УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ

Рака Д. Јовановић

ПОБОЉШАЊЕ МЕТОДА МРАВЉИХ КОЛОНИЈА ЗА РЕШАВАЊЕ НП-КОМПЛЕТНИХ ОПТИМИЗАЦИОНИХ ЗАДАТАКА НА ГРАФОВИМА

докторска дисертација

Београд, 2012

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Raka D. Jovanovic

IMPROVEMENT OF THE ANT COLONY OPTIMIZATION FOR SOLVING NP-COMPLETE PROBLEMS ON GRAPHS

Doctoral Dissertation

Beograd, 2012

Комисија:

др Милан Туба, редовни професор, Факултет за компјутерске науке, Мегатренд универзитет

др Ђорђе Дугошија, редовни професор, Математички факултет, Београдски универзитет

др Миодраг Живковић, редовни професор, Математички факултет, Београдски универзитет

20. Јун 2012.

Захвалница

Желео бих да захвалим професору др Милану Туби, ментору још од израде магистарског рада са којим сам објавио двадестак радова и био на великом броју научних скупова. Захваљујем професору др Ђорђу Дугошији на несебичној помоћи око израде тезе. Захвалио бих се професору др Миодрагу Живковићу на прегледу тезе и препорукама које су из тога проистекле. Такође се захваљујем др Драгани Јовић и др Александри Стринић на саветима и препорукама приликом писања тезе. Захвалност дугујем и професору Рудолфу Лорензу који ми је омогућио рад на тези приликом мог боравка на Тексас АМ Унивезитету у Дохи.

Захваљујем се и својој породици и пријатељима на дугогодишњој подршци и вери у мене током мог истраживачког рада.

ПОБОЉШАЊЕ МЕТОДА МРАВЉИХ КОЛОНИЈА ЗА РЕШАВАЊЕ НП-КОМПЛЕТНИХ ОПТИМИЗАЦИОНИХ ЗАДАТАКА НА ГРАФОВИМА

Резиме

Ова теза посвећена је решавању тешких оптимизационих проблема из области графова употребом метахеуристике која се заснива на симулацији понашања колоније мрава, односно оптимизацијом мрављим колонијама (ОМК). У оквиру рада даје се преглед постојећих метода за решавање овог типа проблема са акцентом на методе који се заснивају на популацији, попут ројева честица, колонија пчела и ОМК. Приказани су предходно развијени алгоритми ОМК којима су решавани проблем трговачког путника, проблем минималног тежинског чворног покривача графа и проблем минималног повезаног доминирајућег скупа. Први допринос тезе овој области састоји се у решавању нових проблема који нису предходно решавани употребом ОМК, попут проблема минималног тежинског доминирајућег скупа. У случају проблема који су предходно решавани од стране других истраживача, унапређење је извршено увођењем нове врсте хибридизације ОМК. Ова хибридизација се заснива на корекцији феромонског трага у зависности од карактеристика најбољег нађеног решења. Идеја је да се из оваквих решења избаце елементи који имају неке непожељне карактеристике, усмеравајући колонију на области решења у којима је вероватније да се добра решења налазе. У извршеним тестовима је показано да овакав приступ налази решења бољег квалитета и убрзава конвергенцију. Такође је извршена исцрпна анализа различитих варијација алгоритма ОМК. Посебно поглавље тезе је посвећено паралелизацији ОМК са анализом различитих видова комуникације између колонија; у овом поглављу је приказан и један нов приступ овом проблему. Коначно, креиран је и програмски пакет за решавање проблема употребом ОМК. Пакет је имплементиран у облику оквира са проширењима. Уведен је и један вид апстракције кроз низ класа који је погодан за креирање овакавих програма.

Кључне речи: алгоритми инспирисани природом, интелигенција ројева, оптимизација мрављим колонијама, комбинаторни проблеми, графовски проблеми

Научна област: Рачунарске науке и технологија, Рачунарство, Процесирање Података/ Методе за решавање проблема. УДК (004.02)

IMPROVEMENT OF THE ANT COLONY OPTIMIZATION FOR SOLVING NP-COMPLETE PROBLEMS ON GRAPHS

Abstract

This Ph.D. thesis contributes to solving hard optimization problems in the field of graph theory using a metaheuristic that is based on simulation of the behavior of an ant colony, named Ant Colony Optimization (ACO). In the thesis an overview of methods for solving this type of problems is given, with an emphasis on population based methods like Particle Swarm Optimization, Bee Colony Optimization and ACO. In the first part of the text previously developed ACO algorithms for the Traveling Salesman Problem, the Minimum Weight Vertex Cover Problem and the Minimum Connected Dominating Set Problem are presented. The first contribution of the thesis to this field consists of solving new problems that previously have not been solved using ACO, like the Minimum Weight Dominating Set Problem. In the case of problems previously solved by other researchers using ACO, the improvement is in introducing a new type of hybridization of the ACO algorithm. This hybridization is based on pheromone correction which is based on some properties of the best found solution. The idea is to remove from these solutions elements that have some undesirable characteristics, steering the colony to search in areas of the solution space where it is more likely for good solutions to exist. In the conducted tests it has been shown that this approach finds solutions of better quality and increases convergence speed. A thorough performance analysis for different variations of ACO has also been presented. A separate chapter of the thesis is dedicated to the parallelization of ACO with an analysis of different methods of communication between the colonies. In this chapter a new approach has been presented for this problem. Finally, the software framework developed for solving these ACO problems is presented. It is implemented in a form of a plugin system. A model of abstraction through a set of classes which is adequate for this type of problems is also introduced.

Keywords: Nature inspired algorithms, Swarm intelligence, Ant colony optimization, Combinatorial problems, Graph problems

Scientific field: Computer science and technology, Computing, Data processing/ Problem-solving methods. (UDC 004.02)

Садржај

1 Увод **3**
- 1.1 НП-комплетни проблеми 4
- 1.2 Метахеуристике за решавање НП-комплетних проблема . . 5
  - 1.2.1 Исцрпна претрага 5
  - 1.2.2 Грамзиви алгоритми (Greedy) 6
  - 1.2.3 Монте Карло 7
  - 1.2.4 Локална претрага 7
  - 1.2.5 Генетски алгоритми 9
  - 1.2.6 Интелигенција ројева 10

2 Оптимизација мрављим колонијама **13**
- 2.1 Варијације алгоритма ОМК 17
- 2.2 Хибридизације алгоритма ОМК 20

3 Побољшање метода мрављих колонија за решавање проблема путујећег трговца (ПТП) методом корекције феромонског трага **23**
- 3.1 Метод сумњивих елемената 23
- 3.2 Тестови и резултати 27

4 Побољшавање решавања проблема минималног тежинског чворног покривача коришћењем оптимизације мрављим колонијама **29**
- 4.1 Решавање ПМТЧП коришћењем ОМК 30
- 4.2 Варијације алгоритма ОМК за ПМТЧП 32
- 4.3 Анализа проблема стагнације 35
- 4.4 Екстензија метода сумњивих елемената 36
- 4.5 Тестови и резултати за варијације ОМК 39
- 4.6 Тестови и резултати за метод искључивања сумњивих елемената 47

5 Решавање проблема минималног тежинског доминирајућег скупа употребом ОМК 58
5.1 Грамзиви алгоритми за ПМТДС 59
5.2 Примена ОМК на ПМТДС 61
5.3 Тестови и резултати 63

6 Решавање проблема минималног повезаног доминирајућег скупа употребом ОМК 68
6.1 Грамзиви алгоритми за решавање ПМПДС 69
6.2 Имплементација ОМК 71
6.3 Избегавање стагнације ОМК 73
6.3.1 Стратегија корекције феромонског трага 73
6.3.2 Избор иницијалног чвора 75
6.4 Тестови и резултати 76

7 Паралелизација ОМК 83
7.1 Различите врсте топологија за паралелизацију ОМК 84
7.2 Тестови и резултати 86

8 Програмско окружење за решавање проблема употребом ОМК 92
8.1 Анализа потреба оквира за решавање проблема употребом ОМК 93
8.2 Имплементација ПТП коришћењем ГРАФ-АНТ-а 98

9 Закључак 102

Глава 1

Увод

Са појавом рачунара за велики број проблема који су до тада били теоретски али не и практично решени, постало је могуће наћи решења. Разлог ове појаве је што је време које је било потребно човеку за израчунавање таквих проблема могло бити мерено у годинама, а уз помоћ рачунара то време скраћено је на неколико секунди. Овакви проблеми нису обавезно били комплексни, пример је множење два стоцифрена броја, множење или рачунање детерминанти матрица великих димензија. Као последица могућности за брзо израчунање једноставних аритметичких операција рачунари су почели да се користе за решавање све комплекснијих проблема. Први проблеми су се распростирали на велик низ области од физике, математике, грађевинарста, економије, банкарства, итд. Они су најчешће могли да се сведу на одређени математички модел а он опет употребом нумеричких метода да се сведе на једноставне аритметичке операције. Како је расла моћ рачунара и алата за креирање програма они су употребљивани за све детаљније описе реалности. За илустрацију ове појаве може се искористити симулација лета пројектила. Испаљивање пројектила се прво гледало као коси хитац, затим се може додати и отпор ваздуха као сила, и снага ветра. Након тога, за сам пројектил може се користити тродимензионални модел и рачунати отпор ваздуха на основу његове аеродинамичности и брзине. Као следећи корак, може се повећавати резолуција модела. У случају да су математички модел и нумеричка метода добри, што се више повећава резолуција имаћемо све веродостојнију симулацију.

Оваквом логиком се стиче утисак да ако имамо довољно моћан рачунар сваки проблем се може решити у кратком времену. На пример ако се модел повећа n пута довољно је повећати снагу равунара n пута и решење ће бити нађено за исто време. У случају m димензионих проблема снагу рачунара повећамо n^m и проблем је опет решен за еквивалентно време.

n^m расте далеко брже од n али је у питању полиномијалан раст. Међутим постоје проблеми за чије решавање не постоје алгоритми који налазе решење у полиномијалном времену. Један од класичних представника је проблем трговачког путника који има следећу формулацију: "Ако постоји n градова наћи најкраће могуће кружно путовање такво да трговац обиђе све градове тачно једном". Овај наизглед једноставан проблем има $n!$ потенцијалних решења, односно све могуће пермутације низа градова. За њега није познат (а највероватније и не постоји) алгоритам који налази решење у полиномијалном времену. Овај проблем спада у групу оптимизационих проблема, односно проблем налажења најбољег решења од свих могућих прихватљивих решења. Прецизније, када је представљен у облику проблема одлучивања спада у групу НП-комплетних проблема.

1.1 НП-комплетни проблеми

У теорији израчунљивости у рачураству, проблем одлучивања је проблем на који се може одговорите са "ДА" или "НЕ". Пример би био "да ли је број a већи од броја b". Област рачунске сложености категорише проблеме одлучивања по томе колико су тешки за решавање. Тешки се овде користи у смислу колико је рачунарских ресурса потребно за најефикаснији алгоритам за одређени проблем. У теорији комплексности, НП (недетерминистичко полиномијално време) је скуп проблема одлучивања решивих у полиномијалном времену на недетерминистичкој Тјуринговој машини. Еквивалентно, то је скуп проблема чија решења могу да се провере на детерминистичкој Тјуринговој машини у полиномијалном времену. Подскуп НП проблема је скуп НП-комплетних проблема одлучивања и они се описују као најтежи НП проблеми. Они имају особину да се сваки од проблема у њему може у полиномијалном времену превести у неки други који такође елемент НП-комплетних проблема. Они имају особину да ако би постојао алгоритам у полиномијалном времену за макар један од њих, онда би постојао полиномијални алгоритам за све проблеме из класе НП-комплетних проблема. До сада (упркос свим напорима) није пронађен полиномијални алгоритам ни за један НП-комплетан проблем, а обично се сматра да није вероватно да постоји такав алгоритам за решавање овог типа проблема.

Пример је проблем трговачког путника, он се не може решити у полиномијално јер се мора на неки начин проверити $n!$ потенцијалних решења, а може се видети да ли је решење тачно једноставним сабирањем укупне путање шта је могуће у полиномијалном времену.

1.2 Метахеуристике за решавање НП-комплетних проблема

За овакве проблеме је јако тешко наћи оптимално решење, али за проблеме из правог живота које они моделирају често нема потребе за њим. На пример да ли је неопходно да трговачки путник изабере најкраћи могући пут или је задовољавајуће да путује "довољно кратким путем". Идеја налажења приближних решења оваквих проблема је створила велик број различитих концепата, метахеуристика, за креирање алгоритама за њихово решавање. Ове метахеуристике се могу поделити на детерминистичке, и не детерминистичке у зависност да ли се користе нека врста случајних бројева. А по начину налажњења решења их можемо поделити на следећи начин

- Исцрпна претрага
- Грамзиви алгоритми
- Монте Карло
- Методе локалне претраге
  - Табу претрага (Taboo Search)
  - Симулирано каљење (Simulated Annealing)
- Генетски алгоритми
- Интелегенција ројева
  - Оптимизација ројевима честица
  - Оптимизација пчелињим колонијама
  - Оптимизација мрављим колонијама

1.2.1 Исцрпна претрага

Исцрпна претрага је најосновнији начин налажења оптималног решења оптимизационих проблема. Она се зове и метод бруталне силе. Идеја је да се сва могућа решења енумеришу и израчуна квалитет сваког од њих. Оваквим методом се са сигурношћу налази оптимално решење. Исцрпна претрага, је за решавање оптимизационих проблема употребљив само у случају малих димензија. Овај проблем се јасно види на случају проблема трговачког путника. Када имамо 8 градова потребно

је проверити 40 320 могућих пермутација што није проблем на данашњим рачунарима, међутим већ у случају 24 града постоји проближно $6.3 * 10^{23}$ пермутација.

Овакви методи имају добру особину да се обично лако имплементирају. Иако немају практичну употребну вредност код проблема већих димензија овакви алгоритми су важни за развијање комплекснијих, јер нам даје решења мањих проблема које можемо користити за проверу исправности нових методи.

1.2.2 Грамзиви алгоритми (Greedy)

Грамзиви алгоритми покушавају да реше комплексан проблем увођењем неке просте локалне хеуристике. Идеја да се решавање проблема сведе на кораке, и да се на сваком од њих бира локално најбоље проширење. Овакву логику ћемо илустровати на примеру проблема трговачког путника. Градња оптималног кружног пута се своди на бирање града који најближи тренутној локацији на сваком кораку. Овакви алгоритми имају следеће основне елементе

- Скуп кандидата из којих се бирају делови решења.
- Функцију селектовања, која одређује који ће елемент бити следећи изабран.
- Постојање критеријума који одређује ваљаност елемента у смислу да ли се његовим укључивање може добити дозвољени облик решења.
- Функција која одређује квалитет до сада генерисаног парцијелног решења.
- Критиеријум који одређује да ли је тренутно генерисано парцијелно решење комплетно решење.

Предност оваквих алгоритама, у односу на остале типове, је једноставност имплементације и велика брзина. Међутим велики је проблем, то што су "кратковиди" и често не постижу оптимално решење. У неким случајевима, чак налазе јединствено најгоре решење [1]. Овај метод и ако обично не даје оптимална решења, је од велике важности јер су решења обично релативно доброг квалитете. Оваква решења се касније могу користити као почетни корак неког комплекснијег метода о чему ће бити речи у наставку.

1.2.3 Монте Карло

Проблем код грамзивих алгоритама је што се заснивају на одређеној хеуристици и преко ње креирају решења. Овакав приступ има два основна проблема. Први је могућност да изабрана хеуристика једноставно није добра и не може да нађе оптимално решење. Други је што се квалитет решења проверава само за једно потенцијално решење, оно које нам даје хеуристика. Овакве проблеме желимо избећи на неки начин. То се постиже додавањем одређеног нивоа сучајности односно коришћењем недетерминистичког алгоритма. Монте Карло је основни метод овог типа. Он има следећу логику. Из скупа потенцијалних решења на случајан начин бирамо једно. Проверавамо да ли је његов квалитет бољи од до сада најбољег нађеног решења, ако јесте чувамо га и бирамо ново тест решење. У оваквом поступку, што је већи број решења тестиран већа је вероватноћа да смо нашли оптимално решење.

Ефикасност оваквих методи се може илустровати на следећем проблему. Замислимо да имамо скуп од n бројева и желимо да нађемо елемент који припада горњој половини скупа. Иницијална идеја је да упоредимо $n/2$ елемента и највеће међу њима сигурно припада горњој половини скупа. Значи, неопходно је извршити $n/2$ поређења. Монте Карло приступ овом проблему је да на случајан бирамо један елемент, проверимо да ли је већи од до сада нађеног максимума и ако јесте прогласимо га за нови максимум. Каква је вераватноћа да после i корака нисмо нашли елеменат који припада горњој полови? Шансе да први изабрани елемент не припада је 0.5. Након другог покушаја потребно је да оба елемента припадају доњој половин односно 0.5^2. Јасно је да вероватноћа да након i корака нисмо нашли елеменат који припада горњој половини је еквивалентна 0.5^i.

Овај пример илустурује да се оваквим методом релативно брзо, у поређењу са глобалном претрагом, може доћи до решења која имају добар квалитет. Међутим за добијање решења високог квалитета потребно је тестирати велик број потенцијалних решења. Из ових разлога се прост Монте карло метод обично не користи већ се комбинује са локалном претрагом или се избор елемената који се тестирају учини "мање случајним".

1.2.4 Локална претрага

Локална претрага је општа метахеуристика која се користи за поправљање већ нађених решења оптимизационих проблема. Она је најефикаснија када се тражи решење које има максималну вредност неког

критеријума међу свим потенцијалним решењима. Већину оптимизационих проблема је могуће дефинисати у контексту претраге простора решења на неколико начина. Локална претрага се креће од једног до другог решења из скупа свих могућих решења докле год није нађено оптимално решење или је задовољен неки други критеријум за прекид рада алгоритма. Прецизније локална претрага је итеративни процес у ком се креће од неког почетног решења, и померамо се у ново решење у његовој околини које је потенцијално боље. Наравно, неопходно је дефинисати "околину" у зависности од проблема који се тренутно решава. У већини случајева оваква околина има више од једног елемента. Избор следећег решења је начин на који се овакви методи у општем случају дефинишу. Најстандарднији приступ је метод "пењања уз брдо". У њему се увек крећемо на решење из комшилука које има максималну вредност по неком критеријуму, за који очекујемо да ће поправити квалитет решења. Један од примера локалне претраге је 2-ОПТ претрага за проблем трговачког путника [2].

Овакви методи се најчешће користи у комбинацији са другим алгоритмима. На пример, грамзивим алгоритмом нађемо решење који користимо као почетни корак локалне претраге. Овакав метод се јако ефикасно комбинује са Монте Карло методом, случајано изабран елемент постаје корен локалне претраге.

Табу претрага (Taboo Search)

Табу претрага је математички оптимизациони метод који припада класи локаних техника претраге. Ова метахеуристика се показала јако ефикасна у решавању оптимизационих проблема. Табу претрагу је дефинисао Фред Гловер (Fred W. Glover) [3], [4]. Он повећава ефикасност локалне претраге употребом меморијске структуре. Меморија се своди на то да када је нађено потенцијално решење обележава се као "табу" и алгоритам претраге у наставку рада више не посећује то решење.

Симулирано каљење (Simulated Annealing)

Симулирано каљење (СК) је недетерминистичка метахеуристика за налажење глобалних оптимума или бар њихових апроксимација. Она се најчешће користи за тражење решења у дискретизованим просторима. Име метода потиче из његове сличности са каљењем челика који се заснива на узастопним грејањем и ковањем. На овај начин повећава се степен кристализације и смањују дефекти који постоје у металу. Грејањем се појединачни атоми одвајају од својх тренутних позиција, и

напуштају локални минимум енергије, а хлађењем и ковањем се атоми терају у нове позиције, то јест у ново стање локалног оптимума.

Аналогно овом физичком процесу, алгоритам СК смењује две фазе случајна промена тренутног решења и тражења бољег решења у околини те позиције. Прецизније у сваком кораку СК из тренутног стања s се на случајан начин бира ново стање s' у његовој околини. Ако су испуњени одређени услови прелази у то ново стање или у супротном остаје у постојећем стању. Овај услов може бити да је ново стање бољег или лошијег квалитета од тренутног. Овај корак се непрекидно понавља, само што се временом повећава вераватноћа да је следеће одговарајуће стање бољег квалитета од претходног. У крајњој фази, алгоритам се претвара у грамзиви алгоритам.

Овај метод је први пут дефинисан од стране Скот Киркпатрика, К. Данијел Гелата и Марија П. Векија [5] и не зависно од њих од стране Владо Чернија [6]. Овај метод се показао изузетно ефикасна на великом броју типова проблема [7], [8], [9], [10].

1.2.5 Генетски алгоритми

Генетски алгоритми [11] (ГА) су недетерминистичке методе претраживања, које опонашају биолошки процес еволуције. Карактеристично за њега је да се користи већа групу решења проблема, која се назива популацијом. Поједина решења из популације се бирају на основу својих квалитета, и међусобно комбинују како би формирала нову генерацију решења која би потенцијално садржала боље потомке. Да би се у ГА овако нешто постигло потребне су, као у природи, постојање операција укрштања и мутације. Симболично, податак о стању неке промењиве се назива геном, а скуп стања променљивих које дефинишу једно решење се назива хромозомом.

Да би се неки проблем решио овим методом потребно је следеће

- Креирати репрезентацију домена решења у облику гена, најчешће у облику бинарне или неке друге ниске.
- Дефинисати функцију квалитета елемената популације.

Сам алгоритам има следеће корака

- Иницијализација у којој креирамо на случајан начин почетну популацију.
- Фаза селекције у којој на основу квалитета чланова популације бирамо која ће се решења међусобно упаривати.

- Фаза репродукције у којој се од изабраних парова, методом укрштања гена, добијају потомци односно нова решења. На неким потомцима се изврши мутација, односо изврше се неке случајне промене.

- Претходна два корака се непрекидно понављају док се не испуне услови за прекид. Ови услови могу бити: да је постигнуто решење задовољавајућег квалитета, достигнут је максималан дозвољен број генерација, дошло је до стагнације претраге.

Употребом овог метода су успешно решени бројни проблеми попут проблема трговачког путника [12] и проблема чворног покривача графа [13].

1.2.6 Интелигенција ројева

Интелигенција ројева је облик колективног понашања децентрализане, самоорганизоване групе јединки који се појављује у природи. Примери оваквих ројева су колоније мрава, ројеви пчела, јата птица итд. У њима низ простих и ниско интелигентних јединки успева да решава компексне проблеме. На пример, колонија мрава успешно налази јако кратку путању од гнезда до хране. Овај концепт понашања је почео да се користи у решавању оптимизационих проблема. Основна идеја оваквих алгоритама је да постоји одређена популација јединки, у којој свака од њих неком простом методом креира решења. Интелигентно понашање се постиже разменом информација међу њима, које утиче на методу за креирање решења.

Оптимизација ројевима честица

Оптимизација ројевима честица (ОРЧ) слично методима локалне претраге тражи нова боља решења у близини већ постојећих. Код метода локалне претраге се ново решење тражи у околини једног решења, и на основу хеуристике која зависи само од локалних података, се бира где ће се тражити следеће решење. Код оптимизације ројевима честица, упоредо се траже решења у околини већег броја потенцијалних решења. Главна разлика је да хеуристика у зависности од које се бира следеће решење не зависи само од локалних карактеристика већ зависи и од позиције до сада најбољег решења. Овом методом се покушава симулирати понашање јата птица или риба. Откриће ове методе се приписује Кенедију, Ебхарту и Шију [14], [15].

Овај алгоритам је се може описати на следећи начин.

1. Иницијализује се почетна популација честица са нормалном расподелом на целом скупу решења.

2. Свакој честици се даје почетна брзина и правац кретања кроз скуп решења.

3. Одреди се најбоља честица.

4. Свакој честици се коригује правац кретања на основу околине и неке случајне вредности. Додатно се поправи правац кретања у зависности од позиције најбољег решења (честице), односно њено кретање се усмери ка њему.

5. Кораци 2, 3, 4 се понављају док се не постигне решење задовољавајућег квалитета, извршен је максималан дозвољен број корака.

Оптимизација пчелињим колонија

Колонија пчела је у стању да сакупљаја полен са већег броја поља цвећа, која су на великим одстојањима од свог пчелињака. Она је у стању да нађе поља са великом количином нектара, и она са којих је могуће са минималним напором прикупљати нектар. Ово се постиже на следеће начин. Прво се пошаљу пчеле извиђивачи, да пронађу обећавајућа поља. Они случајним поступком иду од једног до другог поља. По повратку у кошницу обавештавају остатак колоније о правцу, удаљености и квалитету поља. На основу ових информација се шаље више или мање нових пчела на неко поље. Нове пчеле при скупљању нектара прате ниво који остаје у цвећу и враћају ову информацију у колонију што омогућава да се смањи број пчела које иду ка лошијим пољима. На овај начин се омогућује колонији да на брз и ефикасан начин прикупља нектар.

Овај процес је инспирисао настанак алгоритма оптимизације пчелињим колонијама у решавању комбинаторних проблема [16]. Овај алгоритам се може описати следећим псеудо кодом

Иницијализовати почетну популацију случајних решења
Израчунати квалитет решења
while (Није Критеријум Прекида) **do**
  Изабери регионе за претрагу околине на основу квалитета
  Пошаљи пчеле у регионе сходно постигнутом квалитету решења
  Изабери пчелу за процену квалитета у сваком од региона
  Пошаљи преостале пчеле у случајну претрагу и оцени њихов квалитет
end while

Овај нов метод се показао јако успешна и примењен је на великом низу проблема попут трасирања возила [17], планирању послова [18], [19].

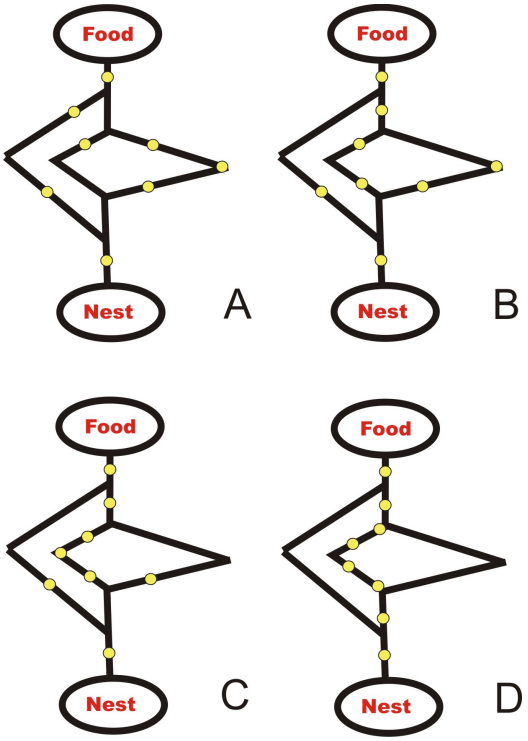
Глава 2

Оптимизација мрављим колонијама

Оптимизација мрављим колонијама (ОМК) је још једна метахеуристика која се заснива на интелигенцији ројева. Она своју инспирацију налази у интелигентном понашању мрављих колонија. Сваки од појединачних мрава у колонији има малу могућност расуђивања и у опште информација о околном свету, међутим целокупна колонија испољава тако рећи интелигентно понашање. Један пример оваквог понашања је налажење кратких путева од мравињака до извора хране. Колонија ово постиже тако што мрави међусобно сарађују, и на тај начин могу да реше комплексан проблем попут налажења најкраћег пута.

У колонији мрава, комуникација се врши преко феромона, хемикалије коју мрави могу да оставе а друге јединке касније да препознају. Употреба феромона ће бити објашњена на примеру како мрави налазе најкраћи пут од гнезда до хране. Ако посматрамо понашање колоније приликом потраге за храном, приметићемо да се у почетној фази мрави крећу доста хаотичним путањама. Како време пролази све већи број њих се креће најкраћом путањом од гнезда до хране, а у касној фази скоро сви.

Овакво понашање је последица тога што путања коју ће изабрати неки мрав зависи од тога колико на одређеном путу има феромона, прецизније мрави више "воле" стазе које поседују више феромона. Сваки мрав приликом тражења хране на својој путањи оставља феромон који полако испарава. Овакво испаравање има за последицу да на дужим стазама, за које је мраву потребно више времена да прође, испари више феромона него на краћим. Тако да следећи пут када мрав бира путању веће су шансе да изабере краћи пут. То за последицу има да се кратке путање све више користе и почињу да имају огромну количину феромона. Овакво понашање се може видети на Слици 2.1.



Слика 2.1: Илустрација понашања колоније мрава кроз време приликом потраге за храном

Предходно понашање је први пут претворено у математички модел од стране Марка Дорига [20] на проблему трговачког путника (ПТП). ПТП се дефинише на графу $G(V, E)$, код ког је свакој ивици $e \in E$, додељена тежина (дужина). Идеја је наћи минималану кружну путању, односно пермутацију скупа V такву да је скуп одговарајућих ивица (оне које повезује узастопне елементе) има минималну тежину.

За креирање симулације мрављe колоније за решавање ПТП потребно је дефинисати вештачке мраве и њихово понашање. Сваки овакав мрав би почео своју путању у случајно изабраном граду, односно чвору графа, и градио своје кружно путовање. Овакву путању гради на основу пробабилистичке функције која зависити од количине феромона који постоји на ивицама и хеуристичке функције која одређује примамљивост ивице. Хеуристичка функција ће бити дужина ивице. Мрави преферирају ивице које имају велику количину феромона и кратке су. На почетку алгоритма, m вештачких мрава је случајно распоређено у изабране градове. При сваком кораку креирања решења, односно при избору следећег града, они мењају количину феромона на изабраној ивици. Ову промену вред-

ности феромона ћемо назвати локално правило ажурирања. Када сви мрави креирају своје путање, потребно је најкраће путање "наградити" односно учинити их пожељнијим повећањем количине феромона на ивицама које им припадају. Овај поступак ћемо назвати глобалним правилом ажурирања.

Да би створили математички модел мрављe колоније који решава ПТП потребно је дефинисати следеће компоненте:

1. Правило избора које појединачни мрави користе за избор следећег дела пута.

2. Глобално правилу ажурирања феромонског трага.

3. Локално правило ажурирања које симулира испаравање феромона.

Вештачки мрави поседују неколико особина које не постоје код природних мрава али су погодне за решавање ПТП. Прецизније то су познавање даљине суседних градова, затим поседују унутрашњу меморију која чува информацију о градовима који су већ били посећени, што им омогућује да не бирају поново исте градове.

Претпоставимо да се тренутно k-ти мрав налази у граду r и желимо да дефинишемо како ће он изабрати следећи град s. Функција избора је представљена следећом једначином

$$s = \begin{cases} \arg\max\limits_{u \notin M_k} \left\{ {\tau_{rs}}^{\alpha} {\eta_{rs}}^{\beta} \right\} & , q \leq q_0 \\ S & , q > q_0 \end{cases} \tag{2.1}$$

У једначини 2.1 τ_{rs} је количина феромона која тренутно стоји на ивици rs. η_{rs} је хеуристичка функција која је обрнуто пропорцијална дужини ивице rs. Параметри α и β нам служе да би се одредила важност (утицај) феромона и хеуристике на избор следећег града. q је случајна промењива која има униформну расподелу на интервалу $[0, 1]$, а q_0 је фиксиран параметар из истог интервала. q и q_0 нам служе да се усклади однос истраживање и експлоатације приликом тражења решења. S је случајна промењива која има расподелу дату следећом једначином, која фаворизује кратке ивице са доста феромона

$$p_{rs}^k = \begin{cases} \frac{{\tau_{rs}}^{\alpha} {\eta_{rs}}^{\beta}}{\sum\limits_{u \notin M_k} {\tau_{ru}}^{\alpha} {\eta_{ru}}^{\beta}} & , s \notin M_k \\ 0 & , s \in M_k \end{cases} \tag{2.2}$$

У једначини 2.2 p_{rs}^k представља вероватноћу да k-ти мрав који је у граду r изабере да се помери у град s ако је претходно посетио градове у M_k.

Као што је већ поменуто феромонски траг треба ажурирати на локалном и глобалном нивоу. Глобално ажурирање се користи да награди ивице које припадају добрим путевима. Ово се постиже тако што мрави по завршетку своје путање, одлажу феромон на ивицама који су у њеном саставу у количини $\Delta\tau^k$ чија вредност је обрнуто пропорцијална дужини целог пута (остале ивице остају не промењене). Овај процес се користи за симулацију процеса акумулирања феромона на добрим путањама, која је код реалних мрава последица повезаности дужине пута и времена потребног за њен прелазак. Ово правило ажурирања је дато следећом једначином

$$\tau_{ij} = (1-\gamma)\tau_{ij} + \gamma\Delta\tau^k \qquad ,\forall ij \in B^k \tag{2.3}$$

Једначина 2.3 нам даје формулу како се мења феромонски траг τ_{ij}. γ је фиксирани параметар који одређује утицај нових решења. $\Delta\tau^k$ је квалитет решења коју је постигао k-ти мрав. B^k је скуп који садржи све ивице које се налазе у решењу које је конструисао k-ти мрав. У практичној примени ОМК не остављају сви мрави феромон већ једино онај који је постигао најбоље решење или нека друга варијација, о чему ће бити више речи у наредној секцији.

Коначно нам остаје да дефинишемо локално правило ажурирања које има сврху да не дозволи стварања ивица које имају јако пуно феромона и као последица тога бивају изабране од стране свих мрава. Овај тип ажурирања је симулација процеса испаравања феромона који постоји у природи. Сваки пут када неки мрав изабере ивицу rs количина феромона на тој ивици се смањује применом следеће формуле

$$\tau_{ij} = (1-\delta)\tau_{ij} + \delta\tau_0 \tag{2.4}$$

У једначини 2.4 δ је предефинисана константа из интервала $[0,1]$, τ_0 је квалитет решења које је постигнуто коришћењем грамзивог алгоритма за решавање ПТП у ком је хеуристичка функција била дужина ивице.

ОМК која је дефинисана на претходно приказан начин има следеће добре карактеристика:

- Свестрана је, у смислу да се лако може проширити на сличне верзије истог проблема. Пример је ПТП у асиметричној верзију.
- Робусна је, у смислу да се уз помоћ минималних измена може применити на друге оптимизационе проблеме попут проблема распореда, покривања и других.

- Алгоритам се заснива на популацији. Ова карактеристика је интересантна зато што омогућује експлотацију позитивних повратних информација као механизам претрага. Такође интуитивно нас наводи на идеје паралелне имплементације.

2.1 Варијације алгоритма ОМК

Као што је предходно поменуто, избор мрава и начин на који ће они остављати феромон јако утиче на перформансе ОМК. Доринго је на почетку представио три варијације алгоритма: мрављи циклус (Ant Cycle), квантитативни мрави (Ant Quantity) и мрављa густина (Ant Density) [21]. Касније је развијен низ варијације ОМК које унапређују основну идеју попут рангирног мрављег система [22], система мрављe колоније [23], елитистичка стратегија [24], мин макс мрављи систем (MMAC) [25]. У овој секцији ће бити дат детаљан опис најзначајнијих варијација, илустрован на примеру ПТП.

Мравњи систем је најједноставнија верзија овог алгоритма и у њему мрави једни за другим рачунају решење проблема. Након тога сваки од њих у зависности од квалитета нађеног решења ажурира феромонску матрицу користеће формуле описане у предходној секцији. У овом облику не постоји локано ажурирање матрице, односно симулације испаравања феромона. Мрави приликом креирања решења, бирају градове увек на основу вероватноће и никад не користе само хеуристику. Мравњи систем се може описати псеудо кодом приказаним на слици 2.2

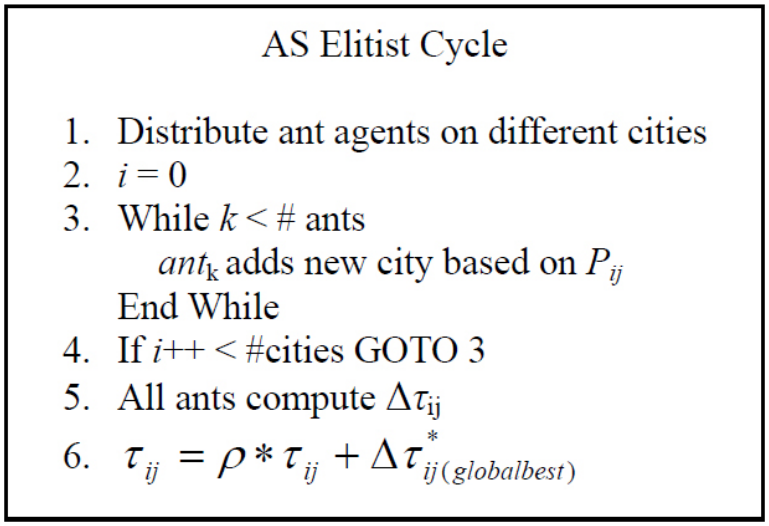
Ant System (AS) Cycle

1. Distribute ant agents on different cities
2. $i = 0$
3. While $k <$ # ants
   add an new city based on probability
   k++
   End While
4. If i++ < #cities GOTO 3
5. All ants compute $\Delta\tau_{ij}$
6. $\tau_{ij} = \rho * \tau_{ij} + \Delta\tau_{ij}$

Слика 2.2: Псеудо код за мрављи систем. Слика преузета из [26]

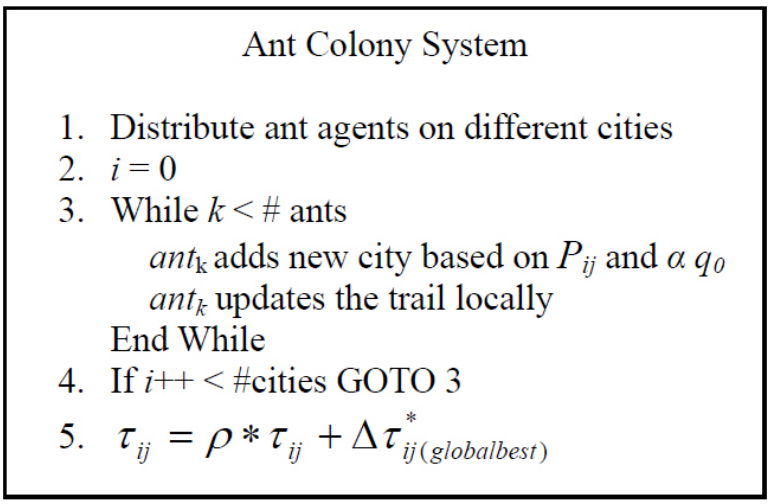
Мрављи систем има велику слабост да није довољно грамзив односно простор у ком се тражи решење остаје велик током целог алгоритма. Проблем овог приступа је то што најбоље решење није довољно награђено.

Грамзивији облик ОМК је елитистичка стратегија. У овој варијацији сви мрави остављају феромон након креирања решења али најбоље нађено решење се додатно појачава приликом сваке итерације колоније. Псеудо код елитистичке стратегије је приказана у Слици 2.3



Слика 2.3: Псеудо код за елитистичку стратегију. Слика преузета из [26]

Систем мравље колоније је најстандарднији облик ОМК. У њему феромон не остављају сви мрави већ једино глобално најбоље решење. Он уводи коришћење локалног ажурирања. Локално ажурирање се изводи тако што сваки мрав након избора следећег града у својој путањи мења количину феромона на ивици коју тренутно користи. Још једна измена у односу на предходне алгоритме је да мрави не креирају своје решења један за другим у смислу да тек након што је један мрав завршио своју путању следећи започиње. У систему мравље колоније решења свих мрава у колоније се креирају напоредо односно сви мрави изаберу први град, затим сви изаберу други и тако даље док сви не креирају сопствено решење. У систему мравље колоније је први пут уведено и коришћење параметра q_0 (описаног у предходној секције) који омогућује балансирање између истраживања и експлоатације у правилу избора за појединачне мраве. Псеудо код ове варијације се може видети на Слици 2.4



Слика 2.4: Псеудо код за систем мравље колоније. Слика преузета из [26]

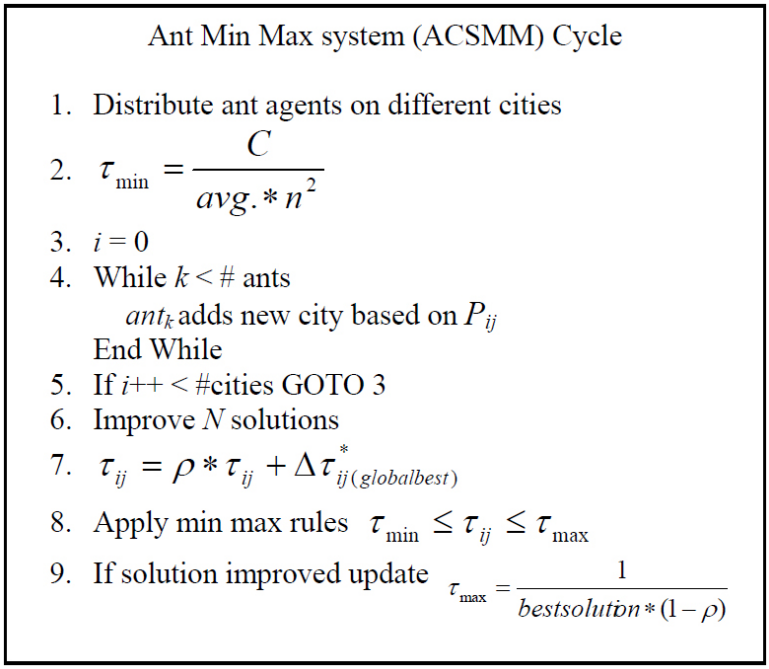
Проблем код система мравље колоније је појављивање ивица које имају јако велику количину феромона и постају саставни део скоро свих ново креираних решења. Други проблем је то што ивице које нису често биране временом губе све више феромона и у каснијој фази алгоритма јако тешко поново бивају саставни део нових решења. Решење ова два проблема је понуђено у мин макс мрављем систему. Идеја је ограничити опсег у коме се могу налазити вредности у феромонској матрици на неки интервал $[\tau_{min}, \tau_{max}]$. У случају ПТП вредности τ_{min} и τ_{max} се добијају из следећих формула

$$\tau_{max} = \frac{1}{L_{gb} * (1 - \gamma)} \tag{2.5}$$

$$\tau_{min} = \frac{c}{avg * n^2} \tag{2.6}$$

Приликом извршавања алгоритма τ_{max} се мења у зависности од квалитета глобално најбољег решења L_{gb} на начин представљен у једначини 2.5. τ_{min} се рачуна по формули датој у једначини 2.6. Његова вредност је фиксирана током целокупне претраге и зависи од квалитета решења које се добије употребом грамзивог алгоритма за ПТП.

Још једна варијација ОМК која је често у употреби је рангирани мрављи систем. Ова варијација је мање грамзива од ММАС-а. У њему већи број мрава оставља феромон, међутим њихов утицај варира од



Слика 2.5: Псеудо код за мин макс мрављи систем. Слика преузета из [26]

квалитета решења, глобално правило ове варијације је дато у следећој једначини.

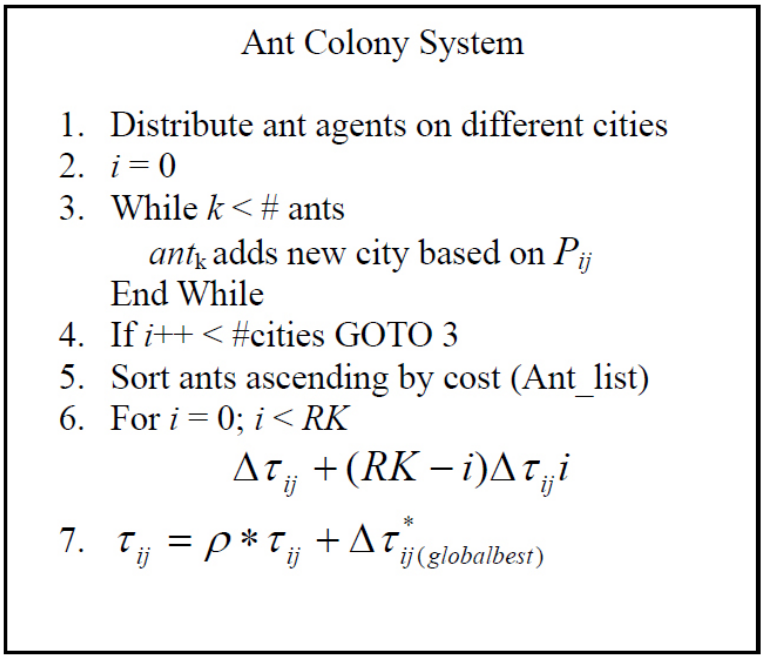
$$\tau_{ij} = (1 - \gamma\frac{r^k}{w})\tau_{ij} + \gamma\frac{r^k}{w}\Delta\tau^k \qquad ,\forall ij \in B^k \tag{2.7}$$

У једначини 2.7 w је број мрава који ће остављати феромон. r^k представља ранг решења које је креирао k-ти мрав. На овај начин се околина већег броја добрих решења истражује. Псеудо код ове варијације ОМК је дат на слици 2.6.

Важно је напоменути да не постоји варијација ОМК која је најбоља за све типове проблема већ перформансе зависе од особина проблема.

2.2 Хибридизације алгоритма ОМК

Ефикасност ОМК је могуће унапредити различитим хибридизација. Варијације ОМК, као што је предходно речено, се заснивају на промени неких делова основног алгоритма. Алгоритам се овим поступком чини више или мање грамзивим и на тај начин постижу боља решења. Хибри-



Слика 2.6: Псеудо код за рангирни мрављи систем. Слика преузета из [26]

дизације представљају далеко радикалнију промену ОМК. Два најчешће коришћена типа хибридизација су комбиновање са локалном претрагом или са неким другим комплекснијим алгоритмом.

Хибридизација локалном претрагом има следећи приступ. Када колонија нађе неко ново решење то решење се користи као почетна тачка неког алгоритма локалне претраге. На пример када се ОМК користи за решавање ПТП, нова решења које налази колоније могу се користити као почетне тачке 2-ОПТ [27], 3-ОПТ [28] или неке друге локалне претраге. Овакав приступ је успешно коришћен на великом броју проблема попут [29], [30]. И ако у највећем броју случајева, додавање локалне претраге на ОМК брзина налажења добрих решења и њихов квалитет бивају побољшани, показано је да локалне претраге могу да имају за последици заробљавање претраге у неком локалном минимуму [31].

Чест начин хибридизације ОМК је његово комбиновање са неким другим методом за решавање оптимизационих проблема. Идеја овог приступа је да се искористе позитивне особине оба метода које се користе и постигну резултати који су бољи од употребе сваке од методи појединачно. ОМК је јако успешно комбинована са генетским алгоритмима [32], [33], оптимизацијомројевима честица и другим методима [34].

Оба представљена приступа унапређењу ОМК су се показала ефикасним у побољшању перформанси али често нису погодни за имплементацију. У случају хибридизација оваквог типа, проблем је што је неопходно развити и програмски имплементирати два независна алгоритма један за ОМК и други за метод са којим се он комбинује. У случају једноставнијих проблема попут ПТП повећање комплексности није велика потешкоћа, међутим у случају проблема који су у својој основи сложени, овакве хибридизације могу да постану изузетно тешке за програмирање. Хибридизације иако су генерално ефикасне, остављају утисак да се на овај начин ОМК на неприродан начин унапређује.

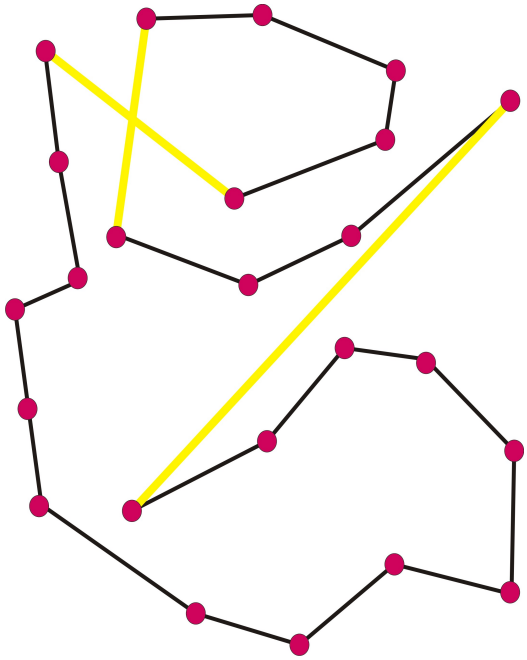
Глава 3

Побољшање метода мрављих колонија за решавање проблема путујећег трговца (ПТП) методом корекције феромонског трага

3.1 Метод сумњивих елемената

У овом поглављу ће бити представљен нов тип хибридизације за унапређење ОМК на ПТП. Унапређење се заснива на увођењу метода који омогућује ОМК да се ослободи заробљавања у локално оптималним решењима, односно спречава стагнацију у потрази за глобалним оптимумом. Овај метод врши корекцију феромонског трага. Калкулација корекције се заснива на анализи особина до сада најбољег нађеног решења. Идеја овог метода је да се смањи вероватноћа налажења ивица са непожељним особинама у оптималном решењу. У даљем делу ове секције ће бити показано да овакав метод може на једноставан начин да се имплементира, није скуп што се тиче процесорског времена и унапређује перформансе MMAC верзије ОМК. Он такође постиже боље резултате од оптимизацијеројевима честица на овом проблему.

Анализом решења која се постижу употребом ОМК за решавање ПТП у случају заробљавања у локалним оптимумима, може се уочити које корекције је потребно учинити да би се поправио квалитет решења. Прецизније, лако је могуће уочити шта не треба да се налази у најкраћој путањи. Постоје два врло једноставна критеријума: јако дугачке



Слика 3.1: Пример непожељних ивица код локалног оптимума проблема трговачког путника

ивице и укрштајуће ивице ће се са јако малом вероватноћом налазити у оптималној путањи. Пример непожељних ивица се може видети на слици 3.1. Следећи корак је како на једноставан начин, који ће у што је могуће мањој мери променити алгоритам ОМК, уклонити ове сумњиве елементе из претраге. Решење је да се јако смањи количина феромона на случајно изабраним сумњивим ивицама које припадају тренутно најбољем нађеном решењу и након тога дозволити колонији да настави потрагу за оптималном путањом.

Корекција је подељена на два дела: у првом се узимају у обзир дужине ивица, док је други део повезан са укрштајућим ивицама. Све укрштајуће ивице ће се сматрати јако сумњивим и на њима ће бити извршена корекција феромонског трага. Ивице код којих ће се корекција феромона вршити као последица њихове дужине ће бити дефинисан на следеци начин. Први корак је дефинисање хеуристике која ће нам одређивати непожељност односно сумњивост ивице. Хеуристичка функција ће бити дужина ивице као у једначини 3.1.

$$Sus(rs) = length(rs) \tag{3.1}$$

Користећи ову хеуристику дефинишемо вератноћу да ивица rs буде

изабрана за корекцију феромонског трага следећом једначином.

$$p_{selected}(rs) = \frac{RK - RankSusp(rs)}{2 * RK} \tag{3.2}$$

У једначини 3.2 се уместо коришћења вреднсти хеуристике Sus за ивице, користимо функцију $RankSusp$ која престставља ранг ивице на основу сумњивости. RK је максимални број ивица који се узима у обзир за корекцију. Последњи корак је смањивање вредности феромонског трага на изабраним ивицама:

$$(\forall(rs) \in Selected)(\tau_{rs} = \delta\tau_{rs}) \tag{3.3}$$

У једначини 3.3 τ_{rs} представља количину феромона депоновану на ивици rs, а δ је фиксирана мала костанта која се користи да би се значајно смањила количина феромона на ивици rs.

Овакав вид избора сумњивих ивица (оне над којима се врши корекција) није у потпуности ефикасна из разлога што ће се оне бирати из исте група ивица све док се не пронађе неко ново најбоље решење. Да би се оваква појава избегала уводи се нови унапређени критеријум сумњивости:

$$CorSusp(rs) = Susp(rs) * ExSusep(rs) \tag{3.4}$$

Унапређење се састоји у праћењу које су ивице и колико често биране и преферирање избора нових ивица. Да би ово постигли, додајемо нови низ $ExSuspect$ чији сви елементи на почетку алгоритма имају вредност 1. Ако је ивица rs изабрана, извршава се следећа корекција:

$$ExSusp(rs) = ExSusp(rs) * \lambda * \frac{len(rs)}{len(MaxEdge)} \tag{3.5}$$

У једначини 3.5 $\lambda \in (0, 1)$ је фиксиран параметар а $MaxEdge$ је најдужа ивица која се појављује у до сада најбољем нађеном решењу. Разломак у једнакости постоји да би количина укоњене сумњивост била сразмерна дужини ивице. На овај начин је омогућено да се дугачке ивице чешће бирају и након великог броја извршених корекција. У случају да је ново најбоље решење нађено све вредности низа $ExSuspect$ се ресетују на вредност 1.

Могуће је увести комплекснији критеријум за сумњивост ивица који боље анализира особине најбољег нађеног решења. Циљ приступа корекције феромона је да се на једноставан начин унапреди ОМК, из тог разлога је изабрана релативно проста хеуристика за коју су експерименти показали да унапређује ОМК.

Описан метод корекције феромона се користи у случају да је дошло до стагнације у потрази за решењем од стране колоније мрава. Критеријум за стагнацију који се корисити је следећи: Стагнација је наступила у случају да није дошло до промене најбољег решења у последњих n итерација. Представљено унапређења ОМК је дато у следећем псеудо коду:

```
 Reset Solution for AllAnts
while(! AllAntsFinished) do
   for AllAnts do if(AntNotFinished) do begin
      Add new edge rs to the solution based on
        probability
      Local update rule for rs
   end for;
end while;
Compute Global Update
if (Stagnation) and (UsingSuspisionImprovement)
   Use SuspisionCorrectionMethod
```

Важна особина оваквог унапређења ОМК је да није скупо у смислу времена извршавања. Време израчунавања корекције је занемарљиво у односу на остали део алгоритма ОМК. У наставку дајемо следећу оцену времена израчунавања.

Време потребно једном мраву да креира решење је реда величине n^2. Ова вредност је последица тога што је мраву потребно n корака да би креирао целокупно решење (на сваком кораку се додаје по један чвор) а на сваком од њих мора у просеку да рекалкулише $n/2$ вероватноћа избора за чворове и корекција феромонског трага.

Време израчунавања корекције феромонског трага је такође реда n^2. Корекција се састоји из три корака од којих први има доминантан утицај на време израчунавање. Први корак је провера које се ивице односно парови чворова укрштају за шта је потребно $(n-1)*n/2 \sim n^2$ операција. Важно је напоменути да се овај корак чак и не извршава сваки пут када је наступина стагнација већ једино у случају да је дошло до промене најбољег нађеног решења од последње корекције. У другом корак је потребно сортирати све ивице које се налазе у решењу за шта нам је асимптотски потребно $n * log(n)$ операција и коначно за избор сумњивих елемената је потребно још n операција. Значи тотално време које је потребно за израчунавање је $n^2 + n * log(n) + n \sim n^2$, или чак мање у случајевима када није потребно рачунати први терм.

Приликом практичне употребе приказаног метода за корекцију феромона, време потребно за његово израчунавање је еквивалентно (или

мање) времену које је потребно једном мраву да креира решење. Пошто се оваква корекција врши веома ретко, у извршеним тестови критеријум стагнације је био 30 итерације колоније без промене најбољег решења, а у свакој итерацији учествује 10 мрава, корекција се користи 1/300 укупног времена рачунања односно свега 0.33%.

3.2 Тестови и резултати

У овој секцији су представљени резултати употребе унапређене верзије ОМК на тест проблемима из бенчмарк библиотеке TSPLIB [35] (http:// www2. iwr. uni-heidelberg. de /groups /comopt /software /TSPLIB95 /tsp/) која представља стандард за ПТП. Сви тестови су извршени на симетричној верзији ПТП. Унапређена верзија ОМК се пореди са алгоритмом оптимизацијеројевима честица приказаном у раду [36], и са једноставном верзијом ОМК у MMAC верзији. За сваки од три метода поредимо најбоље нађено решење, квалитет просечног решења и стандардну девијације решења за већи број експеримената . Резултати за ОМК су добијени коришћењем софтверског пакета који је представњен у тези. Резултати су приказани у табели 3.1, колоне 3, 5 и 7 (Best) приказује најбоље нађено решење, док колоне 4, 6, и 8 (AvgError) приказују квалитет просечног решења за све експерименте дате следећом формулом:

$$AvgErr = (Avg - Opt)/Opt * 100\% \tag{3.6}$$

У првих пет тест проблема, резултати су добијени за 100 покретања алгоритма ОМК да би резултати били упоредиви са оним из [36]. У тестове је такође укључено и 6 већих проблема код којих смо анализирали резултате покретања 10 колонија. Смањен је број тест колонија јер није било значајнијих квалитативних промена резултата. За сваку од тест колонија је конструисано 200,000 тура односно свака колонија је извршила 20,000 итерација. Све мравље колоније су имале исте параметре са следећим вредностима: $\alpha = 4$ и $\beta = 1$ параметри који контролишу утицај феромона и хеуристике (дужине ивица) на транзиционо правило, $q_0 = 0.9$ одређује однос истраживања/експлоатације, $\gamma = 0.1$ и $\delta = 0.1$ одређују глобално и локално правило ажурирања, почетна вредност феромона у феромонској матрици је дата као $\tau_0 = 1/n * L_{nn}$ где је L_{nn} дужина решења које је добијено коришћењем грамзивог алгоритма заснованог на хеуристичкој функцији најближег комшије. Параметри за дефинисање метода корекције феромона засноване на сумњивости елемената су имали следеће вредности $RK = ProblemSize/10$, $\lambda = 0.9$, $\delta = 0.1$ и $n = 30$. У тестовима смо користили по 10 колонија за основни

Табела 3.1: Поређење оптимизације ројевима честицам (PSO), MMAC варијација ОМК (ACO) и MMAC комбинованог са искључивањем сумњивих елемената (SEE), најбоље решење (Best), просечна грешка (AvgErr)

Problem	OPT	PSO		ACO		SEE	
		Best	AvgErr(%)	Best	AvgErr(%)	Best	AvgErr(%)
eil51	426	427	2.58	427	0.52	427	0.23
berlin52	7542	7542	3.85	7542	0.28	7542	0.13
st70	675	675	3.34	676	1.50	675	1.36
eil76	538	546	4.17	538	1.21	538	1.19
pr76	108159	108280	3.82	108359	2.94	108358	2.62
kroa100	21282	-	-	21282	0.77	21282	0.72
lin105	14379	-	-	14379	0.38	14379	0.38
pr124	59030	-	-	59385	0.84	59030	0.72
pr136	96772	-	-	96785	1.09	96781	0.70
u159	42080	-	-	42080	1.27	42080	0.45
kroa200	29368	-	-	29532	1.21	29490	0.93

ОМК и ОМК унапређен методом за корекцију феромонског трага. У оба случаја су коришћени вредности од 0 до 9 за вредности семена генератора псеудослучајних бројева. Оваквим избором семена је омогућено да се на добар начин види ефекат корекције феромона. Део алгоритма који имплементира корекцију феромонског трага је имао засебни генератор случајних бројева шта омогућује да мрави имају што сличније понашање у оба два теста.

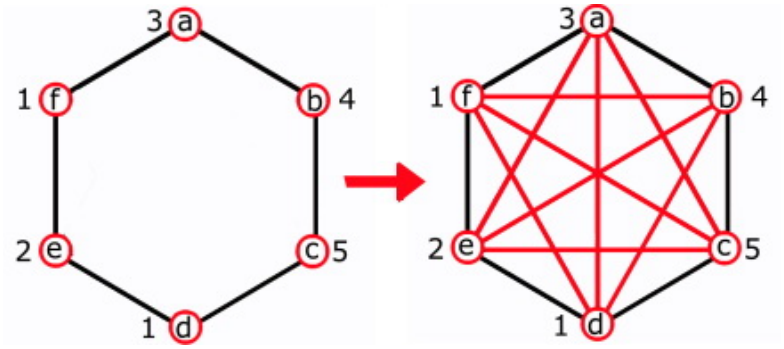
У првих пет проблема који су исти као у раду [36], постигнути резултати нашом методом су бољи када се пореде са методом ројева честица. Најбоље постигнуто решење је једнако у три случаја, боље у једно и лошије у једном случају. Када се пореде резултати просечног квалитета резулата представљена метода поправља значајно квалитет резултата у свих пет примера. На тест примерима је извршено и поређење са ОМК и ОМК унапређене корекцијом феромона. У овом случају је најбоље решење унапређене ОМК дало боље резултате у 5 случаја, и ни у једном није смањен квалитет решења. Када се пореди просечан квалитет решења унапређен метод је постигао боља решења у 10 случајева и био једнак у само једном.

Глава 4

Побољшавање решавања проблема минималног тежинског чворног покривача коришћењем оптимизације мрављим колонијама

Један од класичних проблема из теорије графова је проблем минималног тежинског чворног покривача (ПМТЧП). Проблем је дефинисан на неорјентисаном графу $G = (V, E)$, где V представља скуп чворова а E је скуп ивица графа, а сваком чвору у графу је додељена тежина. Чворни покривач графа G је подскуп скупа чворова $V' \subseteq V$ такав да има особину да за сваку ивицу $(v1, v2) \in E$ бар један од чворова $v1$ или $v2$ pripada V'. Минимални тежински чворни покривач графа је покривач који има најмању суму тежина чворова који му припадају. Предходно је показано да је овај проблем НП-компетан чак и у случају да постоји рестрикција да су све тежине једнаке и да је максимални степен (број конекција) чвора једнак 3 [37].

Велик број проблема из свакодневног живота се може свести на ову форму. Један пример би могао бити постављање депонија у околини града. За овај тежак НП-комплетан проблем у случају већих инстанци проблема није могуће наћи оптимално решење у израчунљивом времену, па је зато развијен низ алгоритама који му налазе субоптимална решења. Доказано је да за ПМТЧП, у верзији са рестрикцијама званој реални чворни покривач, у ком је укупна тежина решења ограничена са k и сваки чвор има тежину већу или једнаку 1, решење може бити нађено са асимптотским временом $O(1.3954^k + kn)$, gde je $n = |V|$, или $O(1.3788^k +$



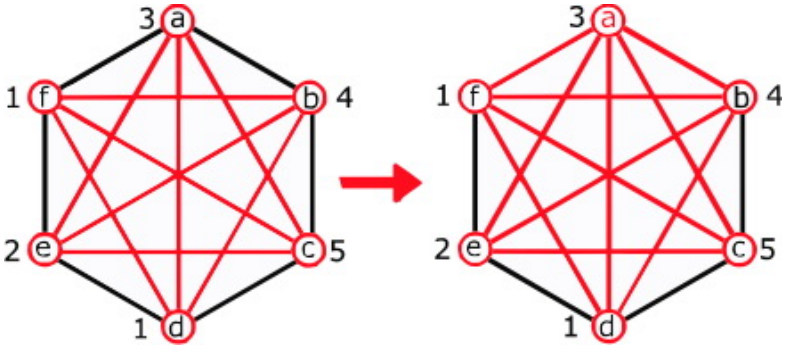
Слика 4.1: Претварање почетног графа у потпуно повезани код ког ивице имају тежине. Црне ивице имају тежину 1, црвене 0

kn) у случају да се употреба меморије повећава споро експоненцијално [38]. У случају да су једине дозвољене тежине позитивни целобројни бројеви проблем је могуће решити истом брзином као верзија проблема из које су искључене тежине за $O(1.2738^k + kn)$. И у овом случају се експоненцијало повећава употреба меморије [39], [40]. Из ових разлога је развијен низ методи које рачунају решења која се близу оптималном. Неке од њих су грамзиви алгоритми [41], [42], генетски алгоритми [43], метахеуристике засноване на гравитицији [44]...

4.1 Решавање ПМТЧП коришћењем ОМК

Примена ОМК на ПМТЧП има две значајне разлике у односу на њену примену на ПТП. У случају ПТП решење је пермутација скупа градова. Код ПМТЧП решење је подскуп скупа чворова графа у ком је редослед неважан. У ПТП хеуристичка функција је константна и представља раздаљину између градова и не мења се током креирања путање. Супротно у случају ПМТЧП хеуристичка функција представља однос тежине чвора и тренутног степена чвора (број ивица које садрже чвор а нису већ покривене чворовима који су већ у конструисаном делу решења), она је динамична јер како се нови чворови додају решењу све већи број ивица бива покривен. Ове две разлике утичу на основни алгоритам у два правца: мрави остављају феромон на чворовима а не на ивицама, а хеуристичка функција се коригује током креирања решења.

ПМТЧП је претходно успешно решаван и методом ОМК од стране Шјуа [45]. Постигнути су бољи резултати него методама локалне претраге попут табу претраге и симулираног каљења. Шјуов метод ће бити представљен у даљем делу ове секције. Први корак при решавању овог



Слика 4.2: Пример ажурирања графа приликом додавања нових чворова скупу решења. Приказано је ажурирања након додавања чвора a

проблема је наћи репрезентацију која нам омогућује брзо и једноставно рачунање динамичне хеуристичке функције. Пошто мрави током креирања решења приликом сваког корака могу да изаберу сваки чвор графа који није део до сада креираног парцијеног решења, природно је користити потпуно повезани граф $G_c(V, E_c)$ који је изведен из почетног графа G. Овај корак можемо видети на слици 4.1. Ивицама новог графа су додељене тежине 1 ако је постојао у G і 0 у супротном случају. Као што је већ поменуто граф је потребно ажурирати како се нови чворови додају скупу решења. Ово је урађено на следећи начин: када се чвор a дода, све ивице у G_c које су повезане са a се поставе на 0, овај корак је илустрован на слици 4.2.

Дефинишимо $G_{ck}(V, E_{ck})$ као стање графа након што је k чворова додато решењу, и одговарајућу функцију

$$\psi_k(i, j) = Value(E_{ck}(i, j)) \tag{4.1}$$

Сада је могуће дефинисати динамичну хеуристичку функцију на следећи начин

$$\eta_{jk} = \frac{\sum_{(i,j)\in E_c} \psi_k(i, j)}{w(j)} \tag{4.2}$$

У једначини 4.2 $w(j)$ представља тежину чвора j. Користећи хеуристику η_{jk} могуће је дефинисати транзиционо правило за појединачне мраве следећом једначином:

$$p_j^k = \begin{cases} 1 & , q > q_0 \ \ \& \ \ j = \arg\max\limits_{i \in A_k} \tau_i \eta_{ik}^\alpha \\ 0 & , q > q_0 \ \ \& \ \ j \neq \arg\max\limits_{i \in A_k} \tau_i \eta_{ik}^\alpha \\ \frac{\tau_j \eta_{jk}^\alpha}{\sum_{i \in A_k} \tau_j \eta_{ik}^\alpha} & , q \leq q_0 \end{cases} \tag{4.3}$$

У једначини 4.3 исто као у случају ПТП q_0 представља константу која дефинише однос истраживање/експлоатација, а q случајну промењиву која одређује који тип селекције ће се вршити у неком кораку креирања решења. A_k представља листу слободних, предходно небираних чворова, који су потенцијални кандидати. За разлику од ПТП транзиционо правило не зависи од последњег изабраног чвора и из овог разлога користимо τ_j уместо τ_{ij}. Да би у потпуности дефинисали систем мрављe колоније неопходно је још дефинисати глобално (када сви мрави заврше креирање свог решења) и локално (када мрав изабере појединачан чвор) правило ажурирања:

$$\Delta\tau_i = \begin{cases} 0 & , i \notin V' \\ \frac{1}{\sum_{j \in V'} w(j)} & , i \in V' \end{cases} \tag{4.4}$$

$$\tau_i = (1 - p)\tau_i + p\Delta\tau_i \tag{4.5}$$

Параметар $\Delta\tau_i$ представља меру квалитета глобално најбољег решења које представља подскуп V' које садржи чвор i. Он је дефинисан у једначини 4.4, који се потом се користи и за дефинисање глобалног правила ажурирања у једнакости 4.5. Параметар p овде представља утицај ново нађеног решења на промену феромонског трага.

Правило локалног ажурирања је дато следећом стандардном формом:

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0 \tag{4.6}$$

Мера квалитета за решење које је добијено коришћењем грамзивог алгоритма (у ком се чвор који има најбољу пропорцију степена и тежине бира) је изабран за вредност τ_0. Параметар φ се користи да би одредио колики је утицај локалног правила ажурирања.

4.2 Варијације алгоритма ОМК за ПМТЧП

Као што је већ напоменуто, перформансе ОМК могу се унапредити употребом различитих варијација основног алгоритма. Предходно је показан ефекат ових варијација па ПТП где су различити стратегије остављања феромона давале видно другакчији квалитет резултата. Из

овог разлога је урађена њихова имплементација на ПМТЧП, ради анализе резултата на овом значајно другачијем проблему.

Као што је већ поменуто главна разлика међу различитим варијацијама алгоритама ОМК је у начину на који мрави остављају феромон на чворовима односно у глобалном правилу ажурирања датом у једначини 4.5. Прецизније ова једначина остаје иста само се на различит начин рачуна вредност $\Delta\tau_i$. У наредном делу ове секције биће приказане различите формула за рачунање $\Delta\tau_i$ за различите варијације.

Мрављи систем (МС) је најосновнији облик имплементације ОМК, и у овој верзији алгоритма сви мрави остављају једнаку количину феромона. Он је дефинисан једначинама 4.5 и 4.7. У једначини 4.7 *Ants* је скуп свих решења које колонија креира при једној итерацији. V'_i представља решење које је мрав i креирао.

$$\forall i \in \bigcup_{V'_i \in Ants} V'_i$$

$$\triangle\ \tau_i = \sum\nolimits_{V_k' \in AntS} \frac{1}{\sum_{j \in V'_k} w(j)} \tag{4.7}$$

Елитистичка стратегија (ЕС) је варијација ОМК у којој је претрага додатно појачана око глобално најбољег решења. Ова варијације је иста као и мрављи систем само што се при свакој итерацији колоније депонује додатни феромон на чворове глобално најбољег решење. Рачунање $\Delta\tau_i$ у овој варијацији је дата једначином 4.8

$$\forall i \in V'_{gb} \cup \bigcup_{V'_i \in Ants} V'_i$$

$$\triangle\ \tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum\nolimits_{V_k' \in AntS} \frac{1}{\sum_{j \in V'_k} w(j)} \tag{4.8}$$

У једначини 4.8 V'_{gb} представља најбоље нађено решење. У неким верзијама ове варијације уместо глобално најбољег решења се користи најбоље решење нађено приликом једне итерације колоније.

Система мравље колоније (СМК) је још једна варијација ОМК у којој мрави не остављају аутоматски феромон. У свакој итерацији колоније, или другим речима када су сви мрави креирали своја решења, само глобално најбоље решење оставља феромон. На овај начин претрага се фокусира око глобално најбољег решења и самим тим се претрага чини грамзивијом. У овом случају $\Delta\tau_i$ се рачуна следећом формулом.

$$\triangle\ \tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)}, \forall i \in V'_{gb} \tag{4.9}$$

Као и у случају имплементације ОМК на ПТП и у случају ПМТЧП долази до појаве да се појављују делови глобално најбољег решења имају јако велику количину феромона и као последица тога бивају бирани од стране свих мрава. Као и у случају ПТП и приликом решавања овог проблема је могуће адаптирати Мин макс систем мрављe (ММАС) колоније. Ово се постиже тако што се ограничи распон у ком могу да се налазе вредности τ_i на неку област $[\tau_{min}, \tau_{max}]$. τ_{max} је и овом случају потребно рачунати динамички на основу глобално најбољег решења и то је је учињено по формули датој у једначини 4.10. У овој једанчини p има исту вредност као и у глобалном правилу ажурирања.

$$\tau_{\max} = \frac{1}{(1-p)} \triangle \tau_{gb} \tag{4.10}$$

τ_{min} се рачуна једном и то на почетку алгоритма формулом 4.11. avg представља просечан број чворова који може бити изабран на сваком кораку, p_{best} је вероватноћа да глобално оптимално решење буде нађено. τ_0 је иницијална вредност феромона.

$$\tau_{\min} = \frac{\tau_0(1 - \sqrt[n]{p_{best}})}{(avg-1)\sqrt[n]{p_{best}}} \tag{4.11}$$

Рангирани мрављи систем (РМС) је варијација ОМК у којој осим квалитета (у смислу бројне вредности) решења се такође користи и ранг решења R. Ранг је дефинисан на основу квалитета решења које су мрави у колонији креирали приликом исте итерације колоније. Ова варијација је дефинисана једначином 4.12

$$BRank = \{V | (R(V) < RK) \wedge (V \in AntS)\}$$
$$\forall i \in V'_{gb} \cup \bigcup_{V'_i \in BRank} V'_i$$
$$\triangle \tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum_{V_k' \in BRank} \frac{(RK - R(V))}{RK} \frac{1}{\sum_{j \in V'_k} w(j)} \tag{4.12}$$

Приликом имплементације ове варијације од изузетне важности је колико ће се најбољих нађених решења узимати у обзир приликом глобалног правила ажурирања. У једначини 4.12 RK се користи ради дефинисања колико ће најбоље рангираних мрава утицати на феромонски траг. Овај параметар се обично одређује од стране корисника и његова вредност јако утиче на ефикасност алгоритма. У екстремном случају када је вредност RK једнака 0, рангирани систем мрављe колоније се претвара у систем мрављe колоније.

4.3 Анализа проблема стагнације

Алгоритам објашњен у секцији описа ОМК одговара систему мрављe колоније. У овој верзији алгоритма ОМК једино глобално најбоље решење (у неким варијацијама само најбоље решење итерације) оставља феромон. На овај начин се повећава ефикасност основне верзије ОМК тако што се интензивира претрага у околини најбољег решења, или другим речима претрага се учини грамзивијом. Један од проблема оваквог приступа је рана стагнација алгоритма. Ово је последица чињенице да потенцијални делови решења која нису близу до сада најбољег нађеног решења губе све више феромона при свакој итерације услед његовог испаравања. После дужег времена извршавања алгоритма постаје скоро немогуће да мрави изаберу овакве чворове. Овај проблем је у случају ПМТЧП још већи у поређењу са ПТП јер вредности феромона се спуштају до нуле.

Као што је већ поменуто ММАС је унапређење система мрављe колоније који одговара предходно представљеном приступу имплементације ОМК на ПМТЧП. Још једном би напоменуо да се код ММАС-а унапређење перформанси алгоритма постиже ограничавањем могућих вредности феромон на неки интервал $\tau_i \in [\tau_{min}, \tau_{max}]$. На овај начин се избегава појављивање изузетно популарних чворова који постају делови сваког решења, и да вредност феромона за неке чворове постане блиска нули шта за последицу има да они не могу више бити изабрани приликом креирања решења. И поред овог унапређења и даље долази до стагнације ОМК. Проблем је у томе што вредност τ_{min} мора да буде мала да би алгоритам остао грамзив. Међутим последица овога је да када вредност феромана за неки чвор дође до вредности τ_{min}, он у даљим итерацијама алгортима бива биран јако ретко. Или прецизније претрага се никад не интезивира у околини таквог чвора осим у случају да постане део глобално најбољег решења. Основно решење овог проблема је да се дода одређен критеријум за стагнацију алгоритма и у случају да је он задовољен да се ресетује феромонска матрица на почетну вредност као што је урађено у раду [46].

Изузетно интересантан приступ проблему стагнације представљен је од стране Вонга и Сиа под називом стратегија минималног феромонског прага (СМФП) (Minimum pheromone threshold strategy (MPTS)) чија је ефикасност показана на проблему квадричне доделе (quadric assignment problem) [47]. У овој стратегији ОМК се проширује додавањем вредности минималног прага феромона τ_{mt} који је ограничен са $\tau_{min} < \tau_{mt} < \tau_{max}$. На почетку алгоритма τ_{mt} додељује се одређена вредност која се адаптира током извршавања претраге у зависност од перформанси алгоритма. У пракси корекција вредности τ_{mt} се врши дељењем са неким

фиксном параметром k на сваких n итерација ОМК. Ова вредност се користи за ажурирање феромонског трага. Приликом извршавања претраге у случају да вредност τ_i постање мања од τ_{mt}, τ_i се додељује вредност τ_{max}. На овај начин се избегава реинцијализација феромонске матрице, а претрага простора решења се извршава на далеко систематичнији начин. У овом методу не долази до губитка информација које постоје у феромонском трагу а опет су сачуване добре особине ММАС.

4.4 Екстензија метода сумњивих елемената

У овој секцији ће бити представљен метод корекције феромонског трага који се користи за избегавање стагнације претраге ОМК. Идеја је заправо адаптација метода избацивања сумњивих елемената која је у предходном делу текста приказана за проблем ПТП. Основна идеја овог приступа је да се искористе неке информације о глобалном најбољем нађеном решењу и употребе се за корекцију феромонског трага. Прецизније идеја се састоји у дефинисању сумњивост елемената које припадају најбољем нађеном решењу или другим речима њихове пожељности у добрим решењима. Циљ овог приступа је да се претрага за оптималним решењем усмери у делове скупа решења у којима се налазе мање сумњиви покривајући скупови. Овакав ефекат се постиже тако што се вредност феромона јако смањи у сумњивим (непожељним) чворовима.

Пре представљања детаља примене ове методе на ПМТЧП биће указано на важне разлике у односу на проширење ММАС коришћењем СМФП и методе сумњивих елемената. Коришћењем ММАС, побољшавање перформанси ОМК се постиже повећањем разноликости претраге неселективно на цео простор решења, прво додавањем доње границе вредности феромона τ_{min}, а у случају стагнације ресетовањем феромонског трага. Када се ММАС прошири са СМФП избегава се реиницијализација феромонског трага провером да ли је вредност феромона τ_i за чвор i постала мања од вредности τ_{mt} и ако јесте се поставља на вредност τ_{max}. Овако изабрани чворови су додати малој групи популарних чворова које мрави често бирају. Велики проблем је што се оваквим поступком, чворови бирају на доста случајан начин јер је једини критеријум за повећање вредности феромона то што имају малу вредност τ_i. Резултат овог поступка је да се често чворови који не би требали да припадају добрим решењима поново појављују у потрази. Када се једном повећа вредност феромона која одговара чвору потребан је велики број итерација ОМК да он напусти популарну групу чворова. Супротно овом приступу када се користи концепт сумњивих елемената не додају се

нови чворови популарној групи већ се из ње избацују они који поседује неке непожељне карактеристике и на овај начин се ова група смањује. У наредним итерацијама мрави прво бирају елементе који припадају популарном скупу, а од тренутка кад таквих више нема бирају се они чворови који имају боље особине. На овај начин се претрага усмерава на области у којима су решења мање сумњива односно имају мање непожељних особина.

Приликом посматрања решења која су генерисана у првим итерацијама алгоритма ОМК за ПМТЧП, може се приметити да се у најбољим нађеним решењима често налазе чворови који немају никакав утицај. "Немање утицаја" се користи у смислу да ако би се ти чворови избацили из решења, преостали чворови би такође представљали покривач графа. Користећи ову обзервацију, развијен је следећи критеријум за рачунање сумњивости чворова који припадају решењу:

$$Sus(i, V') = w(i)NCE(V' \setminus \{i\}) \tag{4.13}$$

Сумњивост је дефинисана у једначини 4.13 функцијом Sus. $w(i)$ представља тежину која одговара чвору i. Функција NCE нам даје број ивица који поскуп скупа чворова графа $G(V, E)$ покрива. За ивицу e ћемо рећи да је покривена од стране скупа V' ако је задовољен следећи услов:

$$\exists(a \in V')(\exists b)((a, b) = e\,)\,) \tag{4.14}$$

Као што се може видети у једнакости 4.13, чвор i који има велику тежину $w(i)$ а скуп $V'/\{i\}$ покрива велик број ивица ће се сматрати јако сумњивим. Слично као у случају ПТП следећи корак је избор случајаног броја чворова (RK), који имају највећу вредност функције Sus. Вероватноћа да неки чвор i који припада до сада најбољем нађеном решењу буде изабран је дат следећом једначином:

$$p_i(selected) = \frac{RK - RankSus(i)}{RK} \tag{4.15}$$

У једначини 4.15 се уместо вредности функције Sus која одговара чвору, користи $RankSus$ који представља ранг чворова по вредности функције Sus. RK представља максимални број чворова који ће бити узети у обзир за корекцију. Последњи корак је спуштање вредности феромонског трага за изабране чворове.

$$\begin{aligned} &\forall i \in Selected \\ &\tau_i = \delta\tau_i \end{aligned} \tag{4.16}$$

Као и у случају ПТП појављује се проблем да ако би се користила само функција Sus за одређивање сумњивости елемената дошли би у ситуацију да се стално бирају исти елементи решења до тренутка када је нађено боље покривање. Из тог разлога се уводи унапређена верзија функције сумњивости на следећи начин

$$CorSus(i, V') = Sus(i, V') * ExSusepect(i) \tag{4.17}$$

Унапређење се састоји у праћењу колико је који чвор пута био биран и преферирању избора нових чворова. Да би ово постигли уводимо низ $ExSuspect$ чији се сви елементи на почетку алгоритма поставе на вредност 1. Када је елемент i изабран низ $ExSuspect$ се ажурира на следећи начин

$$\begin{aligned} &0 < \lambda < 1 \\ &ExSuspect(i) = ExSuspect(i) * \lambda \end{aligned} \tag{4.18}$$

У случају налажења новог најбољег решење, сви елементи низа $ExSuspect$, се реиницијализију на вредност 1. За критеријум сумњивости било је могуће користити и неки комплекснији приступ попут провере парова или тројки чворова, међутим овакви критеријуми нису изабрани из разлога избегавања повећања комплексности имплементације и времена израчунавања.

Последњи корак потребан да би се у потпуности описало унапређење овог типа је дефинисање критеријума стагнације. У случају ПМТЧП коришћен је следећи критеријум: стагнација је наступила када није дошло до промене глобалног најбољег решења n итерација, а најбоље решење постигнуто приликом итерације није промењено у последњих m итерација. Ако су ови критеријуми задовољени спроводи се предходно дефинисан метод корекције феромонског трага. Псеудо код који одговара једној итерацији ОМК са оваквом хибридизацијом је следећи:

```
Reset Graph Info
Reset Solution for all Ants

 while  (! AllAntsFinished)
   For All Ants
     If (AntNotFinished)
       begin
         add new vertex A to solution
         based on probability
         correct ants cover graph data
```

```
        calculate new heuristic
        local update rule for A
      End
    End for
  End while

Compute DeltaTauI
Compute TauI

If (Stagnation)and(UsingSuspisionImprovemnet)
  Use SuspisionCorrectionMethod
```

4.5 Тестови и резултати за варијације ОМК

У овој секцији је представљено напоредно поређење перформанси различитих варијација ОМК на ПМТЧП. Програм у ком су рађени експерименти поређења је писан у програмском језику C#, а детаљи самог софтвера ће бити дати у последњем поглављу ове тезе. Сама апликација је креирана као систем са проширењима који у себи садржи оквир који имплементира општи облик за сваку од различитих варијација алгоритма ОМК.

Табела 4.1: Број чворова 25, Број ивица 71, Вредност решења постигнутог употребом грамзивог алгоритма 1088, Максимални број итерација колоније 1250

Варијација	Најбоље решење	Просечно решење
MC	839	871.6
EC	779	834.7
PMC	787	856.3
CMK	779	827.1
MMAC	779	830.6

Резутати ових експеримената су дати у Табелама 4.1, 4.2, 4.3, 4.4,4.5, 4.6, 4.7, 4.8, 4.9 и 4.1. У њима варијација система мрављe колоније одговара алгоритму датом у Шјуовом раду [48]. У овом раду је показана ефикасност употребе ОМК за решевање овог проблема.

У експериментима је тестиран већи број различитих графова који садрже различит број чворова и ивица. У сваком од тестова колоније су имале по 10 мрава. Параметар истраживања је имао вредност $q_0 = 0.1$,

Табела 4.2: Број чворова 25, Број ивица 131, Вредност решења постигнутог употребом грамзивог алгоритма 1135, Максимални број итерација колоније 1250

Варијација	Најбоље решење	Просечно решење
MC	952	986.7
CMK	952	985.3
EC	952	994.1
PMC	952	957.6
MMAC	952	983.6

Табела 4.3: Број чворова 50, Број ивица 172, Вредност решења постигнутог употребом грамзивог алгоритма 2238, Максимални број итерација колоније 2000

Варијација	Најбоље решење	Просечно решење
MC	1736	1740.1
CMK	1554	1597.4
EC	1716	1744.3
PMC	1650	1694.7
MMC	1556	1589.3

параметар утицаја хеуристике $\alpha = 1$, параметри који одређују понашање глобалног и локалног правила ажурирања су имали следеће вредности $\varphi = 0.1$ и $p = 0.1$. У случају рангираног мрављег система коришћена је вредност $RK = 5$ за број најбољих резултата који ће се користити по итерацији колоније. Вредност τ_0 је израчуната из решења које је постигнуто употребом грамзивог алгоритма описаног на почетку овог поглавља. Приликом дефиниције MMAC коришћена је следећа вредност $p_{best} = 0.05$.

Табела 4.4: Број чворова 50, Број ивица 374, Вредност решења постигнутог употребом грамзивог алгоритма 2238, Максимални број итерација колоније 2000

Варијација	Најбоље решење	Просечно решење
MC	1861	1918.6
CMK	1833	1876.3
EC	1861	1885.8
PMC	1833	1872.2
MMAC	1833	1882.2

Табела 4.5: Број чворова 150, Број ивица 562, Вредност решења постигнутог употребом грамзивог алгоритма 6782, Максимални број итерација колоније 2000

Варијација	Најбоље решење	Просечно решење
MC	5827	5951.2
CMK	4920	5117.9
EC	5760	5912.1
PMC	5694	5802.2
MMAC	5002	5169.2

Табела 4.6: Број чворова 150, Број ивица 1470, Вредност решења постигнутог употребом грамзивог алгоритма 6834, Максимални број итерација колоније 2000

Варијација	Најбоље решење	Просечно решење
MC	6303	6354.7
CMK	5688	5872.5
EC	6284	6185.8
PMC	6156	6230.7
MMAC	5756	5889.6

За сваку од варијација, је извршено по десет независних тестова, односно покретања алгоритма ОМК за сваки од тест проблема. У сваком од тестова постављен је максимални дозвољен број итерација колоније и пореде се резултати постигнути до тог броја корака. Анализа је урађена поређењем најбољег нађеног решења, просечног решења и дистрибуције и дисперзије свих десет покретања колоније. Време потребно за извршавања различитих колонија се јако мало разликовало и из тог разлога је искључено из даље дискусије.

Табела 4.7: Број чворова 250, Број ивица 970, Вредност решења постигнутог употребом грамзивог алгоритма 10877, Максимални број итерација колоније 500

Варијација	Најбоље решење	Просечно решење
MC	9814	9961.2
CMK	8962	9259
EC	9958	10006.4
PMC	9686	9761
MMAC	9106	9269.2

Табела 4.8: Број чворова 250, Број ивица 8399, Вредност решења постигнутог употребом грамзивог алгоритма 11239, Максимални број итерација колоније 500

Варијација	Најбоље решење	Просечно решење
MC	10900	10968.4
CMK	10683	10747.4
EC	10915	10957.6
PMC	10882	10940.6
MMAC	10698	10763.2

Табела 4.9: Број чворова 500, Број ивица 18581, Вредност решења постигнутог употребом грамзивог алгоритма 22395, Максимални број итерација колоније 500

Варијација	Најбоље решење	Просечно решење
MC	21900	21997.6
CMK	21499	21558
EC	22014	22057.2
PMC	21963	22005.4
MMAC	21462	21624

Тест проблеми су генерисани на случајан начин. У њима су тежине чворова бирани на случајан начин из целобројног интервала $[20, 70]$ са униформном расподелом. Креирани су графови са 25, 50, 150, 250 и 500 чворова, за сваку од ових величина генерисани су два различита скупа ивица. У алгоритму за генерисање ивица, за сваки чвор креирало би се n ивица до случајно изабраних чворова. n је у случају табела 4.1, 4.3 , 4.5 био случајан број из интервалал $[1, 4]$, у табелама 4.2, 4.4, 4.6, 4.7 је био број из интервала $[1, 10]$, а у таблама 4.8, 4.9 из интервала $[10, 20]$

Табела 4.10: Број чворова 500, Број ивица 50971, Вредност решења постигнутог употребом грамзивог алгоритма 22626, Максимални број итерација колоније 500

Варијација	Најбоље решење	Просечно решење
MC	22421	22458.8
CMK	22344	22440.4
EC	22406	22436
PMC	22430	22463.6
MMAC	22410	22448.8

Прво посматрамо понашање варијација ОМК на случају малих графова у табелама 4.1, 4.2. У овом случају све варијације ОМК осим најосновнијих МС, и ЕС налазе оптимално решење. Оптимално решење је у случају малих проблема нађено методом бруталне силе односно исцрпне претраге.

У случају већих проблема оптимално решење није нађено из разлога што је у овим случајевима време потребно за његово израчунавање методом исцрпне претраге јако дуго. Квалитет решења које су постизали МС и ЕС су били доста лошији, у поређењу са другим варијацијама у случају већих проблема. У случају малих проблема РМС је постизао најбољи квалитет просечног решења, али су за налажење најбољих резултата обично било потребно више итерација колоније у односу на СМК и ММАС.

У случају проблема средњих (Табеле 4.3, 4.4) и великих (Табеле 4.5, 4.6, 4.7, 4.8, 4.9) димензија, ММАС и СМК дају најбоље резултате са тим да је СМК незнатно бољи када се посматрају просечна и најбоља нађена решења. У случају проблема средњих димензија РМС постиже резултати који су нешто гори од ове две варијације ОМК, међутим у случају великих проблема ова разлика би се значајно повећала. У овом случају РМС би постизао резултате који су слични односно упоредиви са оним које постижу МС и ЕС. Овакво понашање се може објаснити чињеницом да је простор решења изузетно велик и зато је потребно имати претрагу која је више фокусирана односно грамзива.

Варијације ОМК СМК и ММАС управо поседују ове особине и као последица тога постижу јако добре резултате. Важно је напоменути да је јако вероватно да су у тестовима коришћено дуже време израчунавања односно да су колоније извршавале већи број итерација РМС би унапредила своје резултате у поређењу са СМК и ММАС. Овакво уверење постоји зато што РМС има спорију брзину конвергенције ка оптималном решењу, али са друге стране је далеко отпорнија на појаву заробљавања у неком од локално оптималних решења.

Табела 4.11: Поређење стандардне девијације различитих варијација на карактеристичним проблемима

Варијација	Стд. Дев. за табелу 4.3	Стд. Дев. за табелу 4.6
МС	20.7354	22.608
СМК	47.2912	135.312
ЕС	38.6891	84.3487
РМС	58.5454	58.5454
ММАС	27.198	37.4019

Ради боље процене квалитета перформанси различитих варијација ОМК такође се анализира и дисперзија решења и њихова дистрибуција. Као мера дисперзије се користи стандардна девијација рачуната следећом формулом.

$$s = \sqrt{\frac{\sum \left(\bar{W} - W_i\right)^2}{n-1}} \tag{4.19}$$

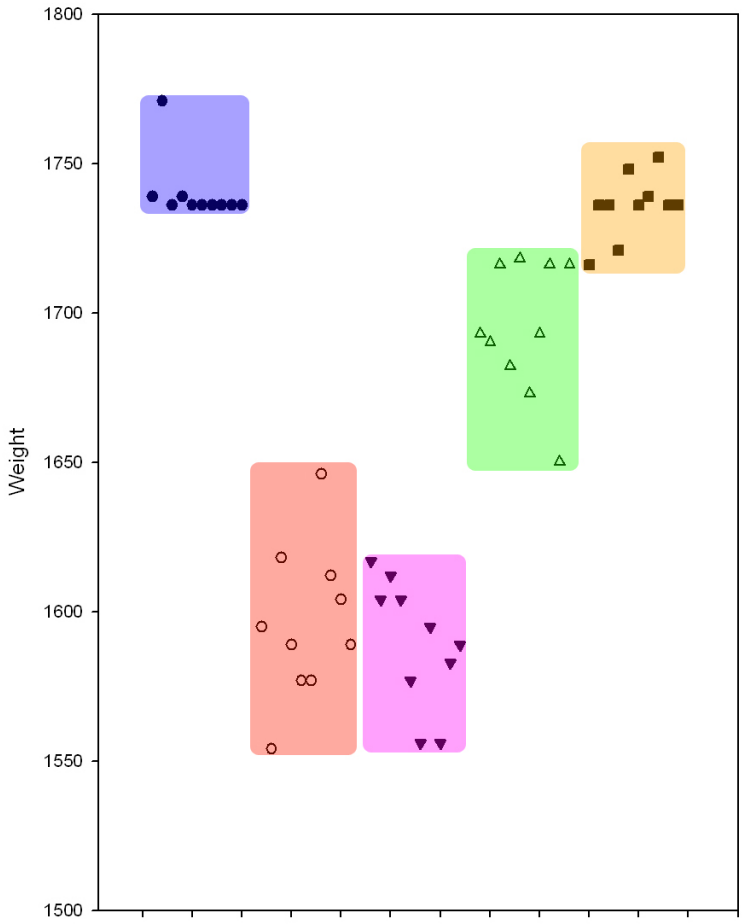
Вредности стандардне девијације су били јако слични за све тестиране инстанце проблема, из тог разлога приказујемо само њихову вредност за проблеме из табела 4.3 и 4.6. За ове проблеме вредности стандардне девијације се могу видети у табели 4.11.

Прво можемо уочити да је у случају МС и ЕС имају малу девијацију, шта се може објаснити чињеницом да ове варијације бивају заробљене у локалним минимумима већ у првим итерацијама колоније. Варијације са добрим перформансама имају уочљиво веће вредности девијације. Значај ове вредности се лакше може разумети ако се посматра дистрибуција решења у сликама 4.3, 4.4.

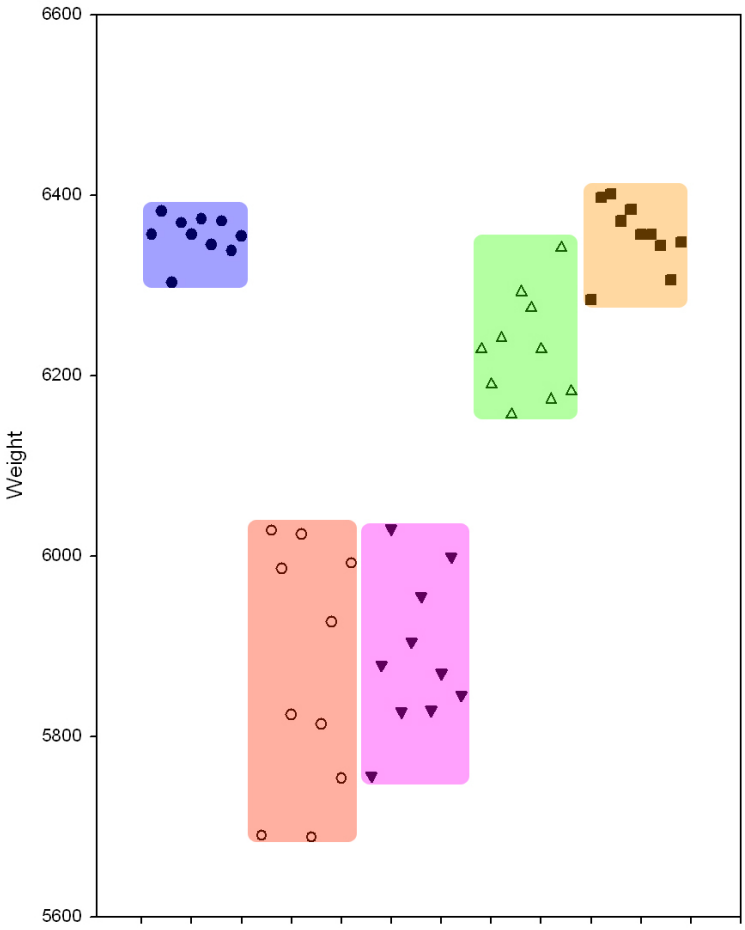
Из ових слика се може видети разлика у понашању МС и ЕС. Ове две варијације постижу сличан квалитет најбољег нађеног и просечног решења, али дистрибуције се јако разликују. МС даје далеко мањи број различитих решења него ЕС.

Изненађујуће је понашање, да фокусирањем претраге у околини добрих покривања расте дисперзија решења, шта је супротно понашање од оног које нам се чини интуитивним. Последица никаквог, или пак слабог појачавања најбољег решења итерације или глобалног решења, је да мрави претражују већу област решења. Проблем је у томе што када је ново најбоље решење нађено феромонски траг се неће довољно брзо променити. Из овог разлога већина мрава неће прећи у област простора решења које је близу новог најбољег покривања. Оваква логика објашњава и чињеницу да РМС спорије конвергира ка добрим решењима у односу на СМК и ММАС варијације али је често у стању да постигне боље резултате. У РМС појачавање феромона које се поклапа са глобалним најбољим решењем је слабије него код ове две методе. Из овог разлога су околине најбољих решења темељније истражена, али је време потребно за померање центра претраге на околину новог најбољег решења потребно више времена.

Још један закључак који можемо извући из анализе дисперзије и дистрибуције решења јесте да је ММАС поузданији за постизање добрих резултата од СМК. СМК има већу дисперзију решења од ММАС. Ово значи да и ако постоји шанса да ће СМК наћи свеукупно најбоље решење,



Слика 4.3: Дистрибуција решења за проблем чији су резултати приказани у Табели 4.3. MC-попуњени кргови, CMK – празни кругови, MMAC – попуњени троуглови, PMC- празни троуглови, EC-попуњени квадрати



Слика 4.4: Дистрибуција решења за проблем чији су резултати приказани у Табели 4.6. MC-попуњени кргови, CMK – празни кругови, MMAC – попуњени троуглови, PMC- празни троуглови, EC - попуњени квадрати

већа је вероватноћа да ће добро решење бити нађено од стране MMAC.

Приликом практичне употребе ових варијација, ако ће бити покренут велики број независних извршавања колонија за тражење најбољег резултата боље је користити CMK, међутим у случају малог броја покушаја MMAC је бољи избор.

Из поређења резултата ових варијација је могуће извести следеће закључке. Две основне методе MC и EC постижу доста горе резултате од осталих метода и нису добар избор за решавање овог проблема. Не постоји све укупно најбоља варијација OMK, већ њен избор зависи од димензија проблема и расположивих ресурса.

4.6 Тестови и резултати за метод искључивања сумњивих елемената

У овој секцији се приказује поређење перформанси чисте OMK примењене на ПМТЧП и унапређене OMK коришћењем метода искључивања сумњивих елемената за корекцију феромонског трага. Циљ ових тестова је да покажу да ова хибридизација OMK поправља резултате основне верзије OMK на ПМТЧП. Параметри који се користе за дефинисање OMK су највећим делом преузети из рада Шјуа [48] пошто су коришћени исти тест проблеми а желимо да покажемо да позитивне ефекте ове хибридизације. У сваком од спроведених експеримената колоније су имале по 10 мрава. Параметар који одређује ниво истраживања је био $q_0 = 0.1$, брзина испаравања приликом глобалног и локалног ажурирања су имали следеће вредности $\varphi = 0.1$ и $p = 0.1$.

Вредности неких параметара OMK се разликују од Шјуа; јер у каснијим истраживањима од стране Гилмора и Драса су нађене вредности за неке од њих које побољшавају резултате постигнуте коришћења OMK [49], [50]. Унапређење је постигнуто променом параметра који одређује утицај хеуристике $\alpha = 5$, ова вредност параметра се користи и у овде приказаним тестовима

Параметри који дефинишу методу искључивања сумњивих елемената су добијени експерименталним путем кроз велики број тестова. Показало се да алгоритам није нарочито осетљив на мале квантитативне промене вредности ових параметара. Изабране су следеће вредности параметара:

Коефицијент за корекцију феромонског трага, који мора да припада интервалу $(0, 1)$, је имао вредност $\delta = 0.0001$. Њиме се множи и на тај начин смањује вредност феромона који одговара неком чвору у феромонској матрици. Овај параметар не сме имати вредност један јер

у том случају се не врши никаква корекција феромонске матрице, и не сме бити једнака 0 јер би тада неком чвору вредност феромона постала једнака нули и било би немогуће изабрати овај чвор приликом креирања решења, и ако је то можда неопходно.

Максимални број чворова који може бити изабран за смањивање вредности феромона је одређен следећом формулом

$$RK = \frac{|V|}{s} \tag{4.20}$$

У једнакости 4.20 параметар s је случајна целобројна промењива са униформном расподелом из интервала $[2, 10]$. Ако је вредност параметра $s = 10$ тада је вредност $RK = 10\%$ броја чворова у решењу, а ако је $s = 2$ онда је $RK = 50\%$. Када је вредност RK између 10% и 50% тада је у просеку од 5% до 25% чворовоа изабрано за смањивање вредности феромона. Проценат је смањен на половину због вероватноће дате у једначини 4.15 која се користи приликом одлуке да ли ће сумњиви чвор бити изабран за корекцију. У случају да је вредност параметра RK превелика онда се покушавају креирати решења без великог броја чворова који постоје у глобално најбољем решењу и као последица обећавајуће области скупа решења бивају брзо напуштена. Са друге стране ако је вредност RK сувише мала утицај методе корекције постаје занемарљив.

Параметар $\lambda = 0.9$ је коришћен ради контороле избегавања поновног бирања истих чворова за корекцију феромона. Вредност 0.9 значи да је вероватноћа да се неки чвор поново изабере једнака 0.9. Ако је вредност овог параметра превише блиска јединици, чворови који имају лоше карактеристике су пречесто бирани, а ако је блиска нули интерес за тај чвор се пребрзо губи и претрага постаје скоро случајна.

Предходно поменути параметри $n = 10$ и $m = 5$ одређују када ће корекција бити вршена. Параметар n броји колико корака глобално најбоље решење није промењено, док параметар m броји колико узастопних итерација је колонија имала исто најбоље решење нађено у једном кораку колоније. Да би дошло до извршења корекције потребно је да су оба услова из претходне секције задовољена. Параметар n омогућава колонији да без спољних утицаја тражи решење када је ново најбоље решење пронађено. Параметар m нам показује да је дошло до стагнације понашања алгоритма. Са оваквим избором параметара је у просеку долазило до корекције сваких 10 итерација колоније. У случају да се m и n повећају долазиће до мање корекција, ако се ови параметри умање корекције ће се извршавати чешће. Ако је број корекција превелик, претрага ће пребрзо напуштати добре области скупа решења, а ако је број

корекција мали тада ће метод избацивања сумњивих елемената имати мали утицај на понашање колоније.

У сваком од тестова колоније су извршавале по 2000 итерација што значи да је било креирано по 20 000 чворних покривача. Коришћен је релативно висок број итерација да би долазило до стагнације алгоритма.

Додатно процесорско време које је било неопходно за рачунање корекције феромона је било занемарљиво приближно 1% укупног времена рачунања независно од величине проблема. Увећање времена извршавања од 1% може се објаснити на следећи начин:

Први корак је процена колика је потребно времена једном мраву да конструише једно решење. Потребно је израчунати динамичку функцију хеуристике чије време израчунавања зависи од димензије проблема и пропорцијална је $|V| * |E|$ а потребно ју је рачунати N пута, где N претставља број чворова у том конкретно креираном решењу. Значи време које је потребно једном мраву да креира једно решење је пропорцијално са $|V| * |E| * N$.

Сада је могуће проценити време које је потребно да се израчуна корекција феромонског трага коришћењем метода сумњивих елемената. За сваки од N чворова које се налазе у најбољем нађеном решењу, потребно је израчунати колико ће се не покривених ивица појавити ако се тај чвор избаци. За један чвор ову вредност је могуће израчунати за време пропорцијално са $N * |E|$ (коришћењем најпростијег алгоритма). Пошто је $N < |V|$, онда је време потребно за рачунање ове функције за све чворове мање од $|V| * |E| * N$. Вредности добијене за све чворове је потребно сортирати шта се може у просеку израчунати за $Nlog(N)$, шта је занемарљиво у поређењу са претходним термом.

Ова анализа показује да алгоритам корекције феромонског трага по асиптомском времену израчунања еквивалентан времену које је потребно једном мраву да креира један покривач. Ова корекција се израчунава ретко (у случају да параметари за критеријум стагнације имају вредности $n = 10$ и $m = 5$), у просеку на сваких десет итерације колоније која се састоји од 10 мрава. Корекција се значи врши једном на сваких 100 креираних решења. Пошто су време потребно за рачунање једног решешења или корекције еквивалентни, добијамо да цена рачунања корекције око 1% укупног времена.

Програм који коришћен у тестовима је писан у програмском језику C#, употребом окружења које ће бити описано у каснијем поглављу тезе. Сви тестови су извршени на разунару са следећом конфигурацијом: процесором Intel(R) CoreTM(2)Duo CPU 8500 @ 4.00 GHz са 4GB RAM на оперативном систему Microsoft Windows 7 Ultimate x64 Edition.

У самим тестовима су коришћени подаци исти као у раду Шјуа [48],

Табела 4.12: Поређење обичне верзије ОМК (ACO) са ОМК унапређеном методом искључивања сумњивих елемената (ACO+SEE) и оптималног решења (OPT) за проблеме малих димензија са тест графовима Тип1

Чворови	Ивице	ACO	ACO+SEE	OPT
10	10	284.0	284.0	284.0
	20	398.7	398.7	398.7
	30	431.3	431.3	431.3
	40	508.5	508.5	508.5
15	20	441.9	441.9	441.9
	40	574.2	570.4	570.4
	60	729.0	726.2	726.2
	80	814.6	807.5	807.5
	100	880.0	880.0	880.0
20	20	473.0	473.0	473.0
	40	661.4	660.3	659.3
	60	861.8	861.8	861.8
	80	905.4	899.9	898.8
	100	1026.8	1026.2	1026.2
	120	1041.5	1038.2	1038.2
25	40	756.6	756.6	756.6
	80	1009.6	1008.1	1008.1
	100	1107.4	1109.1	1106.9
	150	1264.0	1264.0	1264.0
	200	1377.7	1373.4	1373.4

које су касније коришћени и у радовима [43], [44]. Тест проблеми су имали различит број чворова (n) и ивица (m). Проблеми су класификовани по броју чворова n које поседују на следећи начин: проблеми са n између 10 и 25 су сматрани малим, 50-300 су проблеми средње величине, и проблеме од 500 до 1000 чворова сматраћемо великим. За сваку вредност n вршени су тестови за више различитих вредности m. За мале и средње проблеме тежине које одговарају чворовима су генерисане коришћењем два метода. У првом методу (Тип1) тежине су случајно биране из интервала $[20, 120]$ коришћењем униформне расподеле. У другом методу постоји корелација између тежине чвора и степена чвора. Тежине су случајно биране из интервала $[1, d(i)^2]$, где $d(i)$ претставља степен чвора (Тип2). За мале и средње проблеме, 10 различитих семена генератора случајних бројева су коришћене за генерисање различитих тест графова, за сваку од комбинација (m, n). За овако генерисане

Табела 4.13: Поређење обичне верзије ОМК (ACO) са ОМК унапређеном методом искључивања сумњивих елемената (ACO+SEE) и оптималног решења (OPT) за проблеме малих димензија са тест графовима Тип2

Чворови	Ивице	ACO	ACO+SEE	OPT
10	10	18.8	18.8	18.8
	20	51.1	51.1	51.1
	30	127.9	127.9	127.9
	40	268.3	268.3	268.3
15	20	34.7	34.7	34.7
	40	171.5	170.5	170.5
	60	360.8	360.5	360.5
	80	698.7	697.9	697.9
	100	1137.8	1130.4	1130.4
20	20	33.0	32.9	32.9
	40	111.8	111.8	111.6
	60	254.4	254.1	254.1
	80	453.1	452.3	452.2
	100	775.2	775.2	775.2
	120	1125.5	1123.1	1123.1
25	40	98.8	98.7	98.7
	80	373.3	373.0	372.7
	100	595.1	595.1	595.0
	150	1291.7	1290.9	1289.9
	200	2713.1	2709.5	2709.5

проблеме анализиране су вредности просечних решења.

У табелама 4.12, 4.13 приказани су постигнути резултати за мале проблеме.

У овим табелама вредности за ОМК и оптимално решење (OPT) су узети из Шјуовог рада [48]. За проблеме ове величине ОМК постиже јако добре резултате, и у великом броју случајева (16 од 40) постиже оптимално решење. У имплементацији ОМК унапређеном методом искључивања сумњивих елемената, постигнути су бољи просечни резултати у поређењу са Шјуовим резултатима у чак 21 случају, а резултати су погоршани у само једном од њих.

У табелама 4.14, 4.15 су приказани резултати за проблеме средње величине.

У случају ових проблема нису дате вредности оптималних резултата због временске комплексност израчунавања, већ само постоји поређење

са резултатима чисте имплементације ОМК и његове комбинације са методом искључивања сумњивих елемената. Вредности за чисту имплементацију ОМК су и овде преузете из Шјуовог рада. Корекција феромонског трага је унапредила квалитет постигнутих решења у скоро свим тестиран случајевима. У само две димензије тест проблема (за $n = 150$, $m = 50$ и $n = 200$, $m = 50$) квалитет решења је био еквивалентан за ОМК и ОМК унапређен методом корекције феромонског трага. За ове инстанце проблема графови су били јако ретки, шта је јако слично са проблемима мале величине, за које можемо предпоставити да су оба метода нашли оптимално решење. За 71 инстанце тест проблема у само 2 случаја је унапређена верзија ОМК дала лошије резултате од обичне ОМК.

За проблеме великих димензија у Шјуовом раду нису дате вредности резултата, већ је супериорност ОМК у односу на симулирано каљење и табу претрагу је показано поређењем времена израчунавања и релативног квалитета решења. И у овом случају, користе се исте инстанце проблема као и у Шјуовом раду, али се унапређена верзија ОМК пореди са резултатима обичне ОМК који су постигнути коришћењем програма који имплементира Шјуов алгоритам. За сваку од тест проблема покренуто је по 10 колонија за чисту ОМК, са кореном генератора псеудослучајних бројева са вредностима од 0 до 9. Коришћењем истих вредности семена генератора псеудослучајних бројева рачунате су и колоније које су унапређене методом искључивања сумњивих елемената. У табели 4.16 посматрамо просечан квалитет решења које су ове методе постигле. ОМК комбинована са методом искључивања сумњивих елемената је дала боље резултате у свим тестираним случајевима. Такође су поређене и брзине конвергенције за ова два метода. У табели 4.16 је приказан и просечан број итерација за десет колонија које користе чисту ОМК са различитим коренима генератора псеудослучајних бројева, да нађе најбољи резултат за све тестиране инстанце проблема.

Ову вредност поредимо са просечним бројем итерација, који је био потребан одговарајућој (у смислу коришћења истог корена генератора псеудо случајних бројева) колонији која је унапређена коришћењем метода искључивања сумњивих елемената да нађе еквивалентно решење или прво које има бољи квалитет. Овакво поређење показује да је највећа предност овог унапређења: велики пад просечног броја итерација за постизање еквивалентних резултата. Број потребних итерација је смањен у просеку на свега једну четвртину односно 25% у поређењу са чистом ОМК.

Ниво убрзања конвергенције је илустрован на три значајна примера различитих величина. Слика 4.5 приказује квалитет просечног решења

Табела 4.14: Поређење обичне верзије ОМК (ACO) са ОМК унапређеном методом искључивања сумњивих елемената (ACO+SEE) за проблеме средњих димензија са тест графовима Тип1

Чворови	Ивице	ACO	ACO+SEE
50	50	1282.1	1280.9
	100	1741.1	1740.7
	250	2287.4	2280.6
	500	2679.0	2669.3
	750	2959.0	2957.3
	1000	3211.2	3199.8
100	100	2552.9	2544.0
	250	3626.4	3614.9
	500	4692.1	4636.4
	750	5076.4	5082.8
	1000	5534.1	5522.7
	2000	6095.7	6068.3
150	150	3684.9	3676.8
	250	4769.7	4754.9
	500	6224.0	6228.7
	750	7014.7	6996.3
	1000	7441.8	7383.6
	2000	8631.2	8597.2
	3000	8950.2	8940.2
200	250	5588.7	5572.4
	500	7259.2	7233.7
	750	8349.8	8300.3
	1000	9262.2	9208.4
	2000	10916.5	10891.1
	3000	11689.1	11680.4
250	250	6197.8	6169.2
	500	8538.8	8495.9
	750	9869.4	9815.5
	1000	10866.6	10791.0
	2000	12917.7	12827.0
	3000	13882.5	13830.6
	5000	14801.8	14735.9
300	300	7342.7	7326.6
	500	9517.4	9491.9
	750	11166.9	11156.5
	1000	12241.7	12163.7
	2000	14894.9	14834.6
	3000	16054.1	15910.5
	5000	17545.4	17479.8

Табела 4.15: Поређење обичне верзије ОМК (ACO) са ОМК унапређеном методом искључивања сумњивих елемената (ACO+SEE) за проблеме средњих димензија са тест графовима Тип2

Чворови	Ивице	ACO	ACO+SEE
50	50	83.9	83.9
	100	276.2	274.4
	250	1886.8	1870.3
	500	7915.9	7876.7
	750	20134.1	20087.6
100	50	67.4	67.2
	100	169.1	167.8
	250	901.7	895.3
	500	3726.7	3707.0
	750	8754.5	8742.3
150	50	65.8	65.9
	100	144.7	144.1
	250	625.7	624.8
	500	2375.0	2358.6
	750	5799.2	5707.0
200	50	59.6	59.6
	100	134.7	134.6
	250	488.7	487.9
	500	1843.6	1818.7
	750	4112.8	4077.0
250	250	423.2	421.2
	500	1457.4	1454.3
	750	3315.9	3289.4
	1000	6058.2	6040.0
	2000	26149.1	25932.1
	5000	171917.2	171500.7
300	250	403.9	402.7
	500	1239.1	1237.3
	750	2678.2	2674.1
	1000	4895.5	4867.9
	2000	21295.2	21107.7
	5000	143243.5	142292.6

Табела 4.16: Поређење обичне верзије ОМК (ACO) са ОМК унапређеном методом искључивања сумњивих елемената (ACO+SEE) за проблеме великих димензија

Nodes	Edges	Просечно решење		Просечно итерација за еквивалентна решења		
		ACO	ACO+SEE	ACO	ACO+SEE	$\frac{\text{ACO+SEE}}{\text{ACO}} * 100\%$
500	500	12746.2	12687.7	1130.6	345.9	30.5
	1000	16700.4	16574.9	1209.5	138.1	11.4
	2000	21225.9	21093	999.6	257.0	25.7
	5000	27891.7	27585.5	854.8	190.6	22.2
	10000	29991.4	29796.4	401.0	101.7	25.3
800	500	15103.3	15069.9	721.4	508.4	70.4
	1000	22991.1	22852.1	1232.2	192.8	15.6
	2000	32067.6	31786.9	1016.8	228	22.4
	5000	39264.2	38906.7	750.0	129.0	17.2
	10000	45107.0	44691.7	936.8	170.1	18.1
1000	1000	25146.9	24925.4	1360.7	389.3	28.6
	5000	45982.5	45588.7	1238.2	248.5	20.0
	10000	52497.0	52105.0	737.9	168.3	22.8
	15000	59160.5	58654.8	676.2	163.5	24.1
	20000	60760.6	60268.2	518.2	210.1	40.5

за десет колонија као функцију од броја итерација за тест пример са 500 чворова и 2000 ивица (Табела 4.16 ред 3).

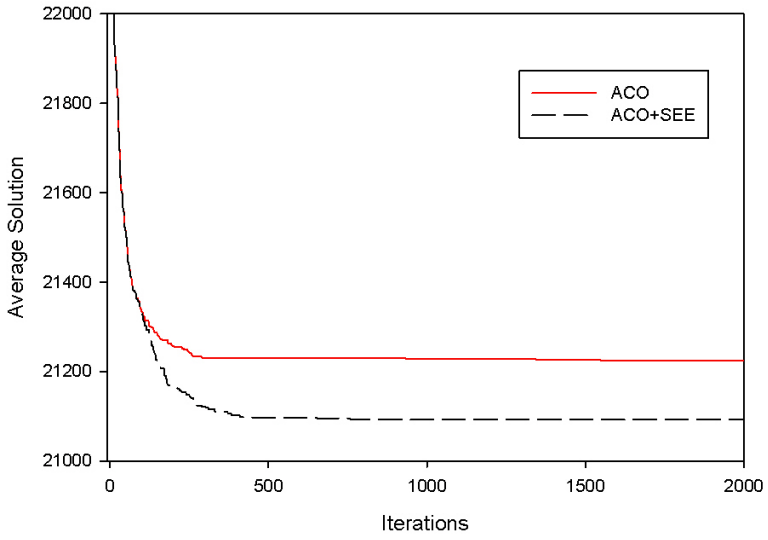
4.5 приказује квалитет просечног решења за десет колонија као функцију од броја итерација за тест пример са 1000 чворова и 1000 ивица (Табела 4.16 ред 11).

4.5 приказује квалитет просечног решења за десет колонија као функцију од броја итерација за тест пример са 800 чворова и 10000 ивица (Табела 4.16 ред 3).

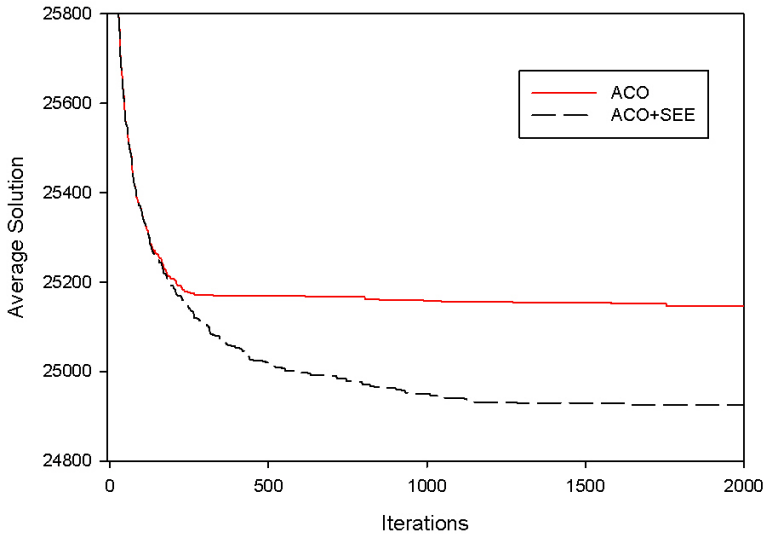
Ови дијаграми показују да се унапређење квалитета решења код обичне ОМК релативно брзо успорава (око 200 итерација), док унапређени алгоритам који користи корекцију феромонског трага наставља да поправља квалитет најбољег нађеног решење још велики број итерација.

У изведеним експериментима, је такође уочљиво да ОМК у случају различитих вредности корена генератора псеудослучајних бројева даје нешто друкчије резултате. Ово може бити објашњење за ретке случајеве (4 од 160) у којима је унапређен метод дао горе резултате од обичног

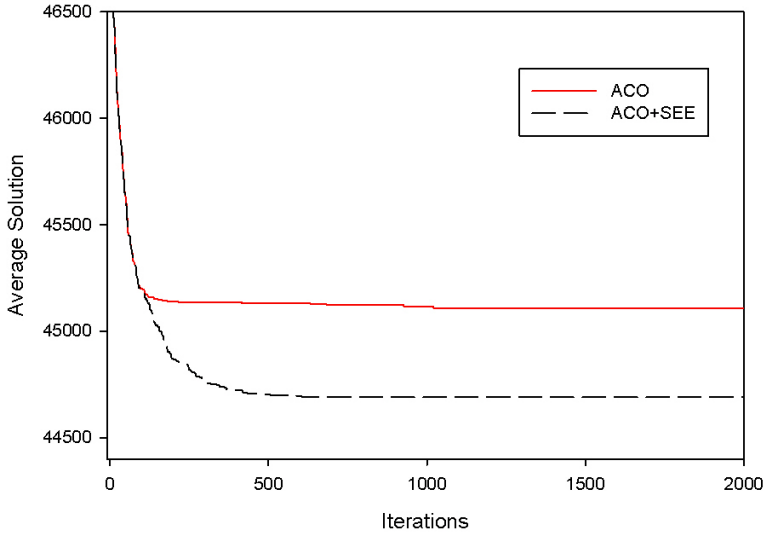
ОМК: несрећан избор корена генератора.



Слика 4.5: Брзина конвергенције просечног решења за десет тест колонија на графу са 500 чворова и 2000 ивица



Слика 4.6: Брзина конвергенције просечног решења за десет тест колонија на графу са 1000 чворова и 1000 ивица



Слика 4.7: Брзина конвергенције просечног решења за десет тест колонија на графу са 800 чворова и 10000 ивица

Глава 5

Решавање проблема минималног тежинског доминирајућег скупа употребом ОМК

Доминирајући скуп за граф $G = (V, E)$ је подскуп скупа чворова $W \subseteq V$ такав да сваки чвор у $V \setminus W$ је суседни неком чвору у W. Чворове u, v се сматрају суседним ако постоји ивица $(u, v) \in E$. Проблем минималног доминирајућег скупа (ПМДС) је наћи најмањи могући доминирајући скуп у графу. Проблем минималног тежинског доминирајућег скупа (ПМТДС) је дефинисан у случају да је тежина c_i додата сваком чвору i и уместо проналажења W са минималним бројем елемената ми тражимо W који има минималну суму тежина. Провера постојања доминантног скупа са k елемената је један од класичних НП-комплетних проблема одлучивања [51]. ПМДС за генералне графове је еквивалентан проблему покривајућег скупа.

Широки опсег практичних проблема може бити трансформисан у ову форму, од позиционирања малопродајних радњи, сензорских мрежа и Мобилних ад хок мрежа (MANETs). Различите екстензије ПМДС-а попут повезаног или тежинских, су коришћене да прецизније дефинишу особине проблема који се решавају [52]. Због чињенице да налажење оптималног решења за ПМДС (и његових варијација) не може бити урађено у полиномијалном времену, широк опсег методи је развијен да би се постигла решења која су близу оптималних. Неке од оваквих методи су грамзиви алгоритам [53], константан-фактор апроксимација [54], енергијски функцијски алгоритам [55], колаборативни хеуристички покривач [56] и полиномијална језгра [57].

ПМДС је решаван и коришћењем алгоритама базираних на популацији попут оптимизације мрављим колонијама (ОМК) и генетских алгоритама при чему је ОМК дала боље резултате [58]. У овој секцији проширујемо апликацију ОМК-а на ПМТДС. Такође се уводи нова хеуристика која узима у обзир тежине чворова који су прекривени приликом додавања нових чворова у скуп решења. Креиран је и одговарајући грамзиви алгоритам. У овој секцији такође приказујемо предности коришћења ОМК-а уместо грамзивих алгоритама за ПМТДС.

5.1 Грамзиви алгоритми за ПМТДС

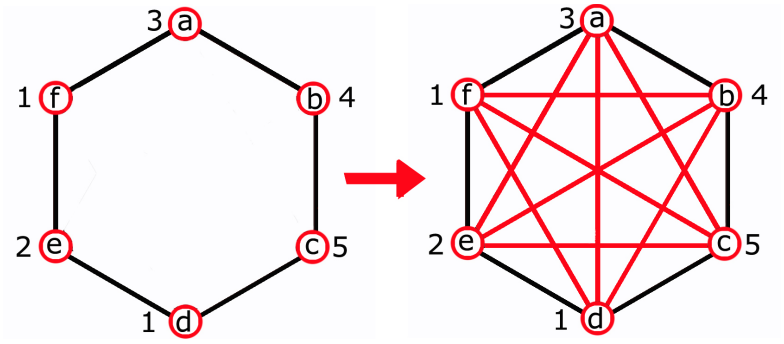
Идеја основног грамзивог алгоритма, на начин на који је презентован у раду [53], за ПМДС, је да се у свакој итерацији додаје нови чвор у W који покрива највише чворова, све док W не формира доминанирајући скуп.

У наредном тексту користимо термин да је чвор j покривен ако $j \in W$ или је j сусед неком чвору $i \in W$, и откривен у супротном случају. У n-тој итерацији, додајемо нови чвор $i \in V_n$ у W који покрива максимални број откривених чвоворова. V_n је скуп чворова који нису селектовани у првих n итерација. Алгоритам је завршен када су сви чворови покривени. Ова хеуристика би требала бити проширена тако да узима у обзир и тежине које одговарају чворовима.

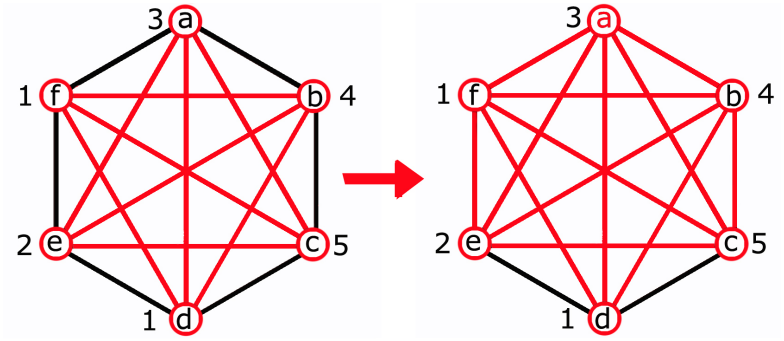
Први корак, који је потребан да би се имплементирао овакав алгоритам је представљање проблема на начин који нам омогућује да се рачунање динамичне хеуристике, и праћење покривености чворова ради на једноставан начин. Да би се ово постигло користимо приступ који је сличан моделу представљеном од стране Шјуа за проблем минималног чворног тежинског покривача [48], који је тежинска верзија хеуристике коришћене у [53]. У Шјуовом раду се користи потпуно повезаном граф $G_c(V, E_c)$ који је изведен из G. У чланку [48], се предлаже додавање тежине 1 ивици у потпуно повезаном графу G_c у случају да он постоји G или 0 ако не постоји у оригиналном графу. Усвојени приступ је илустрован у слици 5.1

Као што је већ напоменуто, неопходно је ажурирати граф приликом додавања нових чворова резултирајућем скупу. Ово се чини на следећи начин, када додамо чвор a, свим ивицам у G_c које су повезане са чвором a или са неким од његовим суседима, постављамо тежину на 0. Овај корак се илуструје на слици 5.2

Сада је могуће дефинисати $G_k(V, E_{ck})$ као стање графа након што је k чворова додато скупу решења, и одговарајућу функције у једначини



Слика 5.1: Претварање почетног графа у потпуно повезани код ког ивице имају тежине. Црне ивице имају тежину 1, црвене 0



Слика 5.2: Пример ажурирања графа приликом додавања нових чворова скупу решења. Приказано је ажурирања након додавања чвора a

5.1.

$$\psi_k(i,j) = Value(E_{ck})(i,j) \tag{5.1}$$

Овако дефинисано правило ажужирања чини могућим да се динамички евалуира (оцени) пожељност чворова уз помоћ функције ψ_k. Сада можемо да дефинишемо динамичну хеуристку

$$Cov = \sum\nolimits_{(i,j)\in E_c} \psi_k(i,j) \tag{5.2}$$

$$\gamma_{jk} = \frac{1 + Cov}{w(j)} \tag{5.3}$$

Хеуристика γ_{jk} у једначини 5.2 исказује да су чворови са малом тежином и великим бројем не покривених суседа веома пожељни. Додавање вредности 1 у суми је коришћено да би се направила разлика између чворова

који немају конекције али нису још увек покривени. Хеуристика узима у обзир само тежину чворова који је тренутно селектован али не тежине чворова који се покривају. Овде се уводи нова хеуристика (Грамзиви 2) која и ове тежине узима у обзир.

$$\eta_{jk} = \frac{Cov(1 + \sum_{(i,j)\in E_c} w(i)\psi_k(i,j))}{w(j)} \tag{5.4}$$

η_{jk} исказује да су пожељнији чворови мале тежине са великим бројем веза који покривају чворове са високом сумом тежина.

Да би се ове две верзије грамзивог алгоритма користиле, мора постојати начин да се препознамо да ли су сви чворови покривени. Ово се постиже додавањем скупа $Y_0 = V$ на почетку алгоритма који се ажурира на сваком кораку. Ако је у кораку i изабран чвор a, имамо следећу корекцију над скупом Y_i

$$Y_i = \{v|(v,a) \in E \vee (a,v) \in E\}/Y_{i-1} \tag{5.5}$$

Алгоритам је завршен када $Y_i = \emptyset$ односно сви чворови су покривени.

5.2 Примена ОМК на ПМТДС

Употреба ОМК се показала ефикасном на разним типовима проблема од економског оптерећења опреме [59], проблемима заказивања [60], чак и у обради слика [61]. Апликација ОМК на ПМТДС разликује се у два кључна аспекта од оригиналне апликације на проблем трговачког путника (ПТП). У ПТП-у решење је низ свих градова који се појављују у проблему или другим речима решење је пермутација скупа чворова. У случају ПМТДС решење је подскуп скупа чворова графа, у којима је редослед неважан. У случају ПТП хеуристична функција, која се користи је статична у смислу да представља удаљеност између чворова и не мења се током генерисања решења. Супротно, за ПМТДС, хеуристика је однос између тежине и суме тежина суседних чворова, који је променљив односно динамичан. Ова сума се мења како додајемо нове чворове скупу решења јер се све више чворова покрива.

Ове разлике, слично као и у случају ПМТЧП, утичу на основни алгоритам на два начина. Прво мрави остављају феромон на чворовима уместо на ивицама. Друго, ми динамички ажурирамо граф, и са њим, хеуристичку функција на начин који је приказан у претходном одељку. Користећи хеуристику дефинисану са η_{jk} у једначини 5.4, можемо дефинисати транзиционо правило за појединачне мраве.

$$p_j^k = \begin{cases} 1 & , q > q_0 \ \ \& \ \ j = \arg\max\limits_{i \in A_k} \tau_i \eta_{ik}^{\alpha} \\ 0 & , q > q_0 \ \ \& \ \ j \neq \arg\max\limits_{i \in A_k} \tau_i \eta_{ik}^{\alpha} \\ \frac{\tau_j \eta_{jk}^{\alpha}}{\sum_{i \in A_k} \tau_j \eta_{ik}^{\alpha}} & , q \leq q_0 \end{cases} \tag{5.6}$$

У једначини 5.6 q_0 је стандардни параметар у ОМК, који прецизира однос експлоатације / истраживања за појединачне мраве. q је случајна промењива на основу које се бира тип селекције на сваком кораку. Y_k је листа доступних чворова. Супротно ПТП-у, транзиционо правило не зависи од последњег селектованог чвора и зато имамо τ_i уместо τ_{ij}

Следећи корак је дефинисање глобалног (када мрав креира решење) и локалног (када мрав бира нови чвор) правила ажурирања. Улога глобалног правила ажурирања је да чворови који припадају решењима већег квалитета постану пожељнији, или другим речима, да се интензивира експлоатацију.

$$\Delta\tau_i = \frac{1}{\sum_{j \in W} w(j)} \quad , \forall i \in W \tag{5.7}$$

$$\tau_i = (1 - p)\tau_i + \Delta\tau_i \tag{5.8}$$

У једначинама 5.7, 5.8 је дефинисано глобално правило ажурирања. У њему $\Delta\tau_i$ представља меру квалитета решења (подскупа) W, који садржи чвор i, и помоћу њега се дефинише глобално правило ажурирања у једначини 5.8. Мера квалитета је обрнуто пропорцијална укупној тежини решења. Параметар p као и у претходно описаним проблемима се користи да би се одредио утицај ново нађених решења на феромонску матрицу.

Локално правило ажурирање служи за спречавања појављивање чворова са јако великом количином феромона које сви мрави бирају. Овако нешто се постиже тако што се чворови чине мање пожељним како их нови мрави бирају. Овако се подржава истраживање у колонији. Локално правило ажурирања је дато у стандардној форми:

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0 \tag{5.9}$$

За вредност τ_0 се узима мера квалитета решења које се постиже употребом другог грамзивог алгоритма (Грамзиви2). Параметар φ одређује утицај локалног правила ажурирања.

5.3 Тестови и резултати

У овој секцији даје се поређење резултата коришћења стандардног грамзивог алгоритма (Грамзиви 1) на ПМТДС, предходно представљене унапређене верзије грамзивог алгоритма (Грамзиви 2) и ОМК која користи исту хеуристику као Грамзиви2. Програм који имплементира ове методе је написан коришћењем оквира за решавање проблема употребом ОМК, односно креирањем одговарајућег проширења који ће бити представљен у каснијем делу тезе.

У тестовима су коришћене следеће вредности параметара који дефинишу ОМК. Колоније су се састојале од десет мрава. Параметар истраживања q_0 је имао вредност 0.1. Параметри за дефинисање глобалног и локалног правила ажурирања су имали следеће вредности $p = 0.1$, $\varphi = 0.1$. Параметар који одређује утицај хеуристике α је имао вредност 4. ОМК је имплементирана у MMAC варијацији. Почетно најбоље решење V' за колоније је добијено употребом алгоритма Грамзиви2, а иницијална вредност феромона за сваки од чворова је дата је једначини 5.10.

$$\tau_0 = \frac{1}{n \sum_{j \in V'} w(j)} \tag{5.10}$$

У једначини 5.10, n представља број чворова који постоји у доминирајућем скупу V'. Свака колонија би извршина по 10 000 итерација.

Генерисана су, на случајан начин, два типа инстанци проблема који су коришћене у тестовима, слично као и у случају ПМТЧП. Ради прегледности овде ће бити поновљен опис начина генерисања. У првом типу графова, целобројне тежине чворова су на случајан начин бирани из интервала $[20, 70]$ са униформном расподелом (Тип1). У другом типу, је постојала корелација између степена чвора v, односно броја ивица чији је део, и у овом случају тежина чвора би се на случајан начин бирала из интервала $[1, e(v)^2]$ са униформном расподелом (Тип2). $e(v)$ представља број конекција који одговарају чвору v.

Генерисани су графови који су имали од 50 до 1000 чворова са различитим бројем случајно генерисаних ивица. Графови су увек били повезани. За сваки од парова број чворова, број ивица генерисано је по десет различитих тест проблема и посматран је просечан квалитет решења које су три анализирана алгоритма постизала. Резултати ових тестова се могу видети у табелама 5.1, 5.3, 5.2 и 5.4.

Када се пореде два грамзива алгоритма, прво се уочава да је унапређена хеуристика која узима у обзир и тежину ново покривених чворова постиже скоро униформно боље резултате, за оба два типа генерисаних

Табела 5.1: Поређење постигнутих резултата за графове малих и средњих димензија генерисане методом Тип2

Димензије	Грамзиви1	Грамзиви2	ОМК
50*50	610.3	609.9	539.8
50*100	509.5	472.7	391.9
50*250	262.6	260.2	195.3
50*500	157.7	155.5	112.8
50*750	114.1	99.8	69.0
50*1000	86.2	83.0	44.7
100*100	1232	1223.7	1087.2
100*250	864.4	819.7	698.7
100*500	564.1	554.3	442.8
100*750	448.2	413.5	313.7
100*1000	352.6	336.4	247.8
100*2000	195.3	210.6	125.9
150*250	1799.5	1758.6	1630.1
150*250	1548.0	1496.4	1317.7
150*500	1064.2	1051.8	899.9
150*750	870.0	840.3	674.4
150*1000	704.4	685.8	540.7
150*2000	415.2	366.3	293.1
150*3000	288.8	283.9	204.7
200*250	2329.2	2274.1	2039.2
200*500	1729.2	1707.8	1389.4
200*750	1349.4	1324.9	1096.2
200*1000	1124.6	1102.0	869.9
200*2000	703.5	665.3	524.1
200*3000	513.1	523.9	385.7

Табела 5.2: Поређење постигнутих резултата за графове малих и средњих димензија генерисане методом Тип2

Димензије	Грамзиви1	Грамзиви2	ОМК
50*50	73.7	72.2	62.3
50*100	137.7	126.6	98.4
50*250	383.3	362.8	202.4
50*500	872.7	750.6	312.9
50*750	1358.7	1227.2	386.3
100*100	150.6	144.2	126.5
100*250	327.5	299.4	236.6
100*500	794.3	725.9	404.8
100*750	1075.3	998	615.1
100*1000	1358.4	1243.9	697.3
100*2000	3398.7	3203.2	1193.9
150*250	223.4	214.2	190.1
150*250	323.6	318	253.9
150*500	652.9	609.4	443.2
150*750	1023.3	900.6	623.3
150*1000	1426.1	1265.2	825.3
150*2000	3233.8	2701.9	1436.4
150*3000	5041.6	4356.2	1751.9
200*250	348.9	335.5	293.2
200*500	618.7	589.1	456.5
200*750	913.3	867.3	657.9
200*1000	1342.2	1222.6	829.2
200*2000	2620.7	2433.6	1626
200*3000	4450.8	3702.0	2210.3

Табела 5.3: Поређење постигнутих резултата за графове великих димензија генерисане методом Тип1

Димензије	Грамзиви1	Грамзиви2	ОМК
500*500	6046.2	5944.6	5476.3
500*1000	4785.8	4664.2	4069.8
500*2000	3248.0	3140.8	2627.5
500*5000	1712.0	1689.8	1398.5
500*10000	990.6	1006.1	825.7
800*1000	9160.3	8953.4	8098.9
800*2000	6729.8	6597.6	5739.9
800*5000	3833.4	3747.5	3116.5
800*10000	2325.2	2248.9	1923
1000*1000	12146.3	11987.7	10924.4
1000*5000	5595.4	5501.3	4662.7
1000*10000	3550.5	3414.1	2890.3
1000*15000	2562.6	2428.1	2164.3
1000*20000	2017.8	1918.1	1734.3

проблема. Алгоритам Грамзиви2 је постизао резултате који су били бољи и до 10%. Унапређење је било најмање у случају ретких графова.

Алгоритам ОМК је као иницијално најбоље решење користио доминирајући скуп који је креиран коришћењем алгоритма Грамзиви2, са чијим резултатима ће бити поређен. Прво је важно напоменути да је ОМК унапредио квалитет решења у свим тестираним случајевима. Ниво унапређења квалитета резултата би варирао у зависности да ли су тестирани графови били густи или ретки. У случају графова са малом густином, или другим речима са малим просечним бројем ивица по чвору, унапређење би било најмање и у просеку би износило од 10-20%.

Ефикасност ОМК би расла са порастом густине, и у случајевима где је просечан чвор био повезан са 20 ивица унапређење би било од 20% па чак и до 70%. Овакви резултати показују да је употреба ОМК на ПМТДС од изузетно велике користи, а са друге стране да оба представљена грамзива алгоритми имају релативно лоше понашање у случају густих графова.

У овој секцији је приказана имплементација ОМК за ПМТДС. Представљена је нова хеуристика који узима у обзир тежине чворова који се покривају. Показано је да је коришћење те хеуристике у грамзивом алгоритму, дало побољшање у односу на стандардан приступ. Ова нова

Табела 5.4: Поређење постигнутих резултата за графове великих димензија генерисане методом Тип2

Димензије	Грамзиви1	Грамзиви2	ОМК
500*500	743.7	705.7	651.2
500*1000	1294.7	1198.1	1018.1
500*2000	2632.2	2378.1	1871.8
500*5000	6642.8	5714	4299.8
500*10000	14619.9	14163.3	8543.5
800*1000	1375.8	1325.5	1171.2
800*2000	2551	2359.3	1938.7
800*5000	6643.7	5957.5	4439.0
800*10000	14068.4	12443.1	8951.1
1000*1000	1479.0	1432.0	1289.3
1000*5000	6722.9	6082.9	4720.1
1000*10000	14352.8	13141.7	9407.7
1000*15000	23032.8	20476	14433.5
1000*20000	32295.5	28427.8	19172.6

хеуристика је коришћена у имплементацији ОМК. Тестови за ове методе су извршени на графовима различитих величина, ивичних густина и алгоритама за генерисање одговарајућих тежина. Резултати показују да ново уведена хеуристика побољшала перформансе грамзивог алгоритма. ОМК се показао као веома ефикасан на ПМТДС и значајно је побољшао квалитет резултата посебно у густим графовима.

Глава 6

Решавање проблема минималног повезаног доминирајућег скупа употребом ОМК

У овом поглављу ће бити представљена примена ОМК на решавање проблема минималног повезаног доминирајућг скупа. Као што је већ речено доминирајући скуп за граф $G = (V, E)$ је подскуп скупа чворова $D \subseteq V$ такав да сваки чвор у $V \setminus D$ је суседан неком чвору у D. Ми зовемо чвореве u, v суседним ако постоји ивица $(u, v) \in E$. Проблем минималног доминирајућег скупа (ПМДС) је наћи најмањи могући доминирајући скуп у графу. Изузетно важна варијација овог проблема је његова повезана верзија. Доминирајући скуп називамо повезаним у случају да има особину да се из сваког чвора $n \in D$ може стићи до били ког чвора $m \in D$ путањом коју чине само ивице које се састоје из чворова који се налазе у оквиру D. Односно, D индукује повезани подграф графа G. Проблем минимални повезаног доминирајућег скупа (ПМПДС) је наћи овакав скуп који има најмање елемената. ПМПДС је такође НП-комплетан проблем.

Решавање овог проблема је добило на популарности у скорије време из разлога своје тесне везе са мобилним ад хок мрежама и сензорским мрежама. У практичним проблемима који се могу представити у облику ПМПДС најчешће нема потребе да се нађе оптимално решење већ је у већини случајева довољно наћи и решење које је близу њега. Постоје два главна правца у приступу решавања овог проблема и они су: централизовани и дистрибуирани. Сваки од њих је у вези са типом конкретне апликације у којој се примењује. У овој секцији ће фокус бити постављен

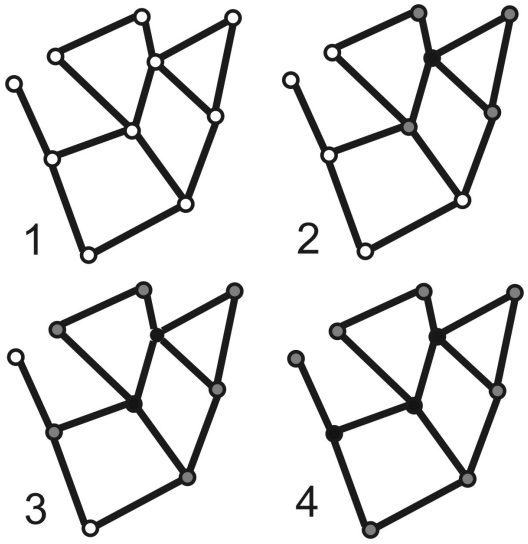
на централизовану верзију решавања овог проблема. До сада је развијен широк спектар метода које налазе решења која су по квалитету близу оптималног. Неколико различитих хеуристика и одговарајућих грамзивих алгоритама је развијено да би се решио ПМПДС [62], [63], [64], [65], [66], [67]. Више корачни колаборативни покривајући хеуристички приступ је коришћен у раду [68]. Овај проблем је такође решаван комбинацијом симулираног каљења и табу претраге [69], неуронских мрежа [70] и параметризоване апроксимације [71].

Овај проблем је предходно био решаван и коришћењем ОМК приликом имплеметнације кластера у мрежи [72]. У методи представљеној у овом раду је коришћена ОМК која се састојала из два корака у првом се налазио доминирајући скуп за дати граф. У другом су додавани нови чворови тако да се добијени подграф учини повезаним. Овакав приступ, који се састоји из два корака, није у потпуности природан за коришћење ОМК. Из тог разлога у саставу ове тезе је развијен ОМК алгоритам који се састоји из једног корака за решавање проблема. У раду [72] је изабран комплекснији двокорачни приступ из страха од раног заробљавања колоније у локалном оптимуму. У развијеном једнокорачном алгоритму ОМК, избегавање ране стагнације претраге се постиже адаптацијом методе сумњивих елемената на ПМПДС. Понашање ОМК је даље унапређено развијањем ефикасног метода за бирање иницијалних услова претраге појединачних мрава.

Из анализе извршених тестова ће бити показано да је овај метод добар избор у односу на постојеће приступе решавању овог проблема. Такође ће бити показано да се додавањем корекције феромонског трага на основу сумњивости елемената перформансе ОМК у ММАС варијацији значајно побољшавају.

6.1 Грамзиви алгоритми за решавање ПМПДС

Постоји више различитих концепата за креирање грамзивих алгоритама за решавање ПМПДС. Може се користити поступак који користи само једну хеуристику и у једном кораку креира решење. Други приступ је да се користи грамзиви алгоритам који се састоји из две фазе или корака. У овом случају у првој фази се решава неки међупроблем попут максималног независног скупа, и у другој фази се он претвара у МПДС, овакав приступ је коришћен у следећим радовима [65] [63], [64]. Двокорачне методе обично дају боље резултате, али по цену повећања компл-



Слика 6.1: Пример креирања повезаног доминирајућег скупа коришћењем грамзивог алгоритама 1) Улазни граф, 2)Иницијални корака 3, 4) Каснији кораци алгоритма

ексности имплементације алгоритма. Побољшање квалитета резултата је последица веће слободе приликом избора нових чвора за проширење тренутно генерисаног делимичног решења у првој фази алгоритма. Коришћење алгоритма оваквог типа као основе за развијање ОМК, не чини се природним, или прецизније потребно је развити засебне ОМК за сваки корак грамзивог алгоритма.

Из ових разлога, у имплементацији ОМК у оквиру ове тезе је изабран једнокорачни грамзиви алгоритам као основа. Прецизније прихваћен је грамзиви алгоритам представљен од стране Гухе [65]. Идеја овог алгоритма је следећа: алгоритам се иницијализује избором чвора $v_0 \in V$ такавог да има највиши степен. Степен чвора v_0 и овде означава број ивица такав да је v_0 њихов део. Сада нам v_0 служи као почетак дрвета V'. При сваком кораку алгоритма бира се чвор w такав да је суседан неком од чворова унутар V', који покрива највећи број не покривених чворова. И у овом случају сматрамо чвор v покривеним ако је $v \in V'$ или ако постоји такав $w \in V'$, за који је $(v, w) \in E$. Овај поступак се понавља све док сви чворови графа G не буду покривени.

Да би се имплементирао овакав алгоритам неопходно је да на једноставан начин могу да се разликују суседни, покривени и непокривени чворови. Ово је могуће постићи процесом представљеном од стране Гухе. Приликом иницијализације проблема сви се чворови боје у бело. Када се

неки чвор дода дрвету V' тада њега бојимо у црно а све његове суседе који нису претходно обојени у црно сивом бојом. У следећем кораку бирамо сиви чвор које повезан са највећим бројем белих чворова. Алгоритам се завршава када не постоји више белих чворова у графу. Илустрација оваквог алгоритма је дата на слици 6.1.

Као што је примећено од стране Гухе и Кулера, када се овај приступ користи за развој грамзивог алгоритма он бива лако заробљен у локално оптималним решења што је последица његове кратковидости. Из овог разлога су развијени комплекснији алгоритми који избегавају овај проблем. Гуха је сам унапредио почетни метод тако што уместо избора појединачних чворова бирао парове њих [65]. У раду [62] коришћена је хеуристика која прати број црних и сивих чворова али и број раздвојених скупова црних чворова за развој грамзивог алгоритма. У радовима [66], [67] приступ са избацивањем чворова је коришћен, у коме се најмање важни чворови прво избацују. Сви ови алгоритми су су тешки за имплементацију и спорији приликом извршавања. У овом поглављу ће бити показано да мане коришћења Гухине једноставне хеуристике се у великој мери могу избећи када се она комбинују са ОМК.

6.2 Имплементација ОМК

Као и у случају ПМТЧП и ПМТДС и у случају ПМПДС решење проблема представља подскуп скупа чворова графа. Хеуристичка функција која се користа је супротно од ПТП динамична односно она се мења приликом генерисања решења као и у случају ПМТЧП и ПМТДС. Коначно у случају ПТП (и ПМТЧП и ПМТДС) при свакој итерацији ОМК сви претходно не посећени чворови су потенцијални кандидати док у случају ПМПДС само сиви чворови могу бити бирани, односно постоји рестрикција скупа кандидата. Значи алгоритам ОМК ће имати следеће карактеристике: феромон ће се остављати на чворовима а не на ивицама, хеуристичка функција ће се динамички ажурирати да прати стање графа и листа потенцијалних кадидата ће бити праћена.

Приликом имплементације ОМК први корак је представити проблем на начин који омогућује рачунање динамичке функције на брз и ефикасан начин. Ово се постиже на следећи начин. На почетку алгоритма сваком чвору i се за вредност хеуристичке функције η_i^0 даје његов степен, или другим речима број чворова са којим је повезан. Такође се креирају три помоћна скупа, први је W_0 који ће на почетку држати све чворове, и два иницијално празна скупа B^0 који ће садржати црне чворове и Gr^0 који ће садржати сиве чворове. Као што је предходно поменуто, хеуристичка

функција је динамичка и она ће бити ажурирана када се нови чворови додају у скуп решења. Ако приликом корака j додамо чвор v тада ће се степен свих његових суседа смањити за један дајући нову хеуристичку функцију η^j. У истом кораку чвор v ће бити пребачен из скупа Gr^j u B^j, а сви његови суседи пребачени из W^j u Gr^j.

Да би алгоритам ОМК био дефинисан на неком проблему потребно је дефинисати правило избора за појединачне мраве, правило локалног и глобалног ажурирања. Прво се дефинише правило избора коришћењем хеуристичке функције η^j на следећи начин

$$p_j^k = \begin{cases} 0 & , j \notin Gr_k \\ prob_j^k & , j \in Gr_k \end{cases} \tag{6.1}$$

$$prob_j^k = \begin{cases} 1 & , q > q_0 \ \ \& \ \ j = \arg\max\limits_{i \in Gr_k} \tau_i \eta_i^k \\ 0 & , q > q_0 \ \ \& \ \ j \neq \arg\max\limits_{i \in Gr_k} \tau_i \eta_i^k \\ \frac{\tau_j \eta_j^k}{\sum_{i \in Gr^k} \tau_j \eta_i^k} & , q \leq q_0 \end{cases} \tag{6.2}$$

У једначини 6.2 параметар q_0 одређује однос истраживања / експлоатације приликом претраге појединачних мрава. На стандардни начин са њим је повезана случајна промењива q. Исто као и у случају ПМТЧП и ПМТДС у транзиционо правило не зависи од чвора који је последњи изабран већ само на тренутно стању помоћног графа па се опет користи τ_j уместо τ_{ij} за вредност феромонског трага. Исто важи и за хеуристичку функцију η_j. Након дефинисања транзиционог правила потребно је дефинисати глобално и локално правило ажурирања.

$$\Delta\tau_i = \begin{cases} 0 & , i \notin V' \\ \frac{1}{|V'|} & , i \in V' \end{cases} \tag{6.3}$$

У једначини 6.3 $\Delta\tau_i$ је мера квалитета глобалног најбољег нађеног решења V' који садржи чвор i, а помоћу њега је дефинисано глобално правило ажурирања у једначини 6.4. Параметар p се овде, као и у предходним проблемима, користи за дефинисање утицаја ново нађених решења на феромонски траг.

$$\tau_i = (1 - p)\tau_i + \Delta\tau_i \tag{6.4}$$

Потребно је назначити да је вредност $\Delta\tau_i$ за већину чворова једнака нули, шта има за последицу да се вредност феромона смањује до нула за чворове који нису саставни део глобално најбољег решења кроз низ итерације колоније.

Формула за локално правило ажурирања има следећу стандардну форму

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0 \tag{6.5}$$

Мера квалитета које одговара решењу проблема које је постигнуто предходно описаним Гухиним грамзивим алгоритмом је узето τ_0. Параметар φ се и овде користи да одреди снагу локалног правила ажурирања

6.3 Избегавање стагнације ОМК

Када се ПМПДС решава применом ОМК заснованој на хеуристици датој од стране Гухе постоји велика вероватноћа да ће алгоритам бити заробљен у неком од локалних оптимума. За овакву појаву постоје два главна узрока. Први је, што је то стандардан проблем код употребе ОМК, због начина на који се креира феромонска матрица. Са друге стране начин на који Гухин грамзиви алгоритам функционише, који је основа за представљену имплемнтацију ОМК, почетни избор чворова има јако велики утицај на коначни резултат.

6.3.1 Стратегија корекције феромонског трага

Слично као и у предходно решаваним проблемима тако и у случају ПМПДС долази до стагнације ОМК. Дискусија аналогна предходно описаној о употреби ММАС варијације ОМК и СМПФ може се применити и на случају ПМПДС. Из ових разлога, у циљу унапређења употребе ОМК за дати проблем желимо да адаптирамо стратегију корекције феромонског трага. Идеја је и у случају ПМПДС, иста у смислу да када дође до стагнације алгоритма желимо да променимо феромонску матрицу помоћу анализе особина глобално најбољег нађеног решења. Адаптација предходно описаног метода се извршава на следећи начин.

Прво ћемо дефинисати функцију $\eta(v, V')$ која нам даје број чворова таквих да их $v \in V'$ јединствено покрива. Користећи ову функцију деминисаћемо функцију сумњивости односно непожељности елемента у коначном решењу.

$$Sus = \frac{1}{1 + \eta(v, V')} \tag{6.6}$$

У једначини 6.6 Sus представља непожељност елемента v у оквиру решења. Следећи корак у стратегији корекције феромона је избор случајног броја RK чворова који су део глобално најбољег нађеног решења

таквих да самостално покривају најмањи број чворова. За сваки чвор i вероватноћа да буде изабран за феромонску корекцију је следећа:

$$p_i(selected) = \frac{RK - RankSus(i, V')}{RK} \tag{6.7}$$

Као и у случају предходно решаваних проблема уместо вредности функције Sus за чвор, користи се његов ранг по овој функцији. RK је максимални број чворова које узимамо у обзир за корекцију феромона. Следећи корак је смањивање вредности феромона за изабране чворове:

$$\begin{aligned} &\forall i \in Selected \\ &\tau_i = \delta\tau_i \end{aligned} \tag{6.8}$$

Употреба функције Sus за избор непожељних чворова има исти проблем као и у случају ПТП и ПМТЧП да долази до појављивања мале групе јако непожељних чворова који бивају стално бирани. Тако да се и у овом случају уместо SuS користи нова унапређена функција која прати колико су често бирани поједини чворови и преферира избор нових:

$$CorSus(i, V') = Sus(i, V') * ExSusepect(i) \tag{6.9}$$

Праћење учесталости бирања се врши увођењем помоћног низа $ExSu-spect$, чији сваки елемент одговара неком од чворова графа. Иницијално се сваком елементу овог низа додељује вредност 1. У случају да је чвор i изабран врши се ажурирање низа $ExSuspect$ на следећи начин.

$$\begin{aligned} &0 < \lambda < 1 \\ &ExSuspect(i) = ExSuspect(i) * \lambda \end{aligned} \tag{6.10}$$

Овакав приступ корекцији феромонског трага где се јако смањује вредност феромона на непожељним сегментима решења се показао јако ефикасним у случају ПМТЧП. Колонија на овај начин, у наредним итерацијама избегава да дода непожељне чворове у ново конструисана решења. Међутим овакав приступ не унапређује ефикасност ОМК у случају ПМПДС, јер је у овом случају потребно да изабрани чворови буду повезани. Проблем који се појављује у случају када решење (подграф) мора бити повезан је што је јако вероватно да чворови који бивају избачени остављају остатак графа не повезаним. У наредним корацима алгоритма, мравима који граде решење је тешко да нађу нова добра решења због проблема одржавања повезаности.

Да би се избегла ова појава додаје се још један ниво корекције феромонског трага који омогућује лакше креирање нових добрих повезаних решења. Овакав вид корекције се постиже повећавањем феромона на

чворовима који нису састаставни део глобално најбољег решања а велика је вероватноћа да ће се налазити. Чворове који нису део најбољег решења а покривају велики број чворова који се у њему налазе, сматраћемо да су јако вероватни у новим добрим решењима.

Користећи предходну дискусију могуће је дефинисати метод за корекцију феромона за чворове који се не налазе у глобално најбољем нађеном решењу. Прво ћемо дефинисати функцију $Des(v, V')$ као број чворова који су у саставу најбољег решења V' таквих да су повезани са $v \notin V'$.

Следећи корак у стратегији корекције феромона је да се изабере случајан број RK' чворова који покривају највећи број чворова који су у саставу најбољег нађеног решења, или другим речима имају највећу вредност функције Des. За сваки овакав чвор i вероватноћа да буде изабран за корекцију феромонског трага је дата следећом формулом:

$$p_i(selected) = \frac{RK' - RankDes(i, V')}{RK'} \tag{6.11}$$

У једначини 6.11 се такође уместо вредност фукције Des за чворове користи $RankDes$, ранг по пожељности. RK' је највећи број елемената који ће бити узети у обзир за корекцију феромона. Последњи корак је ажурирање вредности за изабране чворове у феромонској матрици, односно њено повећање, које се извршава на следећи начин:

$$\begin{gathered} \forall i \in Selected \\ \tau_i = \frac{\tau_{max} + \tau_{min}}{2} \end{gathered} \tag{6.12}$$

За чворове чија је вредност феромона увећана, и у овом случају ће се пратити колико су често бирани употребом низа $ExSuspect$. Помоћу њега биће дефинисана унапређена функција пожељности $CorDes$ на исти начин као и у избору чворова који су у саставу најбољег решења.

Коначно потребно је још дефинисати критеријум стагнације у потрази за решењем. Критеријум који је коришћен у овом случају је да није дошло до промене глобално најбољег решења у последњих n итерација. У имплементацији овог алгоритма ће се користити две различите вредности n и n' за два метода корекције феромона.

6.3.2 Избор иницијалног чвора

Када је почетна основа за креирање алгоритма ОМК, за решавање ПМПДС, Гухин грамзиви алгоритама, перформансе су јако зависне од иницијално изабраног чвора. Разлог за овакво понашање је што решење

полако расте од иницијаног чвора кроз избор неког од његових суседа. Као што је предходно поменуто, хеуристичка функција је динамична, тако да предходно изабрани чворови не одређују само скуп кандидата, већ утиче и на то који ће од њих бити изабран. На овај начин долази до ефекта "снежне лавине" приликом креирање коначног решења. У случају ПТП овај проблем, али мање екстреман, је решен тако што се први чвор на случајан начин бира из скупа свих чворова графа. У случају ПМПДС ово није добар приступ јер релативно мали број чворова се налази у саставу минималног доминирајућег скупа па овакав поступак чини претрагу превише широком. У случају ПМТЧП, где је решење мали подскуп скупа чворова графа, на случајан начин се бира један елемент тренутно најбољег решења за иницијални чвор. Ако исти принцип употребимо у случају ПМПДС претрага ће бити превише сужена. Ово је последица тога што у случају ПМТЧП предходно изабрани чворови једино утичу на динамичку хеуристичку функцију, док у случају ПМПДС они утичу и на скуп потенцијалних кандидата.

Приликом избора иницијалног чвора потребно је направити баланс између ова два приступа, шта се може урадити на следећи начин:

$$InitVertex = \begin{cases} Random(V') & , s < s_0 \\ Random(V, \tau) & , s \geqslant s_0 \end{cases} \tag{6.13}$$

У једначини 6.13, s је случајна промењива са униформном расподелом, која одређује који ће се тип селекције вршити. s_0 је фиксиран параметар који одређује колика ће бити учесталост избора иницијалног чвора из глобално најбољег решења V' или из свих чворова графа V. У случају да је избор из V, расподела вероватноће је једино зависна од феромонског трага који одговара појединачним чворовима.

6.4 Тестови и резултати

Програм за тестирање представљеног алгоритма за решавање ПМПДС је писан у C#, и коришћено је исто програмско окружење као и у осталим тестовима. Сам програм имплементира следећи псеудо код:

```
Reset Graph Info
Reset Solution for all Ants
Select Initial Vertex for all ants

 while  (! AllAntsFinished)
   For All Ants
```

```
    If(AntNotFinished)
      begin
        add new vertex A to solution
        based on probability
        correct ants cover graph data
        calculate new set of candidates
        calculate new heuristic
        local update rule for A
      End
    End for
  End while

Compute DeltaTauI
Compute TauI

If(Iteration % n)
  Use CorSus for Pheromone Correction
If(Iteration % n')
  Use CorDes for Pheromone Correction
```

Да би се потврдила ефикасност представљене унапређене имплементације ОМК на ПМПДС извршена су два типа тестова. У првом се анализира ефикасност овог метода на стандардним тест проблемима за која су позната оптимална решења. У другој групи су генерисани тест графови на начин како је препоручен у раду [73], који одговарају проблему креирања кластера у ад хок мрежама.

ОМК алгоритам је имплементиран у ММАС варијацији. За обе верзије ОМК, са и без употребе корекције феромонског трага, извршено је по десет одвојених симулација. Поређен је просечни квалитет решења и стандардна девијација на свакој од инстанци тест проблема. Све колоније су имале следеће параметре: $q_0 = 0.9$ одређује однос истраживање/ експлоатације код правила одлучивања појединачних мрава, $p = 0.1$ и $\varphi = 0.1$ које одређују утицај глобалног и локалног правила ажурирања. Вредност параметра који дефинише правило избора за иницијални чвор је имао вредност $s_0 = 0.2$. Параметри који су коришћени за стратегију корекције феромонског трага имали су следеће вредности: коефицијент за смањивање вредности феромонског трага је био $\delta = 0.0001$, максимални број чворова који ће бити изабран за корекцију је био $RK = \frac{|V'|}{s}$ при чему је s био случајан целобројна вредност са униформном расподелом из интервала [2,10] а параметар за корекцију низа $ExSuspect$ је био

$\lambda = 0.9$. Параметри за критеријум стагнације су били $n = 20$ и $n' = 40$. У сваком од тестова је покренуто 10 колонија за оба случаја, прецизније за ОМК и ОМК комбинованом са стратегијом корекције феромона. У оба случаја су коришћене вредности од 0 до 9 за корене генератора псеудослучајних бројева.

У првој серији тестова коришћени су стандардни тест примери који су се појави на конференцији за логичко програмирање (Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09))) [74]. Највећи дозвољени број итерација по свакој колонији је био 350, односно 3500 решења је креирано. Упоређена су решења постигнута употребом ОМК комбиноване са стратегијом корекције феромонског трага, ОМК и са основном верзијом Гухиног грамзивог алгоритма. Резултате овог поређења могу се видети у табели 6.1.

Табела 6.1: Поређење ефикасности ОМК, ОМК са корекцијом феромонског трага (ОМК+КФТ), Гухиним грамзивим алгоритмом, и оптималним резултатима постигнутим са конференцији за логичко програмирање (LPNMR)

Проблем	LPNMR	Грамзиви	ОМК		ОМК+КФТ	
Димензије	Резултат		Просек	Ст. Дев	Просек	Ст. Дев
40*200	5	10	5.8	0.60	5.3	0.45
45*250	5	15	5.8	0.40	5.5	0.50
50*250(1)	8	15	8.1	0.54	8.0	0.00
50*250(2)	7	17	7.5	0.50	7.1	0.30
55*250	8	20	8.8	0.98	8.3	0.45
60*400	7	15	7.0	0.00	7.0	0.00
70*250	13	32	14.2	0.74	13.9	1.04
80*500	9	20	10.0	0.44	9.8	0.40
90*600	10	19	10.9	0.83	10.6	1.01
	8.00	18.11	8.68		8.34	

У табели 6.1 су једино дати резултате за проблеме који су имали задовољавајуће резултате (била су позната оптимална решења) који су дати у бенчмарк подацима са LPNMR'09. Најбоља решења која су нађена од стране колоније, која се уобичајено налазе у оваквим табелама, овде нису приказана јер су на свим тест примерима колоније успеле да нађу оптимална решења. Прво шта уочавамо приликом анализе ових резултата је да основни Гухин грамзиви алгоритам даје слабе резултате и

у просеку има грешку од 126% у поређењу са познатим оптималним решењем. Употреба ОМК у верзији MMAC даје велико унапређење у односу на ову вредност и грешка је свега 8.5%. Овакав резултат указује на то да употреба ОМК, са пажљивим избором иницијалних параметара, успева да превазиђе кратковидост грамзиве методе на којој је заснован. Коначно резултати који су постигнути додавањем стратегије корекције феромског трага на ОМК успевају даље да унапреде перформансе ОМК и у овом случају је грешка свега 4.25%.

За илустрацију ефикасности употребе ове верзије ОМК урађено је и поређење резултата постигнутих за овај проблем у случају да је он посматран као проблем одлуке. Резултати су узети од од следећих метода: одговорним скуповним програмирањем (Answer Set Programming (ASP)), методом задовољавања претпоставки (Propositional Satisfiability (SAT)) и методом спутаног програмирања (Constraint Programming (CP)) чији су резултати дати на веб адреси конференције LPNMR'09. У овом случају су се бенчмарк тестови стајали из 21 инстанце проблема повезаног доминирајућег скупа за графове различитих величина, и за сваки од њих је питање да ли постоји такав скуп са одређеним бројем чворова. За сваки од тестираних проблема покренуте су по две симулације колоније и у случају да би бар једна од њих постигла, до фиксног броја итерација (350), решење са одговарајућим бројем чворова сматрали би да је проблем задовољен и супротно у случају да није нађено такво решење. Тестови су извршени на сличном рачунару (рачунар коришћен за креирање резултата за ОМК је био незнатно бољи) као и на конференцији LPNMR'09. Рачунари су имали следеће конфигурације: случају LPNMR'09 је коришћен Dell OptiPlex 745, 1 CPU with 2 cores: GenuineIntel Intel(R) Core(TM)2 CPU 6600 @ 2.40GHz 4 GB RAM, док је у случају ОМК Dell OptiPlex 755, 1 CPU with 2 cores: GenuineIntel Intel(R) Core(TM)2 CPU E8500 @ 3.16GHz 3 GB RAM.

Програми за тестирање метода који су коришћене на LPNMR'09 су креирани коришћењем програмског језика C++, док је апликација ОМК урађена употребом C# шта програмина коришћеним на конференцији даје предност у брзини извршавања. Време које је било потребно представљеној имплементацији ОМК комбиноване са корекцијом феромонског трага за извршавање било је једнако 52 секунде за решавање тачно свих 21 инстанци проблема. У поређењу са овим резултатом, најбржи програм са LPNMR'09 је успео да реши ове проблеме за 36 секунди, док је следећи најбржи програмима било потребно редом 128, 169, 465 и 535 секунди. И ако овакво поређење није у потпуности прецизно ипак нам показује да је употреба ОМК за решавање овог проблема добар избор када је брзина извршавања главни критеријум.

У другој групи тестова, генерисани су графови на исти начин као што је препоручено у Чиновом раду [73]. Графови су генерисани на следећи начин. На фиксираном простору димензија $N*N$, одређен број случајних тачака са нормалном расподелом је изабран. Ове тачке ће представљати чворове графа. Ако је растојање између чворова i и j мањи од неке вредности R у том случају граф ће садржати ивицу (i, j). Проблеми су генерисани са различитим бројем чворова и различитим густинама ивица (вредности параметра R) и на њима су поређени резултати постигнути коришћењем ОМК и ОМК комбинованим са методом корекције феромонског трага. Сви параметри ОМК су исти као и у случају прве групе тестова осим броја дозвољених итерација колоније који је у овом случају био 5000, односно 50 000 решења. Број дозвољених итерација је повећан јер су димензије проблема у овом случају значајно веће. Резултате ових тестова можемо видети у табелама 6.2, 6.3.

За сваку од 41 тест инстанце проблема, поредимо најбоље нађено решење и просечно решење за ОМК и ОМК комбиновано са стратегијом корекције феромонског трага. Прво би напоменуо да и у случају проблема већих димензија, прости грамзиви алгоритам постиже лоше резултате. Оба типа коришћења ОМК унапређују минимално нађено решење 2 до 3 пута када се пореде са грамзивим алгоритмом. ОМК комбинован са корекцијом феромонског трага је поправило најбоље нађено решење у чак 18 случајева а покварило квалитет резултата у само три. Када се посматра квалитет просечног решења тада је резултат поправљен у 33 инстанце проблема а покварен у 6, док у два није дошло до промене. Предност коришћења стратегије корекције феромона је израженија у случају мањих и средњих димензија проблема. Ова чињеница се може објаснити тиме што су параметри који одређују понашања стратегије корекције феромона били одређени из анализе перформанси алгоритма на малим проблемима. Очекивано је да би се исти ниво побољшања могао постићи и у проблемима већих димензија ако би се изабрале боље вредности параметара.

Табела 6.2: Поређење ОМК и ОМК унапређене корекцијом феромонског трага (ОМК +КФТ) на тест проблемима малих димензија, генерисаним као у раду [73]

Рег. (N*N) Чворови	R	Грамзиви	ОМК		ОМК + КФТ	
			Мин	Просек	Мин	Просек
400	60	48	20.0	21.6	**1**9	**21.2**
80	70	33	16	17.0	**15**	**16.2**
	80	35	12	14.0	12	**13.1**
	90	41	11	11.8	11	**11.6**
	100	23	8	9.0	8	**8.9**
	110	25	8	8.5	8	8.5
	120	17	7	7.5	7	**7.2**
600	80	38	23	24.7	**22**	**23.6**
100	90	40	22	23.8	**21**	**23.6**
	100	38	17	20.0	17	**19.0**
	110	35	15	17.2	15	**16.8**
	120	36	15	16.2	**14**	**15.5**
700	70	96	46	50.7	46	**49.6**
200	80	89	41	**43.7**	41	43.9
	90	84	34	36.0	**33**	**35.7**
	100	75	28	**30.8**	28	31.0
	110	70	23	27.4	**22**	**26.4**
	120	68	21	23.6	21	**23.4**

Табела 6.3: Поређење ОМК и ОМК унапређене корекцијом феромонског трага на тест проблемима великих димензија, генерисаним као у раду [73]

Рег. (N*N) Чворови	R	Грамзиви	ОМК		ОМК + КФТ	
			Мин	Просек	Мин	Просек
1000	100	96	46	50.7	46	**49.6**
200	110	92	43	44.9	**42**	**44.8**
	120	82	37	39.9	37	**39.8**
	130	91	32	34.7	32	**34.9**
	140	76	30	31.3	**29**	31.3
	150	83	28	29.6	**26**	**28.8**
	160	86	**24**	26.6	25	**26.5**
1500	130	158	60	64.5	60	**64.3**
250	140	144	53	57.2	**52**	**57**
	150	170	51	54.9	51	**54.4**
	160	151	47	50.5	**45**	**49.8**
2000	200	178	55	**58.6**	**52**	58.8
300	210	151	51	53.5	**50**	**52.8**
	220	140	47	48.9	**45**	**48.4**
	230	166	44	47.5	44	**46.9**
2500	200	198	79	82.0	79	**81.5**
350	210	185	75	79.1	**74**	**78.2**
	220	205	**68**	**72.6**	69	73.8
	230	193	66	69.2	66	**68.9**
3000	210	259	99	**101.6**	**98**	104.0
400	220	225	**88**	**95.4**	91	97.6
	230	205	86	91.4	86	**90.3**
	240	210	82	85.8	**80**	**84.1**

Глава 7

Паралелизација ОМК

У општем случају када имамо систем од n паралелних процесора, од којих сваки има брзину једнаку k, њихова ефикасност је мања од једног процесора који има брзину $n * k$. Оваква појава је последица чињенице да се део процесорског времена мора утрошити за синхронизацију и комуникацију између процесора. Међутим, овакви паралелни системи су обично далеко јефтинији за креирање од једног моћног процесора, и зато је истраживање паралелизације од изузетне важности. Паралелизација се показала јако моћном у случају алгоритама који су засновани на популацији попут генетских алгоритама и ОМК [75].

Више различитих приступа је коришћено приликом паралелизације алгоритама ОМК. Показано је да је паралелизација коришћењем већег броја колонија ефикаснија у односу на када се она уводи додељивањем засебних процеса појединачним мравима. У случају генетских алгоритама имамо сличну појаву, да се најбољи резултати употребе паралеизације постижу када се популација подели на низ независних група, односно острва, и користи одређен вид комуникације међу њима. Када се овакав приступ примени на алгоритме заснована на популацији, често се постиже чак супер линеарно унапређење (ефикасност већа од једног процесора са брзином $k * n$) када се примењује на одређене проблеме [76].

У случају када се ОМК паралелизује употребом мулти колонијалних система, тип података који ће се користити за комуникацију је од јако велике важности. Нађена решења, феромонске матирце и разни други параметри су били тестирани као информација која ће се размењивати међу колонијама [77], [78], [79]. Показало се да се добри резултати постижу када се размењује најбоље до тада нађено решење, овакав приступ је коришћен и у експериментима чији опис је саставни део овог поглавља.

Да би се паралелизација ОМК дефинисала, поред информација које

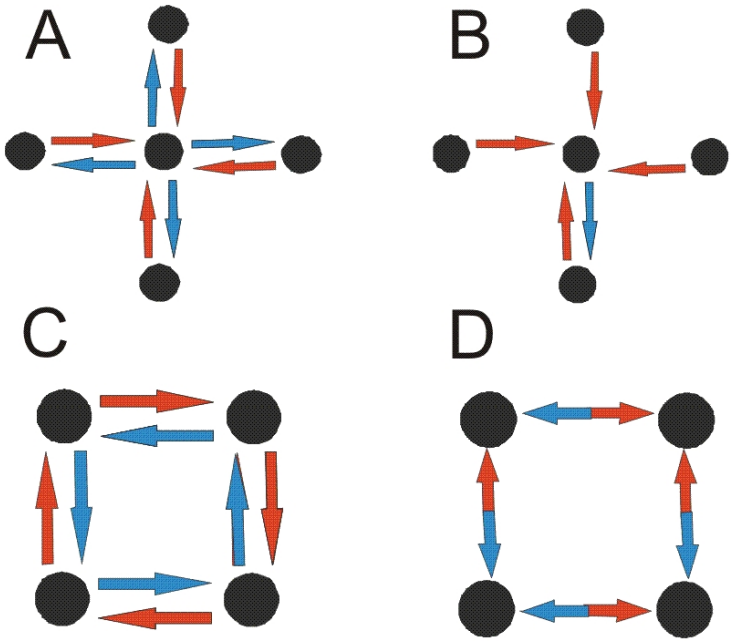
ће колоније размењивати међусобом, потребно је описати и метод комуникације и тачне интеракције међу њима. Прецизније потребно је описати могуће топологије, односно везе које постоје међу колонијама и алгоритме који потичу из ових веза. Стандардне топологије које се користе код паралелног ОМК су: потпуно повезана, замена најгорег решења, прстен и независна паралелна извршавања. У овом поглављу ће бити прецизно описани одговарајући алгоритми и њихове перформансе ће бити поређене на ПМТЧП.

7.1 Различите врсте топологија за паралелизацију ОМК

Као што је већ поменуто паралелни алгоритми су од изутетно велике важности за оптимизационе методе засноване на популацији јер често доводе до супер линеарног повећања ефикасности. Овај ниво побољшања се постиже у случају ОМК када се користе системи засновани на већем броју колонија. Овакви алгоритми се називају острвским. У њима је сваком острву односно колонији додељен засебан процес. Комуникација се заснива на размени најбољих решења које су појединчне колоније успеле да нађу. Анализа се у овом поглављу посвећује различитим типовима комуникације међу колонијама или прецизније топологији. Под топологијом се подразумева које ће колоније бити међусобно повезане и размењивати податке. У наставку ће бити описане делимично промењене топологије и начини комуникације из рада [80].

Потпуно повезана топологија

У потпуно повезаној топологији, постоји n колонија, и свака од њих је покренута са различитим семеном генератора псеудослучајних бројева. Колоније комуницирају и размењују информације у циљу проналажења најбољег решења. Интеракција међу колонијама се изводи на следећи начин. Најбоље нађена решења се узимају од свих колонија. Најбоље од свих решења, или у неким случајевима индекс најбоље колоније, се проналази и прослеђује до свих n колонија. Свака од ових колонија замењује своје најбоље решење добијеним. Илустрација ове комуникације се може видети на слици 7.1.



Слика 7.1: Различите топологије које се користе приликом паралелизације ОМК. А) потпуно повезана топологија В) замена најгорег решења С) комуникација прстена D) комуникација прстена размене

Замена најгорег решења

И овом случају, се налази најбоље решење које су постигле n колонија, али уместо да се најбоље решење прослеђује свим колонијама, оно се шаље само колонији која је имала најгоре решење. Ова колонија замењује своје решење новим бољим. Овај приступ има предност у односу на потпуно повезану топологију што је број порука које морају да се шаљу међу колонијама смањен на приближно пола (слика 7.1).

Комуникација прстена

Овај вид комуникације се заснива на топологији прстена. У њој је свака колонија повезана једино са својим суседима. Прецизније у систему са n колонији, при чему је свакој од њих додељен јединствени индекс i са вредношћу од 0 до $n-1$. Колонија са индексом i ће једино дати своје решење колонији са индексом $(i-1) mod\, n$ а добити решење од колоније са индексом $(i+1) mod\, n$. У случају да колонија добије решење које је боље од њеног најбољег нађеног решења, она га замењује новим (слика 7.1). Оваквом комуникацијом се даље смањује број порука који је потребно

разменити међу колонијама, и нестаје потреба за процесом који треба да анализира податке од свих колонија.

Комуникација прстена размене

Овај облик комуникације је јако слична комуникацији прстена у смислу да колоније коминицирају само са својим суседима. Дефиниција ове нове топологије је саставни део тезе. Као што је предходно поменуто у топологији комуникација прстена, од две суседне колоније, она са лошијим решењем замењује своје решење бољим. У прстену замене уместо преписивања ради се замена односно лошија колонија замењује своје решење бољим, док боља колонија чини супротно замењује своје боље решење лошијим. На овај начин лошије решење није у потпуности изгубљено, већ се комбинује са феромонском матрицом које је створила колонија са бољим решењем. У пракси колонија i једино размењује решење са колонијом $(k-1) mod\, n$ (слика 7.1).

Независна паралелна извршавања

У овој имплементацији паралелног ОМК не постоји никаква комуникација међу колонија. Извршавају се засебни секвенцијални алгоритми са различитим коренима генератора псеудослучајних бројева у n различитих процеса. Када све колоније стигну до максималног броја итерација узима се најбоље решење од свих њих. Ова метода има предност што није потребано никакво додатно програмирање и даје нам добру основу за поређење понашања осталих метода.

7.2 Тестови и резултати

У овој секцији ће бити анализирани методи комуникације за паралелне алгоритме ОМК, представљене у предходном одељку, примењене на ПМТ-ЧП. Програм за тестирање ових алгоритама ће бити представљен у наредном поглављу.

У креираном софтверу, паралелизација је имплементирана тако да је свака колонија у засебној прогамској нити (thread). Поред нити за сваку колонију, постоји још једна нит која представља кластер колонија, односно, мастер класу која је задужена за одржавање комуникације међу нитима. Оваква имплементација није савршена репрезентација правог паралелног система извршавања различитих топологија на мултипроцесорском систему или рачунарској мрежи. Главне слабости (разлике) оваквог начина представљања реалног паралелног система су што се

коминикација између колонија извршава без кашњења, не може доћи до нестанка података приликом комуникације међу колонијама или да дође до неочекиваног прекида рада неке од колонија. Из овог разлога фокус алализе ће бити на квалитету резултата које овакве топологије постижу а не на брзини извршавања.

У тестовима се користе проблеми различитих величина од 50, 100 и 150 чворова. Такође су тестиране различите величине кластера колонија, који су се састојале од 5 или 10 колонија које се истовремено извршавају. У сваком од експеримената свака од колонија имплементирана у MMAC варијацији OMK. У табелама 7.2, 7.3, 7.4, 7.5, 7.6 свакој од топологија је дато једнако време извршавања а размена информација међу колонија је била вршена сваких n временских интервала.

Када се пореди секвенцијални алгоритам OMK са паралелизованим верзијама, коришћен је стандардни приступ за давање истог време извршавања. Пореди се квалитет најбољег нађеног решења за паралелни алгоритам F_t^k, где k представља број покренутих колонија које међу собом комуницирају по принципима описаним у претходној секцији, а свака од њих добија t процесорског времена. Ови резултати се пореде са секвенцијалном алгоритмом F_{kt} која има укупно време извршавања $k * t$. Овакво поређење ставља секвенцијални алгоритам у нешто неповољнији положај у поређењу са паралелним алгоритмима због релативно дугог времена извршавања. Као што је поменуто у раду [80] секвенцијални алгоритми се понашају ефикасније у односу на паралелне када је време извршавање краће. Резултати оваквих тестова за секвенцијалне алгоритме и кластере од 5 колонија се могу видети у Табелама 7.1, 7.2 и 7.3. У овим табелама су дати резултати за покретање 5 независних паралелних и секвенцијалних алгоритама. Поређење се врши по најбоље нађеним решењима за свих пет пуштања и њиховим просечним решењима.

Табела 7.1: Број чворова 50, број ивица 209, број колонија 5, квалитет решења постигнутог грамзивим алгоритмом 2038, време извршавања 1 минута, фреквенција комуникације 10 секунди

Топологија	Најбољи	Просек
Потпуно повезана	1712	1738.8
Замена најгорег	1701	1725
Прстен	1660	1729.2
Прстен замене	1667	1704.6
Независна паралелна извршавања	1695	1719.2
Секвенцијални алгоритам	1730	1749.2

Табела 7.2: Број чворова 100, број ивица 450, број колонија 5, квалитет решења постигнутог грамзивим алгоритмом 4548, време извршавања 2 минута, фреквенција комуникације 10 секунди

Топологија	Најбољи	Просек
Потпуно повезана	3470	3521.6
Замена најгорег	3493	3519
Прстен	3464	3516.8
Прстен замене	3464	3503.6
Независна паралелна извршавања	3503	3540
Секвенцијални алгоритам	3542	3583.4

Табела 7.3: Број чворова 150, број ивица 450, број колонија 5, квалитет решења постигнутог грамзивим алгоритмом 6782, време извршавања 4 минута, фреквенција комуникације 30 секунди

Топологија	Најбољи	Просек
Потпуно повезана	5672	5763.2
Замена најгорег	5634	5735.6
Прстен	5601	5726.
Прстен замене	5643	5738.8
Независна паралелна извршавања	5746	5735.6
Секвенцијални алгоритам	5788	5852.6

Анализом резултата у случају малих тест проблема са 50 чворова у Табели 7.1, прво уочавамо да су сви паралелни алгоритми дали боље резултате од секвенцијалног. Главни разлог оваквог односа је чињеница да код секвенцијалног алгоритма, као последица великог броја извршених итерација, претрага релативно рано улази у стање стагнације. Да је у овом тесту коришћено краће време извршавања ова разлика би била далеко мања.

У случају малих тест проблема, алгоритам који одговара потпуно повезаном приступу даје лоше резултате у поређењу са осталим топологијама. Овакво понашање се може објаснити тиме што је у неким случајевима цео кластер фокусирао своју претрагу у лошој области скупа решења коју је тешко напустити. Са друге стране у случају да је почетак претраге у доброј области нема потребе за великим бројем итерација у току претраге због малих дименизија проблема. Можемо закључити да је чување разноликости приликом потраге за решењем од јако велике важности, јер топологије које су имале овакву особину су постизале боље резултате.

У случају већих димензија проблема, фокусирање претраге је давало боље резултате, јер је фокусирана област скупа решења довољно велика да различите колоније не претражују исте области. У овом случају топологије које су интензивирале претраге у добрим областима су постизале сличне добре резултате. У случају великих проблема алгоритам независних пуштања заостаје за другим алгоритмима када се пореди квалитет решења, али и даље постиже добре резултате.

Табела 7.4: Број чворова 50, број ивица 209, број колонија 10, квалитет решења постигнутог грамзивим алгоритмом 2038, време извршавања 1 минута, фреквенција комуникације 10 секунди

Топологија	Најбољи	Просек
Потпуно повезана	1735	1747
Замена Најгорег	1701	1730.6
Прстен	1672	1722.6
Прстен замене	1673	1712.4
Независна паралелна извршавања	1699	1719.8
Секвенцијални алгоритам	1730	1749.2

Табела 7.5: Број чворова 100, број ивица 450, број колонија 10, квалитет решења постигнутог грамзивим алгоритмом 4548, време извршавања 2 минута, фреквенција комуникације 20 секунди

Топологија	Најбољи	Просек
Потпуно повезана	3462	3502.8
Замена најгорег	3499	3524.8
Прстен	3499	3507.8
Прстен замене	3460	3491
Независна паралелна извршавања	3493	3505.8

У другој групи експеримената, тестиран је ефекат повећања броја колонија са 5 на 10, док је укупно време калкулација остало исто. У случају малих проблема (Табеле 7.1, 7.4), ово повећање није добар избор и није повећало квалитет резултата. Оваква појава се може објаснити тиме што је простор потенцијалних решења релативно мали, и због тога већи број колонија истражује исту област решења. У случају проблема средње величине (табеле 7.2, 7.5) резултати су били слични нешто бољи у случају већег броја колонија. Коначно, У случају великих проблема (табеле 7.3, 7.6) повећавањем броја кооперативних колонија је значајно погоршало квалитет постигнутих решења, чак се приближавају резултатима постигнутим од стране секвенцијалног алгоритма. У овом случају,

Табела 7.6: Број чворова 150, број ивица 450, број колонија 10, квалитет решења постигнутог грамзивим алгоритмом 6782, време извршавања 4 минута, фреквенција комуникације 30 секунди

Топологија	Најбољи	Просек
Потпуно повезана	5779	5823
Замена најгорег	5800	5836.6
Прстен	5786	5821.4
Прстен замене	5765	5804
Независна паралелна извршавања	5754	5788.2

овакво понашање је вероватно последица тога што ни једна колонија нема довољно времена да интензивира претрагу у близини добрих резултата. Уместо тога само се велики део простора решења површно претражује. Оваква појава нам указује на зависност која постоји између величине проблема и броја колонија које интерагују, односно да постоји оптималан однос између времена израчунавања, величине проблема и броја колонија.

Када се пореди ефикасност различитих топологија, имамо резултат да комуникација прстена размене даје у просеку најбоље резултате. Само у једном случају овај тип топологије је имао лошије резултате, шта је могућа последица релативно малог броја тестова, односно да је у питању статистичка грешка.

Добро понашање овог вида комуникације међу колонија се може објаснити особинама произилазећег алгоритма. Прво, разноврсност претраге се не губи брзо зато што ни једно нађено решење се не одбацује и губи преписивањем. Ово је предност у односу на видове комуникације који фокусирају своју претрагу само у областима у којима се налазе добра решења попут потпуно повезане, замене најгорег и комуникације прстена. Ипак ефекат размене решење усмера претрагу у правцу добрих решења, и не задржава се у истим областима као у случају независних паралелних пустања.

Понашање се може лакше разумети ако посматрамо шта се дешава приликом размене решења између две колоније A и B. Предпоставимо да колонија A има боље решење од колоније B. Након размене лошија колонија B ће добити решење које је бољег квалитета, и у каснијим итерацијама када је довољно феромона депоновано почеће да претражује исти регион простора решења као и колонија A пре размене. У случају колоније A која добија решење које је лошијег квалитета у односу на оно које је већ имала, могуће је да ће завршити тако што ће претраживати исти простор као и колонија B пре размене, после великог броја итерација.

Међутим ово није вероватна последица размене решења. Појединачни мрави не зависе директно од најбољег до тада нађеног решења већ од феромонске матрице. Феромонски траг ће се полако трансформисати од оног које одговара добром решењу (које потиче од колоније A) ка лошијем трагу (које одговара колонији B) и у овом периоду је јако вероватно да ће неки од мрава наћи неко решење које је боље од решења од оног које на почетку имала колонија B.

Закључак извршених тестова је потврдио да и у случају ПМТЧП паралелизација ОМК, чак и у случају изостанка комуникације међу процесима, добар избор и даје боље резултате од секвенцијалоног алгоритма. Из изведених тестова било је могуће уочити да постоји зависност између броја колонија, величине проблема и расположивог процесорског времена за постизање оптималних резултата. Показано је, да не постоји један универзално најбољи метод комуникације између колонија, али је ипак ново предложени приступ комуникације прстена размене дао боље резултате у просеку. Важно је напоменути да на Виндоус платформи кроз коришћење нити паралелизација даје боље резултате од секвенцијалног алгоритама.

Глава 8

Програмско окружење за решавање проблема употребом ОМК

Велики број проблема је решаван употребом оптимизације мрављим колонијама. Међутим софтвери који су прављени у ову сврху су најчешће специфично конструисани за тачно одређен специфичан проблем. Неке од апликација су АЦОТСП (ACOTSP), АНТНЕТ (ANTNET) и ГУИАнт-Минер (GUI ANT-Miner) [81] које се доступне путем интернета и које се могу користити под условима ГПЛ (General Public Licence). Овакви софтвери могу бити коришћени као путокази за креирање новог софтвера за решавање проблема коришћењем ОМК. Њихови програмски кодови могу чак бити и модификовани у ову сврху, али је обично једноставније написати цео програм изпочетка. Интересантан је пројекат под називпм АНТ-Ц (Ant-C) који је отворен за проширење имплементацијама за ОМК за специфичне проблеме. Нажалост овај пројекат има значајну отежавајућу околност да је писан у програмском језику С и као последица тога није објектно орјентисан.

Добро је позната чињеница да је објекто орјентисан приступ, најпогоднија методологија за креирање софтвера који има могућност реупотребљивости и модуларности, када желимо да решавамо већи број сличних проблема. Уго Чирико је креирао објекто орјентисан оквир за решавање проблема коришћењем ОМК. Назив његовог оквира је ЈФАЦС (Java framework for Ant Colony Systems) [82]. ЈФЦАС има две велике мане. Прво је да не постоји никакво графичко окружење и да је концепт паралелизације везан за мраве а не колоније. Прецизније паралелизација је изведена тако што се за сваког мрава унутар колоније креира процес а не на далеко ефикаснији начин где су процеси везани за засебне колоније.

У овом поглављу се представља ново креирани објектно орјентисан оквир за решавање проблема употребом ОМК, који је коришћен у свим експериментима који су саставни део ове тезе. Назив овог пројекта је ГРАФ-АНТ (Graphical Framework for Ant Colony Optimization). Овај пројекат је писан у C# програмском језику.

8.1 Анализа потреба оквира за решавање проблема употребом ОМК

Оквир је специфичан тип софтверске библиотеке који је сличан АПИ-ју (Application program interface) у смислу да је то пакет који служи за брзи развој нових апликација. Основна разлика је у томе што је АПИ скуп функција које позивамо, а оквир се састоји из хијерархије абстрактних класа. Корисник оквира креира програмски код за одређене методе ових класа. Оквире карактерише употреба обрнутог контролног механизма који се заснива на концепту "Не зовите нас, ми ћемо звати Вас" за комуникацију са корисничким кодом, односно функцијама креираним од стране корисника. Значи оквир обезбеђује целокупну контролну структуру а од корисника тражи једино да имплементира детаље специфичне за решавани проблем. Овакав приступ је природан за имплементацију ОМК.

Када се развија овакав систем јако се корисно имати ефикасно графичко окружење које нам могућује праћење понашања колоније приликом решавања задатог проблема. Ефикасност ОМК алгоритма је јако зависна од вредности својих улазних параметара, а ове параметре је најчешће једино могуће добити из великог низа тестова. Процес налажења ових вредности је далеко једноставнији уз добро графичко окружење. Одговор на овакве потребе је био избор C# као језика за развој ГРАФ-АНТ оквира, пошто омогућаје једноставно креирање моћног ГКИ.

ГРАФ-АНТ је конструисан као низ класа које омогућавају креирање апликације које користе ОМК. У њему постоји основна абстрактна класа RAntVisualiserAbstract која је задужена за основне детаље око приказа проблема. На овај начин је се избегава редундантност програмског кода и убрза развој нових модула. Ова класа осим самог графичког приказа је и задужена за чување података који нису директно повезани са ОМК. Пример оваквих информација је позиција чворова графа у проблему трговачког путника, јер за ОМК су важне једино растојања међу њима, а опет у неким алгоритмима саме позиције се могу искористити.

Појављује се и потреба за засебном класом која ће чувати информа-

ције везане за граф које су директно везане за ОМК и њих ћемо сместити у класу под називом RAntGraphAbstract. Ова класа ће држати податке попут вредности хеуристике, феромонског трага и тежина. Веза између ове две класе је јака и превазилази само графички приказ решења. У неким случајевима је чак неопходно израчунати параметре ОМК из дате визуелизације. У овом оквиру овај вид комуникације међу класама је отворен.

Следећи важан део алгоритма ОМК је имплементација понашања појединачних мрава овај задатак је повезан са абстрактном класом RAntAbstract. У њој постоје абстрактне методе које је потребно имплементирати за специфичне проблеме попут GetNextHeuristicStep(), CalcProbabilityDistribution(). Ова класа такође имплементира и неке стандардне потребе за разне типове проблема попут ажурирање локалне путање, добијање конкретних вредности из расподеле вераватноћа, праћење стања траженог решења.

У највећем делу проблема који се решавају употребом ОМК решење није једноставан низ индекса чворова графа већ комплексна структура која има своје карактеристике и рестрикције. Неколико корака имплементације ОМК користе путање креиране од стране мрава на разне начине, а и потребно је да графичко окружење може да га визуелизује. Ове особине нам указују на потребу за дефинисањем још једне абстрактне класе RAntGraphPathAbstract која ће имплементирати неке од стандардних потреба за манипулацијом путања које креирају појединачни мрави и која се користи за комуникацију између различитих класа унутар ГРАФ-АНТ-а.

Класа RAntColonyAbstract је задужена за стандардне потребе мрављих колонија попут праћења до сада најбољег решења, бројање итерација, провера стагнације алгоритма. Као што је предходно већ поменуто, постоје разне варијације алгоритма мрављих колонија попут мрављег система, елитистичке стратегије, система мрављe колоније, мин макс система мрављe колоније, рангираног мрављег система. Овакве варијације су највећим делом независне од проблема који се решава и генерално су виша апстракција која је везана ОМК. RAntColonyAbstract управо имплементира овакве варијације које се касније могу користити за конкретне имплементације ОМК.

Различитим хибридизацијама се унапређују перформансе ОМК. Хибридизације се најчешће заснива на комбиновању ОМК са неком другом методом за решавање оптимизационих проблема које су предходно поменуте. У ГРАФ-АНТ-у је омогућена хибридизација у два правца. Први је додавањем локалне претраге да би се унапредила решења нађена од стране појединачних мрава. Други вид хибридизације који је предвиђен

приликом развијање овог оквира је корекције феромонсоког трага у случају да је дошло до стагнације у потрази за решењем. Класа RAntColonyAbstract омогућује додавање хибридизације имплементацијом одговарајућих метода.

ОМК се јако успешно паралелизује и овим поступком се постиже чак супер линеарност перформанси. Из овог разлога је било изузетно важно додати и ову могућност у наш оквир. Паралелизација је спроведена користећи модел острва односно везивањем засебних колонија за засебне процесоре. Заправо ГРАФ-АНТ симулира мулти процесорску машину креирајући низ процеса који се паралелно извршавају под контролом оперативног система. Класа која је посвећена регулисању паралелизације је RAntColonyClusterAbstract. Комуникација између колонија код паралелног извршавања је од јако велике важности [83]. Она има две основне карактеристике и то су подаци који ће се слати од једно колоније до друге и које ће колоније размењивати податке. У ГРАФ-АНТ-у је изабрана размена најбољих пронађених решења међу колонијама. RAntColonyClusterAbstract имплементира више стандардних модела комуникација међу колонијама, односно топологија. Ове топологије су сличне оним у раду [80] које су ближе описане у поглављу о паралелизацији ОМК.

Као што је предходно поменуто мравље колоније имају различито понашање и брзину конвергенције у зависности од својих параметара и варијације основног алгоритма. У случају система са више колонија имамо ситуацију да су неке од њих међусобно комплементарне. Пример су колоније са ниским и високим нивоом истраживања. Повољна комбинација колонија у оваквом систему може значајно повећати ефикасност целокупног система. Из овог разлога у ГРАФ-АНТ-у постоји могућност засебног подешавања параметара појединачних колонија у оваквом систему. Препознавање добрих комбинација параметара је олакшано лаким приступом параметарима, и визуализацијом претраге у реалном времену.

Из ових разлога је ГКИ ГРАФ-АНТ-а посвећена значајна пажња, и препознате су следеће важне карактеристике које су и имплементиране:

- Лако и једноставно подешавање параметара који припадају различитим групама. Групе параметара су следеће: одлучивања појединачних мрава, варијације алгоритма мрављих колонија, хибридизација, визуализација и мулти колонија.

- Могућност контроле семена генератора случајних бројева ради боље евалуације ефекта промене вредности параметара.

- Лак приступ појединачним колонијама унутар система са више колонија. Овакав приступ је потребан за подешавање параметара

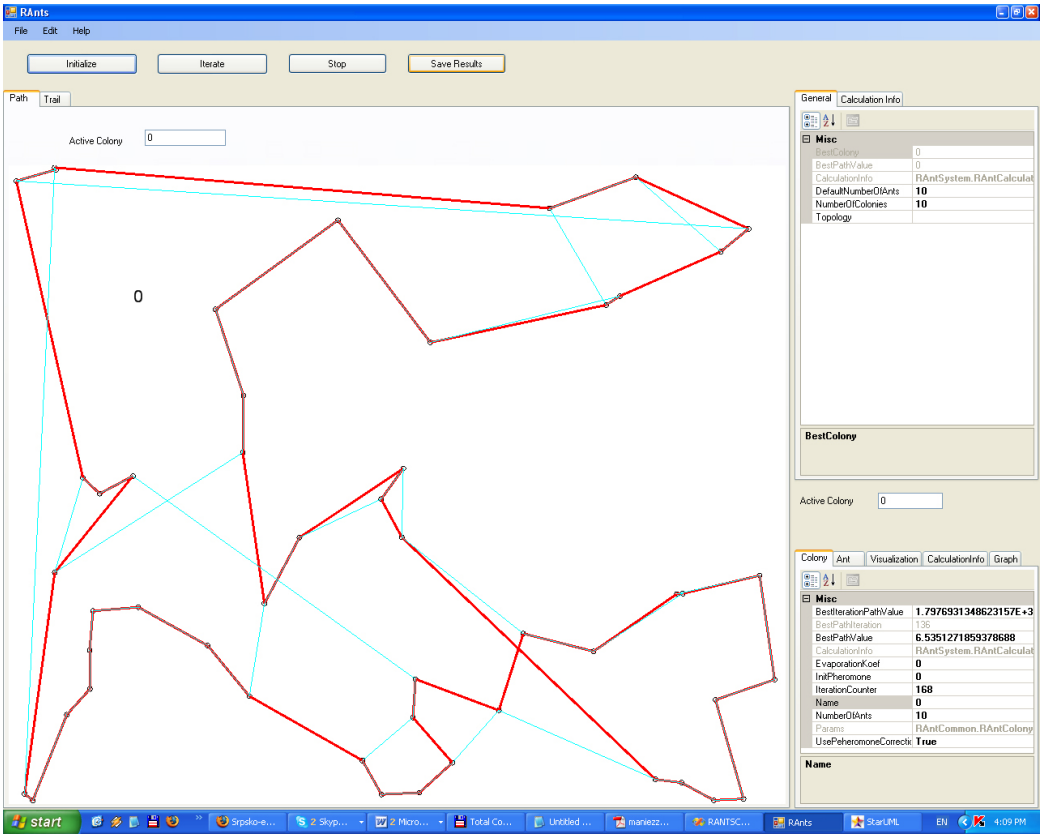
и визуелизацију.

- Могућност симултане промена параметара за већи број колонија унутар система са више колонија.
- Могућност праћења стања феромоског трага.
- Постојање реакције у реалном времену на промену вредности параметара колоније.
- Могућност визуелног поређења резултата различитих колонија унутар система са више колонија.
- Адаптација ГКИ за широк спектар различитих проблема које је могуће решити уз помоћ ОМК.
- Избор модула који решавају различите проблеме употребом ОМК.
- Могућност чувања и учитавања скупова параметара.
- Чување резултата и стања ОМК алгоритама за даљу анализу.

У ГРАФ-АНТ-у су имплементиране све ове карактеристике као саставни део ГКИ-а. ГКИ је повезан са модулом који дефинише одређен проблем кроз низ предходно описаних класа. ГРАФ-АНТ је развијен у C# језику јер је било очигледно да ће бити неопходна надоградња основног ГКИ за специфичне решаване проблеме, а овако нешто је могуће извести употребом проперти грид (property grid) компоненте. Ова компонента аутоматски повезује ГКИ и пропертије које постоје у класама.

Строго говорећи, ГКИ није саставни део окружења, из разлога што би то јако ограничило потенцијалне употребе. Са друге стране, поменуте ГКИ карактеристике су задовољавајуће за већину корисника. Чак и у случају корисника којима овај вид визуелизације није у потпуности довољан, овај ниво ће бити користан у раним фазама тестирања и развијања алгоритма. Дизајнирање и програмирања ГКИ одузима велики део времена развоја неког софтвера. Из овог разлога је јако корисно имати овај део програма предходно спремним за коришћење.

У тренутној фази ГРАФ-АНТ окружење и одговарајући ГКИ је у облику Microsoft Visual Studio 2010 C# пројекта. Пројекат се састоји из низа датотека у којем се налазе абстрактне класе које чине окружење. Пројекат такође садржи шаблон класе које садрже имплементацију ПТП. У случају решавања новог проблема употребом ОМК, најједноставније је адаптирати овај шаблон.

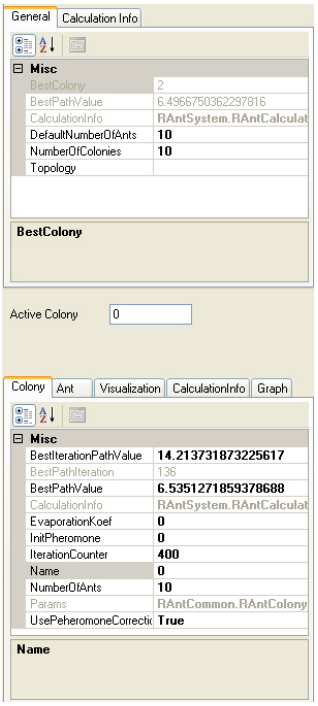


Слика 8.1: Изглед главног екрана ГРАФ-АНТ окружења са просирењем за ПТП

На сликама 8.1, 8.2 може се видети изглед ГКИ за ПТП. Може се приметити ра се параметри за појединачне колоније и системе колонија налазе у одвојеним таб контролама. За сваки од њих групе сродних параметра су стављене у посебне таб странице (tab-pages). На левој страни слике 8.1 може се видети тренутно најбоље решење у црвеној боји а најбоље решење нађено у последњој итерацији плавом бојом.

Екран који приказује ове резултате је анимиран у реалном времену, шта је важна особина јер нам омогућује праћење рада алгоритма и убрзава његов развој. Анимација у реалном времену је могуће искључити ради повећања брзине израчунавања, шта је и уобичајенији начин употребе код решавања проблема у фази када је основни алгоритам у потпуности развијен.

ГКИ има могућност основне контроле програмских нити који је неопходан у случају коришћења система колонија који се паралелно извршавају. Над нитима које одговарају појединачним или свим колонијама у



Слика 8.2: Изглед прозора за подешавање параметара унутар ГРАФ-АНТ-а

система је могуће извршити следеће команде: засутављање, паузирање, покретање, реиницијализација и чување тренутног стања система. Операције са датотекама прате стандардне команде и могу се извршити преко менија ГКИ.

8.2 Имплементација ПТП коришћењем ГРАФ-АНТ-а

У овој секцији се илуструје употреба ГРАФ-АНТ окружења на примеру ПТП. Слика 8.3 представља УМЛ (unified modeling language) дијаграм класа самог оквира и његова екстензија класама које су неопходне за имплементацију ОМК на ПТП. Прво, треба напоменути да у дијаграму постоји велики број конекција међу класама у ГРАФ-АНТ-у као што су композиције, агрегације и зависности. Ово је последица комплексности која постоји у системима који се састоје из више колонија. Корисник окружења је изолован од овога, једино што се од њега тражи је да имплементира специјализоване класе које су неопходне за решавање кон-

кретног проблема.

Када се креира модул за решавање ПТП прво је потребно креирати класе наслеђене од абстрактих класа ГРАФ-АНТ-а као што је приказано на слици 8.3. Прво се креира класа $RAntVisualiserTSP$ наслеђена од $RAntVisualiserAbstract$. У овом случају није потребна имплементација наслеђених метода $Visualise$ и $ShowPath$ које служе за визуелизацију проблема и решења (у овом случају путање) које креирају мрави јер су предходно имплементиране у абстрактној класи.

Основна класа за визуелизацију имплементира исцртавање чворова на одређеним позицијама, а већ имплементирано исцртавање ивица између чворова као претпостављени метод визуелизације решења. У неким случајевима, визуелизација тренутног стања феромонског трага током извршавања ОМК, може бити од велике користи током развијања алгоритма. У ГРАФ-АНТ-у је то могуће учинити имплементацијом методе $ShowTrail$. Није обавезно имплементирати овај метод, шта није учињену у случају ПТП.

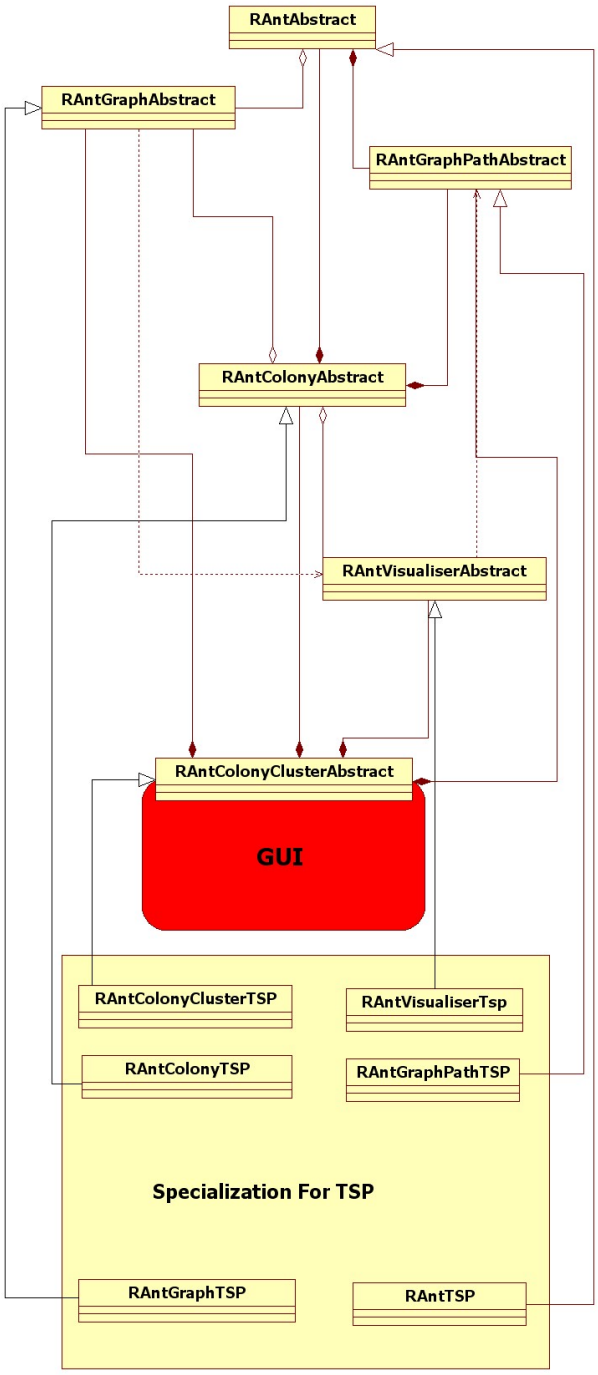
Такође је потребно имплементирати класу $RAntGraphTSP$ класа која контролише понашање феромонског трага. Да би ово било могуће потребно је користити информацију о удаљености односно цени ивица које спајају одређене чворове. Ови подаци су саставни део $RAntVisualiserTSP$, у случају ПТП најједноставније је као цену користити еуклидску удаљеност између градова односно чворова. У класи $RAntGraphTSP$ постоји констуктор чији је параметар објекат класе $RAntVisualiserAbstract$. Овим конструктором на једноставан начин се могу проследити све битне информације за иницијализовање класе феромонског трага.

$RAntGraphPathTSP$ је исти као и основна абстрактна класа из разлога што не постоје била каква ограничења за чворове унутар решења осим да су јединствена. У случају решавања проблеме у ком постоје ограничења потребно је имплементирати метод $CheckNewNode$ која проверава валидност чвора који је потенцијални део новог решења.

У класи $RAntTSP$, која регулише понашање појединачних мрава, потребан је додатни програмски кода у конструктору који повезује податке о графу, путањи и омогућава једноставну визуелизацију. Начин иницијализације мрава у колонији је још један корак који треба регулисати у зависности од конкретног проблема. Овај поступак је подељен на две виртуалне функције $Reset$ и $ResetPath$. $Reset$ служи за реиницијализацију објекта који представњају мраве, а $ResetPath$ ресетује податке везане за делимично констуисана решења. У случају ПТП потребно је једино почетни град поставити на случајно изабрани град односно чвор.

Класе $RAntColonyTSP$ и $RAntColonyClusterTSP$ су исте као и у основној абстрактној класи осим метода које служе за креирање одгова-

рајућих мрава и колонија. У класи $RAntColonyTSP$ било је потребно имплементирати виртуелну методу $CreateAnts$ која креира објекте класе $RAntTSP$ који су у колоније. Слично, у класи $RAntColonyClusterTSP$ било је потребно имплементирати метод која креира објекте који представљају мравље колоније одговарајућег типа. Када су испрограмиране све класе потребне за решавање одређеног проблема, за повезивање ГКИ није потребно ни мало додатног програмског кода. Класа $RAntColony-Cluster$ повезује колоније и њихове групе параметара са ГКИ.



Слика 8.3: УМЛ дијаграм за абстрактне класе унутар ГРАФ-ANT-а и класе унутар проширења за ПТП

Глава 9

Закључак

У овој докторској дисертацији је представљен један приступ решавању тешких оптимизационих проблема из области графова. Прецизније, више различитих проблема је решавано новом и актуелном метахеуристиком која се заснива на опонашању колоније мрава односно употребом оптимизације мрављим колонијама. У уводу рада су приказани и други савремени методи који решавају оптимизационе проблеме, од којих су неки касније поређени са ОМК.

У овом раду ОМК је коришћена за решавање проблема трговачког путника, проблема мининалног тежинског чворног покривача графа, проблема минималног тежинског доминирајућег скупа и проблема минималног повезаног доминирајућег скупа. У случају два проблема ПТП и ПМТЧП који су предходно већ били решавани употребом ОМК, извршена је детаљна анализа употребе различитих варијација алгоритма ОМК. Из овакве анализе је закључено да не постоји једна најбоља варијација ОМК која се може користити на сваком проблему већ постоји зависност од типа проблема, његових димензија и расположивих рачунарских ресурса. Описане карактеристике варијација ОМК на овим проблемима нам омогућују да у случају решавања сличних проблема можемо искористити и убрзати развој алгоритма.

У овој тези је употребом ОМК решаван и ПМТДС. Он, по знању аутора, никад раније није решаван употребом ове метахеуристике. Приликом решавања овог проблема развијена је нова хеуристика која се користи за стварање грамзивог алгоритма. Употребом ове хеуристике, грамзиви алгоритам постаје ефикаснији у односу на постојеће, што је потврђено у извршеним експериментима. Коначно, ОМК која се заснива на овом грамзивом алгоритму постиже далеко боље резултате од њега самога.

У случају ПТП, ПМТЧП и ПМПДС који су предходно решавани

употребом ОМК, унапређена је њена употреба на неколико начина. Прво, у случају ПТП и ПМТЧП перформансе ОМК су унапређене хибридизацијом основног алгоритма. Хибридизација се заснива на употреби информација које имамо о до тада најбољем нађеном решењу. Идеја је да се на основу односа које постоје између делова решења (ивица у ПТП, чворова у ПМТЧП) усмери колонија мрава на делове скупа решења у којим је већа вероватноћа да постоје решења високог квалитета. Прецизније, уводи се метод корекције феромонске матрице која се користи када је дошло до стагнације у потрази за бољим решењима. Она користи апостериори хеуристику на елементима да сада најбољег нађеног решења.

Корекција феромонске матрице се заснива на концепту непожељности одређених елемената у добрим решењима. Пример непожељних елемената у случају ПТП су јако дугачке или укрштене ивице. У случају ПМТЧП непожељни су чворови који имају велику тежину а самостално (у односу на остале чворове решења) покривају мали број чворова. На непожељним деловима решења би се јако смањила количина феромона и на тај начин би се колонија усмерила да тражи нова решења у областима скупа која имају пожељније особине.

У експериментима је показано да овакав приступ проналази решења значајно бољег квалитета. Као другу добру особину, могуће и значајнију, истичемо број итерација потребних за конвергенцију до добрих решења који је значајно смањен. Важно је напоменути да стратегија корекције феромонске матрице која се заснива на искључивању непожељних елемената, није значајно повећавала време извршавања алгоритма ОМК (мање од 1%) у тестираним проблемима. У поређењу са постојећим хибридизацијама ОМК има предност у једноставности имплементације. Класичне хибридизације у већини случајева састоје се у комбиновању ОМК са неком врстом локалне претраге или генетским алгоритмима. Оне су изузетно тешке за практичну имплементацију јер је неопходно развити два засебна алгоритма, један за ОМК и други за метод са којом се комбинује. За разлику од њих, хибридизација употребом метода корекције феромонског трага заснованом на искључивању непожељних елемената је једноставна. Код ње је једино потребно на постојећи алгоритам ОМК додати критеријум за стагнацију претраге и хеуристику непожељности. Значи, оваква хибридизација се може на једноставан начин додати за бројне проблеме који су већ решени употребом ОМК, шта јој даје посебан значај.

Ефикасност методе корекције феромонског трага је посебно евидентна на ПМПДС. Овај проблем је предходно био решаван комплексном верзијом ОМК која се састојала из две фазе. У првој фази је био креиран доминирајући скуп а у другој би се он и учинио и повезаним. Природнији

приступ ОМК који се састоји из једне фазе предходно није коришћен из разлога што је овакав алгоритам био кратковид и лако би био заробљен у неком од локално оптималних решења. Решење за ПМПДС употребом ОМК, које је саставни део тезе, је изведено употребом једно-фазног приступа. Проблем кратковидости, лаког заробљавања у неком од локалих оптимума, се превазилази употребом методе корекције феромонског трага и пажљивим избором почетних корака колоније. Сама имплементација корекције феромонског трага искључивањем непожељних елемената решења је у овом случају значајно другачија због потребе чувања повезаности скупа. У изведеним експериментима оваква имплементација ОМК се показала јако ефикасном и постизала резултате упоредиве са водећим методама које решевају овај проблем.

У тези је вршено и истраживање паралелизације ОМК. Паралелизација ОМК је анализирана на ПМТЧП. Она је изведена креирањем већег броја колонија а свакој од њих је одговарао засебан процес. Паралелизација ОМК је од јако великог значаја за ову метахеуристику јер омогућују чак супер линеарност унапређења перформанси. У анализи је посвећена пажња комуникацији између одвојених процеса у смислу информација које ће колоније размењивати и топологије, односно повезаности међу њима. Упоређене су стандарде топологије односно потпуно повезана, замена најгорег резултата и топологија прстена. Уведена је нова топологија која је варијације топологије прстена у којој уместо преписивања решења код суседних колонија врши се њихова замена. Ова ново уведена топологије је у извршеним тестовима постигла најбоље резултате.

За успешну анализу и развој алгоритама ОМК било је неопходно креирати одговарајућу програмску апликацију. Креирани софтвер је писан у C# у облику оквира са проширењима. Разлог за овакав избор програмског система је чињеница да је велик број проблема решаван употребом ОМК, сваки од њих је имао део који је строго везан за решавани проблем и део који је саставни део сваког алгоритма који се заснива на ОМК. Идеја је била издвојити опште потребе попут графичког корисничког интерфејса и чување резутата симулације колоније. Са друге стране и велики део алгоритма ОМК је са мањим варијацијама исти код свих алгоритама који се решавају уз помоћ ОМК. Ови заједнички делови алгоритама који се заснивају на ОМК су издвојени у низ апстрактних класа, чија употреба касније омогућује да се на лак и једноставан начин стварају проширења које решавају нове проблеме без потребе за писањем редундантног програмског кода. Управо као проширења за основни софтвер су написани модули који решавају све проблеме који се појављују у представљеној тези.

Представљени резултати су били презентовани на неколико конфе-

ренција и објављени у одговарајућим зборницима радова [84], [85], [86], [87]. Одређени део ових резултата је такође објављен у најугледнијим међународним часописима [88], [89], [90], [91], [92], [93]. Квалитет и важност извршеног истраживања је потврђена већим бројем цитата у другим научним радовима.

Библиографија

[1] A. Y. G. Gutin, A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp, Discrete Applied Mathematics 220, 4598 (2002) 81–86.

[2] G. A. CROES, A method for solving traveling salesman problems, Operations Research 6 (1958) 791–812.

[3] F. Glover, Tabu search — part i, ORSA Journal on Computing 1 (3) (1989) 190–206.

[4] F. Glover, Tabu search — part ii, ORSA Journal on Computing 2 (3) (1990) 4–32.

[5] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science, Number 4598, 13 May 1983 220, 4598 (1983) 671–680.

[6] V. Cerny, Optimization by simulated annealing, A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm 45 (1985) 41–51.

[7] J. R. Slagle, A. Bose, P. Busalacchi, C. Wee, Enhanced simulated annealing for automatic reconfiguration of multiprocessors in space, in: Proceedings of the 2nd international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 1, IEA/AIE '89, ACM, New York, NY, USA, 1989, pp. 401–408. doi:http://doi.acm.org/ 10.1145/ 66617.66666.
URL `http://doi.acm.org/ 10.1145/ 66617.66666`

[8] P. Siarry, G. Berthiau, F. Durdin, J. Haussy, Enhanced simulated annealing for globally minimizing functions of many-continuous variables, ACM Trans. Math. Softw. 23 (1997) 209–228. doi:http://doi.acm.org/ 10.1145/ 264029.264043.
URL `http://doi.acm.org/ 10.1145/ 264029.264043`

[9] Y. Liang, L. Xu, Mobile robot global path planning using hybrid modified simulated annealing optimization algorithm, in: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC '09, ACM, New York, NY, USA, 2009, pp. 309–314. doi:http://doi.acm.org/ 10.1145/ 1543834.1543876.
URL http://doi.acm.org/ 10.1145/ 1543834.1543876

[10] D. R. Greening, F. Darema, Rectangular spatial decomposition methods for parallel simulated annealing, in: Proceedings of the 3rd international conference on Supercomputing, ICS '89, ACM, New York, NY, USA, 1989, pp. 295–302. doi:http://doi.acm.org/ 10.1145/ 318789.318821.
URL http://doi.acm.org/ 10.1145/ 318789.318821

[11] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[12] M. Nojoumian, D. K. Nair, Comparing genetic algorithm and guided local search methods by symmetric tsp instances, in: Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO '08, 2008, pp. 1131–1132.

[13] M. Pelikan, R. Kalapala, A. K. Hartmann, Hybrid evolutionary algorithms on minimum vertex cover for random graphs, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, GECCO '07, 2007, pp. 547–554.

[14] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol. 4, 1995, pp. 1942 –1948 vol.4. doi:10.1109/ ICNN.1995.488968.

[15] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Evolutionary Computation, IEEE International Conference on, 1998, p. 69–73.

[16] D. T. Pham, E. Koc, S. Otri, S. Rahim, M.Zaidi, The bees algorithm - a novel tool for complex optimisation problems, in: Proceedings of IPROMS 2006 Conference, 2006, pp. 454–661.

[17] S. Häckel, P. Dippold, The bee colony-inspired algorithm (bcia): a two-stage approach for solving the vehicle routing problem with time windows, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09, ACM, New York, NY, USA, 2009, pp. 25–32. doi:http://doi.acm.org/ 10.1145/ 1569901.1569906.
URL http://doi.acm.org/ 10.1145/ 1569901.1569906

[18] L.-P. Wong, C. Y. Puan, M. Y. H. Low, C. S. Chong, Bee colony optimization algorithm with big valley landscape exploitation for job shop scheduling problems, in: Proceedings of the 40th Conference on Winter Simulation, WSC '08, 2008, pp. 2050–2058.

[19] C. S. Chong, A. I. Sivakumar, M. Y. H. Low, K. L. Gay, A bee colony optimization algorithm to job shop scheduling, in: Proceedings of the 38th conference on Winter simulation, WSC '06, 2006, pp. 1954–1961.

[20] M. Dorigo, L. M. Gambardella, Ant colonies for the travelling salesman problem, Biosystems 43 (2) (1997) 73–81.

[21] D. Maniezzo, M. Dorigo, V. Maniezzo, A. Colorni, Ant system: An autocatalytic optimizing process (1991).

[22] B. Bullnheimer, R. F. Hartl, C. StrauЯ, A new rank based version of the ant system - a computational study, Central European Journal for Operations Research and Economics 7 (1997) 25–38.

[23] M. Dorigo, L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation.

[24] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26 (1) (1996) 29 –41. doi:10.1109/3477.484436.

[25] T. Stützle, H. H. Hoos, Max-min ant system, Future Gener. Comput. Syst. 16 (9) (2000) 889–914.

[26] D. Asmar, A. Elshamli, S. Areibi, A comparative assessment of aco algorithms within a tsp environment, Dynamics of Continous Discrete and Impulsive Systems-Series B-Applications and Algorithms 1 (2005) 462–467.

[27] G. A. CROES, A method for solving traveling salesman problems, Operations Res. 6 (1958) 791 –812.

[28] S. LIN, B. W. KERNIGHAN, An effective heuristic algorithm for the traveling-salesman problem, Operations Res. 21 (1973) 498–516.

[29] Q. Chengming, An ant colony algorithm with stochastic local search for the vrp, in: 3rd International Conference on Innovative Computing

Information and Control, IEEE Computer Society, Los Alamitos, CA, USA, 2008, pp. 464–468.

[30] J. Levine, F. Ducatelle, Ant colony optimization and local search for bin packing and cutting stock problems, Journal of the Operational Research Society 55 (2004) 705–716.

[31] F. Neumann, D. Sudholt, C. Witt, Rigorous analyses for the combination of ant colony optimization and local search, in: ANTS '08: Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 132–143.

[32] H. hao Zuo, F. lun Xiong, Time ant colony algorithm with genetic algorithms, in: International Conference on Information Acquisition, IEEE Computer Society, Los Alamitos, CA, USA, 2006, pp. 1057–1061.

[33] Z.-J. Lee, S.-F. Su, C.-C. Chuang, K.-H. Liu, Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment, Applied Soft Computing 8 (1) (2008) 55–78.

[34] P. Shelokar, P. Siarry, V. Jayaraman, B. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Applied Mathematics and Computation 188 (1) (2007) 129 – 142. doi:DOI: 10.1016/j.amc.2006.09.098.
URL `http:// www.sciencedirect.com/ science/ article/ pii/ S0096300306013294`

[35] G. Reinelt, Tsplib - a traveling salesman problem library, ORSA Journal on Computing 3 (4) (1991) 376–384.

[36] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, Q. X. Wang, Particle swarm optimization-based algorithms for tsp and generalized tsp, Information Processing Letters 103 (5) (2007) 169–176.

[37] R. Karp, In R.E. Miller and J.W. Theater, Complexity of Computer Computations, Plenum Press, New York, 1972.

[38] On efficient fixed-parameter algorithms for weighted vertex cover, Journal of Algorithms 47 (2) (2003) 63 – 77. doi:DOI: 10.1016/ S0196-6774(03) 00005-1.

[39] J. Chen, I. Kanj, G. Xia, Improved parameterized upper bounds for vertex cover, in: R. Krб lovic, P. Urzyczyn (Eds.), Mathematical Foundations of Computer Science 2006, Vol. 4162 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006, pp. 238–249.

[40] M. Cygan, L. Kowalik, M. Wykurz, Exponential-time approximation of weighted set cover, Information Processing Letters 109 (16) (2009) 957 – 961.

[41] V. Chvatal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235.

[42] K. Clarkson, A modification of the greedy algorithm for vertex cover, Information Processing Letters 16 (1) (1983) 23–25.

[43] A. Singh, A. K. Gupta, A hybrid heuristic for the minimum weight vertex cover problem, Asia-Pacific Journal of Operational Research (APJOR) 23 (2) (2006) 273–285.

[44] S. R. Balachandar, K. Kannan, A meta-heuristic algorithm for vertex covering problem based on gravity, International Journal of Mathematical and Statistical Sciences 1 (3) (2009) 130–136.

[45] J. Shyu, P. Yin, B. Lin, An ant colony optimization algorithm for the minimum weight vertex cover problem, Annals of Operations Research 131 (2) (2004) 283–304.

[46] T. Stützle, H. Hoos, Improvements on the ant system: Introducing the max min ant system, in: Third International Conference on Artificial Neural Networks and Genetic Algorithms, Springer Verlag, University of East Anglia, Norwich, UK, 1998, pp. 245—249.

[47] K. Y. Wong, P. C. See, A new minimum pheromone threshold strategy (mpts) for maxвЂ“min ant system, Applied Soft Computing 9 (3) (2009) 882–888.

[48] J. Shyu, P. Yin, B. Lin, An ant colony optimization algorithm for the minimum weight vertex cover problem, Annals of Operations Research 131 (2) (2004) 283–304.

[49] S. Gilmour, M. Dras, Understanding the pheromone system within ant colony optimization, in: AI 2005: Advances in Artificial Intelligence, Springer Berlin/Heidelberg, 2005, pp. 786–789.

[50] S. Gilmour, M. Dras, Kernelization as heuristic structure for the vertex cover problem, in: Ant Colony Optimization and Swarm Intelligence, Springer Berlin/Heidelberg, 2006, pp. 452–459.

[51] M. R. Garey, D. S. Johnson, Computers and Intractability. A Guide to the Theory of NPCompleteness., W. H. Freeman and Company, New York-San Francisco, 1979.

[52] J. Blum, M. Ding, A. Thaeler, X. Cheng, Connected Dominating Set in Sensor Networks and MANETs, Academic Publishers, Kluwer, 2004.

[53] A. K. Parekh, Analysis of a greedy heuristic for finding small dominating sets in graphs, Inf. Process. Lett. 39 (5) (1991) 237–240, 121382.

[54] C. Ambuhl, T. Erlebach, M. Mihalбk, M. Nunkesser, Constant-factor, Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs, 2006.

[55] X. Xu, Z. Tang, W. Sun, X. Chen, Y. Li, G. Xia, W. Bi, Z. Zon, An algorithm for the minimum dominating set problem based on a new energy function, in: SICE Annual Conference, Sapporo, Japan, 2004, pp. 924–926.

[56] R. Misra, C. Mandal, Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks, IEEE Transactions on Parallel and Distributed Systems 21 (2010) 292–302.

[57] S. Gutner, Polynomial kernels and faster algorithms for the dominating set problem on graphs with an excluded minor, in: Parameterized and Exact Computation: 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers, Springer-Verlag, 2009, pp. 246–257, 1696691.

[58] C. K. Ho, Y. P. Singh, H. T. Ewe, An enhanced ant colony optimization metaheuristic for the minimum dominating set problem, Applied Artificial Intelligence 20 (2006) 881–903.

[59] A. Vlachos, An ant colony optimization (aco) algorithm solution to economic load dispatch (eld) problem, WSEAS Transactions On Systems 5 (8) (2006) 1763 – 1771.

[60] F. Kolahan, M. Abachizadeh, S. Soheili, A comparison between ant colony and tabu search algorithms for job shop scheduling with sequence-dependent setups, WSEAS Transactions on Systems 12 (2006) 2819–2824.

[61] N. Mastorakis, X. Zhuang, Image processing with the artificial swarm intelligence, WSEAS Transactions on Computers 4 (4) (2005) 333–341.

[62] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, K.-I. Ko, A greedy approximation for minimum connected dominating sets, Theor. Comput. Sci. 329 (1-3) (2004) 325–330.

[63] X. Cheng, M. Ding, D. Chen, An approximation algorithm for connected dominating set in ad hoc networks, in: Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peerto-Peer Networks (TAWN, 2004.

[64] M. Min, H. Du, X. Jia, C. X. Huang, S. C.-H. Huang, W. Wu, Improving construction for connected dominating set with steiner tree in wireless sensor networks, Journal of Global Optimization 35 (1) (2006) 111–119.

[65] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.

[66] S. Butenko, X. Cheng, C. Oliveira, P. Pardalos, A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks, in: Recent Developments in Cooperative Control and Optimization, Kluwer Academic Publishers, 2004, pp. 61–73.

[67] S. Butenko, C. Oliveira, P. Pardalos, A new algorithm for the minimum connected dominating set problem on ad hoc wireless networks, in: CCCT'03, International Institute of Informatics and Systematics (IIIS), 2003, pp. 39–44.

[68] R. Misra, C. Mandal, Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks, IEEE Transactions on Parallel and Distributed Systems 21 (3) (2010) 292–302.

[69] M. Morgan, V. Grout, Metaheuristics for wireless network optimisation, in: Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on, 2007, pp. 15 –15. doi:10.1109/ AICT.2007.28.

[70] H. He, Z. Zhu, E. Makinen, A neural network model to minimize the connected dominating set for self-configuration of wireless sensor networks, IEEE Transactions on Neural Networks 20 (6) (2009) 973–982.

[71] R. G. Downey, M. R. Fellows, C. McCartin, F. A. Rosamond, Parameterized approximation of dominating set problems., Information Processing Letters (2008) 68–70.

[72] C. Zhang, Q. Xu, Clustering approach for wireless sensor networks using spatial data correlation and ant-colony optimization, in: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing - Volume 01, IEEE Computer Society, Washington, DC, USA, 2009, pp. 538–541. doi:10.1109/NSWCTC.2009.350.

[73] C. K. Ho, Y. P. Singh, H. T. Ewe, An enhanced ant colony optimization metaheuristic for the minimum dominating set problem, Applied Artificial Intelligence 20 (10) (2006) 881–903.

[74] Asp competition:submitted benchmarks (2009).
URL `http://dtai.cs.kuleuven.be /events /ASP-competition /encodings.shtml`

[75] R. Tanese, Parallel genetic algorithms for a hypercube, in: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, 1987, pp. 177–183.

[76] T. Stutzle, Parallelization strategies for ant colony optimization, in: Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, Springer-Verlag, 1998, pp. 722–731.

[77] M. Middendorf, F. Reischle, H. Schmeck, Multi colony ant algorithms, Journal of Heuristics 8 (2002) 305–320.

[78] B. Bullnheimer, G. Kotsis, C. Strauss, Parallelization strategies for the ant system, in: High Performance Algorithms and Software in Nonlinear Optimization, Applied Optimization, Dordrecht: Kluwer, 1998, pp. 87–100.

[79] S. Benkner, R. Doerner, K.F.and Hartl, G. Kiechle, M. Lucka, Communication strategies for parallel cooperative ant colony optimization on clusters and grids, in: Complimentary Proceedings of PARA'04 Workshop on State-of-the-Art in Scientific Computing, 2004, pp. 3–12.

[80] U. L. D. Bruxelles, A. F. D. Roosevelt, M. Manfrin, M. Birattari, T. Stьtzle, M. Dorigo, Parallel ant colony optimization for the traveling salesman problem, in: Proc. of the 5th Int. Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS), Springer Verlag, 2006, pp. 224–234.

[81] P. Balaprakash, M. A. M. de Oca, Aco:public software (2006).
URL http:// iridia.ulb.ac.be/ mdorigo/ ACO/ aco-code/ public-software.html

[82] U. Chirico, A java framework for ant colony systems (2009).
URL http://www.ugosweb.com/ Documents/ jacs.aspx

[83] I. Ellabib, P. Calamai, O. Basir, Exchange strategies for multiple ant colony system, Inf. Sci. 177 (5) (2007) 1248–1264. doi:http:// dx.doi.org/ 10.1016/ j.ins.2006.09.016.

[84] R. Jovanovic, M. Tuba, A comparative assessment of ant colony optimization algorithms for the minimum weight vertex cover problem, in: AIKED'09: Proceedings of the 8th international conference on Artificial intelligence, knowledge engineering and data bases, Cambridge, United Kingdom, February 2009, pp. 490–494.

[85] R. Jovanovic, M. Tuba, D. Simian, Ant colony optimization applied to minimum weight dominating set problem, in: Proceedings of the 12th international conference on Automatic control, modelling and simulation, Catania, Italy, May 2010, pp. 322–326.

[86] R. Jovanovic, M. Tuba, D. Simian, Analysis of parallel implementations of the ant colony optimization applied to the minimum weight vertex cover problem, in: Proceedings of the 9th international conference on Simulation, modelling and optimization, Budapest, Hungary, September 2009, pp. 254–259.

[87] R. Jovanovic, M. Tuba, D. Simian, Developing an object-oriented framework for solving problems using ant colony optimization, in: Recent Advances In Mathematical And Computational Methods In Science And Engineering, Bucharest, Romania, November 2008, pp. 528–533.

[88] R. Jovanovic, M. Tuba, An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem, Applied Soft Computing 11 (8) (2011) 5360–5366.

[89] R. Jovanovic, M. Tuba, Ant colony optimization algorithm with pheromone correction strategy for minimum connected dominating set problem, Computer Science and Information Systems (ComSIS), accepted for publishing.

[90] R. Jovanovic, M. Tuba, Improved ant colony optimization algorithm with pheromone correction strategy for the traveling salesman problem,

Journal of Systems Engineering and Electronics, submitted for publishing.

[91] R. Jovanovic, M. Tuba, D. Simian, Comparison of different topologies for island-based multi-colony ant algorithms for the minimum weight vertex cover problem, WSEAS Transactions on Computers 9 (2010) 83–92.

[92] M. Tuba, R. Jovanovic, An analysis of different variations of ant colony optimization to the minimum weight vertex cover problem, WSEAS Transactions on Information Sciences and Application 6 (2009) 936–945.

[93] R. Jovanovic, M. Tuba, D. Simian, An object-oriented framework with corresponding graphical user interface for developing ant colony optimization based algorithms, WSEAS Transactions on Computers 7 (12) (2008) 1948–1957.