Решавање проблема максималног ахроматског броја

Научни рад у оквиру курса Рачунарска интелигенција
Универзитет у Београду, Математички факултет

Душан Пантелић
pantelic.dusan@protonmail.com

27. август 2020.

Сажетак

Кроз овај рад читалац ће бити упућен у проблем максималног ахроматског броја, као и у решавање истог. Пре свега, шта представља максимални ахроматски број, његов значај, примену, али и проблеме при његовом одређивању. Представљени су алгоритми за решавање проблема и њихова компарација, као и њихове позитивне и негативне стране. Читалац ће такође добити увид у експерименталне податке извршавања представљених алгоритама са примењеним статистичким методама зарад стварања шире слике о практичној примени. На крају рада сумирани су закључци истраживања и изложени правци за даље истраживање и унапређивање.

Садржај

1 Увод

Познавање општих концепата теорије графова је неопходно за разумевање рада, тако да се пропоручује претходно упознавање са материјом помоћу релевантне литературе [1].

1.1 Бојење графа и ахроматски број

Правилно бојење графа представља додељивање боја чворовима графа, тако да никоја два суседна чвора немају исту боју. Граф можемо бојити на више начине, стога бојење графа не мора бити јединствено, већ може садржати различити број боја, где избор боја није релевантан. Бојење графа са n боја називамо n-бојење, а уколико важи да за сваки пар различитих боја постоје два суседна чвора графа којима су те боје додељене, то бојење називамо комплетно n-бојење. Хроматски број графа G, у ознаци $\chi(G)$, представља минималан број n тако да постоји комплетно n-бојење тог графа. Ахроматски број графа G, у ознаци $\psi(G)$, представља максималан број n тако да постоји комплетно n-бојење тог графа.

1.2 Особине ахроматског броја

Нека је n број чворова неког графа G, тада можемо уочити да важи следеће: $1 \leq \chi(G) \leq \psi(G) \leq n$. Ахроматски број такође можемо представити као максимални број партиција у свим партиционисањима низа чворова, при чему унутар партиције не сме бити суседних чворова и за сваки пар партиција мора постојати барем једна грана која повезује те две партиције. Горе наведена дефиниција је погоднија за касније имплементирање алгоритамских решења. Јанакакис и Гаврил су доказали да је проблем налажења ахроматског броја НП-комплетан [7].

1.3 Значај налажења ахроматског броја

Решавање проблема ахроматског и хроматског броја су специјални случајеви проблема бојења графа, где уз правилно бојење захтевамо да број боја буде максималан односно минималан. Једно правилно бојење графа може бити решење судоку слагалице, разних проблема планирања, као и проблема алокације регистра [1]. Кроз пример планирања летова авиона где сваки чвор представља лет, а грана између два чвора представља да се летови преклапају, видимо да правилно бојење овако представљеног графа распоређује авионе групи летова међу којима нема преклапања летова. Приметимо такође да тражењем ахроматског броја у претходном примеру заправо тражимо максималан број авиона да покрију све групе летова где између група летова мора бити преклапања(ради мањег чекања) чиме добијамо растерећење авио саобраћаја, док тражењем хроматског броја тражимо најмањи број авиона који покривају летове.

[1] Пример литературе [6].

2 Решење проблема

2.1 Алгоритам провере исправности бојења

Овај алгоритам не решава проблем ахроматског броја, али се користи у свим осталим алгоритмима који ће бити обрађени, тако да ћемо га одвојити као целину и приказати његов псеудо-код.

```
function proveri_bojenje(bojenje):
  for (svaka boja u bojenju):
    if (boja sadrzi susedne cvorove):
      return False

  for (svaki par boja u bojenju):
    if (ne postoji grana izmedju parova boja ):
      return False

  return True
```

Listing 1: Псеудо-код провере исправности бојења

2.2 Алгоритам исцрпне претраге

Најпоузданији али и временски најзахтевнији начин решава проблема ахроматског броја је алгоритам исцрпне претраге. Идеја алгоритма је да се за сваки могући број боја генришу сва могућа бојења, при чему тражимо највећи број боја за који постоји валидно бојење.

```
function nadji_ahromatski_broj(graf):
  ahromatski_broj = -1
  for (svaki moguci broj boja grafa br):
    for (bojenje grafa u svim mogucim bojenjima sa br boja):
      if (proveri_bojenje(bojenje)):
        ahromatski_broj = br
        break

return ahromatski_broj
```

Listing 2: Псеудо-код алгоритма исцрпне претраге

Предности алгоритма: Решење је оптимално.
Мане алгоритма: Експоненцијална сложеност.

2.3 Генетски алгоритам

Решење проблема генетским алгоритмом је направљено модификацијом алгоритма приказаном у [5]. У наставку је приказан псеудокод, као и описи свих битнијих карактеристика алгоритма.

```
function genetski_algoritam():
  populacija = random_bojenja_grafa()
  while (nije ispunjen maksimalan broj iteracija i nije dostignuto
      optimalno resenje):
    sort(populacija)
    zadrzi_cetvrtinu_najboljih_jedinki()
    for (ostatak jedinki):
      selktuj_roditelje()
      ukrsti_roditelje()
      izvrsi_mutaciju()
  return populacija
```

Listing 3: Псеудо-код генетског алгоритма

Критеријум заустављања: Алгоритам се зауставља када је достигнут максималан број итерација, или у случају када је познато оптимално решење зауставља се када нека од јединки садржи оптимално решење.

Сортирање популације: Популацију сортирамо на основу прилагођености јединке коју добијамо функцијом прилагођености опадајуће.

Функција прилагођености: Функцију прилагођености рачунамо као количник броја боја коришћеног при бојењу и броја грешака које доводе до неисправног бојења. Уколико је број грешака једнак нули јединку награђујемо тако што множимо број боја са константом већом од један. Треба бити обазрив прилоком избора награде за валидно бојење. Премала награда доводи до тога да валидна решења испадну из популације, док превелика награда фаворизује валидна решења и не дозвољава да се развију можда оптималнија решења чиме популација конвергира ка локалном оптимуму.

Елитизам: Задржавамо четвртину најбољих јединки како би обезбедили да наставе живот непромењене.

Селекција: Бирамо насумично две јединке, при чему свака јединка има исту шансу да буде изабрана.

Укрштање: Низ чворова графа делимо на три дела насумичне величине. Прво дете узима за први део чворова боје првог родитеља, за други боје другог родитеља и за трећи опет боје првог родитеља. За друго дете важи обратно.

Мутација: Свака јединка са задатом вероватноћом мења боју насумичног чвора у такође насумичну боју.

Предности алгоритма: Извршава се у разумном временском периоду.

Мане алгоритма: Решење није увек оптимално, а некада решење није валидно бојење.

Могућа унапређења: Уколико након завршетка алгоритма популација не садржи валидно решења један од начина превазилажења овог проблема је коришћење алгоритма мудрости роја честица описаног у [3]. Овај алгоритам није имплементиран, али идеја је да се најбоља половина решења локално претражује до проналаска валидног решења тако што ће чворови најбољег решења који нарушавају валидност променити боју у најзаступљенију боју за тај чвор у осталим јединкама.

2.4 Хибридни генетски алгоритам

Претходно описан алгоритам можемо надоградити тако што ћемо приликом сваке итерације најбољу и најгору јединку обрадити алгоритмом симулираног каљења. Симулирано каљење, иако се некада користило као самосталан алгоритам за проблем бојења графа, не даје довољно добра решења у пракси [2]. Без обзира на горе наведено, конвергенцију популације можемо убрзати хибридизацијом генетског алгоритма тј. додавањем симулираног каљења.

```
1000 function simulirano_kaljenje():
       while (nije dostignut maksimalan broj iteracija za kaljenje):
1002     if (validno_bojenje):
           podeli cvorove nasumicne boje na dva dela i drugom delu
         dodaj novu boju
1004     else:
```

```
      zameni boje na dva nasumicno izabrana cvora

    if (promena donosi bolju prilagodjenost):
      prihvati_promenu
    else if (kontrolni_parametar > nasumicnog_broja):
      prihvati_promenu()
    else:
      odbaci_promenu()
    azuriraj_kontrolni_parametar()
```

Listing 4: Псеудо-код симулираног каљења

Напомена: Контролну променљиву ажурирати тако да како итерације одмичу тако је и мања вероватноћа прихватања неповољне промене. Не треба ни одмах бити сувише мала да не би остали у локалном оптимуму.

Предности алгоритма: Може убрзати конвергенцију популације ка оптималном решењу, али и извадити популацију из локалног оптимума.

Мане алгоритма: Могуће је изгубити оптимално решење прихватањем неповољније промене.

3 Експериментални резултати

У наставку ће бити приказани експериментални резултати, њихова визуелизација као и поређење резултата. За тест примере коришћени су насумично генерисани графови са бројем чворова између 2 и 14. Графови су сачувани у gml формату и можете их пронаћи на https://github.com/pantelic-dusan/maximum-achromatic-number/random_graphs. Тестови су извршавани на платформи https://cgc.sbgenomics.com на инстанци x1.32xlarge(3840 SSD, 128vCPUs, 1952GB RAM), и сви тест резултати су доступни на https://github.com/pantelic-dusan/maximum-achromatic-number/test_results у JSON формату.

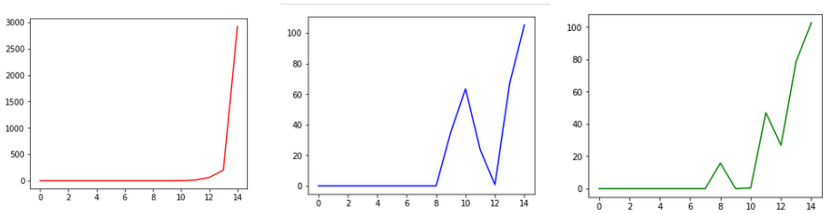
3.1 Визуализација података

Део резултата, који поред броја чворова и грана за сваки од горе наведених алгоритама приказује и редом резултат њиховог извршавања као и време протекло током извршавања, приказан је на 1.

NODES	EDGES	BF_OUT	BF_TIME	GA_OUT	GA_TIME	GAA_OUT	GAA_TIME
7.0	0.0	1.0	0.043	1.0	0.022	1.0	0.023
10.0	29.0	6.0	0.616	6.0	0.275	6.0	0.255
4.0	0.0	1.0	0.001	1.0	0.015	1.0	0.026
2.0	1.0	2.0	0.000	2.0	0.017	2.0	0.021
8.0	6.0	4.0	0.054	4.0	0.078	3.0	79.071
10.0	34.0	7.0	0.742	6.0	99.525	7.0	0.470
13.0	55.0	9.0	88.663	9.0	12.028	8.0	164.089
12.0	4.0	3.0	229.384	3.0	0.038	3.0	0.038
2.0	0.0	1.0	0.000	1.0	0.020	1.0	0.023
7.0	6.0	4.0	0.010	4.0	0.020	4.0	0.107
9.0	4.0	3.0	0.602	3.0	0.027	3.0	0.027
6.0	14.0	5.0	0.003	5.0	0.017	5.0	0.017
8.0	12.0	5.0	0.027	5.0	0.110	5.0	0.022
14.0	52.0	8.0	249.689	7.0	171.753	7.0	169.527
9.0	27.0	7.0	0.111	6.0	88.201	7.0	0.486
4.0	0.0	1.0	0.001	1.0	0.015	1.0	0.020
2.0	1.0	2.0	0.000	2.0	0.016	2.0	0.019
14.0	12.0	5.0	4973.395	5.0	0.442	5.0	0.044
4.0	5.0	3.0	0.000	3.0	0.014	3.0	0.018
3.0	2.0	2.0	0.000	2.0	0.016	2.0	0.018

Слика 1: Пример резултата

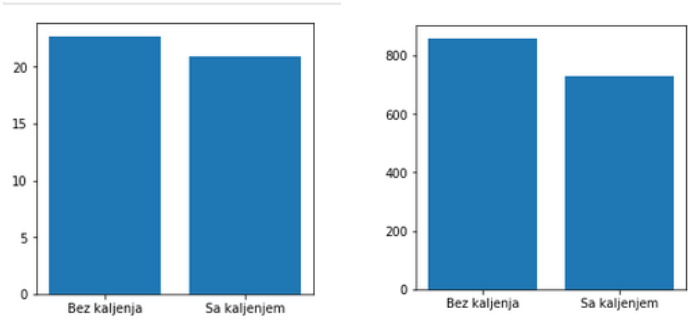
Такође приказивањем података о времену извршавања алгоритама редом на 2, можемо видети напредак друга два алгоритма у односу на алгоритам исцрпне претраге.



Слика 2: Просечно време извршавања у односу на број чворова

3.2 Поређење генетског алгоритма са и без симулираног каљења

Поређењем редом просечног времена извршавања и просечног броја итерација гентских алгоритама са и без симулираног каљења на тест резултатима, долазимо до закључка да додавањем симулираног каљења незнатно побољшавамо перформансе генетског алгоритма, што можемо видети на слици 3.

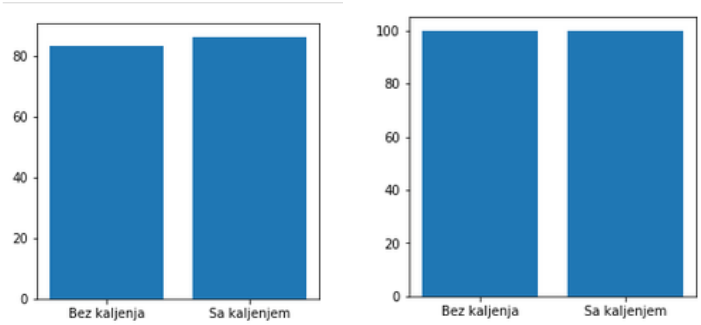


Слика 3: Поређење перформанси генетског алготитма са и без каљења

Такође поређењем, као на слици 4, редом процента исправних решења и процента оптималних решења, долазимо до закључка да генетски алгоритам са каљењем благо побољшава вероватноћу добијања оптималног решења и да оба алгоритма дају исправна решења.

4 Закључак

Собзиром на значај обрађене теме потребно је стално унапређивати постојеће методе за решавање овог проблема. Обрађена су три алгоритма, која са свим својим предностима и манама, нису довољно добра за проблеме који захтевају изузетну прецизност и брзину решавања. Побољшања су могућа тако што ћемо на основу статистичких резултата кориговати константе алгоритма као што су број итерација, величина популације, фактор мутације, израчунавање прилагођености, или у случају опције са симулираним каљењем тражимо



Слика 4: Поређење оптималности генетског алготитма са и без каљења

најповољнији број итерација и фактор опадања контролне променљиве.

Читаоцу се препоручује да настави самостално унапређивање обрађених алгоритама, али и покушај комбиновања са другим методама. Такође је препорука да се изуче и друге врсте алгоритама попут Дејвид-Путманове методе описане у [2], оптимизација колонијом мрава описане у [4], али и других доступних решења.

sdsdfsdfsdfsd dsdsdfdsfs

Литература

[1] A.K.Bincy and B.Jeba Presitha. *Graph Coloring and its Real Time Applications anOverview.* International Journal of Mathematics And its Applications.

[2] Huberto Ayanegui and Alberto Chavez-Aragon. *A complete algorithm to solve the graph-coloring problem.* Facultad de Ciencias Basicas, Ingenieria y Tecnologia.

[3] Musa M. Hindi and Roman V. Yampolskiy. *Genetic Algorithm Applied to the Graph Coloring Problem.*

[4] Jonathan M. Thompson Kathryn A. Dowslanda. *An improved ant colony optimisation heuristic for graph colouring.*

[5] Bipin Thanki Sandeep R. Vasant. *Optimization of Graph Coloring Problem using Hybrid Selection: A Genetic Algorithm Approach.* Ahmedabad University, Ahmedabad, Gujarat, India.

[6] Robin J. Wilson. *Introduction to Graph Theory.* Longman Group Ltd, 4th edition, 1996.

[7] M. Yannakakis and F. Gavril. *Edge Dominating Sets in Graphs.* Society for Industrial and Applied Mathematics, 1980.