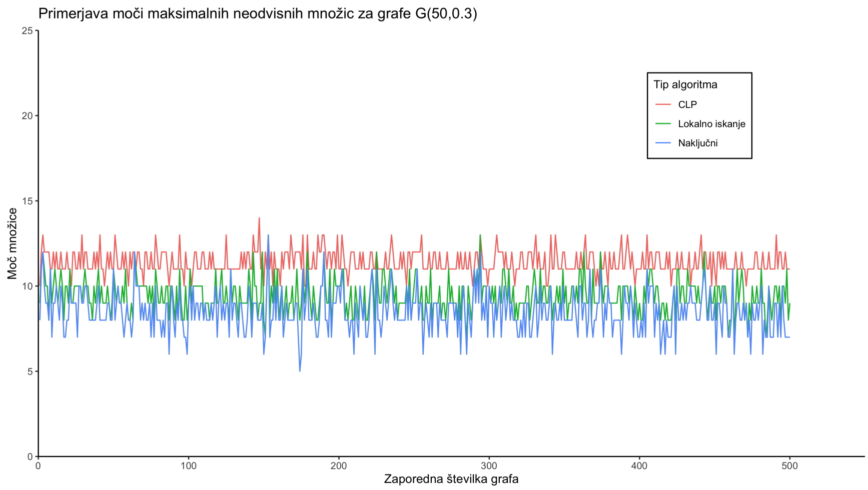
```
import networkx as nx

def tightness(graf, mnozica, v):
    t = 0
    for w in graf.neighbors(v):
        if w in mnozica:
            t += 1
        else:
            pass
    return t

def lokalno_iskanje(G, I):
    V = list(G.nodes)
    for v in I:
        L = []
        for w in G.neighbors(v):
            if G.neighbors(v) != []:
                if tightness(G, I, w) == 1:
                    L.append(w)
                else:
                    pass
            else:
                pass
        while len(L) >= 2:
            for v1 in L:
                L.remove(v1)
                for w1 in L:
                    if w1 not in G.neighbors(v1):
                        I.remove(v)
                        I = I + [v1, w1]
                        break
                    elif w1 in G.neighbors(v1):
                        pass
                if w1 not in G.neighbors(v1):
                    break
            if w1 not in G.neighbors(v1):
                break
    return I
```

3.3 Analiza rezultatov

3.3.1 Analiza algoritmov na grafih $G(50, 0.3)$

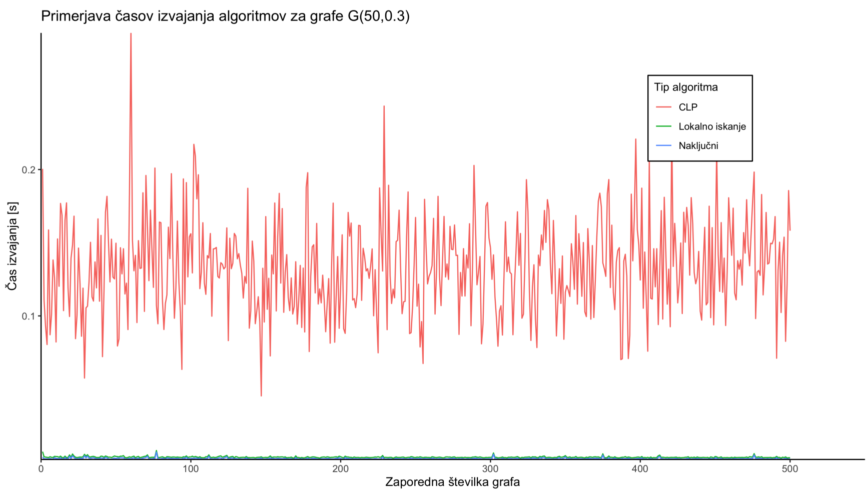
Najprej bomo primerjali algoritme na Erdős-Rényijevih $G(50, 0.3)$. Na tak način sva generirala 500 grafov in na njih izvedla algoritma $CLP(G)$, $nakljucni\_MIS(G)$, za vsak graf pa sva slednjega poskušala izboljšati še z algoritmom $lokalno\_iskanje(G, I)$, ki poleg grafa G sprejme še $I = nakljucni\_MIS(G)$.

Opazimo, da neodvisne množice največjih moči najde CLP.



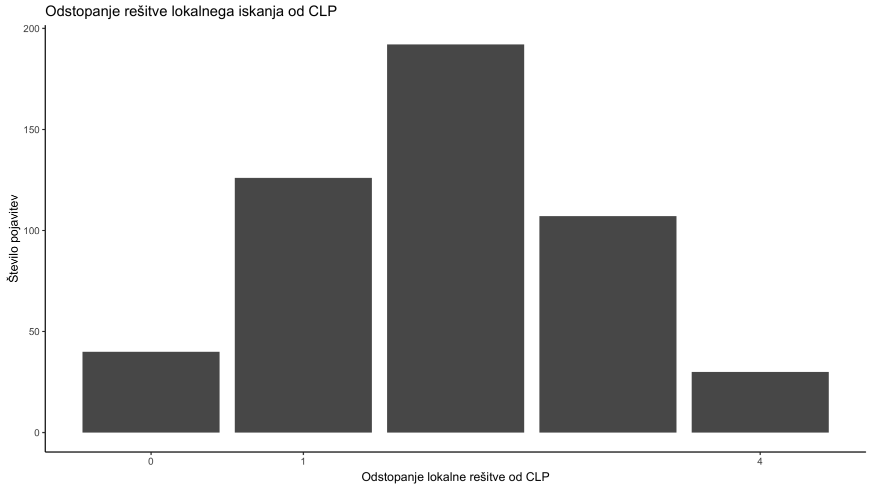
Slika 1: Moči neodvisnih množic za grafe $G(50, 0.3)$

Kljub temu, da je, za generirane grafe, CLP najpogosteje vrnil najboljšo rešitev, lahko v naslednjem grafu vidimo njegovo slabost, CLP je občutno počasnejši od preostalih dveh algoritmov.



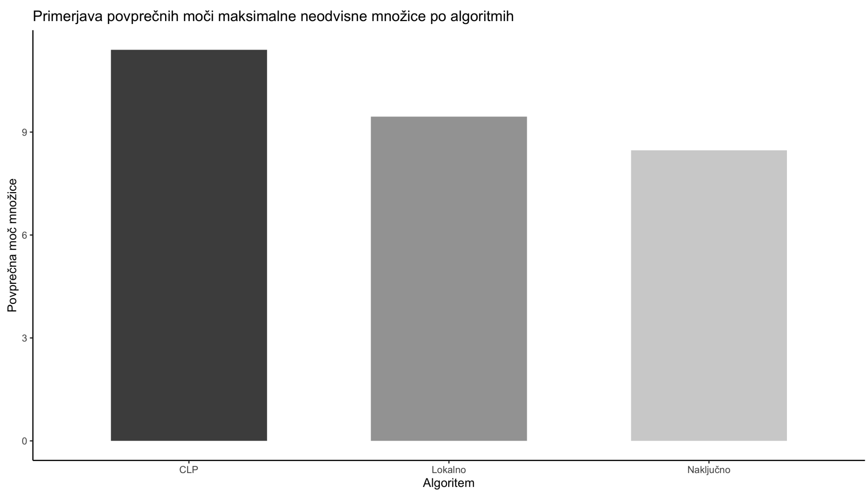
Slika 2: Časovne zahtevnosti algoritmov za $G(50, 0.3)$

Če si pogledamo še odstopanja rešitev *lokalnega_iskanja* od *CLP* opazimo, da je bila rešitev *lokalnega iskanja* največkrat za dve vozlišči manjša od rešitve *CLP*.



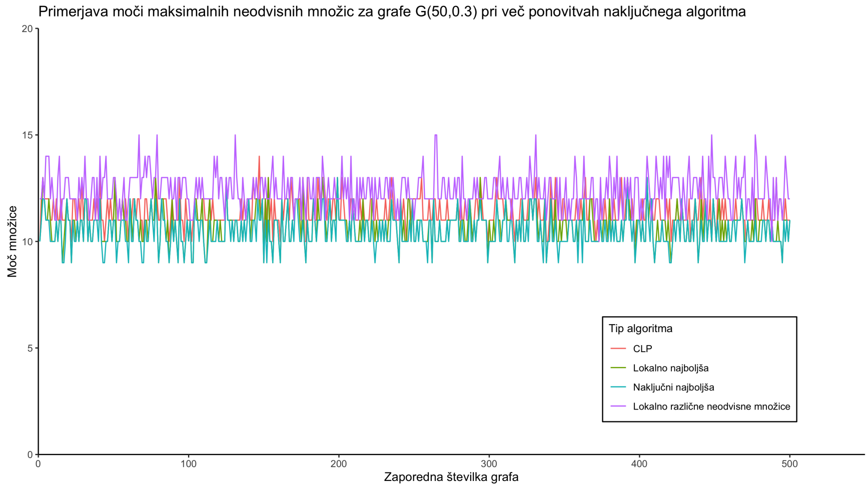
Slika 3: Odstopanja rešitev lokalnega iskanja do CLP.

Kot omenjeno, je maksimalne neodvisne množice z najvišjimi močmi vračal *CLP*, v povprečju so bile te množice velikosti 12. Druge največje množice je vračalo *lokalno iskanje*, katerega množice so imele v povprečju 9 vozlišč, za eno več kot od povprečja moči neodvisnih množic pridobljenih z algoritmom *nakljucni_MIS*. Rezultati so vidni na spodnjem grafu.

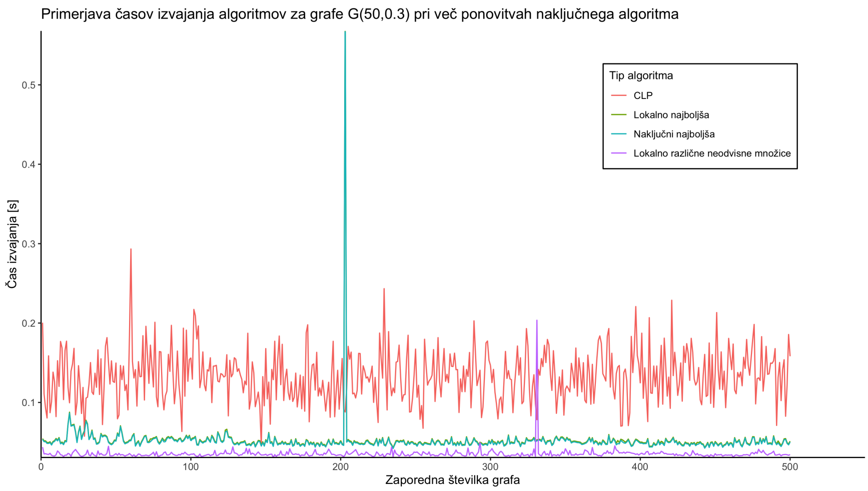


Slika 4: Povprečne vrednosti rezultatov posameznih algoritmov

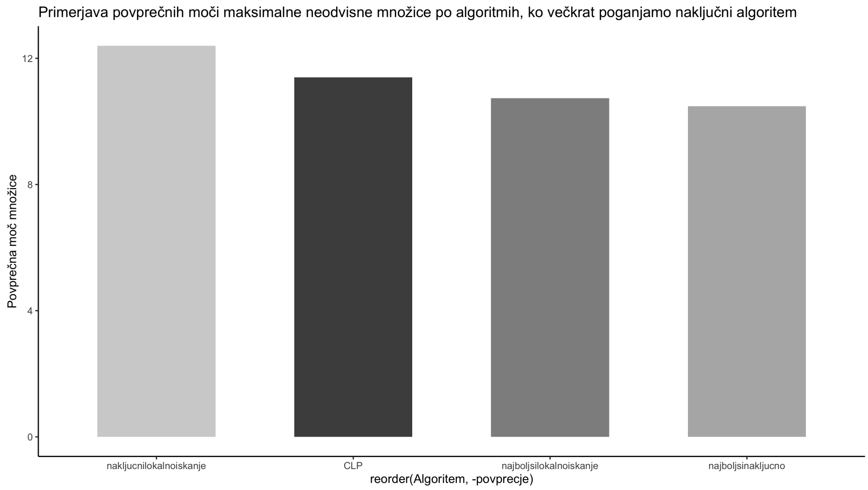
3.3.2 Analiza izboljšanih algoritmov na grafih $G(50, 0.3)$



Slika 5: Vrednosti rezultatov izboljšanih posameznih algoritmov



Slika 6: Časovne zahtevnosti izboljšanih posameznih algoritmov

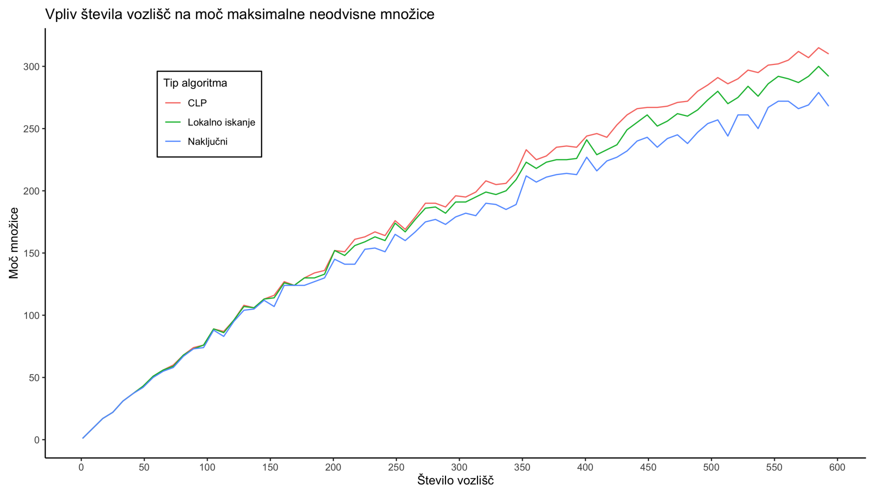


Slika 7: Povprečne vrednosti rezultatov izboljšanih posameznih algoritmov

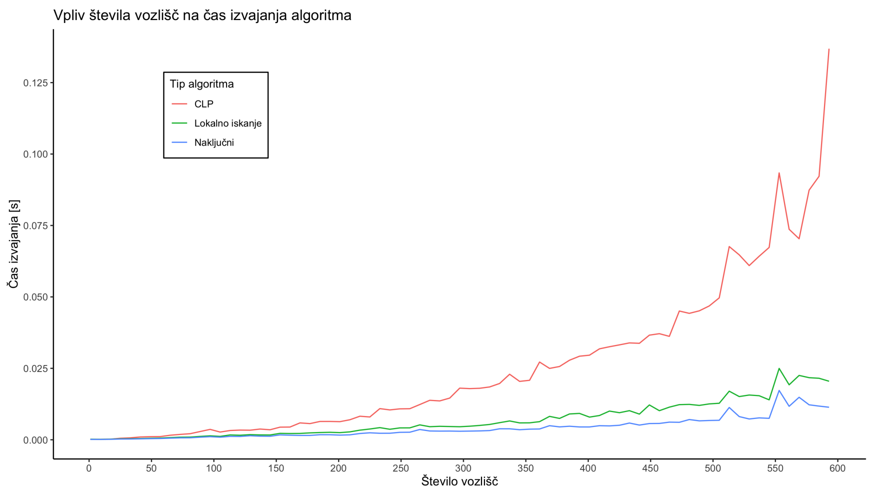
3.3.3 Primerjava algoritmov na grafih s konstantno verjetnostjo p in spremenljivim številom vozlišč n

Drugo skupino grafov sva, tako kot prvo, generirala z Erdős-Rényijevim $G(n, p)$ modelom, vendar pri tem spreminjala število vozlišč v grafih, verjetnost pa ohranila konstantno $p = 0.05$. Algoritme sva izvajala na grafih velikosti od 1 do 600 vozlišč.

Na spodnjem grafu si lahko ogledamo kako se, s povečevanjem števila vozlišč, spreminja moč maksimalnih neodvisnih množic. Pri konstantni verjetnosti p, se moč maksimalne neodvisne množice, v odvisnosti od moči množice vozlišč V, povečuje skoraj linearno, vendar moč maksimalnih neodvisnih množic dobljenih s CLP narašča hitreje od tistih pridobljenih z $nakljucni\_MIS$ in *lokalnim iskanjem*.



Slika 8: Velikosti neodvisnih množic v odvisnosti od števila vozlišč n



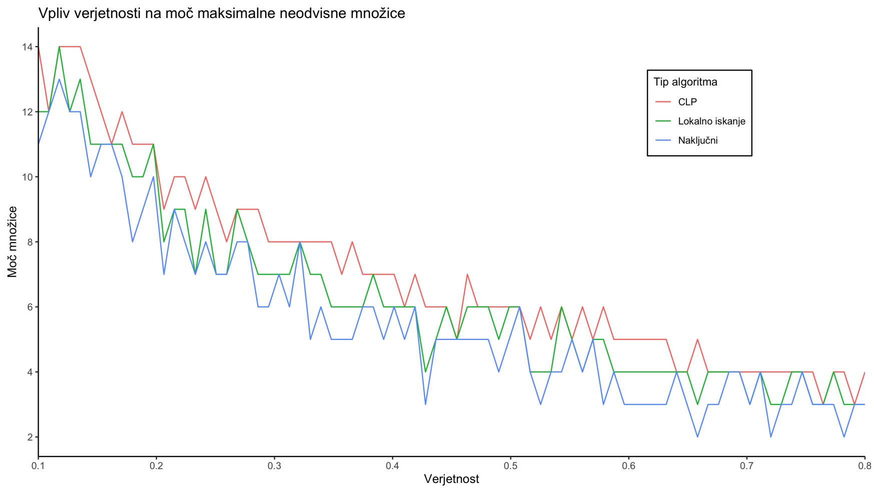
Slika 9: Spreminjanje časovne zahtevnosti algoritmov v odvisnosti od števila vozlišč n

Na zgornjem grafu ponovno opazimo največjo slabost iskanja maksimalne neodvisne množice s CLP, tj. časovno zahtevnost. Čas, porabljen za izvajanje CLP je, v odvisnosti od števila vozlišč v množici V, naraščala eksponentno. Prav tako je naraščala časovna zahtevnost algortima $nakljucni\_MIS$ in $lokalnega\_iskanja$, vendar so bila naraščanja, na opazovanih grafih, občutno manjša. Glede na to, da je iskanje maksimalne neodvisne množice NP-težek problem lahko pričakujemo, da bi z nadaljnjim povečevanjem števila vozlišč v množici V, časovna zahtevnost začela naraščati eksponentno tudi za zadnja dva algoritma. Zametki tega so opazni, ko gledamo časovne zahtevnosti algoritmov $nakljucni\_MIS$ in $lokalnega\ iskanja$ za največje moči množice V.

3.3.4 Primerjava algoritmov na grafih s konstantim številom vozlišč n in spremenljivo verjetnostjo p

Tudi zadnjo skupino grafov sva generirala z Erdős-Rényijevim $G(n, p)$ modelom, pri tem pa ohranila konstantno velikost množice vozlišč V in spreminjala verjetnosti p. Vrednosti verjetnosti sva dobila s klicom funkcije $numpy.linspace(0.1, 0.8, 80)$.

Če je, s povečevanjem velikosti množice V, naraščala tudi velikost maksimalne neodvisne množice, opazimo, da je v primeru povečevanja verjetnosti p vzorec nasproten, saj velikost maksimalne neodvisne množice pada, ko povečujemo verjetnost p in posledično število povezav v grafu. Torej velikost maksimalne neodvisne množice ni odvisna le od števila vozlišč v grafu, ampak tudi od števila povezav. Ugotovitve so opazne na spodnjem grafu.

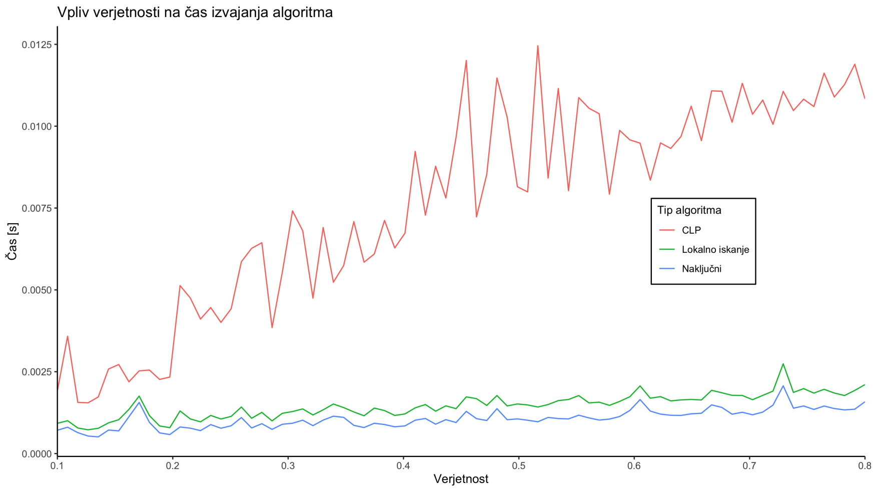


Slika 10: Vpliv verjetnosti na velikost neodvisnih množic

Tako kot velikost maksimalne neodvisne množice, je od števila povezav v grafu (oz. verjetnosti p), odvisna tudi časovna zahtevnost algoritmov. Časovna zahtevnost algoritmov ob povečevanju števila povezav v grafu se obnaša podobno, kot se je obnašala pri povečevanju vozlišč. Časovna zahtevnost vseh treh algoritmov narašča skupaj s verjetnostjo p, vendar pri CLP hitreje kot pri $nakljucni\_MIS$ in $lokalnim\ iskanjem$.

4 Zaključek

Iz analize rezultatov lahko zaključimo, da v primeru generiranja grafov z Erdős-Rényijevim $G(n, p)$ modelom, na velikost maksimalne neodvisne množice v danem grafu vplivata tako število vozlišč v grafu, kot tudi število povezav. Kot alternativo iskanju maksimalne neodvisne množice s CLP-jem, smo uspeli implementirati algoritem $nakljucni\_MIS$ in ga kasneje izboljšati z lokalnim iskanjem. Čeprav so rešitve CLP malenkost boljše v večini primerov, sta $nakljuci\_MIS$ in $lokalno\ iskanje$ boljša za grafe z večjim številom vozlišč, ali pa z večjim številom povezav, saj sta manj časovno zahtevna.



Slika 11: Vpliv verjetnosti na časovno zahtevnost algoritmov

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)`.