Природно-математичи факултет, Универзитет у Бањој Луци

студијски програм Математика и информатика

смјер Информатика

Предмет: Операциона истраживања

Тема: The Minimum Weakly Connected Dominating Set Problem

Студенти:

Сара Талић и

Никола Ђајић

Предметни професор:

др Марко Ђукановић, доц.

септембар 2023.

Садржај

1. Увод

У овом семинарском раду бавићемо се проблемом The Minimum Weakly Connected Dominating Set Problem, скраћено MWCDS проблем.

Weakly Connected Dominating Set распрострањено се користи за кластеровање мобилних бежичних ad hoc мрежа и у дистрибуираним сензорским мрежама.

Појам MWCDS се први пут помиње од стране Jean E. Dunbar-a 1997. године.

За почетак, упознаћемо се самом позадином проблема, те ћемо кроз наставак рада представити концепте рјешавања датог проблема кроз методе похлепног и генетског алгоритма, као и кроз PuLP рјешавач.

У завршници приложићемо наше експерименталне резултате датог проблема, кроз споменуте методе, те дати наш закључак.

Имплементацију алгоритама можете видјети на https://github.com/NikolaDjajic/OI-git.

2. Опис проблема

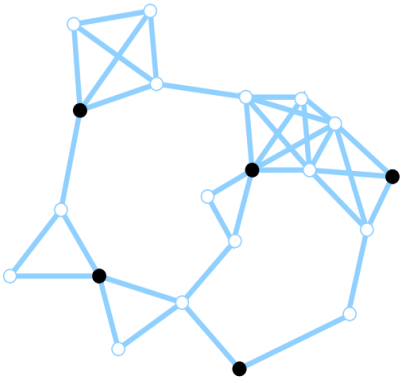
Прије него што дефинишемо шта заправо овај проблем обухвата, потребно је да се упознамо са неким основним појмовима о графовима.

Граф, једноставно, можемо дефинисати као скуп тачака (чворова или тјемена) које су међусобно повезане (не нужно све) линијама (ивицама или гранама).

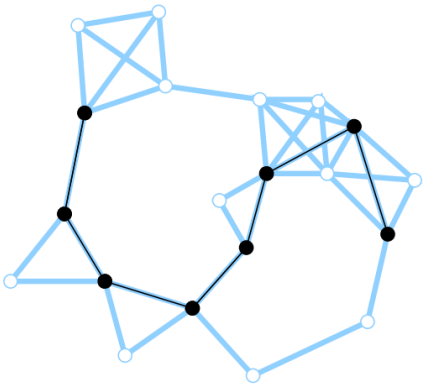
Посматраћемо повезан, неусмјерен граф $G = (V, E)$. Скуп V представља скуп чворова, док скуп E представља скуп грана датог графа. За граф G кажемо да је повезан уколико за било која два чвора из скупа чворова $(u, v) \in V$ постоји пут који почиње у чвору u, а завршава се у чвору v.

Дефинишимо доминантни скуп. Доминантни скуп је подскуп $S \subseteq V$, гдје сваки чвор графа $v \in V$ или припада доминантном скупу $v \in S$ или постоји грана $(u, v) \in E$ таква да $u \in S$. Ако је доминантни скуп повезан, онда га називамо повезаним доминантним скупом (енг. Connected Dominating Set), тј. ако за свако $(u, v) \in S$ постоји пут од u до v који пролази искључиво кроз чворове доминантног скупа.

Једноставније објашњено, циљ је да имамо скуп чворова од којих можемо доћи до свих осталих чворова који нису у доминантном скупу. Као што можемо да видимо у примјеру на Слици 2.1, преко чворова означених црном бојом, можемо да досегнемо до свих осталих чворова који нису у доминантном скупу (необојени чворови). Док на Слици 2.2 видимо примјер повезаног доминантног скупа.

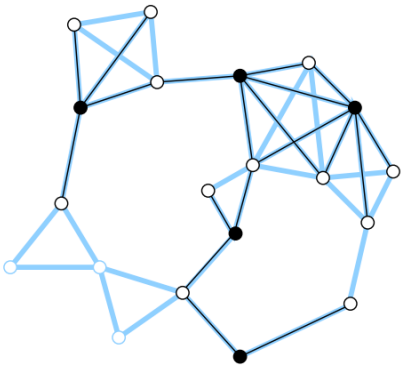


Слика 2.1 [4]



Слика 2.2 [4]

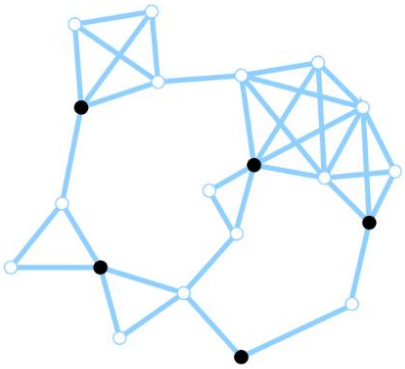
Са друге стране, постоји и слабо повезани доминантни скуп (енг. Weakly Connected Dominating Set). Подграф слабо индукован од стране W ($W \subseteq V$) се може записати као $S_w = (N[W], E \cap (W \times N[W]))$. N[W] садржи чворове удаљене један корак од W и себе. Примјер таквог подграфа можемо видјети на Слици 2.3.



Слика 2.3 [4]

Доминантни скуп W је слабо повезан доминанти скуп уколико је подграф слабо индукован од W повезан.

На Слици 2.4 приказан је примјер слабо повезаног доминантног скупа. За разлику од повезаног доминантног скупа, са слике видимо да чворови доминантног скупа не морају нужно бити директно повезани.



Слика 2.4 [4]

Наш задатак јесте да пронађемо такав слабо повезан доминантни скуп, али минималан.

3. Похлепни алгоритам

Похлепни алгоритам представља програмску парадигму која се заснива на итеративном рјешавању проблема, тј. корак по корак. Током сваке итерације потребно је да забиљежимо оптимално најбољи избор.

Користи се за рјешавање тежих оптимизационих проблема. Међутим, потребно је бити опрезан јер се многи проблеми и не могу оптимално ријешити путем похлепног алгоритма. Али, тада похлепни приступ може да нам послужи ради добијања приближних рјешења или полазних рјешења која ће се другим методама унаприједити.

Неки од примјера коришћења похлепног алгоритма су:

- Дајкстрин алгоритам,
- Хафманов алгоритам,
- Крускалов алгоритам,
- Примов алгоритам и многи други.

3.1 Похлепни алгоритам за MWCDS

Размотрићемо идеју за реализацију нашег проблема минималног слабо повезаног доминантног скупа путем похлепног алгоритма.

Корак 1

Почетни чвор, од којег крећемо рјешавање нашег проблема, бирамо по принципу чвора са највећим бројем сусједа у графу, тј. почетни чвор ће бити чвор највећег степена у графу. Овај чвор смо означили као тренутни чвор.

Корак 2

Потребно је да дефинишемо три листе:
Листа за чворове доминантног скупа,
Листа означених чворова, у коју спадају доминантни чворови и њихови сусједи и
Листа која ће представљати чворове за обилазак, тачније листу чворова међу којима се налазе потенцијална рјешења проблема.

Почетни чвор додајемо у доминанти скуп, као и у скуп означених чворова. Све сусједе почетног чвора додајемо и у листу означених и у листу за обилазак.

Додавањем сусједа у листу за обилазак обезбјеђујемо повезаност изабраних чворова у коначном рјешењу.

Корак 3

Постојаће петља у којој ће се више-мање понављати претходни кораци.

Петља ће се одвијати све док постоје чворови у листи за обилазак, када наведена листа постане празна петља се прекида.

Унутар петље прво проналазимо нови тренутни чвор. Нови тренутни чвор биће чвор највећег степена унутар листе за обилазак. Тренутни чвор ћемо потом обрисати из листе обилазака, јер смо га управо „обишли“.

Потом, тражимо листу сусједа тренутног чвора. Ако постоје сусједи тренутног чвора који нису у листи означених, додаћемо их у листу, као и у листу за обилазак.

На крају, ако тренутни чвор није доминантан, или неко од његових сусједа није доминантан, тренутни чвор ћемо додати у листу доминантних.

4. Генетски алгоритам

Генетски алгоритам припада хеуристичким методама (методе које не гарантују оптимално рјешење). Инспирација за овај алгоритам лежи у Теорији еволуције Чарлса Дарвина.

Неколико појмова требало би да дефинишемо када причамо о генетском алгоритму:

- Јединка = свака јединка је могуће рјешење проблема.
- Популација = скуп јединки који представља подскуп простора претраге.
- Прилагођеност (енг. fitness) = оцјена квалитета дате јединке.
- Генетски код = начин на који се представља једна јединка.
- Гени = дијелови генетског кода.

Такође, постоје и генетски оператори, а то су:

- Селекција

Овим процесом се бирају јединке над којима ће се послије вршити укрштање и мутација. Јединке се бирају на основу индивидуалне прилагођености, оне са већом прилагођеношћу су у предности да буду одабране за наставак.

Неке од начина селекције су рулетска, турнирска, елиминациона и друге.

- Укрштање

Слично као и у самој еволуцији, укрштање представља мијешање гена јединки. Као резултат ове операције добијамо нову јединку која носи потенцијално добре гене својих родитеља.

- Мутација

Овај процес представља мијењање гена. Тиме се омогућава враћање добрих гена који су се изгубили приликом претходних радњи.

Посљедица извршавања претходно дефинисаних радњи јесте настанак нове генерације.

4.1 Генетски алгоритам за MWCDS

Размотрићемо идеју за реализацију MWCDS проблема путем генетског алгоритма.

Корак 1

Креирамо почетну популацију. Насумично ћемо бирати чланове популације. На почетку, сваки чвор има 50% шансе да се пронађе у свакој јединци.

Корак 2

Потом, ту популацију сортирамо растуће на основу прилагођености. Прилагођеност се мјери на основу броја чворова, уколико је та јединка потенцијално рјешење за слабо повезан доминантни скуп. У супротном, дајемо јој велику оцјену (конкретно у нашем случају 1000).

Корак 3

Техника селекције коју користимо је елитизам, што подразумијева да двије најбоље јединке из сваке популације аутоматски прелазе у нову генерацију.

Корак 4

Због двије елитне јединке додане у претходном кораку, улазимо у for петљу у опсегу до (укупног броја популација умањеног за два)/2. Затим, унутар наведене петље бирамо рулетском селекцијом два родитеља из популације, и над њима вршимо операцију укрштања. Кориштено је једнопозиционо укрштање.

Процесом укрштања добијамо двије нове јединке над којима вршимо операције мутације. Што се тиче мутације, она се врши на основу унапријед задане вјероватноће мутације, задане при покретању програма. Након извршене мутације, добијене двије јединке додајемо у нову генерацију.

Корак 5

Послије извршавања for петље из корака број 4, тренутна популација заправо ће бити изведена нова генерација.

Корак 6

Кораци почев од 2 до 5 понављају се унутар while петље, која зависи од унапријед заданог временског интервала.

5. PuLP

PuLP алат представља оптимизациони рјешавач. Уз помоћ овог алата у могућности смо да рјешавамо LP, ILP и MILP моделе. Овај алат се позива у програмском језику Python-у.

Прије коришћена датог алата, потребно је увести га, а то ћемо извршити сљедећом наредбом

```
from pulp import *.
```

Модел пишемо по јасно дефинисаним правилима која ћемо размотрити конкретно кроз наш примјер минималног слабо повезаног доминантног скупа у секцији 5.2.

5.1 ILP модел за MWCDS

Прије него што коментаришемо PuLP за MWCDS , потребно је дефинисати ILP модел истог.

Dangdang Niu и Minghao Yin су дефинисали ILP модел за MWCDS у свом дјелу „A Self stabilizing Memetic Algorithm for Minimum Weakly Connected Dominating Set Problems“, те ћемо сада исти да размотримо.

Сам граф G биће представљен на сљедећи начин:

V = скуп свих чворова графа,

E = скуп свих грана графа,

E_{weak} = дводимензионална матрица сусједства, за коју вриједи

$$\mathrm{E_{weak}[u,v]} = \begin{cases} 1, & \text{ако за } u, v \in V \text{ , постоји грана } uv \\ 0, & \text{ако не постоји грана} \end{cases}$$

Затим, дефинисаћемо двије варијабле:

$$x_{uv} = \begin{cases} 1, & \text{ако је } (u,v) \in E \text{ грана одабрана за оптимално рјешење} \\ 0, & \text{уколико није} \end{cases}$$

$$y_t = \begin{cases} 1, & \text{ако је чвор } t \in V \text{ одабран за оптимално рјешење} \\ 0, & \text{уколико није} \end{cases}$$

Овај модел је проблем минимизације, те је функција циља:

$$1 + min \sum_{u,v \in V} x_{uv}$$

Са сљедећим ограничењима:

$$x_{uv} \leq E_{weak}[u,v] \qquad \forall u,v \in V \quad (1)$$

$$x_{uv} = x_{vu} \qquad \forall u,v \in V \quad (2)$$

$$y_u + y_v \geq 2 * x_{uv} \qquad \forall u,v \in V \quad (3)$$

$$\sum_{u,v \in S} x_{uv} \leq |S| - 1 \quad (4)$$

$$\forall S \subseteq V$$

$$\sum_{u,v \in V} x_{uv} = \sum_{t \in V} y_t - 1 \quad (5)$$

$$\sum_{t \in N[V]} y_t \geq 1 \quad (6)$$

Прва два ограничења (1) и (2) нам осигуравају да је скуп одабраних грана у сваком тренутку подскуп грана E_{weak} скупа. Треће ограничење (3) нам показује везу између грана и чворова у оптималном рјешењу, тачније за сваку одабрану грану, крајњи чворови морају бити одабрани. Четврто ограничење (4) нам гарантује да подграф није циклус, док заједно са петим ограничењем (5) чини да је подграф стабло.
И на крају, шесто ограничење (6) нам омогућава доминантност, тачније, за сваки чвор барем један његов сусједни чвор мора бити одабран.

5.2 PuLP за MWCDS

У овој секцији ћемо дефинисати правила за превођење модела којег смо описали у секцији 5.1.

Првенствено, креирање нашег модела вршимо сљедећим кодом:

```
model = LpProblem("MinimumWeaklyConnectedDominatingSet", LpMinimize).
```

Наш проблем је проблем минимизације, те смо зато користили аргумент LpMinimize.

Сљедећи корак је дефинисање наших варијабли:

```
x = LpVariable.dicts("x", Eweak, 0, 1, LpBinary)

y = LpVariable.dicts("y", V , 0, 1, LpBinary).
```

Обје наше варијабле су бинарне, односно могу имати вриједност или 0 или 1, што смо и написали приликом дефинисања истих.

Сада можемо да имплементирамо нашу функцију циља на сљедећи начин:

```
model += 1 + lpSum( x [ (u, v) ]  for u in V for v in V ).
```

И за крај, имплементираћемо сва ограничења проблема, која смо описали у секцији 5.1:

```
# ogranicenja 1, 2 i 3
for u in V:
    for v in V:
        if u!=v:
            model += x[ (u, v) ] <= Eweak[u,v]
            model += x[ (u, v) ] == x[ (v, u) ]
            model += y[u] + y[v] >= 2*x[ (u, v) ]
```

```
# ogranicenje 4
for S in powerset(V):
    if len(S) > 1:
        model += lpSum( x[ (u, v) ] for u in S for v in S if u!=v) <= len(S)-1
```

```
#ogranicenje 5
model += lpSum( x[ (u,v) ] for u in V for v in V if u!=v) ==
lpSum( y[t] for t in V) - 1
```

```
#ogranicenje 6
for v in V:
    model += lpSum(y[t] for t in neighbors(v)) >= 1
```

Уз напомену да су функције `powerset()` и `neighbors()`, функције које смо дефинисали за проналажење подскупова скупа чворова V и сусједа одређеног чвора из скупа V.

6. Експериментални резултати

инстанца	*|V|*	*|E|*	d_{avg}	*Похлепни*		*Генетски*				*PuLP*	
				резултат	**вријеме**	**најбољи резултат**	**средњи резултат**	**девијација**	**вријеме**	**резултат**	**вријеме**
graf-10	10	18	1,8	5	0,001	3	3,0	0,0	0,101	2	0,206
graf-11	11	20	1,81	3	0,001	3	3,3	0,948	0,097	3	0,428
graf-12	12	40	3,3	4	0,001	3	3,3	0,479	0,135	3	0,803
graf-14	14	44	3	3	0,002	3	3,7	0,479	0,167	3	4,325
graf-17	17	38	2,2	7	0,000	7	7,4	0,510	0,206	4	69,708

Табела 1 – веома мале инстанце

инстанца	*|V|*	*|E|*	d_{avg}	*Похлепни*		*Генетски*				*PuLP*	
				резултат	**вријеме**	**најбољи резултат**	**средњи резултат**	**девијација**	**вријеме**	**резултат**	**вријеме**
johnson8-2-4	28	420	15	3	0,010	3	3,0	0,0	0,387	-	600,0
MANN-a9	45	918	40	3	0,005	2	2,0	0,0	0,873	-	600,0
hamming6-4	64	704	22	12	0,005	4	4,7	0,825	1,192	-	600,0
hamming6-2	64	2000	57	2	0,013	2	2,0	0,0	1,425	-	600,0
johnson8-4-4	70	2000	53	5	0,013	2	2,0	0,0	1,296	-	600,0

Табела 2 – мале инстанце

инстанца	*|V|*	*|E|*	d_{avg}	*Похлепни*		*Генетски*				*PuLP*	
				резултат	**вријеме**	**најбољи резултат**	**средњи резултат**	**девијација**	**вријеме**	**резултат**	**вријеме**
keller4	171	9K	110	9	0,114	2	2,9	0,735	7,304	-	600,0
c-fat200-5	200	8K	84	3	0,095	3	3,2	0,412	12,120	-	600,0
hamming8-4	256	21K	163	16	0,297	2	2,8	1,029	17,683	-	600,0
MANNa-27	378	71K	373	3	1,586	2	2	0,0	38,859	-	600,0
johnson32-2-4	496	108K	435	3	2,984	3	3,0	0,0	97,167	-	600,0

Табела 3 – средње инстанце

инстанца	*|V|*	*|E|*	d_{avg}	*Похлепни*		*Генетски*				*PuLP*	
				резултат	**вријеме**	**најбољи резултат**	**средњи резултат**	**девијација**	**вријеме**	**резултат**	**вријеме**
p-hat700-2	700	183K	347	16	4,929	7	8,6	1,507	427,372	-	600,0
keller5	776	226K	582	11	10,656	2	3,2	0,632	243,315	-	600,0
brock800-3	800	207K	518	9	9,652	5	5,5	0,529	342,380	-	600,0
MANNa-45	1000	533K	1000	3	39,845	2	2,0	0,0	476,780	-	600,0
phat1500-1	1500	285K	379	21	22,348	19	26,5	7,059	600,0	-	600,0

Табела 4 – велике инстанце

У претходним табелама приказани су експериментални резултати наших алгоритама.

Веома мале инстанце су графови до 20 чворова, мале инстанце су графови од 20 до 100 чворова, средње инстанце су од 100 до 500 чворова и велике су са више од 500 чворова.

Мале, средње и велике инстанце кориштене су са https://networkrepository.com/dimacs.php.

Објашњење табела:

|V| = представља укупан број чворова графа,

|E| = представља укупан број грана графа,

d_{avg} = представља просјечан степен чвора у графу,

резултат = укупан број чворова у минималном слабо повезаном доминантом скупу и

вријеме = вријеме потребно за извршавање програма изражено у секундама.

Експерименти су вршени на рачунару са спецификацијама:

Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz

RAM 16.0 GB (15.7 GB usable)

64-bit operating system, x64-based processor.

Похлепни алгоритам и PuLP су извршавани једанпут, те смо биљежили добијени резултат, као и вријеме потребно да се реализује рјешење.

Са друге стране, генетски алгоритам је извршаван десет пута. Генетски алгоритам је временски био ограничен на десет минута као и PuLP рјешавач. Биљежили смо најбољи резултат, те смо пронашли и просјечни резултат за свих десет извршавања. Вријеме извршавања представљено код генетског алгоритма је просјечно вријеме извршавања свих десет покушаја. Такође, рачунали смо и стандардну девијацију по сљедећој формули:

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - s)^2}{n-1}}$$

гдје

X_i представља і-ти резултат извршавања,

S представља средњи резултат и

n представља укупан број извршавања.

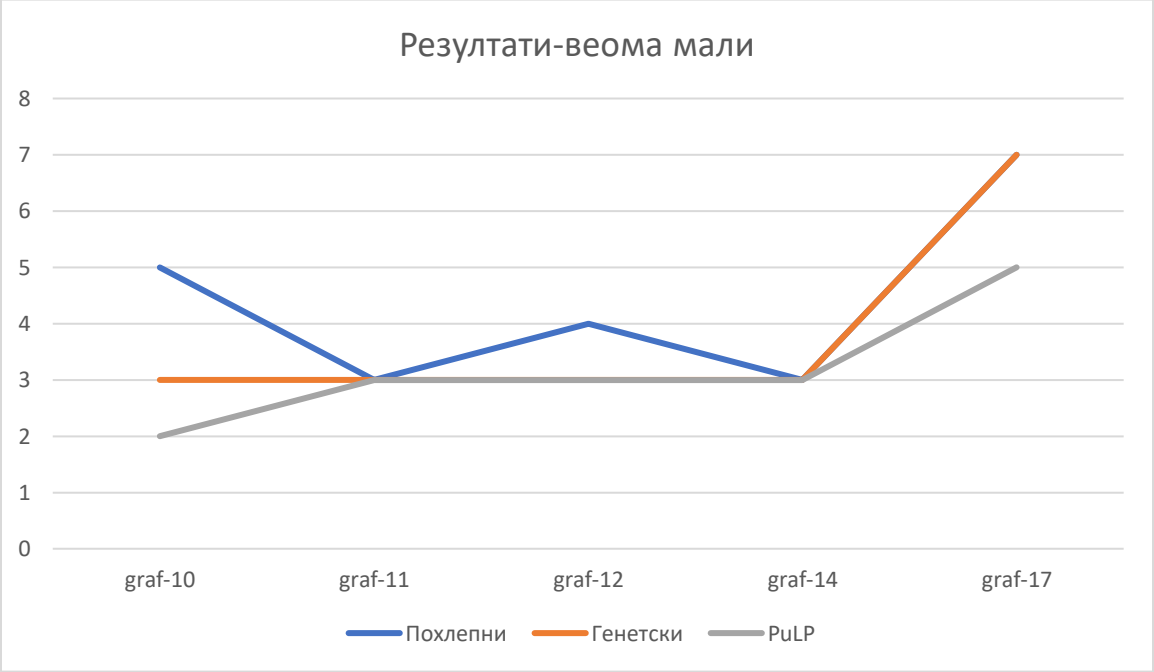
Као што можете да видите, PuLP рјешавач није пронашао резултате код малих, средњих и великих инстанци у временском оквиру од десет минута.

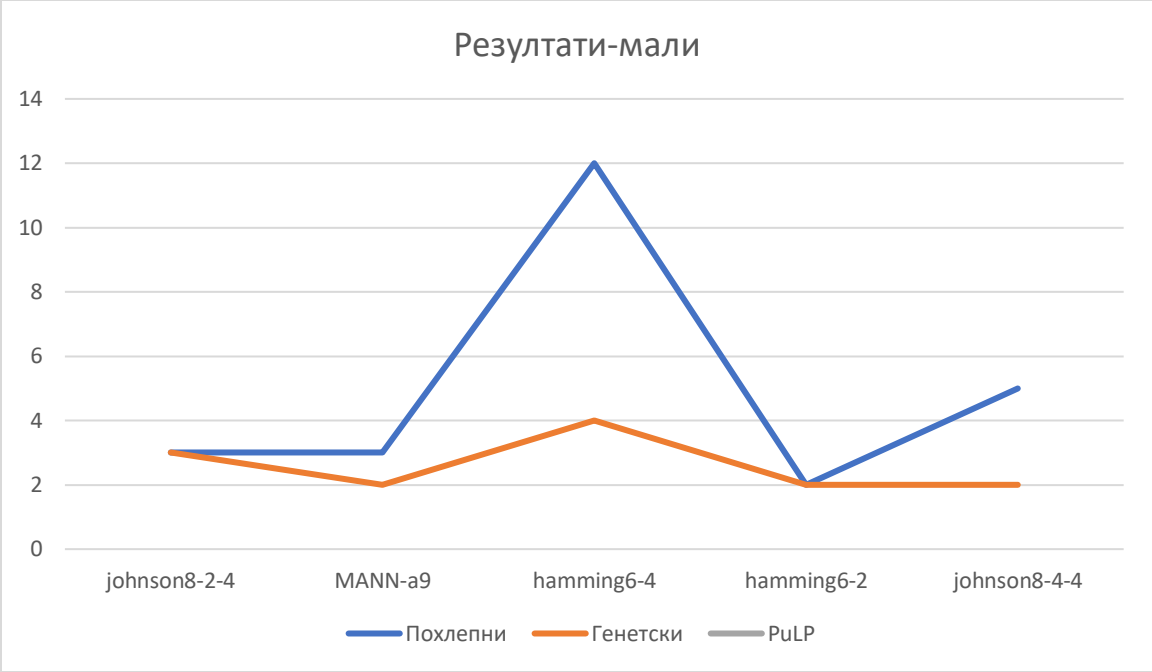
Зашто нам се то десило? Наиме, ако погледамо ограничење (4) за ILP модел нашег проблема у секцији 5.1 (страна 10), можете да примијетите да је потребно да прођемо кроз све подскупове скупа чворова. Ако знамо да је број подскупова једног скупа са n чланова 2^n, за граф са нпр. 28 чворова (мали граф) постоји 268435456 поскупова, што је огроман број, те у овом кораку долази до великог временског застоја и немогућности извршења програма у задатом року. Код мањих графова, као што можемо видјети у Табели 1, извшавање је успјешно.

6.2 Графички приказ резултата

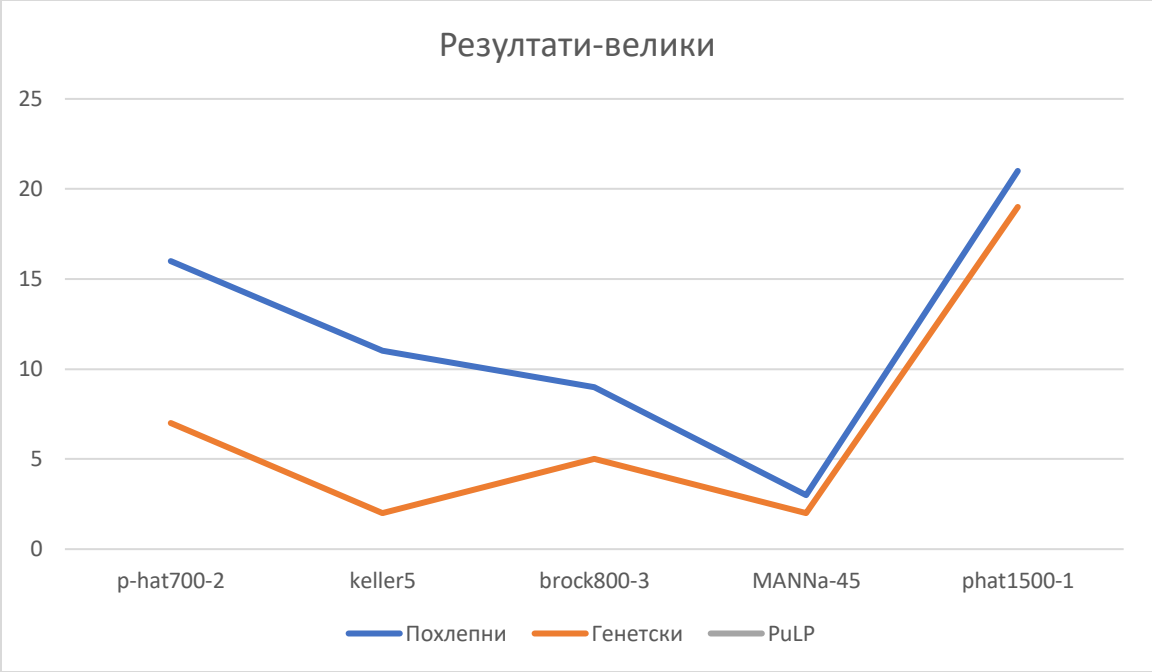
У овом дијелу приказани су графички прикази добијених резултата. Посебно ћемо посматрати за веома мале, мале, средње и велике графове. Прва четири графика представљају поређења добијених резултата. За генетски алгоритам посматрамо најбољи резултат.

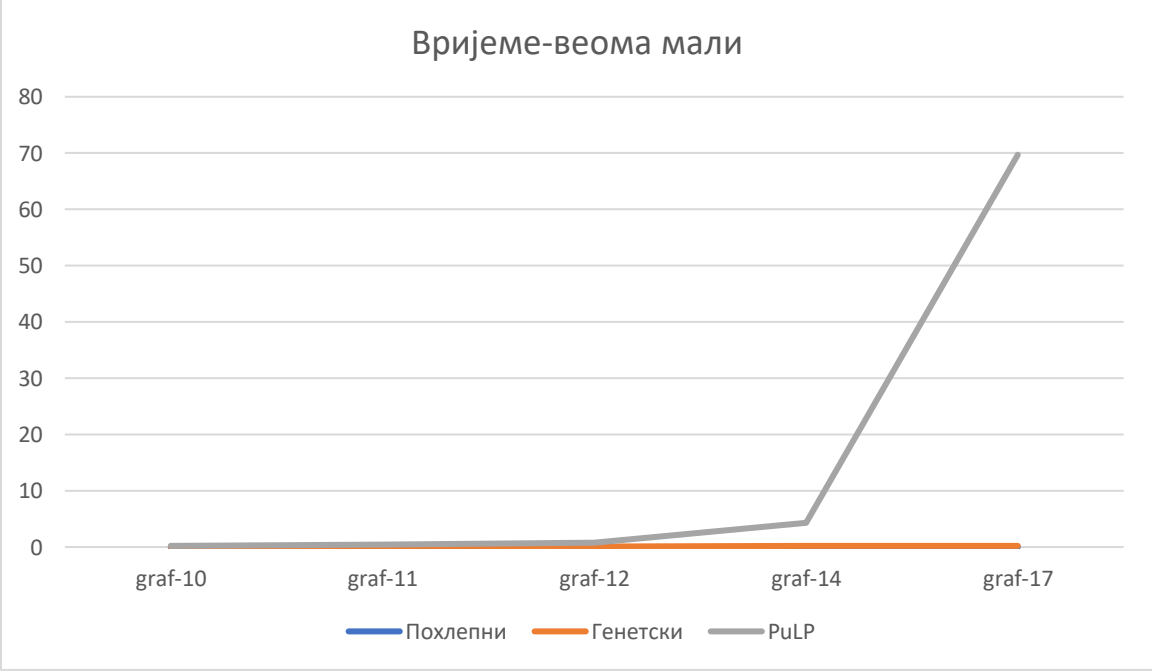
Остала четири графика представљају поређење времена извршавања изражено у секундама.

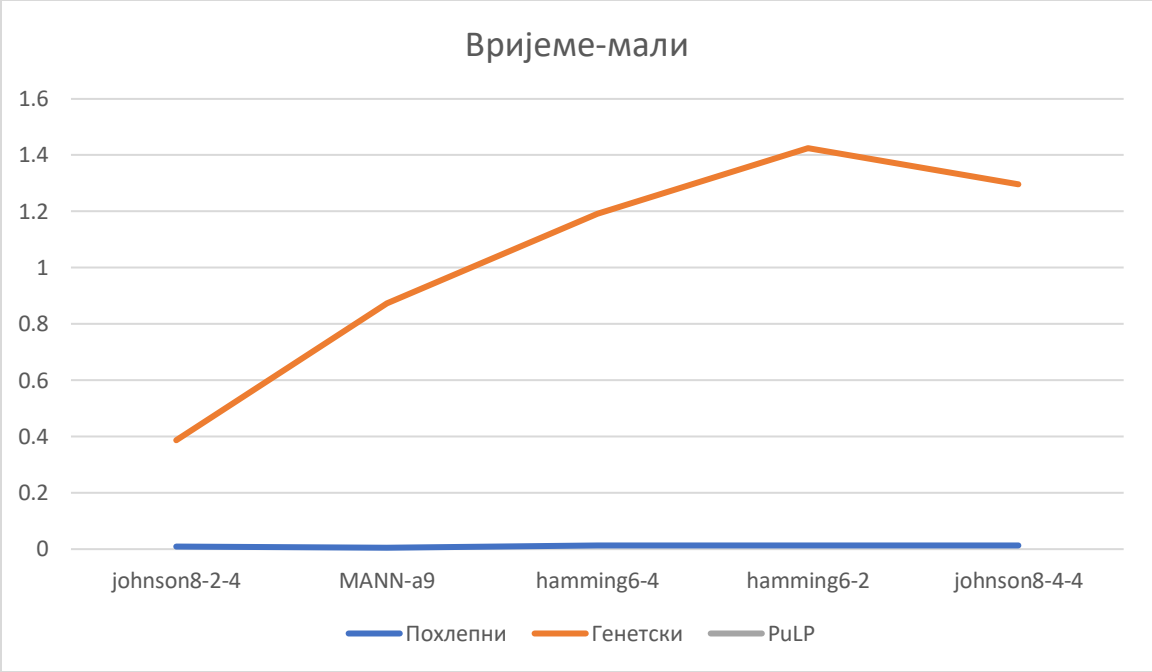


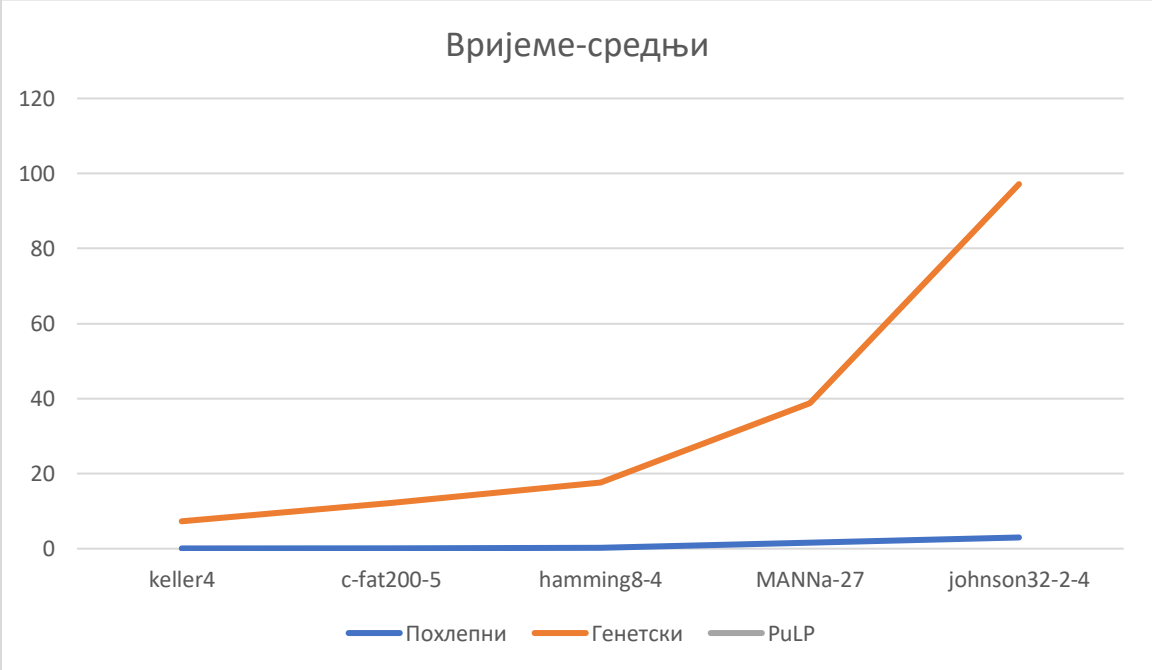

Резултати-мали
14
12
10
8
6
4
2
0
johnson8-2-4
MANN-a9
hamming6-4
hamming6-2
johnson8-4-4
Похлепни
Генетски
PuLP

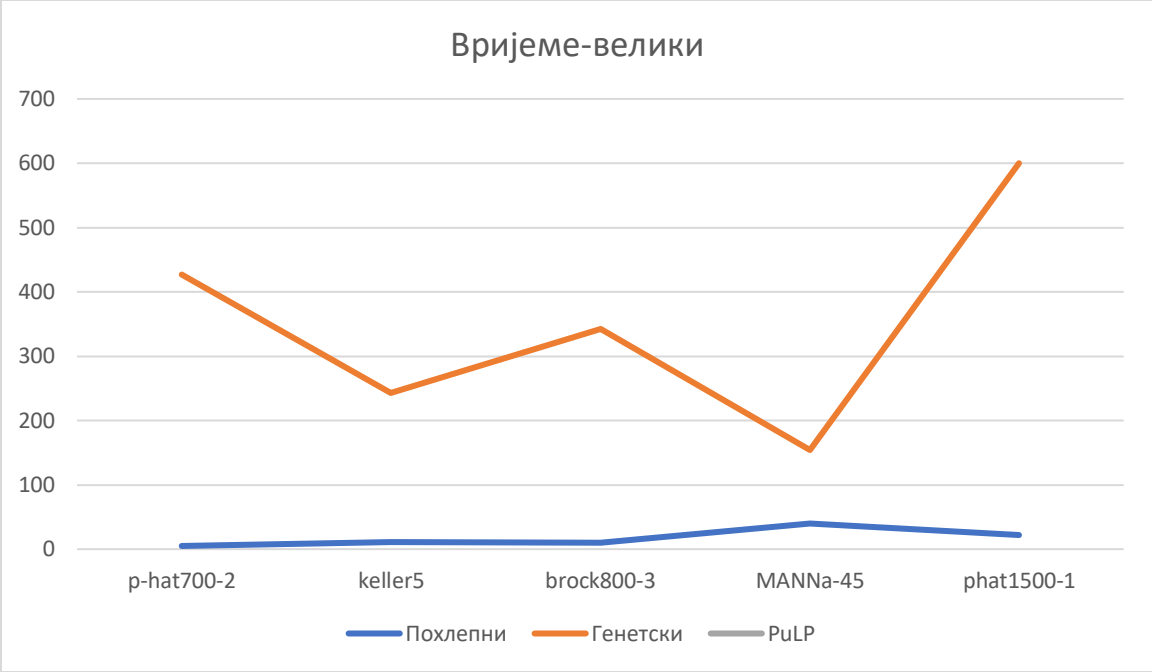

Резултати-средњи
18
16
14
12
10
8
6
4
2
0
keller4
c-fat200-5
hamming8-4
MANNa-27
johnson32-2-4
Похлепни
Генетски
PuLP


Резултати-велики
25
20
15
10
5
0
p-hat700-2
keller5
brock800-3
MANNa-45
phat1500-1
Похлепни
Генетски
PuLP


Вријеме-веома мали
80
70
60
50
40
30
20
10
0
graf-10
graf-11
graf-12
graf-14
graf-17
Похлепни
Генетски
PuLP


Вријеме-мали
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
johnson8-2-4
MANN-a9
hamming6-4
hamming6-2
johnson8-4-4
Похлепни
Генетски
PuLP


Вријеме-средњи
120
100
80
60
40
20
0
keller4
c-fat200-5
hamming8-4
MANNa-27
johnson32-2-4
Похлепни
Генетски
PuLP


Вријеме-велики
700
600
500
400
300
200
100
0
p-hat700-2
keller5
brock800-3
MANNa-45
phat1500-1
Похлепни
Генетски
PuLP

7. Закључак

Проблем који смо анализирали јесте NP тежак проблем.

Много је фактора који утичу на рјешења и вријеме извршавања, број чворова графа, повезаност графа, број грана и многи други.

У зависности од свих фактора сваки вид рјешавања овог проблема, од наша три, имаће своје предности и мане.

Очекивано, похлепни алгоритам нам је најбрже од свих давао резултате, али, такође, понекад веома незахвалне резултате. Код мањих инстанци видимо да може да парира како генетском алгоритму, тако и PuLP рјешавачу. Како се графови повећавају тако и похлепни алгоритам полако даје слабије резултате у односу на друге. Оно што можемо да закључимо јесте да код слабије повезаних графова, код оних чији је средњи степен мањи, похлепни је знатно лошији од генетског алгоритма. Код јаче повезаних графова можемо рећи да веома солидно рјешава наш проблем.

Генетски алгоритам показује се као бољи избор него похлепни. Покретањем десет пута били смо у могућности да видимо благе осцилације у рјешењима. Што се тиче временских трајања извршавања доста је дуже потребно времена него за похлепни алгоритам, што је, донекле, и очекивано.

Нажалост, остаје велики недостатак усљед немогућности извршавања PuLP-а. Код веома малих графова видјели смо да уз помоћ PuLP-а добијамо најбоље резултате. Вјерујемо да би се PuLP и са већим графовима истакао својим резултатима.

8. Литература

[1] Ђукановић М. , Матић Д. , Увод у операциона истраживања, 2022. година

[2] Dangdang N. , Minghao Y. , A Self-stabilizing Memetic Algorithm for Minimum Weakly Connected Dominating Set Problems, https://hsi-workshop.github.io/hsi-2022-camera-ready/A_Self-stabilizing_Memetic_Algorithm_for_Minimum_Weakly_Connected_Dominating_Set_Problems.pdf

[3] Dangdang N. , Xiaolin N. , Lilin Z. , Hongming Z. , Minghao Y. , A greedy randomized adaptive search procedure (GRASP) for minimum weakly connected dominating set problem, 2022. година, https://www.sciencedirect.com/science/article/abs/pii/S0957417422023569

[4] Chen Y. , Liestman A. , Clustering Algorithms for Ad Hoc Wireless Networks, 2004. година, https://www.researchgate.net/publication/2915870_Clustering_Algorithms_for_Ad_Hoc_Wireless_Networks

[5] https://en.wikipedia.org/wiki/Dominating_set

[6] https://en.wikipedia.org/wiki/Connected_dominating_set