Природно-математички факултет

Универзитет у Бањој Луци

смјер Информатика

Предмет: **Операциона истраживања**

Тема: **The Minimum Sum Coloring Problem**

Студенти:

Ивана Крминац

Радомир Тамбурић

Предметни професор:

др Марко Ђукановић, доц.

Садржај

Увод

Проблеми бојења графа представљају важан, популаран и веома истражен предмет у операционим истраживањима у пољу теорије графова. Један од најзанимљивијих проблема јесте Проблем минималне суме бојења (енг. Minimum Sum Coloring Problem, у даљем тексту MSCP).

Овај проблем представља изазов за истраживаче у областима рачунарства, математике и операционих истраживања. Има широк спектар примјена у различитим областима као што су распоређивање ресурса, распоређивање распореда рада и управљање транспортом.

Појам MSCP је први увео Supowit 1987. године користећи терминологију која није теорија графова, а први пут је овај проблем проучавала Ewa Kubicka 1989. године у својој докторској тези у теорији графова.

Кроз овај рад ћемо се детаљније упознати са проблемом и представити различите приступе за рјешавање овог проблема кроз методе похлепног и генетског алгоритма, и кроз PuLP рјешавач. На крају ћемо тестирати сваки од поменутих алгоритама над тестним инстанцама, те дати наш закључак.

Гитхаб репозиторијум можете пронаћи овде.

Опис проблема

Циљ MSCP је пронаћи најмањи могући збир бројева боја, који су додијељени чворовима графа, тако да немамо ни један пар сусједних чворова који су обојени истом бојом.

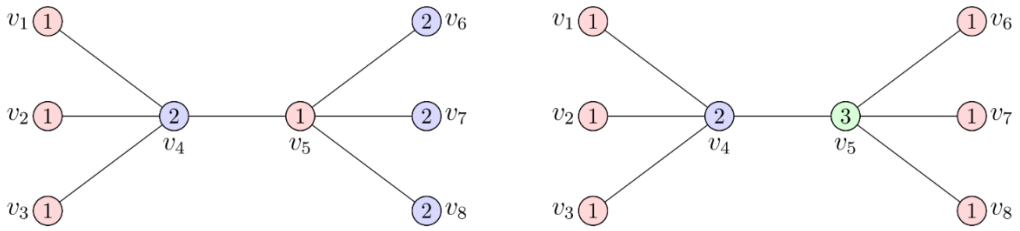
Формално, нека је G = (V, E) неусмјерен граф, гдје је V скуп чворова, а E скуп ивица графа G. Нека је правилно бојење C графа G партиција $\{V_1, \ldots, V_k\}$ од V у k независних подскупова тј. подскупова упарених несусједних чворова графа. Сви чворови који припадају подскупу V_i су обојени бојом i ($i \in \{1, \ldots, k\}$) тј. боја i је означена цијелим бројем i. Сума боја, бојења C, је дата функцијом:

$$f(C) = \sum_{i=1}^{k} i \cdot |V_i|.$$

Дакле, MSCP се своди на проналажење бојења C графа G са минималном вриједности функције f(C). Минимална вриједност се означава са Σ(G) и назива се хроматска сума (енг. chromatic sum). Број боја употребљен за бојење таквог графа се назива *јачина* (енг. strength) графа G означена са s(G). Са друге стране, минималан број боја са којима можемо да обојимо граф се означава са χ(G).

Такође, MSCP је *NP*-тежак проблем и уско повезан са класичним проблемом бојења графа.

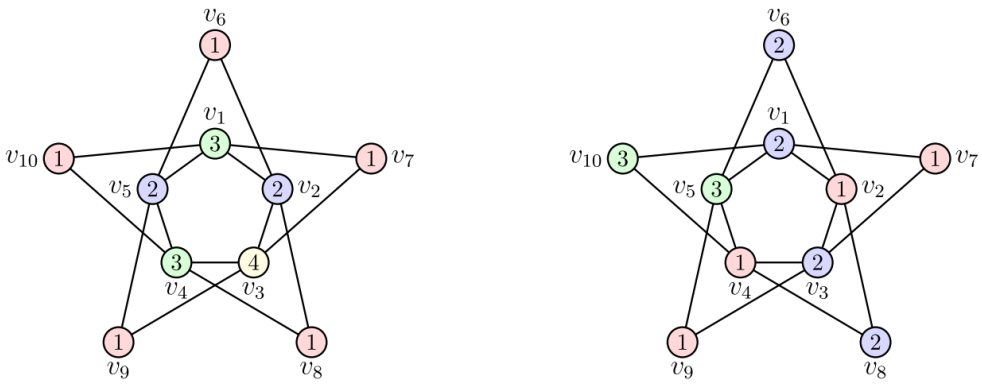
Запажање 1. Да би добили оптимално рјешење, s(G) може да буде веће од χ(G). Погледајмо следећи примјер на слици 1.



Слика 1

Бојење графа C_1 (на лијевој страни) користи χ(G) = 2 боје за бојење графа, са минималном сумом $f(C_1) = 12$, док бојење C_2 (на десној страни) има минималну суму $f(C_2) = 11$ док користи једну боју више тј. χ(G) је овде 3. Овим запажањем стоји да су независни скупови поредани по нерастућој кардиналности у било ком оптималном MSCP рјешењу.

Запажање 2. У оптималном бојењу C, сваки независан скуп V_j ($j = \{1, \ldots, k\}$) је максималан независан скуп у подграфу $G[\cup V_i]$, али није нужно независан скуп максималне кардиналности овог подграфа. Примјер се налази на слици 2 гдје су приказана два бојења графа G.



Слика 2

Бојење графа које користи максималну кардиналност има $f(C_1) = 19$ (лијево) и оптимално бојење истог графа користећи не-максималну кардиналност независних скупова са $f(C_2) = 18$ (десно).

Сада када смо се мало боље упознали са MSCP, можемо да погледамо имплементацију алгоритама за рјешавање поменутог проблема.

Похлепни алгоритам

Основна идеја похлепних алгоритама је да се у сваком кораку изабере опција која највише приближава крајњем циљу, без обзира на потенцијалне посљедице који могу настати у каснијим корацима.

Иако ови алгоритми не гарантују најбоље могуће рјешење, често су брзи и лаки за имплементацију, што их чини атрактивним за примјену у практичним ситуацијама. Међутим, у неким ситуацијама, они могу произвести прихватљиво или приближно оптимално рјешење у разумном времену.

Похлепни алгоритам за MSCP

Пратимо следеће кораке:

1. Сортирамо чворове графа по њиховом степену (број сусједа чвора) од најмањег ка највећем
2. За сваки чвор *v*, у листи сортираних чворова, радимо следеће:

   2.1. Правимо листу боја сусједа чвора *v* који су већ обојени

   2.2. Правимо листу доступних боја, тако што гледамо у листу већ кориштених боја, са изузетком претходно направљене листе боја свих сусједа

   2.3. Из листе доступних боја бирамо најмању боју *c* (по цјелобројној вриједности) и додјељујемо је чвору *v,* ако није ни једна доступна, додајемо нову боју која долази на ред по цјелобројној вриједности

   2.4. Боју *c* додајемо у листу кориштених боја

3. Када смо обишли све чворове, вратимо мапу чворова и њихових боја.

Генетски алгоритам

Ови алгоритми се заснивају на принципима природне селекције и генетике. Они генеришу рјешења за оптимизацију проблема коришћењем техника инспирисаних природном еволуцијом, као што су насљеђивање, мутација, селекција и укрштање.

Основни појмови генетског алгоритма:

- Јединка = једно рјешење проблема
- Популација = скуп јединки који представљају скуп рјешења
- Прилагођеност (фитнес функција) = оцјена квалитета јединке
- Генетски код = начин представљања сваке јединке.

Технике:

- Селекција = избор јединки из тренутне популације, на основу функције прилагођености, које ће бити коришћене за добијање наредне популације
- Укрштање = комбиновање гена два родитеља при чему се добијају нови потомци
- Мутација = мијењање гена јединке које омогућава враћање добрих гена уколико су се изгубили током претходних корака
- Елитизам = једна или више најбољих јединки се директно преносе у наредну генерацију да би се избјегле ситуације да се најбоље јединке изгубе.

Генетски алгоритам за MSCP

Размотримо нашу имплементацију генетског алгоритма за овај проблем:

1. Правимо почетну популацију величине која нам је прослијеђена као аргумент наше главне функције
2. Сортирамо популацију по функцији прилагођености (од најбоље ка најлошијој јединки), у нашем случају је то сума графа (функција циља)
3. Имплементација елитизма: двије најбоље јединке пролазе директно у следећу генерацију
4. Укрштање: на рандом начин бирамо два родитеља из тренутне популације, те коришћењем једнопозиционог укрштања добијамо нову јединку
5. Поправка: укрштањем је могло да дође до неправилно обојеног графа, те је потребно, у том случају, да поправимо граф. Прво провјеримо да ли постоје

два сусједна чвора која су обојена истом бојом, ако постоје – такав чвор обојимо најмањом дозвољеном бојом

6. Мутација: за новонасталу јединку бирамо једну рандом одабрану тачку (чвор графа), те над њом и њеним сусједним чворовима покушамо да добијемо бољи резултат (мању суму)
7. Новонасталу јединку пребацујемо у нову генерацију
8. Кораке 4,5 и 6 вршимо (size//2) пута, гдје је „size“ величина популације
9. Услови за завршавање алгоритма су вријеме које је ограничено на 10 минута и број итерација за конструисање нове генерације. Ако је један од ових услова истекао, алгоритам се овде завршава и исписујемо најбољу јединку из популације (по MSCP). У супротном, враћамо се на корак 2.

PuLP

PuLP је Python библиотека која пружа ефикасан начин за формулисање и рјешавање различитих проблема линеарног и цјелобројног програмирања. Прије употребе, потребно га је инсталирати једноставном командом коју позивамо у терминалу:

pip install pulp

ILP модел за MSCP

Овај модел користи бинарну варијаблу $x_{v,i}$ за сваки чвор $v \in V$ и сваку (потенцијалну) боју $i \in [k] := \{1, \ldots, k\}$ која прецизира да ли је чвор v обојен бојом i или не. Вриједност k представља горњу границу *јачине* s(G) графа. На тај начин, MSCP може бити формулисан следећим моделом:

Функција циља:

$$\min \sum_{i \in [k]} \sum_{v \in V} i \cdot x_v^i$$

Ограничење (1)

$$x_u^i + x_v^i \leq 1 \qquad uv \in E, i \in [k],$$

Ограничење (2)

$$\sum_{i \in [k]} x_v^i = 1 \qquad\qquad v \in V,$$

Ограничење (3)

$$x_v^i \in \{0, 1\} \;\; v \in V, i \in [k].$$

MSCP је проблем минимизације, па функција циља минимизује суму бојења. Ограничење (1) нам гарантује да сваки чвор $uv \in E$ и за сваку боју $i \in [k]$, највише један чвор ће бити обојен бојом i тј. ни један пар сусједних чворова неће бити обојен истом

бојом. Ограничењем (2) смо осигурали да сваки чвор мора да буде обојен тачно једном бојом. Ограничење (3) нам говори да је варијабла $x_{v,i}$ бинарна варијабла која узима вриједност или 0 или 1.

Када смо дефинисали ILP, можемо и да напишемо PuLP алгоритам за MSCP:

Прво, дефинишемо наш модел, са аргументом минимизације:

```
problem = pulp.LpProblem('MinimumSumColoringProblem', LpMinimize)
```

Дефинишемо нашу бинарну варијаблу:

```
x = pulp.LpVariable.dicts('x', [(v, c) for v in vertices for c in range(1,k)], cat='Binary')
```

Затим дефинишемо и нашу функцију циља:

```
problem += lpSum(c * x[v, c] for v in vertices for c in range(1, k))
```

Дефинишемо и ограничења.

Ограничење (1):

```
for v in vertices:
    for neighbor in graph[v]:
        for c in range(1,k):
            problem += x[v, c] + x[neighbor, c] <= 1
```

и ограничење (2):

```
for v in vertices:
    problem += lpSum(x[v, c] for c in range(1,k)) == 1
```

Ограничење (3) је само по себи дефинисано јер смо дефинисали бинарну варијаблу.

Позивањем функције solve() над нашим проблемом покренуће извршавање PuLP алата.

Експериментални резултати

инстанце	|V|	|E|	Похлепни		Генетски				PuLP	
			резултат	вријеме	HP	CP	девијација	вријеме	HP	вријеме
queen5_5	25	160	99	0.027	79	80.6	1.0	0.934	**75**	2.70
queen6_6	36	290	168	0.032	148	152.2	2.43	1.576	**139**	598.40
queen7_7	49	476	257	0.185	224	232.8	4.1	2.135	**196**	25.27

Табела 1 – мале инстанце

инстанце	|V|	|E|	Похлепни		Генетски				PuLP	
			резултат	t	HP	CP	девијација	t	HP	t
queen10_10	100	2940	701	0.056	**646**	670.6	10.2	5.862	708	574.53
queen11_11	121	3960	921	0.237	**873**	884.3	9.4	7.451	/	600
david_87	87	406	269	0.162	251	264.8	9.8	1.863	**237**	25.08
huck_74	74	301	247	0.155	244	255.8	9.1	1.396	**243**	15.78

Табела 2 – средње инстанце

инстанце	|V|	|E|	Похлепни		Генетски				PuLP	
			резултат	t	HP	CP	девијација	t	HP	t
anna_138	138	493	302	0.130	289	299.8	14.2	4.478	**276**	10.96
queen13_13	169	6656	1522	0.197	**1426**	1153.5	23.07	12.064	/	600
queen14_14	196	8372	1887	0.215	**1754**	1810.2	37.6	15.089	/	600

Табела 3 – велике инстанце

Гдје је:

|V| = број чворова графа

|E| = број грана графа

HP = најбољи резултат

CP = средњи резултат

t = вријеме представљено у секундама

девијација = стандардна девијација коју смо рачунали по формули:

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

Закључак

На резултате наших тестова алгоритама утиче доста фактора. Поред броја чворова, обратимо пажњу на „густину“ графа, тј. формално речено, на просјечан степен графа.

У већини случајева, очекивано, PuLP рјешавач се показао као најбољи. Међутим, на већим графовима, поменути алгоритам није успио да се заврши у року задатог времена од 10 минута.

У контраст томе, похлепни алгоритам је био најбржи, али већином је давао најлошије резултате. Свакако, добра је опција када желимо да нађемо рјешење у што краћем року, а поготово ако се ради о великим графовима, за које би друга два алгоритма одузели значајно више времена.

Генетски алгоритам се показује као довољно добра замјена за PuLP рјешавач када се ради о великим графовима, када PuLP закаже. Овај алгоритам смо покретали 10 пута на свакој инстанци. Биљежили смо најбољи и просјечан резултат, те стандардну девијацију, по којој видимо да одступања нису велика и да алгоритам ради стабилно.

Литература

- Ђукановић М., Матић Д. , Увод у операциона истраживања, 2022.
- Diego Delle Donne, Fabio Furini, Enrico Malaguti, Roberto Wolfler Calvo, Discrete Applied Mathematics - A branch-and-price algorithm for the Minimum Sum Coloring Problem
- Graph Coloring Instances