УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ПРИРОДНО–МАТЕМАТИЧКИ ФАКУЛТЕТ

Зоран Максимовић

Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова

докторска дисертација

Крагујевац, 2016.

I. Аутор
Име и презиме: Зоран Максимовић
Датум и место рођења: 19. јул 1967, Ужице
Садашње запослење: асистент на Војној академији Универзитета одбране
II. Докторска дисертација
Наслов: Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова
Број страница: VIII+109
Број слика: 4
Број библиографских података: 63
Установа и место где је рад израђен: Природно-математички факултет, Крагујевац
Научна област (УДК): Математика (51)
Ментор: др Јозеф Кратица, научни саветник, Математички институт САНУ
III. Оцена и одбрана
Датум пријаве теме: 20.04.2016.
Број одлуке и датум прихватања докторске дисертације:
Комисија за оцену подобности теме и кандидата: 1. др Јозеф Кратица, научни саветник, Математички институт САНУ 2. др Љиљана Павловић, редовни професор, Природно-математички факултет Универзитета у Крагујевцу 3. др Александар Савић, доцент, Математички факултет Универзитета у Београду 4. др Драган Матић, доцент, Природно-математички факултет Универзитета у Бања Луци
Комисија за оцену докторске дисертације: 1. др Љиљана Павловић, редовни професор, Природно-математички факултет Универзитета у Крагујевцу 2. др Бобан Стојановић, ванредни професор, Природно-математички факултет Универзитета у Крагујевцу 3. др Александар Савић, доцент, Математички факултет Универзитета у Београду 4. др Драган Матић, доцент, Природно-математички факултет Универзитета у Бања Луци
Комисија за одбрану докторске дисертације: 1. др Љиљана Павловић, редовни професор, Природно-математички факултет Универзитета у Крагујевцу 2. др Бобан Стојановић, ванредни професор, Природно-математички факултет Универзитета у Крагујевцу 3. др Александар Савић, доцент, Математички факултет Универзитета у Београду 4. др Драган Матић, доцент, Природно-математички факултет Универзитета у Бања Луци
Датум одбране дисертације:

Садржај

Резиме

У овом раду су разматрана два уопштења NP-тешког проблема максималне бисекције, где се уместо тежина грана као реалних бројева уводи r-торка позитивних реалних бројева. Прво уопштење које се разматра је вишедимензионални проблем максималне бисекције где су тежине на гранама r-торке ненегативних реалних бројева. Друго разматрано уопштење је вишедимензионални проблем максималне бисекције на повезане подграфове где, поред захтева првог уопштења, постоји захтев да подграфови индуковани партицијама скупа чворова буду повезани. Поред наведених уопштења, у раду је разматран и проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена.

Вишедимензионални проблем максималне бисекције се може применити на управљање људским ресурсима. Један од најважнијих аспеката је компатибилност, односно некомпатибилност запослених која, по својој природи, може бити вишедимензионална. Тако је сваки критеријум који утиче на компатибилност представљен једном координатом вектора. Повезаност подграфова (тимова) игра важну улогу, зато што се повезаност запослених у оквиру тима формира на основу склоности ка заједничком раду или претходној сарадњи. Друга могућа примена је у организацији електронских кола. Постоје одређени аспекти који се могу сматрати важним као што је интерференција, коришћена струја, емисија топлоте итд.

За оба предложена уопштења проблема максималне бисекције дате су и формулације мешовитог целобројног линеарног програмирања, као и докази њихове коректности. За проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена дат је нови модел са полиномијалним бројем ограничења. Коришћењем стандардних решавача CPLEX и Gurobi, добијена су оптимална решења за све инстанце малих димензија и неке инстанце средњих

димензија.

За добијање приближних решења на инстанцама великих димензија су развијене следеће метахеуристике за решавање уопштења проблема максималне бисекције: генетски алгоритам, метода променљивих околина и метахеуристика заснована на електромагнетизму. Све методе су тестиране на случајно генерисаним инстанцама до 1000 чворова и 350000 грана. Тестирања су показала да су развијене метахеуристике знатно унапредиле приближне резултате добијене од стране стандардних решавача коришћењем предложених модела. Добијени експериментални резултати показују да је најбоље резултате показао генетски алгоритам.

Кључне речи

комбинаторна оптимизација, целобројно линеарно програмирање, метахеуристике, метода променљивих околина, генетски алгоритми, метахеуристике засноване на електромагнетизму, вишедимензионални проблем максималне бисекције, вишедимензионални проблем максималне бисекције на повезане подграфове, проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена.

Abstract

In this dissertation two generalizations of NP-hard maximum bisection problem, where wights on edges are r-tuples of positive real numbers instead of real numbers, are considered. The first generalization is the multidimensional maximum bisection problem, where weight on edges are r-tuples of non-negative real numbers. The second generalization is the connected multidimensional maximum bisection problem, with additional condition that subgraphs induced by partitions of vertex set are connected. Beside aforementioned generalizations, in this dissertation is considered the maximum degree bounded connected subgraph problem.

Multidimensional maximum bisection problem can be applied in human resource management. One of the most important aspects is compatibility/incompatibility between employees that is, by its nature, multidimensional. Each criteria of compatibility is represented by one coordinate of a vector. The connectedness of subgraphs (teams) plays important role, because teams should be formed by employees that had been working together before as much as possible. Another application is in electronic circuits design. There are certain aspects that can be considered important such as: interference, current, heat dissipation, etc.

For both proposed generalizations of the maximum bisection problem mixed integer linear programming formulations are given with proofs of its correctness. For the maximum degree bounded connected subgraph problem a new mixed integer linear programming formulation with polynomial number of constraints is given. Using standard solvers CPLEX and Gurobi, optimal solutions are obtained for all small-size instances and some medium size instances.

For obtaining solutions on large size instances for solving generalizations of maximum bisection problem, the following metaheuristics are developed: genetic algorithm, variable neighborhood search and metaheuristic based on electromagnetism. All methods were tested on randomly generated instances with up to 1000 vertices and 350000 edges. Tests showed that developed metaheuristics considerably improved solutions obtained using standard solvers and proposed MILP formulations.

Experimental results suggest that genetic algorithm showed the best results.

Keywords

combinatorial optimization, mixed integer linear programming, metaheuristics, variable neighborhood search, genetic algorithms, electromagnetism, graph bisection multidimensional maximum bisection problem, connected multidimensional maximum bisection problem, maximum degree bounded connected subgraph problem.

Предговор

У свим сферама друштва се јавља потреба за неким врстама оптимизације, у процесима производње у индустрији, у управљању људским ресурсима итд. Често се пред одговорна лица поставља задатак да организују било физички процес производње, било организацију самих тимова који су укључени у реализацију неког пројекта. Ови процеси су компликовани зато што укључују велики број међусобно зависних и независних компоненти и проналажење оптималног решења често није могуће. Стога се у решавању таквих проблема често користе методе које проналазе приближна решења, по могућности она која су довољно близу оптималном.

У овом раду се истражују нека уопштења бисекције графова и повезаности подграфова. Уопштења су заснована на приступу да се уместо тежина грана као реалних бројева узимају r-торке ненегативних реалних бројева. Тако су у дисертацији разматране различите методе за решавање следећих проблема: вишедимензионални проблем максималне бисекције, вишедимензионални проблем максималне бисекције на повезане подграфове и проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена.

У првој глави су дати основни појмови о којима ће бити реч у овој дисертацији. У другој глави се разматра вишедимензионални проблем максималне бисекције. Дат је модел мешовитог целобројног линеарног програмирања за његово решавање као и доказ његове коректности. Будући да је проблем NP-тежак, развијене су и три метахеуристике за његово решавање: генетски алгоритам, метода променљивих околина и метахеуристика заснована на електромагнетизму. На крају су разматрани експериментални подаци добијени применом свих разматраних техника. У трећој глави се разматра вишедимензионални проблем максималне бисекције на повезане подграфове, који је сличан вишедимензионалном проблему максималне бисекције само што се захтева да

подграфови индуковани партицијама буду повезани. Као и у претходној глави, дат је модел мешовитог целобројног линеарног програмирања за решавање вишедимензионалног проблема максималне бисекције на повезане подграфове са доказом његове коректности, а након тога су приказане развијене метахеуристике за његово решавање: генетски алгоритам, метода променљивих околина и метахеуристика заснована на електромагнетизму. На крају су, такође, разматрани експериментални подаци добијени применом развијених метода. У четвртој глави је разматран проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена. У литератури је већ био познат генетски алгоритам за решавање овог проблема, па због тога проблем није решаван метахеуристички. Уместо тога је за дати проблем предложена потпуно нова MILP формулација за његово решавање са полиномијалним бројем услова. У раду је дат и математички доказ коректности дате формулације.

Постигнути резултати имају теоријски и практични значај. За проблеме где већ нису постојали MILP модели, они су развијени, а у осталим случајевима су побољшани. Поред тога, развијене су и метахеуристичке методе које су се показале као ефикасне и поуздане што је показано њиховим тестирањем на одговарајућим инстанцама.

Желим да се захвалим ментору, др Јозефу Кратици, који је руководио израдом моје докторске дисертације, као и члановима Комисије: проф. др Љиљани Павловић, проф. др Бобану Стојановићу, доц. др Александру Савићу и доц. др Драгану Матићу на пажљивом читању и корисним сугестијама које су значајно побољшале квалитет овог рада. Дугујем захвалност и др Ђорђу Дугошији, др Милану Дражићу, др Зорици Дражић, др Милени Богдановић, Ани Симић и Јелисавки Милошевић на подршци током рада на овој дисертацији.

Својој породици, супрузи Драгици и деци Вукашину и Ани, дугујем велику захвалност на показаном стрпљењу због одсутности током рада на овој дисертацији.

У Београду/Крагујевцу, јуна 2016. године

Кандидат

мр Зоран Максимовић

Глава 1

Увод

Потреба за ефикасношћу се јавља у свим аспектима друштва: у здравству, школству, привреди, државној управи, итд. Да би се неки ентитет у друштву довео у стање веће ефикасности потребне су две ствари: критеријум ефикасности, и поступак којим се реорганизује ентитет задржавајући функционалност уз захтев да се ефикасност повећа у складу са критеријумом ефикасности. Могу се успоставити различити критеријуми ефикасности, као што су: број потрошених сати, количина утрошеног материјала, степен хабања машине и слично. Поступак реорганизације ентитета је проблем који зависи од самог ентитета.

Проблемима организације и реорганизације се баве људи разних струка: економисти, менаџери, инжењери и, наравно, математичари. Две компоненте највише утичу на решавање проблема ове природе: сама природа проблема и величина проблема. Идентификацијом природе проблема и одређивањем модела који га описује, често долазимо и до напредних метода њиховог решавања. У случају да методе нису познате, могу се пронаћи нове или се могу унапредити постојеће. Са друге стране, у пракси се често дешава да, иако нам је позната природа проблема као и поступак за његово решавање, сама величина проблема може да значајно отежа или чак и онемогући његово решавање. На пример, у интегрисаном колу може да буде и више хиљада елемената, док је у већим компанијама потребно организовати производњу у којој учествује на стотине радника, машина и сл.

Да би се проблем могао уопште решавати, неопходно је прво формирати математички модел. Математички модел образују елементи и релације између

њих. Свим елементима проблема мора одговарати нека математичка структура у математичком моделу. На сличан начин, свим односима елемената проблема мора одговарати нека релација између математичких структура.

Приликом истраживања се, по правилу, тежи да се користе добро познате математичке структуре зато што то омогућава коришћење већ постојећих теоријских резултата. Многи проблеми који се јављају су комбинаторне природе и углавном се могу моделирати структурама из области комбинаторике или теорије графова.

1.1 Комбинаторна оптимизација

Да би се решавали оптимизациони проблеми, потребно је на почетку одредити који параметри учествују у дефиницији проблема. Након тога је потребно одредити скуп променљивих којима се описују елементи проблема, као и природу тих променљивих (да ли променљиве узимају вредности из скупа природних бројева, целих, реалних или можда из неког њиховог подскупа). По природи решења, оптимизационе проблеме делимо у две класе: оне код којих су решења из непребројивих скупова и она код којих су решења из пребројивих или коначних скупова. Последње оптимизационе проблеме често називамо дискретним или проблемима комбинаторне оптимизације.

Проблем комбинаторне оптимизације се може дефинисати на разне начине. Наведимо дефиницију из [7].

Дефиниција 1. *Нека је* $X = \{x_1, x_2, ..., x_n\}$ *скуп променљивих и нека су познати:*

- *домени променљивих* $D_1, D_2, ..., D_n$*, који су пребројиви или коначни;*
- *ограничења на променљивама;*
- *функција циља* f *која се минимизује:* $f : D_1 \times D_2 \times ... \times D_n \to \mathbb{R}^+$.

Допустива тачка је тачка $s = (v_1, v_2, ..., v_n)$ *таква да је* $v_i \in D_i$ *и* s *задовољава сва ограничења. Скуп свих допустивих тачака (или простор решења) се обележава са* S*. Сваки елемент скупа* S *представља потенцијално решење проблема. Циљ оптимизационог процеса* $P = (S, f)$ *је пронаћи оно најбоље решење, тј. проналазак оне допустиве тачке* $s^* \in S$ *за коју функција* f *достиже*

минимум, тј. важи $(\forall s \in S)\ f(s^*) \leq f(s)$. *Добијена тачка минимума* s^* *се назива глобалним оптималним решењем проблема* (S, f).

Поступак за који можемо доказати да ће на крају његовог извршавања добијено решење бити глобално оптимално решење проблема називамо егзактном методом. Тривијални приступ би могао бити да се на неки начин израчуна функција $f(s)$ за све допустиве тачке s и међу њима пронађе најбоље решење. Такав приступ се назива и тотална енумерација односно претраживање исцрпљивањем свих могућности (енг. exhaustive search). Његовом применом се теоријски сигурно може добити оптимално решење, али у случају NP-тешких проблема време извршавања експоненцијално расте са димензијом проблема.

Често су формулације проблема комбинаторне оптимизације једноставне, а поступци њиховог решавања веома сложени. Поменимо само неке од таквих познатих проблема: бојење графова, проблем трговачког путника, проблем суме подскупа, проблем Хамилтоновог пута, итд. Иако су формулације ових проблема једноставне, није познат алгоритам полиномијалне сложености за њихово решавање. Стога, егзактно решавање оваквих проблема за инстанце великих димензија би чак и на најбржим данашњим рачунарима трајало недопустиво дуго.

То је разлог што се, због комплексности поступка решавања, поред тражења оптималног решења, покушава и пронаћи приближно решење које можда није глобални оптимум, али има задовољавајући квалитет. Да би се то постигло развијају се методе за приближно решавање ових проблема.

Један од популарних приступа проналажењу приближних решења проблема је коришћењем апроксимативних алгоритама. За сваки од тих алгоритама је задат фактор апроксимације, односно мера његовог максималног одступања од оптималног решења. Конкретно за проблем максимизације, фактор апроксимације је реална константа c, $c < 1$ таква да, за решење x и његову функцију циља $f(x)$, важи $c \cdot opt \leq f(x) \leq opt$ где је opt оптимално решење проблема. Слично томе, за проблеме минимизације, фактор апроксимације је реална константа c, $c > 1$ таква да, за решење x и његову функцију циља $f(x)$, $opt \leq f(x) \leq c \cdot opt$ где је opt оптимално решење проблема.

Поред апроксимативних алгоритама, за решавање се користе и такозване хеуристичке методе. За резултат који је добијен применом неке хеуристичке

методе се не може гарантовати да ли је оптималан, а није позната ни мера његовог одступања у односу на оптимално решење. Неке хеуристичке методе се могу применити не само на конкретан проблем него и на ширу класу проблема уз релативно мале измене. Такве хеуристике називамо матахеуристичким методама или краће метахеуристикама.

1.2 Метахеуристике

Осман и Лапорте [49] формално дефинишу метахеуристику као итеративни процес који управља подређеним хеуристикама интелигентним комбиновањем различитих концепата истраживања и искоришћавања простора претраге (простора решења) коришћењем стратегије учења за структурисање информација ради ефикасног проналажења решења блиског оптималном.

Треба приметити да добијање оптималног решења није гарантовано, иако је за одређене (ретке) случајеве показана конвергенција оптималном решењу. Међутим, чак и у таквим случајевима, неопходни услови конвергенције ка оптималном решењу обично захтевају или бесконачан број итерација или бесконачне меморијске захтеве. Упркос томе, метахеуристике су се показале као веома ефикасне у проналажењу решења блиских оптималним за неке практичне проблеме.

1.2.1 Класификација метахеуристика

Прва класификација метахеуристика би могла бити по полазној идеји за њихов развој па тако имамо оне које засноване на опонашању природних процеса и оне које су настале људским размишљањем а немају директне везе са природним појавама. У метахеуристике засноване на аналогији са природом спадају генетски алгоритми, генетско програмирање, еволуционе стратегије, алгоритми засновани на мрављим колонијама, алгоритми засновани на пчелињим колонијама, симулирано каљење, метахеуристика заснована на електромагнетизму, итд. Са друге стране, у метахеуристике које нису инспирисане природним појавама спадају: итеративна локална претрага, табу претрага, метода променљивих околина (VNS), итд.

Према броју активних тренутних решења (допустивих тачака) којима се

управља постоје две групе метахеуристика: метахеуристике које раде са једним активним тренутним решењем (енг. single point search) и метахеуристике које раде са више активних тренутних решења за које још кажемо да су засноване на популацији (енг. population based). Алгоритме који раде само са једним тренутним решењем називамо методама трајекторије зато што у сваком тренутку метахеуристика на неки начин одређује у коју следећу тачку ће да пређе у потрази за оптималним решењем, на тај начин одређујући путању по простору претраге. У метахеуристике засноване на једном активном решењу спадају метода променљивих околина, итеративна локална претрага, табу претрага, симулирано каљење итд. Са друге стране метахеуристике засноване на популацији функционишу тако што се у једној итерацији алгоритма разматра више допустивих тачака на основу којих се креирају нове тачке комбиновањем постојећих у циљу задржавања квалитета старих тачака и њиховог унапређења у новим итерацијама. У ове метахеуристике спадају: генетски алгоритми, генетско програмирање, еволуционе стратегије, метахеуристика заснована на електромагнетизму, алгоритми засновани на мрављим колонијама, алгоритми засновани на пчелињим колонијама, итд.

Развијен је велики број различитих метахеуристика, а неке од најчешће коришћених су:

- итеративна локална претрага (енг. Iterated Local Search - ILS);
- симулирано каљење (енг. Simulated Annealing - SA);
- табу претраживање (енг. Tabu Search - TS);
- генетски алгоритам (енг. Genetic Algorithm - GA);
- генетско програмирање (енг. Genetic Programming - GP);
- еволуционе стратегије (енг. Evolutionary Strategies - ES)
- метода променљивих околина (енг. Variable Neighborhood Search – VNS);
- алгоритам заснован на мрављим колонијама (енг. Ant Colony Optimization - ACO);
- Лагранжева релаксација (енг. Lagrangian Relaxation - LR);

- метахеуристика заснована на електромагнетизму (енг. Electromagnetism - EM);
- похлепни алгоритам са прилагођеном случајном претрагом (енг. Greedy Randomized Adaptive Search Procedure - GRASP)
- алгоритми засновани на пчелињим колонијама (енг. Artificial Bee Colony - ABC, Bee Colony Optimization - BCO);
- меметички алгоритми (енг. Memetic Algorithms - MA)
- метода заснована наројевима (енг. Particle Swarm Optimization - PSO).

За решавање проблема у овој дисертацији ће бити коришћене три метахеуристике: генетски алгоритми (GA), метода променљивих околина (VNS) и метахеуристика заснована на електромагнетизму (EM). Ове три методе ће бити детаљније описане у наставку овог поглавља, док детаљни приказ других метода превазилази обим овог рада, а може се наћи у прегледним радовима [7, 49, 23].

1.2.2 Генетски алгоритми

Генетски алгоритми (GA) су метахеуристике засноване на популацији и које опонашају неке аспекте природне еволуције, чије је основе поставио Џон Холанд у [32]. Сваки елемент популације се назива јединком и он представља потенцијално решење. Очигледно да популација представља подскуп простора решења (тј. простора претраге). Генетски алгоритам је итеративни поступак где се у свакој итерацији неке јединке тренутне популације мењају поступцима који опонашају процесе природне еволуције, са циљем да се од скупа јединки из претходне генерације покуша направити нова генерација чије су јединке прилагођеније, тј. одговарају бољим решењима. Да би се могло проценити која јединка је прилагођенија, неопходно је свакој јединки, тј. потенцијалном решењу доделити неку нумеричку вредност коју зовемо прилагођеност (енг. fitness). Функцију која пресликава простор претраге у прилагођености називамо функцијом прилагођености (енг. fitness function). Циљ генетских алгоритама је да се сваком наредном генерацијом проналазе све прилагођеније и прилаго-

ђеније јединке (тј. све боља и боља решења). Поступци којима модификујемо јединке из популације називамо генетским операторима.

Основни објекти којима генетски алгоритам управља су популација и појединачне јединке унутар популације. Популација је сачињена од више јединки чији се број најчешће креће од неколико до неколико стотина. Свакој јединки одговара једно решење полазног проблема. Јединке су представљене генетским кодом (низом гена), који се записује у бинарној, целобројној или некој другој коначној азбуци. Ген треба да буде тако записан да омогућава лако примењивање генетских оператора.

Да би се обезбедила почетна разноврсност генетског материјала, почетна популација се најчешће креира на случајан начин. У неким случајевима се за креирање почетне популације користе и друге хеуристике, обично са циљем постизања бољих почетних решења. За време рада генетског алгоритма на популацију се примењују генетски оператори. Најчешће се користе генетски оператори селекције, укрштања и мутације, мада постоје и неки други оператори.

Аргумент оператора селекције је скуп свих јединки, тј. цела популација, а његов резултат је његов подскуп који представља оне јединке које ће преживети (кроз потомке). Избор јединки које преживљавају се заснива на прилагођености тј. вредности функције прилагођености за конкретну јединку. Тривијалан начин за избор јединки би могао бити да се јединке сортирају по прилагођености и да се изабере потребан број јединки које су најприлагођеније. Постоје разне врсте селекције, као што су: турнирска селекција, рулет селекција, елитистичка селекција итд. Поред тога, могуће је и комбиновање ових селекција. Код турнирске селекције се бира група по група из популације и из тих група се бира најбоља јединка која прелази у следећу генерацију. Рулет селекција обезбеђује да је вероватноћа преживљавања јединке пропорционална прилагођености, чиме се фаворизују натпросечно прилагођене јединке. Код елитистичке селекције један део најбољих јединки директно прелази у следећу генерацију.

Оператор укрштања као аргумент има бар две јединке, а као резултат добијамо потомке који потенцијално прелазе у следећу генерацију. Постоје разни оператори укрштања као што су оператор укрштања у једној тачки, оператор

укрштања у две тачке, вишеродитељски оператор укрштања, униформни оператор укрштања, итд. Код оператора укрштања у једној тачки се на истој позицији (тачки) прекида ген обе родитељске јединке. Након тога се надовезује први део гена прве јединке са другим делом гена друге јединке и први део гена друге јединке са другим делом гена прве јединке. Тако добијамо две јединке са комбинованим генима родитеља.

Мутација је оператор чији је аргумент само једна јединка, а резултат је такође једна јединка. Резултујућа јединка се добија од полазне тако што се ген јединке на случајно изабраној позицији мења. Вероватноћа да ће доћи до промене је обично задата унапред. Циљ овог оператора је да случајном мутацијом спречи да процес превремено конвергира у локалном оптимуму лошег квалитета.

Оператор селекције формира следећу генерацију од потомака и неких јединки из претходне генерације после примене генетских оператора. Процес се понавља израчунавањем прилагођености свих јединки и наставља применом генетских оператора, све док се не испуни критеријум заустављања. Неки често коришћени критеријуми су: заустављање након задатог броја генерација, након задатог времена, након задатог броја генерација без унапређења најбоље јединке, након достизања задате дисперзије функције циља у популацији, итд.

Детаљнији приказ генетских алгоритама превазилази обим овог рада, а може се наћи, на пример, у прегледним радовима [35, 16].

1.2.3 Метода променљивих околина

Методу променљивих околина су увели Ненад Младеновић и Пјер Хансен у [48]. Код ове метахеуристике се претражује сукцесивно систем околина текућег решења у потрази за решењем већег квалитета. Ако је пронађено квалитетније решење, онда се оно проглашава текућим решењем и процес понавља почевши од претраживања прве околине. Показало се да је промена околине важна и приликом "спуштања" до локалног минимума, као и приликом изласка из њега, у потрази за бољим решењем. Ова метахеуристика је успешно примењена у разним областима. Поменимо само неке од њих: локацијски проблеми [29], графовски проблеми [5], проблеми мешовитог целобројног програмирања [40], проблеми паковања [47], итд. Детаљни приказ ове метахеуристике и њених

примена је ван опсега овог рада, а може се наћи, на пример у [28].

VNS метахеуристика даје добре резултате зато што у многим оптимизационим проблемима локални минимуми нису у потпуности случајно размештени у простору претраге, већ постоји веза између њих. Из тог разлога алгоритам не прати неку задату путању, него покушава да мењајући околину најбољег решења коју претражује дође до бољих. На тај начин се увек чува најбоље решење и, заједно са њим, све његове добре особине.

Да би се проширила претрага, VNS у потрази за бољим локалним минимумом обично користи скуп околина растуће кардиналности. Нека је x произвољно решење и N_k, за $k = k_{min}, \cdots, k_{max}$, коначан скуп структура околина. Скуп решења у k-тој околини тачке x обележавамо са $N_k(x)$. Најчешће се користи структура околина за коју важи да $|N_{k_i}(x)| < |N_{k_{i+1}}(x)|$, где је $i = k_{min}, \cdots, k_{max} - 1$, мада се ова метода може примењивати и ако ово није испуњено.

У процесу претраге VNS алгоритам покушава да у тренутној околини најбољег решења S нађе боље решење. То се обично постиже применом две процедуре: размрдавања и локалне претраге. Процедуром "размрдавања" (енг. shaking) се добија кандидат који је потенцијално боље решење у односу на тренутно најбоље, које се даље процесира процедуром локалне претраге.

Процедура размрдавања је директно везана за дефинисани систем околина $N_k(x)$, односно решење x' се бира на случајан начин из $N_k(x)$. Пошто се избор врши на случајан начин, она врло мало утиче на време извршавање целог алгоритма, али може значајно да допринесе диверзификацији решења, односно за спречавање конвергенције у локалном оптимуму лошег квалитета.

Локална претрага је процедура која се користи за побољшање решења, а може користити самостално, а чешће у оквиру неког метахеуристичког алгоритма. Пошто ова процедура испитује могућности за побољшавање конкретног решења, врло је битна ефикасна имплементација ове процедуре, уз истовремено очување потенцијала за добијање бољих решења од задатог. Генерално, постоје две стратегије које се могу применити: стратегија примене првог побољшања (енг. First Improvement Strategy) и стратегија примене најбољег побољшања (енг. Best Improvement Strategy). Коришћењем прве стратегије већ прво побољшање на које смо наишли одмах примењујемо, и процедуру поново примењујемо од почетка. На тај начин су нам побољшања углавном мања, али их

чешће добијамо. У стратегији примене најбољег побољшања се чека док год се не пронађе највеће могуће побољшање за дату тачку, па се тек онда примењује. Тиме су побољшања углавном већа, али их спорије добијамо. Ова стратегија је нарочито погодна када имамо осмишљен брз начин за ажурирање функције циља, у случајевима када се решење врло мало мења. Више детаља о претходно описаним стратегијама се може видети у [27].

Дакле, после примене размрдавања, од решења x се добија неко решење x'. Оно се даље побољшава применом локалне претраге. Нека је x'' решење које се добило њеним завршетком (када се више не може добити побољшање уз помоћ локалне претраге).

Ако је x'' боље решење од x онда се оно прогласи тренутним решењем ($x \leftarrow x''$) и понавља поступак почевши од најмање околине. Ако је x'' лошије решење од x тада се покушава са новом, већом околином од x. У случају да не постоји већа околина онда се поступак понавља од размрдавања у тој највећој околини. Овај процес се наставља док год се не испуни критеријум завршетка. Као и код генетских алгоритама, могу се користити различити критеријуми завршетка рада VNS алгоритма, као што су максимални број итерација, максимално време извршавања или максимални број итерација без побољшања добијеног решења.

1.2.4 Метахеуристика заснована на електромагнетизму

Метахеуристику засновану на електромагнетизму (ЕМ) су увели Бирбил и Фанг у [6]. То је робустан и ефикасан алгоритам који даје квалитетна решења за проблеме глобалне оптимизације [6], а особина му је да брзо конвергира решењу задовољавајућег квалитета. Ова метахеуристика користи популацију и интеракцију између решења, моделовану по принципу привлачења и одбијања наелектрисаних честица у електростатичком пољу. Основна идеја ове методе је да добра решења привлаче сва остала решења, а лоша решења их одбијају. Детаљи конвергенције и остали теоријски аспекти ове методе су изван опсега ове дисертације, а могу се видети у раду [6].

Сваки члан популације који се разматра овом метахеуристиком називамо ЕМ тачком (или решењем), а саму популацију називамо скупом решења. Уопштено, функционисање ове метахеуристике би се могло описати на следећи начин: на почетку се популација поставља на случајан начин узимајући вре-

дности из скупа допустивих решења. Након тога се израчунава функција циља за све ЕМ тачке. На основу функције циља се израчунава наелектрисање за све ЕМ тачке. Користећи наелектрисања, израчунавају се силе привлачења и одбијања за све ЕМ тачке. Осим тачке са најбољом функцијом циља, померају се све ЕМ тачке у правцу израчунатих сила, али случајним кораком померања, чиме се обезбеђује да све ЕМ тачке имају вероватноћу већу од нуле да дођу у до сада непосећени регион. Ако је испуњен критеријум завршетка, завршава се са радом, а у супротном се поступак понавља почев од израчунавања функције циља за све тачке. Алгоритам метахеуристике засноване на електромагнетизму обично садржи и локалну претрагу за сваку ЕМ тачку пре израчунавања наелектрисања. Математички, конкретне формуле зависе од типа оптимизације (минимизација/максимизација), па ће оне бити дате у следећем поглављу.

1.3 Проблеми бисекције и повезаности подграфова

Граф се дефинише као уређени пар $G = (V, E)$, где је V скуп чворова, а E скуп грана. У зависности од тога шта чини скуп грана разликујемо разне врсте графова. Најчешће сматрамо да су гране двоелементни подскупови (неуређени парови) скупа чворова и тада такве графове називамо неоријентисаним графовима (енг. undirected). Приметимо да у неоријентисаним графовима нема петљи и паралелних грана. У случају да су гране уређени парови, кажемо да је реч о оријентисаним (енг. directed). У овом раду ће бити речи о неоријентисаним графовима код којих скупови чворова и грана имају коначан број елемената. За чвор $v \in V$ и грану $e \in E$ кажемо да су инцидентни ако је $v \in e$. За чворове $v_1, v_2 \in V$ кажемо да су суседни ако је $\{v_1, v_2\} \in E$. Ход или шетња (енг. walk) у графу је алтернирајући низ чворова и грана, такав да почиње и завршава чвором, и да су чворови инцидентни грани претходник и следбеник дате гране. Приметимо да се чворови и гране у шетњи могу понављати. Кажемо да су два чвора повезана ако постоји ход у графу који их повезује. Ако су свака два чвора неког графа повезана тада тај граф називамо повезаним. Подскуп скупа чворова $V' \subset V$ датог графа $G = (V, E)$ индукује подграф $G' = (V', E')$ графа тако што за елементе скупа грана E'

бирамо оне гране из E чија оба краја припадају подскупу V'. Повезани подграф $G' = (V', E')$ је подграф графа $G = (V, E)$ који је повезан. Степен чвора (енг. degree) је број грана инцидентних датом чвору.

За дати граф могуће је доделити тежине чворовима и/или гранама. Такав граф називамо тежинским. Ако су тежине додељене само чворовима (гранама) кажемо да је реч о графу са тежинским чворовима (гранама). Тежине су обично бројеви (или вектори бројева као што је случај у овој дисертацији), који се обично бирају из скупа целих или реалних бројева.

Проблеми из теорије графова налазе примену у разним областима математике, али и других наука, као што су рачунарство, операциона истраживања, биохемија, молекуларна биологија, разне области инжењерства итд.

Значајан број графовских проблема је заснован на партиционисању графа. Партиционисање графа подразумева разбијање скупа чворова V на партиције $V_1, V_2, ..., V_k$ и да графови индуковани партицијама имају одређене особине. Ову поделу називамо партиционисањем графа. Ако граф делимо на две партиције онда имамо такозвану бипартицију графа. Неки проблеми бипартиционисања графа су Max Cut, Min Cut, Бисекција, итд. Max Cut проблем је проблем код ког се тражи партиционисање графа на две партиције при чему је број грана чији се крајеви налазе у различитим партицијама максималан. Проблем се може уопштити тако да се посматра граф са тежинама на гранама. У том случају је Max Cut проблем заправо проблем партиционисања графа на две партиције такве да је сума тежина грана између њих максимална. Показано је да је овај проблем NP-тежак [21]. Специјални случај Max Cut проблема је Проблем Максималне Бисекције (енг. Maximum Bisection Problem или Max-Bisection Problem) где се додаје још захтев да партиције имају једнак број чворова. У општем случају и овај проблем је NP-тежак [21].

За граф са тежинама (на гранама или чворовима) се могу разматрати разни проблеми везани за одређивање подграфа са ограниченим степеном чвора (енг. Degree-Constrained Subgraph problems). Ограничење степена чвора може значити да се тражи да степен сваког чвора из подграфа не сме бити већи, или да степен чвора не сме бити мањи од унапред задатог броја d. Поред тога, могу се постављати захтеви у вези природе подграфа. Неки од најпознатијих примера су нпр. проблем разапињућег стабла минималног степена [19] (енг.

Minimum-Degree Spanning Tree), проблем Штајнеровог стабла најмањег степена [20] (енг. Minimum-Degree Steiner Tree), проблем најмањег подграфа са чворовима минималног степена [3] (енг. Minimum Subgraf of Minimum Degree), итд.

Један од проблема везаних за одређивање подграфа са ограниченим степеном чвора је и проблем проналажења повезаног подграфа највеће тежине са чворовима ограниченог степена (енг. Maximum Degree Bounded Connected Subgraph Problem - MDBCSP). Нека је $G = (V, E)$ неповезан граф, $d \geq 2$ природан број и $w : E \to \mathbb{R}^+$ функција тежине. Проблем проналажења повезаног подграфа највеће тежине са чворовима ограниченог степена је проблем проналажења подграфа $G' = (V', E')$, где је $V' \subseteq V$ и $E' \subseteq E$, тако да је подграф G' повезан, степен чворова у G' не прелази d и $\sum_{e \in E'} w(e)$ има максималну вредност.

Глава 2

Вишедимензионални проблем максималне бисекције графа

Мотивација за изучавање проблема максималне бисекције графа потиче из привреде. Посматрајмо следећи организациони проблем. У фабрици са 300 радника треба формирати две осмочасовне смене од 150 радника. Да би се постигли најбољи резултати и учинили рад ефикаснијим, у поређењу са претходним искуствима, надзорник разматра три аспекта сваког пара радника који су радили заједно на неком претходном задатку. Та три аспекта су: психолошки, физички и образовни. Први аспект укључује лакоћу сарађивања у претходним случајевима; други, физичку компатибилност, могућност да радници подједнако деле тешкоће посла и трећи, образовну баријеру која може стајати између пара радника у вези са разумевањем и извршавањем задатка. Надзорник одлучује да примени следећи метод на одлучивање како ће поделити радну снагу. Ако се радници организују у смене, важно је колико су радници који раде у истој смени компатибилни између себе. Један од начина да се повећа компатибилност између радника исте смене је да се смањи компатибилност између радника у супротним сменама. Дакле, повећање некомпатибилности међу различитим сменама може да буде основ за поделу посла. Након нормализовања оцена по различитим аспектима, укупна некомпатибилност се може измерити сумирањем разлика међу паровима радника у различитим сменама. Разлике се сумирају по сваком аспекту, а најмања сума се разматра као основ за поделу због тога што је по свим другим аспектима неслагање веће.

Сада, представљени проблем може бити описан у графовској терминологији.

Сваки радник је представљен чвором у графу, и грана повезује два радника који су радили до сада у некој претходној прилици. Тежина гране је тројка која се састоји од вредности три претходно поменута аспекта. Идеја је да се подели граф у два подграфа са подједнаким бројем чворова, тако да је минимум тежина грана по координатама чији су чворови у различитим подграфовима максималан.

Као што се може видети, ово представља нови проблем, вишедимензионални проблем максималне бисекције (MDMBP), који се формално задаје: За граф $G = (V, E)$ са ненегативним r-торкама као тежинама на гранама и где је $|V|$ паран број, проблем се састоји у проналажењу партиције скупа чворова V у два подскупа S и $V \setminus S$, где је $|S| = |V \setminus S|$, и тежина пресека између скупова максимална. Приметимо да максималној тежини пресека одговара максимална некомпатибилност између одговарајућих скупова радника у претходно наведеном примеру. Тежина пресека је минимум сума координата тежина грана чији крајеви припадају различитим скуповима S и $V \setminus S$.

Проблем описан у претходном пасусу је уведен у [43]. У односу на полазни проблем максималне бисекције графа, MDMBP је знатно тежи за решавање. Очигледно је да је рачунање функције циља временски захтевније, што самим тим отежава и одређивање у који скуп треба ставити који чвор.

Лако је показати да је MDMBP генерализација проблема максималне бисекције (MBP). Проблем максималне бисекције (MBP) је добро познат проблем комбинаторне оптимизације. За граф $G = (V, E)$ са ненегативним тежинама на гранама и где је $|V|$ паран број, проблем максималне бисекције се састоји у проналажењу поделе (енг. partition) скупа чворова V на два подскупа S и $V \setminus S$, где је $|S| = |V \setminus S|$ и где је сума тежина на гранама између скупова максимална. Проблем максималне бисекције се може применити у разним областима попут организације електронских кола [57], процесирања слика [56], оптимизације преводилаца [31], итд.

Проблем максималне бисекције је NP-тежак што је показано у [22]. Комплексност проналажења оптималног односно доброг приближног решења проблема максималне бисекције је довело до развоја разних приступа решавању овог проблема почев од егзактних метода преко апроксимативних алгоритама до метахеуристика.

Једна од широко коришћених квадратних формулација [41] за МВР са бинарним променљивама x_j које су додељене сваком чвору је

$$\max \quad \frac{1}{4}\sum_{i,j\in V} w_{ij}(1 - x_i x_j)$$

тако да важи

$$e^T x = 0$$
$$x_j \in \{-1, 1\} \quad j = 1, \ldots, n$$

где је $e \in R^n$ вектор колона са свим јединицама, ^{T} оператор транспоновања и w_{ij} је тежина гране између чворова i и j. Треба приметити да је $x_j = 1$ или $x_j = -1$ па је $S = \{j|x_j = 1\}$ или $S = \{j|x_j = -1\}$.

Ова формулација омогућава примену апроксимативних алгоритама који су засновани на семидефинитном програмирању. Приступ Goemansa и Williamsona максималној бисекцији из [24] је проширен од Frizea и Jerruma у [18] и добијен је апроксимативни алгоритам са фактором 0.651. У [63] Ye је побољшао фактор на 0.699 модификацијом Frieze-Jerrum приступа. Фактор апроксимације је даље побољшан на 0.7016 од Halperina и Zwicka у [26], укључујући неке неједнакости троугла у семидефинитну релаксацију. Напоменимо да су апроксимативни алгоритми са константним фактором, пре свега, теоријске природе и да нема сврхе експериментално поредити обичне хеуристичке методе са њима. Разлози за то су у сасвим различитој конструкцији ове две класе приближних алгоритама. Док је основни циљ апроксимативних алгоритама са константним фактором добијање решења која никако нису лошија од доње границе која је једнака производу оптималног решења и фактора, дотле је основни циљ метахеуристика добијање што бољег решења на конкретној инстанци. У раду [30] дат је доказ да не постоји полиномијални апроксимативни алгоритам са фактором већим од $\frac{16}{17}$.

Поред горе поменуте квадратне формулације, постоји више других приступа за његово егзактно решавање као што су методе гранања и сечења (Branch-and-Cut), засноване на линеарном односно семидефинитном квадратном моделу [4] и метода заснована на пресеку семидефинитне и полиедарске релаксације [52].

У раду [52], аутори су представили метод за проналажење егзактног решења за Max-Cut проблем заснован на семидефинитној формулацији. Семидефинитна релаксација је коришћена и комбинована са неједнакостима троугла.

Овај приступ користи Лагранжову дуалност за добијање горњих граница у разумном времену. Рачунски интензивнији део њихове процедуре ограничавања је решавање основне семидефинитне релаксације Max-Cut проблема. Аутори такође дискутују применљивост њиховог приступа на специјални случај Max-Cut проблема где је кардиналност партиција једнака, тј. на проблему максималне бисекције.

Други скуп приступа, нарочито погодан за добијање решења инстанци велике димензије, су метахеуристике. Поменимо само неке од широког спектра примењених метахеуристика као што су: меметички алгоритам [59], метода променљивих околина [41], неуронске мреже [60], детерминистичко каљење [13], итд.

Меметички алгоритам представљен у [59] интегрише групишући оператор укрштања и процедуру оптимизације са табу-претрагом. Предложени оператор укрштања врши очување заједничке највеће групе чворова, водећи рачуна о родитељским решењима са истовременом контролом удаљености потомачких решења од родитељских. Експериментални резултати указују да меметички алгоритам побољшава, у великом броју случајева, претходна најбоља позната решења за MBP.

У [41], Ling и остали су применили ограничење $e^T x = 0$ са функцијом циља и добили нови оптимизациони проблем који је еквивалентан проблему максималне бисекције, и тада су искористили посебну технику похлепне локалне претраге на резултујући проблем. Експериментални резултати у том раду указују да је предложена метода ефикасна и да добија квалитетна решења за проблем максималне бисекције.

У [60], алгоритми засновани на Лагранжевим мрежама су предложени за решавање проблема максималне бисекције. Бисекциона ограничења су релаксирана у функцији циља увођењем казнене функције. Решење бисекције је рачунато коришћењем дискретних Хопфилдових неуронских мрежа (DHNN). Растући фактор казнене функције може боље помоћи DHNN да избегне локални оптимум и да добије задовољавајућу конкретну бисекцију. Додатно је дата анализа конвергенције предложеног алгоритма. На крају, експериментални резултати су презентовани на G-set скупу инстанци великих димензија.

У [13] је предложен алгоритам детерминистичког каљења за решавање про-

блема максималне бисекције. Гранична функција је заснована на квадратном корену (енг. square-root barrier function), где се параметар који означава границу постепено смањује слично као температура у методи симулираног каљења. Алгоритам претражује боља решења у правцу допустивог спуштања (енг. feasible descent direction), која имају жељену особину да су доња и горња граница променљивих увек задовољене аутоматски ако је дужина корака број између 0 и 1. Експериментални резултати показују да је алгоритам знатно бржи од једног од набољих постојећих апроксимативних алгоритама при том добијајући решења која су приближно истог квалитета.

2.1 Дефиниција проблема

У овом поглављу ће бити дата дефиниција вишедимензионалне генерализације проблема максималне бисекције (MDMBP), која је уведена у раду аутора ове дисертације [43]. У овом проблему су тежине на гранама уместо реалних бројева, задате као r-торке ненегативних реалних бројева. Тежина пресека је минимум сума тежина по координатама. Задатак је пронаћи поделу скупа чворова V у два дисјунктна подскупа (партиције) са једнаким бројем чворова и максималном тежином пресека. За $r = 1$ MDMBP се своди на стандардни проблем максималне бисекције. Из чињенице да је проблем максималне бисекције NP-тежак, и из чињенице да је проблем MBP специјалан случај MDMBP следи да је и проблем MDMBP такође NP-тежак.

Тежина пресека код вишедимензионалног проблема максималне бисекције се одређује тако што се прво, по координатама, сумирају вектори тежина грана које повезују чворове из S и $V \setminus S$. Након тога се одређује минимум добијених сума. Добијени минимум је тежина пресека.

Сада ће бити дата и формална математичка формулација MDMBP. Нека је $G = (V, E)$ неоријентисани граф, и w је функција која додељује свакој грани $e = \{i, j\}$ неку r-торку ненегативних реалних бројева $(w_{e1}, w_{e2}, \ldots, w_{er})$ и $S \subseteq V$. Пресек $C(S)$ одређен скупом S је дефинисан као

$$C(S) = \{e \in E | e \cap S \neq \emptyset \wedge e \cap (V \setminus S) \neq \emptyset)\}. \tag{2.1}$$

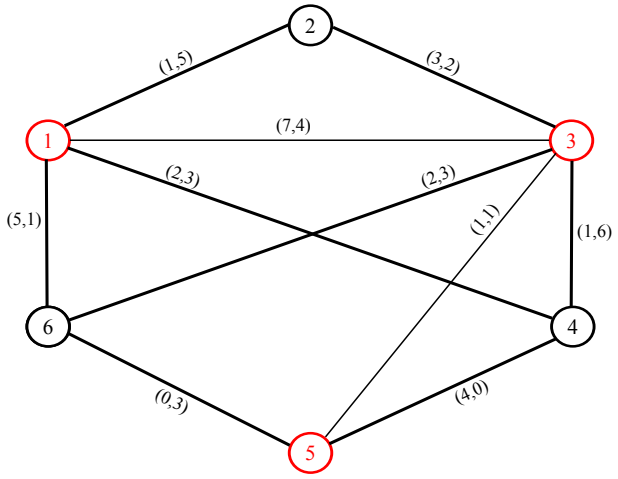
Из дефиниције је очигледно да су пресеци $C(S)$ и $C(V \setminus S)$ исти скупови.

Тежина пресека се дефинише као

$$w(C(S)) = \min_{1 \leq l \leq r} \sum_{\{i,j\} \in C(S)} w_{\{i,j\}l}. \tag{2.2}$$

Задатак MDMBP је да одреди партицију скупа чворова у два дисјунктна скупа S и $V \setminus S$ где је $|S| = |V \setminus S|$ и где је тежина пресека максимална.

Пример 2.1. *Вишедимензионални проблем максималне бисекције може бити илустрован примером датим на слици 2.1, где је оптимално решење дато скупом* $S = \{1, 3, 5\}$. *Скуп* S *генерише пресек* $C(S) = \{\{1,2\},\{1,4\},\{1,6\},\{2,3\},\{3,4\},\{3,6\},\{4,5\},\{5,6\}\}$ *где су суме по координатама* $(18, 23)$ *па је тежина пресека* 18.



Слика 2.1: Пример графа са паровима као тежинама на гранама

Вишедимензионални проблем максималне бисекције се јавља где год су релације између ентитета карактерисане векторима бројева уместо само бројевима. Једна практична примена, поменута у [43], је у оптимизацији организације електронских кола. Електронске компоненте имају одређене аспекте који се могу сматрати тежинама, као што су интерференција, коришћена струја, повезаност, дисипација топлоте, итд. Проблем је доделити електронске компоненте једној од две плоче на такав начин да, на пример, две топле компоненте (најтоплије) буду на различитим плочама.

2.2 Модел мешовитог целобројног линеарног програмирања

У овом потпоглављу уводимо модел целобројног линеарног програмирања за вишедимензионални проблем максималне бисекције.

Нека су $S \subseteq V$, $|V| = n$, и r димензија вектора тежина и нека је w_{el} l-та координата вектора тежине гране e. Дефинишимо бинарне променљиве x и y на следећи начин:

$$x_i = \begin{cases} 1, & i \in S \\ 0, & i \in V \setminus S \end{cases}$$

$$y_e = \begin{cases} 1, & \text{e} \in \text{C(S)} \\ 0, & \text{e} \notin \text{C(S)} \end{cases}$$

Формулација мешовитог целобројног линеарног програмирања за решавање MDMBP може се записати као:

$$\max U \tag{2.3}$$

тако да важи

$$U \leq \sum_{e \in E} w_{el} \cdot y_e, \quad 1 \leq l \leq r \tag{2.4}$$

$$x_{e_i} + x_{e_j} \geq y_e, \quad \{e_i, e_j\} = e \in E \tag{2.5}$$

$$x_{e_i} + x_{e_j} + y_e \leq 2, \quad \{e_i, e_j\} = e \in E \tag{2.6}$$

$$\sum_{i=1}^{n} x_i = n/2, \tag{2.7}$$

$$x_i, y_e \in \{0, 1\}, i \in V, e \in E \tag{2.8}$$

$$U \in [0, +\infty) \tag{2.9}$$

Теорема 1. *Оптимално решење вишедимензионалног проблема максималне*

бисекције постоји ако и само ако постоји оптимално решење MILP формулације (2.3)-(2.9).

Доказ. ($\Rightarrow$)

Претпоставимо да је S оптимално решење и да је његов одговарајући пресек $C(S)$. Доказаћемо да су услови (2.3)-(2.9) задовољени.

Дефинишимо промењљиве x и y као у горе наведеном излагању. То можемо учинити зато што знамо партиционисање скупа V и имамо одређен пресек. Ако је $y_e = 0$ тада су (2.5) и (2.6) очигледно испуњени Ако је $y_e = 1$ тада одговарајућа грана $e = \{i, j\}$ припада пресеку, а тачно један од инцидентних чворова гране e мора да буде у скупу S. Тада или је $x_{e_i} = 1$ или $x_{e_j} = 1$ и самим тим услови (2.5) и (2.6) испуњени.

Услов (2.7) је очигледно испуњен зато што је захтевано да се скуп чворова партиционише на два скупа једнаке кардиналности.

Услови (2.8) и (2.9) су директно задовољени из дефиниције промењљивих x, y, а природа промењљиве U је одређена тиме да тражимо максимум и чињеницом да вектори тежина по гранама имају ненегативне координате.

($\Leftarrow$) Претпоставимо да имамо оптимално решење (x^*, y^*, U^*) MILP формулације (2.3)-(2.9). Партиционисање скупа V у два подскупа S і $V \setminus S$ је тада одређено са $S = \{i \in V | x_i^* = 1\}$, док је пресек дат са $C(S) = \{e \in V | y_e^* = 1\}$.

Из овако дефинисаног $C(S)$ и ограничења (2.4) може се закључити да је

$$U^* \leq \min_{1 \leq l \leq r} \sum_{\{i,j\} \in C(S), i \in S j \notin S} w_{\{i,j\}l} \leq \sum_{\{i,j\} \in C(S)} w_{\{i,j\}l}$$

што значи да је $U^* \leq w(C(S))$. Како је U^* максимум функције циља (2.3) оно је једнако највећој тежини пресека по свим скуповима S где је $|S| = |V|/2$.

Из услова (2.8) се види да су y^* бинарне промењљиве. Ако је $y_e^* = 1$ онда се из услова (2.5) и (2.6) може закључити да инцидентни чворови гране e не могу бити оба у скупу S или оба у скупу $V \setminus S$. Из тога следи да је пресек $C(S)$ добро дефинисан.

Ако је $y_e^* = 0$ тада из ограничења (2.3)-(2.4) следи да оба чвора гране e морају бити у истој партицији (или S или $V \setminus S$). Ако би чворови били у различитим партицијама, онда би се могло закључити да вектор тежина гране e није укључен у тежину пресека, и према томе U^* не би било максимално што

је контрадикција са (2.3)-(2.9). Дакле, закључујемо да у овом случају чворови гране e морају бити у истој партицији.

Из услова (2.7) следи да $|S| = n/2 = |V \setminus S|$, што значи да је скуп чворова заиста партиционисан у два скупа са једнаком кардиналношћу.

Пошто из услова (2.8) следи да су променљиве x бинарне јасно је да ће сваки чвор бити или у S или у $V \setminus S$ што значи да су ова два скупа добро дефинисана. □

Као што се види из саме формулације, број променљивих и број ограничења је полиномијално зависан од димензије самог проблема. Тачније, модел садржи укупно $|V| + |E|$ променљивих и $2|E| + r + 1$ ограничења. Овај, релативно мали број променљивих и ограничења даје добре предуслове за практичну примену, што је и приказано у поглављу 2.6.

2.3 Генетски алгоритам за решавање MDMBP

У овом поглављу ће бити приказан GA за решавање MDMBP.

Општи принципи рада GA описани су у уводу, а предложени GA за решавање овог проблема приказан је алгоритмом 1.

2.3.1 GA репрезентација

За решавање MDMBP коришћено је целобројно кодирање јединки, односно генетски код представља низ гена где сваки ген представља један цео број. Број гена у генетском коду је $n/2$ за овај проблем, где сваки ген одређује један чвор из S. Ови чворови могу бити означени са a_i, $1 \leq i \leq n/2$ и скуп S је дефинисан као $S = \{1 + a_1, \ldots, 1 + a_{n/2}\}$, пошто су чворови графа нумерисани од 1 до n, а елементи a_i су опсегу од 0 до $n - 1$.

У овом GA се користи приступ који гарантује допустивост јединки. Да би се то испоштовало, све вредности a_i морају бити међусобно различите, па најпростија додела $a_i \leftarrow g_i$ није могућа, где је g_i, i-ти ген. У општем случају, формула за ову доделу за a_i мора узети у обзир све претходно изабране чворове $a_1, \ldots, a_{i-1}$. Због тога, ова додела сукцесивно одабира чворове $a_i, 1 \leq i \leq n/2$, у скуп S, водећи рачуна да се приликом одабира новог чвора a_i не разматрају

Algorithm 1: GA псеудо код

```
   Algorithm GA ;
 1 Input_Data() ;
 2 Random_Init() ;
   while not Stopping_Criterion() do
       foreach ind from (N_elite + 1) to N_pop do
           if Exist_in_Cache(ind) then
 3             obj_ind ← Get_Value_From_Cache(ind);
           end
           else
 4             obj_ind ← Objective_Function(ind) ;
 5             LS1(ind,obj_ind) ;
 6             Put_Into_the_Cache_Memory(ind,obj_ind) ;
               if Full_Cache_Memory() then
                   Remove_LRU_Block_From_Cache_Memory();
               end
           end
 7     end
 8     LS2(best,obj_best);
 9     Fitness_Function();
10     Selection();
11     Crossover();
12     Mutation();
   end
13 Output_Data();
```

они чворови који су већ одабрани. Дакле, постоји n могућности за g_1, $n-1$ могућности за g_2 и $n/2+1$ могућности за $g_{n/2}$.

Вредност i-тог гена g_i $(1 \leq i \leq n/2)$ би требало да буде између 0 и $n-i$. Да би ово било постигнуто, користи се додатна формула $g_i \leftarrow g_i \ (mod\ (n-i+1))$. Тада, вредност a_i се добија као елемент са позицијом g_i из листе некоришћених елемената.

Пример 2.2. *Размотримо сад податке из примера 2.1, у коме је* $n = 6$. *На пример, кодирање* $g = (3, 6, 2)$ *представља низ* $a = (3, 1, 4)$ *на следећи начин:*

- *Да би се гарантовала допустивост кодирање* $g = (3, 6, 2)$ *ће бити трансформисано у* $g = (3, 1, 2)$, *пошто је* $g_1 = 3 \ (mod\ 6)$, $g_2 = 6 \ (mod\ 5)$ *и* $g_3 = 2 \ (mod\ 4)$;
- *На почетку, сви чворови су неискоришћени, па је елемент* $a_1 = g_1 = 3$. *Даље, само је елемент* 3 *искоришћен, па је неискоришћени елемент на позицији* $g_2 = 1$ *једнак* $a_2 = 1$. *На крају, елементи* 1 *и* 3 *су искоришћени, па је* a_3 *једнако неискоришћеном елементу на позицији* $g_3 = 2$, *што је* **4**;
- *Одатле,* $S = \{4, 2, 5\}$.

Предност оваквог кодирања је да су све јединке допустиве. Овај принцип рада само са допустивим јединкама примењен је и на операторе укрштања и мутације.

2.3.2 Локална претрага

Процедура локалне претраге се користи да би се унапредила одређена јединка (допустиво решење). У овој имплементацији, постоје две процедуре локалне претраге, означене са LS1 и LS2. Као што се може видети из Алгоритма 1, процедура LS1 се примењује на сваку не-елитну јединку у свакој генерацији. Након процесирања свих не-елитних јединки (елитне јединке задржавају вредности својих решења из претходне генерације), LS2 се примењује на најбољу јединку у целокупној популацији.

Процедура LS1 ја заснована на идеји локалне претраге описане у [59]. На почетку се формира структура података која бележи колики је добитак (добитак може да има негативну вредност) при премештању сваког чвора у супротни

скуп. При томе се посматрају сви елементи скупа V, па уколико је елемент у скупу S рачуна се добитак уколико би он прешао у скуп $V \setminus S$, а у супротном, уколико је у скупу $V \setminus S$ рачуна се добитак уколико би он прешао у скуп S.

У свакој итерацији посматра се чвор са највећим добитком премештаја из S у супротни скуп $V \setminus S$, врши се ажурирање низа добитака, затим се посматра чвор са највећим добитком премештаја из $V \setminus S$ у супротни скуп S и поново врши ажурирање низа добитака. Уколико се функција циља за овакав пар чворова који се премештају повећала, примењује се то премештање, а у супротном се завршава рад процедуре локалне претраге. У процесу формирања поменуте структуре података не постоји никаква временска уштеда, али у сваком следећем кораку се та структура ажурира практично у линеарном времену $O(n)$.

Међутим, за разлику од [59], где је структура једнодимензиони низ, у случају MDMBP ситуација је знатно сложенија па је структура матрица, где сваки ред представља чвор графа, а одговарајућа колона представља r-торку добитака када дати чвор мења скуп. Сада, уместо да се памти добитак за сваки чвор, могуће је памтити добитке по свакој координати тежине гране. Да би се одлучило који чвор ће се померити из скупа S у његов комплемент $V \setminus S$, неопходно је одредити:

$$\max_{v \in S} \left\{ \min_{1 \leq l \leq r} \{sum_l + \Delta_{v,l}\} \right\} \tag{2.10}$$

за свако $1 \leq l \leq r$, $sum_l = \sum_{e \in C(S)} w_{el}$ и $\Delta_{v,l}$ је добитак израчунат са:

$$\Delta_{v,l} = \begin{cases} \sum_{x \in S, x \neq v} w_{(v,x),l} - \sum_{y \in V \setminus S} w_{(v,y),l}, & v \in S \\ \sum_{y \in V \setminus S, y \neq v} w_{(v,y),l} - \sum_{x \in S} w_{(v,x),l}, & v \in V \setminus S \end{cases} \tag{2.11}$$

Сада, чвор v са максималним добитком ће се померити из скупа S у $V \setminus S$. Ако има више чворова за које је добијен максимум, биће изабран један од њих на случајан начин. Након померања, добитак за сваки чвор $u \in V$ се ажурира формулом:

- Случај $u = v$: $\Delta_{u,l} \leftarrow -\Delta_{v,l}$;
- Случај $u \neq v \wedge (u, v \in S \vee u, v \in V \setminus S)$: $\Delta_{u,l} \leftarrow \Delta_{u,l} - 2w_{(u,v),l}$;
- Случај $(u \in S \wedge v \in V \setminus S) \vee (v \in S \wedge u \in V \setminus S)$: $\Delta_{u,l} \leftarrow \Delta_{u,l} + 2w_{(u,v),l}$.

Након што је померај извршен важи $|S| = n/2-1$, па је неопходно померити један чвор из $V \setminus S$ у S. Приликом померања чвора, треба ажурирати структуру података у којој се бележе добици, што се ради аналогно. Временски најзахтевнији део LS1 је прво израчунавање добитака $\Delta_{u,l}$. Сва наредна ажурирања се обављају веома брзо, пошто се добици $\Delta_{u,l}$ мењају само за v и његове суседе. Одатле, LS1 је нарочито ефикасан за ретке графове.

У LS2 процедури локалне претраге, алгоритам покушава да унапреди решење заменом једног елемента $v_1 \in S$ једним елементом $v_2 \in V \setminus S$. Примењује се на сваки пар $(v_1, v_2) \in S \times (V \setminus S)$ са стратегијом првог унапређења, што значи да када је унапређење достигнуто, оно се одмах примењује и локална претага LS2 почиње поново из почетка са унапређеним записом. Ако за сваки пар (v_1, v_2), процедура замене продукује вредности функције циља које су мање или једнаке од оригиналне, локална претрага се завршава без даљих унапређења.

Да би се добила ефикасна процедура локалне претраге, процес поновног израчунавања вредности функције циља у LS2 је унапређен на следећи начин. Размотримо пар $(v_1, v_2) \in S \times (V \setminus S)$ и већ израчунату вредност sum_l. За све чворове $b \in N(v_1)$, где је $N(v_1)$ суседство чвора v_1, низ sum мора бити ажуриран тако да узме у разматрање замену (v_1, v_2):

$$sum_l^{new} = \begin{cases} sum_l + w_{(v_1,b),l}\ , & b \in S \\ sum_l - w_{(v_1,b),l}\ , & b \in V \setminus S \wedge b \neq v_2 \\ sum_l, & b = v_2 \end{cases} \tag{2.12}$$

где $l = 1, \ldots, r$. Сада, $sum_l = sum_l^{new}$, и слична формула је примењена за чворове v_2 и све чворове $b \in N(v_2)$:

$$sum_l^{new} = \begin{cases} sum_l + w_{(v_2,b),l}\ , & b \in V \setminus S \\ sum_l - w_{(v_2,b),l}\ , & b \in S \wedge b \neq v_1 \\ sum_l, & b = v_1 \end{cases} \tag{2.13}$$

Треба приметити да је свака итерација LS2 знатно спорија од процеса ажурирања у LS1, али процедура LS2 може да буде од великог значаја, с обзиром на то да се замена два чвора не може извршити помоћу два LS1 помераја. Да би се направио добар баланс између перформанси алгоритма и корисности примене

LS2, у овом GA се LS2 примењује само на најбољу јединку.

2.3.3 Селекција

Оператор селекције, који бира родитељске јединке за генерисање потомака у следећој генерацији, је унапређени оператор турнирске селекције [15]. Овај оператор користи реални параметар F_{tour} који означава жељену просечну величину турнира. Да би се проценила жељена величина турнира два типа турнира су примењена. Први тип ће бити одржан k_1 пута на $\lfloor F_{tour} \rfloor$ јединки, док ће други тип бити примењен k_2 пута са $\lceil F_{tour} \rceil$ јединки које учествују, тако да $F_{tour} \approx \frac{k_1 \cdot \lfloor F_{tour} \rfloor + k_2 \cdot \lceil F_{tour} \rceil}{k_1 + k_2}$. У нашој имплементацији $F_{tour} = 5.4$ је преузето из [15] и одговарајуће вредности k_1 и k_2 су 30 и 20, редом.

2.3.4 Укрштање и мутација

Након примене оператора селекције, изабране јединке ће бити процесиране оператором укрштања. Јединке су упарене на случајан начин. Једна од основних идеја генетског алгоритма је да кроз оператор укрштања две изабране јединке размене гене и продукују потомке који ће имати побољшане квалитете и нарочито функцију циља у односу на родитељске јединке. Приликом сваког укрштања су генерисана два потомка која ће у наредној генерацији евентуално заменити неке јединке из тренутне генерације. У овом раду је коришћено стандардно укрштање у једној тачки. Овај стандардни оператор бира на случајан начин место у генетском коду у којој ће се обавити укрштање. На том месту дели родитељске јединке и формира јединке потомака тако што ће надовезати први део генетског кода (до места укрштања) прве родитељске јединке са другим делом генетског кода (од места укрштања) друге родитељске јединке, а други потомак се креира тако што се надовеже први део генетског кода друге родитељске јединке са другим делом прве родитељске јединке. У овој имплементацији је ниво укрштања једнак 0.85, што значи да у отприлике 85% случајева парови јединки размењује гене.

У овом раду се користи оператор просте мутације са замрзнутим генима, који су као концепт први пут уведени у раду [53]. Директна примена овог оператора мења случајно изабран ген са неким нивоом мутације. Такође, оператор је модификован да би се избегао проблем замрзнутих гена. Замрзнути

гени су гени који су на одређеној позицији исти за све јединке у популацији. Они не могу бити промењени ни селекцијом ни укрштањем зато што су присутни код свих јединки у популацији. Постојање замрзнутих гена ефективно смањује простор претраге и повећава могућност превремене конвергенције.

Модификација једноставног оператора просте мутације у GA је да је ниво мутације увећан само за замрзнуте гене. У овој имплементацији, основни ниво мутације је $0.4/n$, док је увећани ниво мутације примењен на замрзнутим генима тачно 2.5 већи, и једнак $1.0/n$.

Приметимо да мале промене у гену могу да резултују великим променама у низу a. Следећи пример илуструје такву ситуацију.

Пример 2.3. *Размотримо податке из примера 2.2. Ако родитељ* $g^{(P)} = (5, 3, 1)$ *мутирајући на првој позицији може да продукује потомка* $g^{(O)} = (0, 3, 1)$*, онда њихови одговарајући a-низови могу бити* $a^{(P)} = (5, 3, 1)$ *и* $a^{(O)} = (0, 4, 2)$*. У том случају, као што је поменуто, мутација на позицији 1 продукује промене на све три позиције.*

2.3.5 Функција прилагођености и политика замене генерације

Квалитет појединачне јединке се мери израчунавањем функције циља. За познато S, процес израчунавања функције циља је директно повезан са дефиницијом MDMBP кроз формуле (2.1) и (2.2). Треба поменути да у процесу рачунања (2.2) нема потребе за чувањем $\sum_{\{i,j\} \in C(S)} w_{\{i,j\}l}$ за свако $l = 1, \ldots, r$. Ипак, корисно је запамтити низ ових сума, са циљем убрзања процедуре локалне претраге.

Једна од најзначајнијих карактеристика јединки је вредност њене функције прилагођености. У овој имплементацији, ако је ова вредност довољно велика, онда ће одговарајућа јединка прећи у следећу генерацију. То значи да решење, које одговара јединки са великом вредности функције прилагођености, има задовољавајући квалитет и да га не треба унапред искључити из даљег разматрања. Вредност функције прилагођености, означена са f_{ind}, $ind = 1, 2, \ldots, N_{pop}$ (где је N_{pop} број јединки у популацији), се израчунава скалирањем вредности функције циља јединке на интервал $[0, 1]$. Скалирање вредности функције циља се изводи због тога што на почетку могу постојати јединке са веома великом вредношћу функције циља што може да води доминацији таквих јединки у

процесу претраге, а са друге стране, при крају претраге, разлика између јединки може бити веома мала. Процес скалирања се у овој имплементацији изводи тако да је најбољој јединки ind_{max}, тј. јединки са најбољом функцијом циља, дата вредност 1, док је најгорој јединки ind_{min} додељена вредност функције прилагођености 0. Прецизније,

$$f_{ind} = \frac{obj_{ind} - obj_{ind_{min}}}{obj_{ind_{max}} - obj_{ind_{min}}},$$

где obj_{ind} представља функцију циља јединке ind, односно вредност тежине одговарајућег пресека.

Након одређивања вредности функције прилагођености, јединке су сада уређене у неопадајућем реду по вредности функције прилагођености $f_1 \geq f_2 \geq \ldots \geq f_{N_{pop}}$.

Уобичајена пракса у имплементацији GA је одређивање елитних јединки, тј. јединки које ће директно проћи у следећу генерацију. У највећем броју случајева, број елитних јединки је мали, зато што постоје два начина да квалитетне јединке пређу у следећу генерацију, једном кроз оператор селекције и једном због тога што су елитне и по дефиницији прелазе у следећу генерацију. Да би се спречило фаворизовање елитних јединки и сачувала добра решења у процесу претраге, али и да би се одржао довољан број јединки које нису елитне и да би се обезбедио довољно велики простор претраге, вредност функције прилагођености за елитне јединке се израчунава следећом формулом

$$f_{ind}^{(new)} = \begin{cases} f_{ind} - \overline{f}, & f_{ind} > \overline{f} \\ 0, & f_{ind} \leq \overline{f} \end{cases} \tag{2.14}$$

где је $ind = 1 \ldots N_{elite}$ и N_{elite} је број елитних јединки у популацији и $\overline{f} = \frac{1}{N_{pop}} \sum_{ind=1}^{N_{pop}} f_{ind}$.

Коришћење ове методе омогућава да не само елитне јединке, него и друге јединке, имају шансу да пређу у следећу генерацију. Притисак селекције на мање погодне јединке (оне са мањом функцијом прилагођености) је сада смањен, док је велики елитизам и дање осигуран, што смањује могућност преурањене конвергенције и губитка генетског материјала.

Примена елитистичке стратегије омогућава значајну уштеду у времену из-

вршавања GA. Примарно, генетски оператори се примењују само на јединке које нису елитне, а како је вредност функције циља елитних јединки иста као у претходној генерацији, није је неопходно поново израчунавати.

Појављивање јединки са истом вредношћу функције циља у истој генерацији може да се јави из два разлога. У првом случају, јединке имају исти генетски код, и у складу са тиме имају исту вредност функције циља. У другом случају јединке имају исту вредност функције циља али другачији генетски код. Понављање јединки у првом случају значајно деградира перформансе GA, пошто вишеструка појава истих јединки не доноси ништа ново већ само спречава друге јединке да продукују потомке. Због тога све копије, осим оригинала, исте јединке ефективно избацујемо из популације у наредној генерацији тако што им прилагођеност поставимо на нулу. У другом случају ефекти нису толико драстични, али је потребно ограничити број оваквих јединки да не би преовладале у популацији. Од свих јединки са истом вредношћу функције циља, али различитим генетским кодом, само првих N_{rv} од њих ће имати функцију прилагођености различиту од нуле, где је N_{rv} унапред задат параметар. Ово је веома ефикасан начин за чување алгоритма од преурањене конвергенције док се у међувремену чува разноликост генетског материјала.

Квалитет појединачне јединке се мери израчунавањем функције циља. За познато S, процес израчунавања функције циља је директно повезан са дефиницијом MDMBP кроз формуле (2.1) и (2.2). Треба поменути да у процесу рачунања (2.2) нема потребе за чувањем $\sum_{\{i,j\}\in C(S)} w_{\{i,j\}l}$ за свако $l = 1, \ldots, r$. Ипак, корисно је запамтити низ ових сума, са циљем убрзања процедуре локалне претраге.

У предложеној имплементацији GA, популација у почетној генерацији је генерисана на случајан начин и има $N_{pop} = 150$ јединки. Све наредне генерације се састоје од елитних и не-елитних јединки. Број не-елитних јединки, означен са N_{nonel}, је постављен на 50 јединки, и то је број јединки који ће бити мењан у свакој генерацији, док ће најбољих $N_{elite} = 100$ јединки директно прећи у следећу генерацију омогућавајући очување добрих јединки и њихових гена. Максимални број јединки са истом вредности функције циља али другачијим генетским кодом N_{rv} у овом случају је постављен на 40.

2.3.6 Кеширање GA

Кеширање је широко распрострањена техника која се користи у рачунарству за брз приступ меморијским модулима и/или дисковима. Кеширање GA је техника која се користи да се избегне поновно израчунавање функције циља за јединке које нису елитне, а које су се већ појављивале у процесу рада GA. Као што је показано у [37], техника кеширања може значајно да унапреди укупне перформансе GA.

Израчунате вредности функције циља су смештене у посебну структуру података којом се управља коришћењем стратегије најстаријег коришћеног члана (енг. Least Recently Used). Када се исти код појави поново, уместо поновног израчунавања, вредност се само преузима из структуре података. Да би се обезбедила ефикасност, структура података је заснована на комбинованој структури података хеш табела + ред (енг. hash-queue table). Број кешираних вредности функције циља је ограничен на $N_{cache} = 5000$ у овој GA имплементацији.

2.4 Метода променљивих околина

У овом поглављу ће бити приказана метода променљивих околина за решавање MDMBP. Предложени VNS алгоритам је приказан Алгоритмом 2.

2.4.1 Иницијализација и функција циља

У VNS за решавање MDMBP скуп S је представљен као комбинација од n елемената класе $n/2$. То је имплементирано као низ дужине $n/2$, чији су елементи цели бројеви из скупа $\{1, 2, \ldots, n\}$ тако да су сви међусобно различити.

Решење у почетној итерацији се генерише на случајан начин. Први елемент се генерише случајним избором једног од n елемената, други случајним избором од једног од преосталих $n-1$ елемената, док се последњи елемент на месту $n/2$ генерише случајним избором једног од преосталих $n/2+1$ елемената. Решење x је у ствари горепоменути низ који је представљен скупом S који одговара датом решењу. Функција циља се израчунава директно, на основу дефиниција MDMBP 2.1 и 2.2.

Algorithm 2: псеудо код VNS

```
   Algorithm Function VNS(k_min , k_max , t_tot, p_move) ;
1  x ← Random_Init();
2  k ← k_min ;
   while t < t_tot do
3  |  x' ← Shaking(x, k);
4  |  x'' ← Local_Search(x');
   |  if Compare(x, x'', p_move) then
5  |  |  x ← x'';
6  |  |  k ← k_min ;
   |  end
   |  else
   |  |  if k < k_max then
7  |  |  |  k ← k + 1;
   |  |  end
   |  end
   end
```

Algorithm 3: Псеудо код функције Compare

```
   Algorithm Function Compare(x, x'', p_move) ;
   if Obj(x'') > Obj(x) then
1  |  return true;
   end
   else
   |  if Obj(x'') < Obj(x) then
2  |  |  return false;
   |  end
   |  else
3  |  |  return Random(0, 1) < p_move ;
   |  end
   end
```

2.4.2 Систем околина

За две ниске истих дужина, Хамингово растојање се дефинише као број позиција на којима се дате ниске разликују. Систем околина N_k за проблем MDMBP се дефинише као скуп комбинација од n елемената класе $n/2$ где се свака од њих на разликују од x у тачно k елемената, односно важи $x' \in N_k(x)$ уколико је Хамингово растојање x и x' једнако k. Једноставније речено, x и x' имају тачно k различитих елемената. Приметимо да сваки елемент околине $N_k(x)$ представља једно допустиво решење проблема. Пошто је у овој имплементацији $k < n/2$, јасно је да је $|N_{k-1}(x)| = \binom{n}{k-1} \leq |N_k(x)| = \binom{n}{k}$. На тај начин је задовољено својство да околине имају растућу кардиналност. Напоменимо да ово својство није неопходно за примену VNS, али је пожељно. Функција $Shaking(x, k)$ на случајан начин одређује и враћа једно решење из околине $N_k(x)$.

У овој имплементацији VNS, коришћене су следеће вредности параметара: $k_{min} = 2$, $k_{max} = \min\{80, n/2\}$ и $p_{move} = 0.2$. Дате вредности обезбеђују да за велике димензије графова величина околине k, односно број замена, не прелази број 80. Прелиминарни експериментални резултати на великим инстанцама су указали да VNS даје довољно добре резултате уколико се узме да је $k_{max} = 80$. Вредност параметра p_{move} је вероватноћа да се пређе у друго решење чија је вредност иста као и вредност тренутног решења.

2.4.3 Локална претрага

Због постојања две функције локалне претраге $LS1$ i $LS2$, а у циљу њиховог што ефикаснијег обједињавања, у VNS алгоритму се примењује стратегија спуштања по променљивим околинама (енг. Variable Neighborhood Descent - VND). Предложена VND стратегија је приказана у Алгоритму 4, а у основном VNS алгоритму се примењује у оквиру функције $Local\_Search(x')$. Напоменимо да, пошто сви елементи из околине представљају допустива решења, за њих се налази оно које има највећу вредност функције циља. Основна претпоставка ове стратегије је да су VND околине обавезно неопадајуће (пожељно растуће) кардиналности. Пошто се претраживање по VND околинама обавља систематски (претражују се у целости), на тај начин се применом VND стратегије значајно може побољшати ефикасност целе VNS имплементације. Напоменимо и да су околине код VND независне и могу бити потпуно другачије у односу на

околине VNS. Да би избегли двосмисленост, за њих ћемо користити назив VND околине, док ћемо за VNS околине писати само околине.

Algorithm 4: VND псеудо код

```
1 l ← 1;
  while l < l_max do
2 |  x' ← FindBestSolution(N_l(x));
3 |  if Obj(x') > Obj(x) then
  |  |  x ← x';
  |  |  l ← 1;
  |  end
  |  else
  |  |  l ← l + 1
  |  end
  end
4 return x;
```

У овој имплементацији је $l_{max} = 2$, пошто имамо две процедуре локалног претраживања. VND околина N_1 одговара процедури $LS1$, док N_2 одговара процедури $LS2$. Имајући у виду да се $LS2$ извршава много спорије у односу на $LS1$, али је и многоструко веће кардиналности, то погодује примени VND стратегије. Напоменимо још да су $LS1$ и $LS2$ потпуно исте као у GA, па их није потребно поново описивати, али и да је учестаност њиховог позивања другачија у односу на GA пошто су они у VNS методи примењене у оквиру VND стратегије.

2.5 Метахеуристика заснована на електромагнетизму

У овом поглављу ће бити приказана метахеуристика заснована на електромагнетизму за решавање MDMBP. Предложени EM алгоритам је приказан Алгоритмом 5.

Као што је речено у претходном поглављу, свака EM тачка представља реални вектор који одговара неком решењу датог проблема. У овој имплементацији, све EM тачке су у првој итерацији генерисане на случајан начин из скупа $[0,1]^{|V|}$ (функција Random_Init()). Свака тачка p_k је вектор са $|V|$ координата p_{ks} где је $s = 1, \ldots, |V|$. Елементи вектора p_k су поређани у нерастућем редоследу,

Algorithm 5: EM псеудо код

```
   Algorithm EM ;
 1 Random_Init() ;
 2 iter ← 0;
   while iter < max_iter do
 3 |   iter ← iter + 1;
   |   foreach point p_k in solution_set do
 4 |   |   Objective_function(p_k);
 5 |   |   LS1(p_k);
 6 |   |   if k = 1 then
   |   |   |   LS2(p_k);
   |   |   end
 7 |   |   Scale_solution(p_k);
 8 |   end
 9 |   Compute_charges();
10 |   Compute_forces();
11 |   Move();
   end
```

тј. $p_{ks_1} \geq p_{ks_2} \geq \ldots \geq p_{ks_{|V|-1}} \geq p_{ks_{|V|}}$. Тада дефинишемо скуп

$$S_k = \{p_{ks_1}, \ldots, p_{ks_{|V|/2}}\},$$

којим је одређена бисекција која одговора тачки p_k. У овој имплементацији EM, број EM тачака је постављен на $m = 20$.

Када је дефинисан скуп S, квалитет одређених EM тачака се мери израчунавањем његове функције циља на исти начин као и код GA. Процедуре локалне претраге $LS1$ у $LS2$ су потпуно исте као у случају GA, али се примењују нешто другачије. Начин примене се може видети у алгоритму 5.

2.5.1 Скалирање

Првобитна намена EM је била у налажењу квалитетних решења нелинеарних проблема глобалне оптимизације [6]. У тој верзији EM, за усмеравање у обећавајуће регионе претраге је коришћен основни принцип EM, механизам привлачења/одбијања. У том случају бољи избор није постојао.

У случајевима када се користи процедура локалног претраживања и/или када се проблем дискретне оптимизације решава помоћу EM свођењем на проблем глобалне оптимизације, постоји додатна метода за усмеравање претраге.

Та процедура се назива скалирање ЕМ [17, 38, 36] и покушава да реални вектор p_k скалира ка дискретном или континуалном решењу добијеног дискретизацијом и локалном претрагом (или само једном од њих ако нису примењене обе).

Процедура скалирања, прилагођена за решавање MDMBP се примењује коришћењем следеће формуле

$$p_k \leftarrow \lambda \cdot p'_k + (1 - \lambda) \cdot p_k \tag{2.15}$$

где је p'_k одређено са $p'_{ks} = \begin{cases} 1, & s \in S \\ 0, & s \in V \setminus S \end{cases}$ и $s \in \{1, ..., |V|\}$. Дати скуп S је резултат процедуре локалне претраге. Да би решење p_k било допустиво фактор λ мора припадати интервалу $[0, 1]$.

Избор одговарајућег фактора скалирања λ је значајан за управљање процесом претраге. У екстремном случају, када је λ близу 1, процес претраге ће вероватно упасти у локални оптимум и остати заробљен. Други екстремни случај, када је λ једнако 0, очигледно представља ситуацију без скалирања. Експерименти су показали да је $\lambda = 0.1$ добар компромис који даје задовољавајуће резултате.

2.5.2 Механизам привлачења/одбијања

Коришћењем механизма привлачења/одбијања, алгоритам усмерава све ЕМ тачке ка тачки која представља најбоље решење у текућој итерацији.

Као што је речено у уводу, свака ЕМ тачка се посматра као наелектрисана честица, где добра решења привлаче сва остала, а лоша их одбијају. Математички, наелектрисање се израчунава по формули:

$$q_i = \exp\left(-n \frac{Obj_{best} - Obj_i}{\sum_{k=1}^{m} (Obj_{best} - Obj_k)}\right) \tag{2.16}$$

Ознака Obj_i означава вредност функције циља за i-ту ЕМ тачку, након примене локалне претраге.

По принципу суперпозиције теорије електростатичког поља, сила која делује у некој тачки од стране друге тачке је обрнуто пропорционална квадрату удаљености између тачака и директно пропорционална њиховом производу наелектрисања.

Математички, сила привлачења или одбијања нелектрисања се израчунава на следећи начин:

$$F_i = \sum_{j=1, j\neq i}^{m} F_i^j, \text{ где је}$$
$$F_i^j = \begin{cases} \left(\frac{q_i q_j}{||p_j - p_i||^2}\right) \cdot (p_j - p_i), & Obj_j > Obj_i \\ \left(\frac{q_i q_j}{||p_j - p_i||^2}\right) \cdot (p_i - p_j), & Obj_j \leq Obj_i \end{cases} \tag{2.17}$$

где $\|p_i - p_j\|$ представља Еуклидско растојање између ЕМ тачака p_i и p_j.

Затим се тренутна решења померају ка најбољем коришћењем процедуре Move(), као у Алгоритму 6. Приметимо да је само тренутно најбоље решење стационарно, односно не помера се овом процедуром.

Algorithm 6: Move pseudo code

```
    Function Move() ;
    foreach point p_k in solution_set do
        if p_k ≠ p_best then
1           β ← Random[0, 1];
2           F_k ← F_k/‖F_k‖ ;
3           for s = 1 to |V| do
4               if F_ks > 0 then
                    p_ks ← p_ks + β · F_ks · (1 − p_ks)
                end
5               else
                    p_ks ← p_ks + β · F_ks · p_ks
                end
            end
        end
6   end
```

Као што се може видети из Алгоритма 6, померање сваке ЕМ тачке је у правцу и смеру укупне силе која делује на њу, а интензитет је једнак случајном броју β који се узима из интервала [0,1]. Вредност β се узима на случајан начин да се не би смањила разноликост решења ЕМ, а вредност мора бити мања или једнака 1, да не би напустили скуп допустивих решења.

2.6 Експериментални резултати

У претходним поглављима су изложени како MILP формулација тако и метахеуристички приступи решавања MDMBP. Ово нам омогућава да решавању

различитих инстанци овог проблема приступимо из два правца. Један правац су егзактне методе инкорпориране у оквиру програмских пакета CPLEX [33] и Gurobi [25]. Поред наведених програмских пакета, имплементирана је и егзактна метода тоталне енумерације, где се испитују све бисекције скупа чворова V и памти бисекција са највећом тежином пресека. Они ће нам омогућити да добијемо или оптимална решења или добре процене граница оптималног решења што зависи од дужине времена извршавања. Како се овде ради о NP-тешком проблему, јасно је да егзактне методе могу добити оптимална решења и верификовати њихову оптималност само за инстанце мање димензије. Због тога је потребно извршавати програмске пакете CPLEX и Gurobi применом предложене формулације са ограниченим временом извршавања, као и метахеуристике GA, VNS и EM, за добијање приближних решења на инстанцама велике димензије. Ова два приступа резултатски надопуњују једни друге. За инстанце малих димензија, где се зна оптимално решење, може се тачно оценити квалитет решења метахеуристичких метода, док се за инстанце великих димензија не може оценити квалитет решења метахеуристика у односу на оптимално. Ипак, резултати метахеуристика се могу директно поредити са приближним решењима добијеним уз помоћ CPLEX и Gurobi решавача.

Сви експериментални резултати у овом раду су добијени извршавањем на AMD FX8300, 3.4GHz PC рачунару са 4GB RAM коришћењем само једног језгра. Сви алгоритми су имплементирани у програмском језику C, док су резултати извршавања комерцијалних решавача добијени коришћењем CPLEX 12.6 и Gurobi 5.6.

2.6.1 Експериментални резултати егзактних решавача

Експерименти у овом и следећем поглављу су вршени на скупу од 27 случајно генерисаних графова са различитим бројем чворова и грана, предложених у [43]. Конкретно, разматране су следеће инстанце: графови са 10 чворова и 15, 25 и 40 грана; графови са 20 чворова и 30, 70 и 150 грана; графови са 30 чворова и 50, 150 и 400 грана; графови са 50 чворова и 80, 300 и 1000 грана; графови са 100 чворова и 150, 500 и 3000 грана; графови са 300 чворова и 500, 2000, 10000 и 30000 грана; графови са 500 чворова и 1000, 3000, 10000 и 60000 грана и графови са 1000 чворова и 1500, 10000, 100000 и 350000 грана. У датим

инстанцама постоје подаци где су вектори тежина грана до највише $r = 20$. Експеримент је вршен са димензијама вектора тежина 1, 2, 3, 4, 5, 10, 15 и 20. Сви тестови коришћењем егзактних решавача у овом раду су извршавани са временским ограничењем од 7200 секунди.

Нумерички резултати за инстанце где је оптимално решење добијено су приказани у табелама 2.1−2.5. У табелама 2.6−2.9 дати су нумерички резултати за инстанце где ниједан решавач није успео да заврши свој рад и верификује оптималност добијеног решења. Напоменимо да, у већини случајева инстанци великих димензија, добијена неверификована решења сигурно нису оптимална, пошто су добијена боља решења помоћу неке од метахеуристика.

У свим табелама, значење прве две колоне је исто. У првој колони обележеној са *inst.*, дати су називи инстанци у облику nnn_mmm где је nnn број чворова и mmm број грана. На пример, инстанца 030_400 је граф са 30 чворова и 400 грана. Ако граф има више од хиљаду грана користи се скраћеница k која означава хиљаде, па тако инстанца 500_60k има 500 чворова и 60 хиљада грана. У другој колони, означеној са r, дата је димензија вектора тежина грана.

У табелама $2.1 - 2.5$, у трећој колони, означеној са *opt.*, дата је оптимална вредност. Наредне две колоне садрже информације у вези тоталне енумерације: време када је оптимално решење достигнуто (t) и укупно време извршавања (t_{tot}), у секундама. Последње четири колоне садрже исте информације о временима потребним да CPLEX и Gurobi достигну оптимално решење, односно укупним временима извршавања.

Пошто табеле 2.6 -2.9 садрже податке за инстанце на којима ниједан решавач није верификовао оптимално решења, у трећој колони табеле су наведени максимуми решења добијених егзактним методама, уз временско ограничење од 7200 секунди. У наредне две колоне табела $2.6 - 2.9$ дати су вредност решења (означена са *vred.*) и време неопходно за проналажење тог решења (t) коришћењем тоталне енумерације. Наредне четири колоне садрже исте информације о резултатима рада CPLEX и Gurobi решавача.

Напоменимо да знак "-" у табелама 2.3 - 2.5 означава да одговарајућа метода или није успела да верификује оптималност решења за дату инстанцу, или уопште није достигла оптимално решење. У табелама 2.6-2.9 ниједна метода није верификовала оптималност у временском оквиру од 7200 секунди, па су

приказане неверификоване вредности. Због тога "-" у табелама 2.6 - 2.9 означава да одговарајућа метода уопште није могла да генерише било какво допустиво решење, у временском оквиру од 7200 секунди, најчешће због недостатка меморије.

Табела 2.1: Егзактни резултати решавача на инстанцама $n = 10$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)
010_015	1	68.708	0.001	0.001	0.051	0.155	0.030	0.030
010_015	2	59.971	0.001	0.001	0.046	0.062	0.030	0.030
010_015	3	54.324	0.001	0.001	0.066	0.088	0.040	0.040
010_015	4	54.324	0.001	0.001	0.091	0.121	0.040	0.040
010_015	5	54.324	0.001	0.001	0.085	0.212	0.050	0.050
010_015	10	49.011	0.001	0.001	0.263	0.269	0.050	0.050
010_015	15	49.011	0.001	0.001	0.134	0.141	0.050	0.050
010_015	20	47.347	0.001	0.001	0.165	0.239	0.060	0.060
010_025	1	82.500	0.001	0.001	0.001	0.265	0.030	0.030
010_025	2	82.500	0.001	0.001	0.183	0.187	0.050	0.050
010_025	3	82.500	0.001	0.001	0.236	0.247	0.030	0.030
010_025	4	82.500	0.001	0.001	0.240	0.245	0.030	0.030
010_025	5	82.500	0.001	0.001	0.357	0.374	0.070	0.070
010_025	10	79.842	0.001	0.001	0.196	0.204	0.070	0.070
010_025	15	79.842	0.001	0.001	0.199	0.243	0.070	0.070
010_025	20	79.842	0.001	0.001	0.256	0.263	0.100	0.100
010_040	1	109.743	0.001	0.002	0.185	0.403	0.060	0.060
010_040	2	109.743	0.001	0.001	0.295	0.312	0.070	0.070
010_040	3	109.743	0.001	0.001	0.381	0.391	0.110	0.110
010_040	4	109.743	0.001	0.001	0.377	0.386	0.080	0.080
010_040	5	109.743	0.001	0.001	0.450	0.493	0.110	0.110
010_040	10	109.743	0.001	0.001	0.525	0.538	0.120	0.120
010_040	15	109.743	0.001	0.001	0.415	0.427	0.110	0.110
010_040	20	108.651	0.001	0.001	0.486	0.500	0.160	0.160

Табела 2.2: Егзактни резултати решавача на инстанцама $n = 20$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)
020_030	1	136.696	0.016	0.078	0.068	0.227	0.030	0.030
020_030	2	122.664	0.031	0.078	0.105	0.109	0.040	0.040
020_030	3	122.664	0.031	0.078	0.047	0.194	0.040	0.040
020_030	4	107.134	0.015	0.093	0.214	0.220	0.050	0.050
020_030	5	105.441	0.046	0.109	0.267	0.276	0.090	0.090
020_030	10	105.441	0.062	0.124	0.290	0.298	0.110	0.110
020_030	15	105.441	0.062	0.140	0.196	0.203	0.090	0.090
020_030	20	105.441	0.475	0.171	0.249	0.258	0.090	0.090
020_070	1	250.973	0.016	0.156	0.318	0.620	0.120	0.120
020_070	2	250.973	0.001	0.171	0.425	0.437	0.160	0.160
020_070	3	246.420	0.001	0.187	0.559	0.583	0.190	0.190
020_070	4	246.420	0.001	0.203	0.571	0.588	0.180	0.180
020_070	5	246.420	0.015	0.218	0.639	0.659	0.190	0.190
020_070	10	244.988	0.001	0.265	0.704	0.725	0.240	0.240
020_070	15	244.988	0.015	0.312	0.578	0.619	0.270	0.270
020_070	20	244.988	0.015	0.374	0.808	0.835	0.250	0.250
020_150	1	502.411	0.031	0.359	1.769	2.210	0.840	0.840
020_150	2	476.472	0.187	0.374	1.476	1.529	1.570	1.570
020_150	3	466.079	0.187	0.421	1.193	2.505	2.670	1.000
020_150	4	466.079	0.202	0.421	2.201	2.832	2.350	2.350
020_150	5	455.679	0.171	0.452	4.213	5.076	2	6.760
020_150	10	442.486	0.202	0.561	1.912	3.387	1	2.190
020_150	15	440.870	0.234	0.671	2.353	2.963	1	5.870
020_150	20	440.870	0.296	0.811	3.751	3.948	1	3.070

Очигледно, број чворова и грана значајно утиче на перформансе одређеног решавача. На пример, за инстанцу 030_050 са вектором димензије 1 и CPLEX и Gurobi су завршили тражење оптималног решења за мање од једне секунде, док је за проналажење оптималног решења инстанце 030_400 било потребно 775.5 односно 193.2 секунди. Резултати, дати у табелама 2.1 – 2.9, такође указују да се временска сложеност углавном увећава са повећањем димензије вектора тежина. На пример, за CPLEX решавач и за инстанцу 050_300 са димензијом вектора тежине $r = 1$, $t_{tot} = 23.89$ секунди, и расте до $t_{tot} = 295.3$ секунде за $r = 15$. Ипак, приметимо да време извршавања на истој инстанци у функцији вредности r није строго растућа функција. На пример, за инстанцу 050_300 са димензијама вектора тежине $r = 15$ и $r = 20$ укупна времена извршавања су 295.3 и 236.9 секунди, респективно.

Табела 2.3: Егзактни резултати решавача на инстанцама $n = 30$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)
030_050	1	224.556	47.41	117.6	0.194	0.194	0.050	0.050
030_050	2	224.556	49.82	123.6	0.134	0.141	0.080	0.080
030_050	3	221.761	54.74	136.3	0.183	0.193	0.110	0.110
030_050	4	221.761	54.24	145.1	0.164	0.173	0.090	0.090
030_050	5	214.010	158.1	151.5	0.496	0.511	0.110	0.110
030_050	10	207.983	86.80	198.9	0.382	0.396	0.170	0.170
030_050	15	199.817	92.96	233.2	0.324	0.337	0.170	0.170
030_050	20	199.817	111.7	279.3	0.472	0.488	0.160	0.160
030_150	1	589.593	118.8	345.4	1.610	1.676	0.940	0.940
030_150	2	536.473	130.9	374.0	1.230	1.264	1.050	1.000
030_150	3	533.635	141.8	404.8	3.122	3.223	1.590	1.000
030_150	4	533.635	150.8	433.5	1.608	2.500	1.480	1.000
030_150	5	533.635	156.8	449.4	2.635	2.708	1.270	1.270
030_150	10	525.300	89.9	573.0	3.150	3.238	2	3.490
030_150	15	519.586	106.5	936.8	3.437	4.837	1	4.470
030_150	20	519.586	216.7	1375	3.969	4.641	1	3.120
030_400	1	1331.773	391.5	1327	15.08	775.5	59	193.2
030_400	2	1230.856	539.2	1430	26.87	1928	167	1249
030_400	3	1208.703	125.0	1416	498.5	4033	2293	2889
030_400	4	1187.753	530.8	1520	5138	5382	4389	5527
030_400	5	1181.987	45.5	1602	5521	5563	-	-
030_400	10	1154.353	593.1	2116	-	-	-	-
030_400	15	1142.317	519.2	2611	-	-	-	-
030_400	20	1140.635	605.3	3098	-	-	-	-

2.6.2 Експериментални резултати добијени метахеуристикама

У овом одељку ће бити представљени експериментални резултати добијени уз помоћ предложених метахеуристика: GA, EM и VNS.

Пошто све три примењене метахеуристике имају недетерминистичко понашање, које зависи од почетног стања генератора случајних бројева, свака од њих је извршавана по 20 пута, са 20 различитих почетних вредности генератора случајних бројева (енг. random seed). Треба напоменути да је исти низ од тих 20 почетних вредности коришћен за сваку од метахеуристика. При томе, за сваку од метахеуристика, је бележена вредност најбољег добијеног решења, укупно време извршавања и време потребно да се дође до добијеног резултата.

Параметри примењене GA метахеуристике за овај проблем су: критеријум заустављања је максимални број генерација једнак 5000 или да максимални број генерација без побољшања не прелази 3000. Критеријум заустављања EM и VNS је максимално време једнако просечном укупном времену извршавања

Табела 2.4: Егзактни резултати решавача на инстанцама $n = 50$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)
050_080	1	372.069	-	-	0.396	0.407	0150	0150
050_080	2	346.271	-	-	0.475	0.496	0150	0150
050_080	3	346.271	-	-	0.491	0.513	0.130	0.130
050_080	4	346.271	-	-	0.446	0.465	0.180	0.180
050_080	5	333.258	-	-	0.697	0.720	0.200	0.200
050_080	10	333.258	-	-	0.615	0.632	0.250	0.250
050_080	15	332.970	-	-	0.575	0.634	0.250	0.250
050_080	20	332.970	-	-	0.882	0.911	0.250	0.250
050_300	1	1124.331	-	-	7.828	23.89	15	73.58
050_300	2	1098.891	-	-	5.292	22.85	46	60.92
050_300	3	1098.891	-	-	12.82	31.82	7	60.87
050_300	4	1094.621	-	-	51.75	117.4	21	74.23
050_300	5	1094.621	-	-	9.152	27.05	37	89.12
050_300	10	1076.105	-	-	17.71	32.11	7	75.70
050_300	15	1062.553	-	-	29.32	295.3	45	107.30
050_300	20	1062.553	-	-	63.49	236.9	89	163.70

GA (t_{tot}), за дату инстанцу. Таквим критеријумом заустављања за EM и VNS, све три метахеуристике на располагању су имале исту количину времена за одговарајућу инстанцу, те онда можемо директно оценити квалитет метахеуристика на основу квалитета решења.

Табеле 2.10 - 2.13 садрже податке о експерименталним подацима за инстанце чије је оптимално решење познато (претходног поглавља). Као и раније, прве две колоне садрже име инстанце и вредност r. Трећа колона, означена са *opt.*, садржи одговарајућу оптималну вредност. Следеће три колоне представљају резултате извршавања GA. Четврта колона је обележена са *najb.* и садржи најбоље резултате добијене коришћењем GA, са записом opt ако је оптимално решење достигнуто. У петој и шестој колони, означеним са t и t_{tot}, су записана просечна времена извршавања и просечна укупна времена извршавања. Наредне четири колоне су коришћене за представљање резултата извршавања EM и VNS метахеуристике, на сличан начин као и у случају GA. Треба напоменути да за метахеуристике EM и VNS није било неопходно бележити укупно просечно време извршавања зато што је оно исто као и код GA, због њиховог критеријума завршетка.

Табеле 2.14-2.17 садрже експерименталне резултате за инстанце са непознатима оптималним решењима. Значење прве две колоне је исто као и код претходних табела. Трећа колона, означена са *preth.*, садржи најбоља решења добијена помоћу егзактних решавача, преузета из табела 2.6-2.9. Четврта колона

Табела 2.5: Егзактни резултати решавача на инстанцама $100 \leq n \leq 300$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)
100_150	1	697.973	-	-	0.586	0.612	0.350	0.350
100_150	2	696.787	-	-	0.750	1.106	0.520	0.520
100_150	3	689.403	-	-	1.082	1.124	0.690	0.690
100_150	4	689.403	-	-	0.843	1.126	0.650	0.650
100_150	5	689.403	-	-	1.425	1.425	1.090	1.090
100_150	10	686.703	-	-	1.873	1.945	1.000	1.000
100_150	15	673.056	-	-	1.343	1.343	1.570	1.570
100_150	20	655.263	-	-	1.678	2.410	0.930	0.930
100_500	1	1986.131	-	-	-	-	300	1495
100_500	2	1931.452	-	-	292.8	4481	45	1514
100_500	3	1901.654	-	-	-	-	1104	1998
100_500	4	1901.654	-	-	-	-	1435	2842
100_500	5	1876.418	-	-	-	-	675	3902
300_500	1	2543.862	-	-	13.99	52.52	106	127.40
300_500	2	2524.215	-	-	263.5	265.9	55	279.66
300_500	3	2456.085	-	-	102.5	3455	450	460.41
300_500	4	2456.085	-	-	382.3	1264	151	961.72
300_500	5	2448.516	-	-	116.7	1939	167	752.36
300_500	10	2377.530	-	-	280.0	537.5	29	287.21
300_500	15	2360.732	-	-	591.4	636.5	67	654.83
300_500	20	2352.357	-	-	40.98	1766	101	692.72

садржи ново најбоље решење добијено уз помоћ метахеуристика GA, EM и VNS. Уколико ниједна од три примењене метахеуристике није поправила претходно решење (приказано у трећој колони), у овој колони није приказан никакав податак. Наредних седам колона имају исто значење као и одговарајуће колоне у табелама 2.10-2.13.

2.7 Анализа резултата и завршна разматрања

Као што се може видети из табеле 2.5, оптимална решења су добијена само за ретке графове до 300 чворова и 500 грана и густе графове до само 30 чворова. Из тог разлога је оправдана потреба за коришћењем метахеуристичких метода за решавање овог проблема.

Резултати из табела 2.10-2.13 показују да су, од укупно 109 инстанци са познатим оптималним решењима, GA и VNS достигли 103, а EM је достигао 95 оптималних решења. Укупно време извршавања је било мање од 14 секунди. На тим инстанцама је VNS метахеуристика најбрже дошла до оптималног решења у 87 случајева, док је GA најбрже дошла до оптималног решења у 24 случаја.

Као што се може видети из табела 2.14 - 2.17, од укупно 107 инстанци где

Табела 2.6: Приближни резултати решавача на инстанцама $50 \leq n \leq 100$

inst.	r	najb.	enum.		CPLEX		Gurobi	
			vred.	t (s)	vred.	t (s)	vred.	t (s)
050_1000	1	3099.083	2914.536	5125	3089.713	1531	3099.083	2404
050_1000	2	3016.320	2874.113	1368	3016.320	5257	3013.999	1354
050_1000	3	2977.524	2861.193	2159	2977.524	650.3	2973.205	6592
050_1000	4	2930.936	2826.714	2475	2930.936	3794	2928.399	2284
050_1000	5	2915.139	2821.667	2586	2915.139	1636	2909.632	3419
050_1000	10	2890.945	2775.508	295.9	2890.945	898.6	2827.708	1658
050_1000	15	2853.382	2762.585	3324	2853.382	1401	2833.230	4129
050_1000	20	2840.953	2762.585	4042	2840.953	2392	2827.708	1658
100_500	10	1852.608	1522.286	3768	1843.131	1572	1852.608	5504
100_500	15	1834.379	1480.356	4748	1834.379	223.3	1834.379	392
100_500	20	1809.037	1480.356	5857	1809.037	607.1	1809.037	4827
100_3000	1	8874.360	8505.179	5089	8874.360	6230	8864.220	2720
100_3000	2	8718.983	8436.693	5774	8718.983	5946	8674.793	2500
100_3000	3	8718.983	8386.485	5289	8656.097	7086	8718.983	3267
100_3000	4	8709.104	8386.485	5692	8709.104	421.0	8699.536	30
100_3000	5	8698.801	8296.34	5703	8583.056	219.3	8698.801	3170
100_3000	10	8709.173	8171.752	505.9	8440.079	142.0	8709.173	4914
100_3000	15	8402.444	8117.397	778.9	8402.444	267.7	8375.000	3770
100_3000	20	8521.629	8048.049	4098	8464.797	266.4	8521.629	4894

није познато оптимално решење, метахеуристике су у 91 случају побољшале најбоља приближна решења добијена помоћу егзактних решавача. Од тога је GA добио 86 нових најбољих решења, VNS 25, а EM 14. За инстанце са 50 до 100 чворова за које није познато оптимално решење, GA је најбрже дошао до најбољег решења у 16 случајева, док је VNS најбрже дошао до најбољег решења у 4 случаја. За инстанце са 300 чворова за које није познато оптимално решење, GA достиже најбоље решење за најмање времена у 19 случајева, док EM достиже најбоље решење само за инстанцу 300_$10k$ и $r = 1$. За инстанце са 500 чворова, за које није познато оптимално решење, GA достиже најбоље решење за 23 инстанце, а VNS само за једну. За инстанце са 1000 чворова за које није познато оптимално решење, једино GA достиже најбоље решење за све 24 инстанце.

Напоменимо да су у 14 случајева све три метахеуристике добиле иста, нова најбоља решења.

Време извршавања метахеуристика је углавном достижно, најчешће мање од 10 минута, осим инстанци 500_$60k$ $1k$_$100k$ и $1k$_$350k$, који представљају врло густе графове велике димензије.

Представљени експериментални резултати јасно демонстрирају корисност предложеног MILP модела, и GA, EM и VNS метахеуристика за решавање

Табела 2.7: Приближни резултати решавача на инстанцама $n = 300$

inst.	r	najb.	enum.		CPLEX		Gurobi	
			vred.	t (s)	vred.	t (s)	vred.	t (s)
300_2k	1	7709.498	5744.132	6999	7690.407	6322	7709.498	6637
300_2k	2	7645.349	5742.752	2774	7645.349	6487	7414.360	5032
300_2k	3	7509.420	5740.743	3139	7509.420	7197	7242.234	1234
300_2k	4	7442.000	5739.065	4386	7442.000	6745	7402.851	1412
300_2k	5	7389.467	5733.544	4731	7389.467	6871	7299.509	887
300_2k	10	7416.068	5733.544	7099	7416.068	7097	7168.137	2206
300_2k	15	7307.361	5679.214	198.9	7307.361	6595	7291.780	4513
300_2k	20	7345.519	5669.931	7136	7345.519	6059	7147.080	3663
300_10k	1	30824.693	27486.982	2801	30824.693	0.353	29951.782	33
300_10k	2	29659.475	27486.982	3249	29771.046	4288	29659.475	539
300_10k	3	30759.960	27472.383	3818	29819.923	6494	30759.960	336
300_10k	4	30989.566	27472.383	4302	29359.477	3087	30989.566	188
300_10k	5	30872.004	27246.909	1004	29682.394	4164	30872.004	80
300_10k	10	30968.334	27246.649	931.3	29253.722	6862	30968.334	747
300_10k	15	30748.223	27183.126	2194	29241.136	5092	30748.223	1299
300_10k	20	30923.953	27183.126	2729	29498.737	4891	30923.953	48
300_30k	1	85934.244	83070.887	6456	85934.244	1.225	84451.384	3
300_30k	2	83903.267	83028.62	6874	66065.103	5760	83903.267	7126
300_30k	3	84952.867	83021.191	5183	66814.125	5575	84952.867	1573
300_30k	4	85143.297	83021.191	5804	66514.511	5641	85143.297	4027
300_30k	5	85845.606	83028.62	6141	66007.823	6611	85845.606	3331
300_30k	10	85691.919	82945.149	3427	65971.340	5955	85691.919	2977
300_30k	15	85894.230	82768.105	6617	66204.979	6213	85894.230	2572
300_30k	20	85715.008	82735.728	6759	65252.157	7072	85715.008	3401

проблема MDMBP. Предложени MILP модел са $|V| + |E|$ бинарних променљивих и $r + 2|E| + 1$ ограничења, за који је доказана коректност, представља вредан допринос за егзактно и приближно решавање MDMBP. Експериментални резултати показују да GA даје значајно боље резултате у односу на EM и VNS.

Табела 2.8: Приближни резултати решавача на инстанцама $n = 500$

inst.	r	najb.	enum.		CPLEX		Gurobi	
			vred.	t (s)	vred.	t (s)	vred.	t (s)
500_1k	1	4744.994	2806.965	5913	4740.014	4102	4744.994	2801
500_1k	2	4739.982	2806.965	7054	4739.982	6815	4739.529	6620
500_1k	3	4695.510	2762.100	1590	4695.510	789.2	4692.877	4350
500_1k	4	4688.866	2762.100	1239	4688.866	5575	4680.997	6234
500_1k	5	4687.445	2762.100	1933	4680.664	2883	4687.445	2885
500_1k	10	4636.442	2762.100	2995	4636.442	2509	4623.145	5832
500_1k	15	4622.783	2762.100	3794	4622.783	6846	4619.915	5316
500_1k	20	4602.182	2713.144	3218	4602.182	3253	4589.822	4576
500_3k	1	11372.166	8358.183	7124	11372.166	474.4	11289.831	2687
500_3k	2	11293.525	8352.269	2402	11077.172	2835	11293.525	3663
500_3k	3	11257.063	8352.269	2774	11063.300	727.4	11257.063	3975
500_3k	4	11267.662	8352.269	2279	10974.964	782.7	11267.662	4074
500_3k	5	11214.219	8339.367	3179	10831.476	679.7	11214.219	4509
500_3k	10	10972.091	8339.367	4692	10799.479	912.4	10972.091	5591
500_3k	15	11159.572	8286.828	3631	10751.672	3379	11159.572	6946
500_3k	20	11038.550	8286.828	5905	10812.779	4230	11038.550	6971
500_10k	1	31847.822	27998.844	6100	31547.096	0.040	31847.822	1
500_10k	2	32453.585	27998.844	7183	31547.096	2676	32453.585	509
500_10k	3	32741.433	27986.956	437.3	30709.973	4326	32741.433	272
500_10k	4	32134.073	27602.152	3184	30710.467	3293	32134.073	12
500_10k	5	32261.153	27602.152	6980	30880.867	3453	32261.153	555
500_10k	10	32868.335	27602.152	6403	30610.289	5060	32868.335	416
500_10k	15	32579.889	27556.169	96.57	30226.088	3517	32579.889	479
500_10k	20	32660.868	27556.169	189.3	30007.443	4501	32660.868	919
500_60k	1	173626.906	165928.512	4365	173626.906	3.894	168687.246	79
500_60k	2	169984.258	165762.356	1814	-	-	169984.258	2000
500_60k	3	171399.820	165762.356	2057	-	-	171399.820	2937
500_60k	4	166948.978	165762.356	2351	-	-	166948.978	1678
500_60k	5	172710.669	165762.356	3584	-	-	172710.669	437
500_60k	10	165966.934	165615.397	3365	-	-	165966.934	144
500_60k	15	165512.248	165067.289	3315	-	-	165512.248	14
500_60k	20	166051.696	165067.289	6398	-	-	166051.696	4068

Табела 2.9: Приближни резултати решавача на инстанцама $n = 1000$

inst.	r	najb.	enum.		CPLEX		Gurobi	
			vred.	t (s)	vred.	t (s)	vred.	t (s)
1k_1.5k	1	7830.439	4429.553	5142	7830.439	4199	7830.439	7082
1k_1.5k	2	7660.702	4406.755	3049	7660.702	500.7	7659.381	5093
1k_1.5k	3	7561.257	4331.854	2753	7561.257	394.7	7549.120	5741
1k_1.5k	4	7552.921	4331.854	3761	7552.921	6665	7551.454	2691
1k_1.5k	5	7561.257	4331.854	4126	7561.257	6775	7558.126	4247
1k_1.5k	10	7499.702	4313.156	6082	7486.617	6695	7499.702	5871
1k_1.5k	15	7457.737	4212.573	2637	7457.737	6542	7448.530	5709
1k_1.5k	20	7432.879	4208.582	3862	7422.875	4892	7432.879	6725
1k_10k	1	34497.300	27849.420	4164	34497.300	3619	33468.985	49
1k_10k	2	33431.473	27453.988	3955	33431.473	2859	33308.086	6351
1k_10k	3	35076.961	27449.083	2319	33788.255	2971	35076.961	81
1k_10k	4	34828.276	27436.452	2905	33388.788	3102	34828.276	3937
1k_10k	5	35122.442	27436.452	3200	33158.065	3165	35122.442	1306
1k_10k	10	34552.295	27340.450	7001	33220.939	2952	34552.295	6163
1k_10k	15	34642.263	27327.217	6969	33157.350	4653	34642.263	3900
1k_10k	20	34789.795	27318.103	5733	32978.447	4106	34789.795	2139
1k_100k	1	294083.743	272658.632	3015	294083.743	1.156	284444.969	1254
1k_100k	2	277883.845	272658.632	3860	-	-	277883.845	2339
1k_100k	3	277770.801	272658.632	4468	-	-	277770.801	2931
1k_100k	4	287363.683	272627.483	6531	-	-	287363.683	6850
1k_100k	5	293827.978	272581.574	2464	-	-	293827.978	6160
1k_100k	10	276657.467	272263.665	3562	-	-	276657.467	2569
1k_100k	15	291638.645	272127.084	3985	-	-	291638.645	5470
1k_100k	20	294052.633	272114.368	5228	-	-	294052.633	1382
1k_350k	1	986248.932	964947.365	5146	986248.932	30.42	965489.117	4131
1k_350k	2	963299.917	963270.571	6110	-	-	963299.917	1174
1k_350k	3	962372.761	962136.163	3520	-	-	962372.761	3720
1k_350k	4	965016.683	962136.163	3919	-	-	965016.683	1321
1k_350k	5	963881.119	962136.163	4313	-	-	963881.119	1551
1k_350k	10	962136.163	962136.163	6521	-	-	-	-
1k_350k	15	962052.538	962052.538	3701	-	-	-	-
1k_350k	20	962008.567	962008.567	2578	-	-	-	-

Табела 2.10: Резултати метахеуристика на инстанцама $n = 10$ где је познато оптимално решење

inst.	r	opt.	GA			EM		VNS	
			najb.	t (s)	t_{tot} (s)	najb.	t (s)	najb.	t (s)
010_015	1	68.708	opt	0.001	0.017	opt	0.002	opt	0.001
010_015	2	59.971	opt	0.001	0.017	opt	0.070	opt	0.001
010_015	3	54.324	opt	0.001	0.016	opt	0.080	opt	0.001
010_015	4	54.324	opt	0.001	0.018	opt	0.062	opt	0.001
010_015	5	54.324	opt	0.001	0.029	opt	0.071	opt	0.001
010_015	10	49.011	opt	0.001	0.020	opt	0.074	opt	0.002
010_015	15	49.011	opt	0.003	0.029	opt	0.074	opt	0.001
010_015	20	47.347	opt	0.003	0.030	opt	0.086	opt	0.001
010_025	1	82.500	opt	0.001	0.016	opt	0.084	opt	0.001
010_025	2	82.500	opt	0.001	0.012	opt	0.085	opt	0.001
010_025	3	82.500	opt	0.001	0.019	opt	0.036	opt	0.001
010_025	4	82.500	opt	0.001	0.021	opt	0.041	opt	0.001
010_025	5	82.500	opt	0.001	0.015	opt	0.042	opt	0.001
010_025	10	79.842	opt	0.002	0.025	opt	0.076	opt	0.001
010_025	15	79.842	opt	0.004	0.035	opt	0.047	opt	0.002
010_025	20	79.842	opt	0.005	0.041	opt	0.050	opt	0.001
010_040	1	109.743	opt	0.001	0.019	opt	0.028	opt	0.001
010_040	2	109.743	opt	0.001	0.022	opt	0.026	opt	0.001
010_040	3	109.743	opt	0.001	0.019	opt	0.032	opt	0.001
010_040	4	109.743	opt	0.001	0.023	opt	0.038	opt	0.001
010_040	5	109.743	opt	0.002	0.027	opt	0.009	opt	0.001
010_040	10	109.743	opt	0.004	0.031	opt	0.016	opt	0.001
010_040	15	109.743	opt	0.004	0.039	opt	0.010	opt	0.001
010_040	20	108.651	opt	0.005	0.043	opt	0.088	opt	0.001

Табела 2.11: Резултати метахеуристика на инстанцама $n = 20$ где је познато оптимално решење

inst.	r	opt.	GA			EM		VNS	
			najb.	t (s)	t_{tot} (s)	najb.	t (s)	najb.	t (s)
020_030	1	136.696	opt	0.001	0.035	opt	0.085	opt	0.001
020_030	2	122.664	opt	0.002	0.045	opt	0.040	opt	0.001
020_030	3	122.664	opt	0.003	0.048	opt	0.047	opt	0.001
020_030	4	107.134	opt	0.003	0.052	opt	0.020	opt	0.001
020_030	5	105.441	opt	0.005	0.055	opt	0.054	opt	0.002
020_030	10	105.441	opt	0.005	0.072	opt	0.055	opt	0.002
020_030	15	105.441	opt	0.005	0.084	opt	0.047	opt	0.001
020_030	20	105.441	opt	0.008	0.097	opt	0.057	opt	0.002
020_070	1	250.973	opt	0.001	0.043	opt	0.095	opt	0.001
020_070	2	250.973	opt	0.004	0.049	opt	0.096	opt	0.001
020_070	3	246.420	opt	0.004	0.058	opt	0.092	opt	0.001
020_070	4	246.420	opt	0.005	0.059	opt	0.094	opt	0.001
020_070	5	246.420	opt	0.005	0.06	opt	0.094	opt	0.001
020_070	10	244.988	opt	0.004	0.071	opt	0.059	opt	0.001
020_070	15	244.988	opt	0.008	0.098	opt	0.074	opt	0.002
020_070	20	244.988	opt	0.011	0.118	opt	0.062	opt	0.001
020_150	1	502.411	opt	0.004	0.053	opt	0.093	opt	0.001
020_150	2	476.472	opt	0.005	0.072	opt	0.082	opt	0.001
020_150	3	466.079	opt	0.004	0.087	opt	0.096	opt	0.002
020_150	4	466.079	opt	0.005	0.097	opt	0.095	opt	0.001
020_150	5	455.679	opt	0.005	0.087	opt	0.049	opt	0.002
020_150	10	442.486	opt	0.014	0.170	opt	0.148	opt	0.004
020_150	15	440.870	opt	0.017	0.186	opt	0.141	opt	0.004
020_150	20	440.870	opt	0.021	0.220	opt	0.184	opt	0.003

Табела 2.12: Резултати метахеуристика на инстанцама $n = 30$ где је познато оптимално решење

inst.	r	opt.	GA			EM		VNS	
			najb.	t (s)	t_{tot} (s)	najb.	t (s)	najb.	t (s)
030_050	1	224.556	opt	0.003	0.062	opt	0.022	opt	0.001
030_050	2	224.556	opt	0.004	0.068	opt	0.014	opt	0.002
030_050	3	221.761	opt	0.004	0.074	opt	0.033	opt	0.001
030_050	4	221.761	opt	0.004	0.074	opt	0.023	opt	0.004
030_050	5	214.010	opt	0.005	0.099	opt	0.032	opt	0.002
030_050	10	207.983	opt	0.009	0.113	opt	0.037	opt	0.001
030_050	15	199.817	opt	0.013	0.182	opt	0.085	opt	0.001
030_050	20	199.817	opt	0.013	0.205	opt	0.107	opt	0.005
030_150	1	589.593	opt	0.004	0.062	opt	0.033	opt	0.002
030_150	2	536.473	opt	0.005	0.094	opt	0.069	opt	0.004
030_150	3	533.635	opt	0.004	0.105	opt	0.035	opt	0.002
030_150	4	533.635	opt	0.008	0.115	opt	0.041	opt	0.003
030_150	5	533.635	opt	0.009	0.122	opt	0.033	opt	0.003
030_150	10	525.300	opt	0.013	0.189	opt	0.120	opt	0.004
030_150	15	519.586	opt	0.018	0.217	opt	0.036	opt	0.004
030_150	20	519.586	opt	0.025	0.288	opt	0.075	opt	0.006
030_400	1	1331.773	opt	0.006	0.151	opt	0.145	opt	0.001
030_400	2	1230.856	opt	0.009	0.174	opt	0.163	opt	0.003
030_400	3	1208.703	opt	0.022	0.160	opt	0.069	opt	0.006
030_400	4	1187.753	opt	0.014	0.178	opt	0.078	opt	0.004
030_400	5	1181.987	opt	0.020	0.197	opt	0.112	opt	0.006
030_400	10	1154.353	opt	0.033	0.290	opt	0.095	opt	0.007
030_400	15	1142.317	opt	0.048	0.533	opt	0.181	opt	0.015
030_400	20	1140.635	opt	0.061	0.527	opt	0.274	opt	0.019

Табела 2.13: Резултати метахеуристика на инстанцама $50 \leq n \leq 300$ где је познато оптимално решење

inst.	r	opt.	GA			EM		VNS	
			najb.	t (s)	t_{tot} (s)	najb.	t (s)	najb.	t (s)
050_080	1	372.069	opt	0.010	0.092	opt	0.012	opt	0.005
050_080	2	346.271	opt	0.005	0.148	opt	0.012	opt	0.004
050_080	3	346.271	opt	0.009	0.166	opt	0.009	opt	0.008
050_080	4	346.271	opt	0.009	0.199	opt	0.027	opt	0.005
050_080	5	333.258	opt	0.013	0.197	opt	0.029	opt	0.006
050_080	10	333.258	opt	0.021	0.253	opt	0.031	opt	0.010
050_080	15	332.970	opt	0.031	0.333	opt	0.086	opt	0.016
050_080	20	332.970	opt	0.042	0.409	opt	0.092	opt	0.017
050_300	1	1124.331	opt	0.005	0.158	opt	0.055	opt	0.010
050_300	2	1098.891	opt	0.012	0.231	opt	0.046	opt	0.010
050_300	3	1098.891	opt	0.014	0.271	opt	0.069	opt	0.013
050_300	4	1094.621	opt	0.017	0.336	opt	0.119	opt	0.009
050_300	5	1094.621	opt	0.021	0.367	opt	0.113	opt	0.015
050_300	10	1076.105	opt	0.036	0.642	opt	0.075	opt	0.022
050_300	15	1062.553	opt	0.055	0.694	opt	0.195	opt	0.026
050_300	20	1062.553	opt	0.073	0.894	opt	0.206	opt	0.042
100_150	1	697.973	opt	0.028	0.362	692.227	0.028	opt	0.037
100_150	2	696.787	opt	0.088	0.533	opt	0.067	opt	0.058
100_150	3	689.403	opt	0.131	0.645	686.703	0.064	opt	0.100
100_150	4	689.403	opt	0.133	0.681	686.703	0.074	opt	0.113
100_150	5	689.403	opt	0.203	0.806	686.703	0.046	opt	0.113
100_150	10	686.703	opt	0.241	1.088	opt	0.174	opt	0.173
100_150	15	673.056	opt	0.082	1.246	672.686	0.226	opt	0.146
100_150	20	655.263	opt	0.265	1.631	654.869	0.169	opt	0.286
100_500	1	1986.131	opt	0.032	0.493	opt	0.117	opt	0.111
100_500	2	1931.452	opt	0.024	0.722	opt	0.193	opt	0.093
100_500	3	1901.654	opt	0.042	0.802	opt	0.217	opt	0.143
100_500	4	1901.654	opt	0.056	0.883	opt	0.331	opt	0.195
100_500	5	1876.418	opt	0.077	0.994	opt	0.211	opt	0.207
300_500	1	2543.862	opt	0.839	3.088	2538.876	0.195	2542.756	1.689
300_500	2	2524.215	2521.305	1.930	5.236	2490.934	0.292	2521.305	2.973
300_500	3	2456.085	2454.950	3.043	6.837	2427.728	0.565	2451.659	2.247
300_500	4	2456.085	2455.162	1.947	6.413	2417.477	0.411	2453.503	2.755
300_500	5	2448.516	2447.802	1.042	6.002	2412.232	0.495	opt	2.840
300_500	10	2377.530	2375.956	2.793	9.242	2346.121	0.743	2373.107	3.328
300_500	15	2360.732	opt	4.691	12.30	2328.276	1.032	2357.859	4.400
300_500	20	2352.357	2349.655	4.359	13.99	2332.311	1.060	opt	5.659

Табела 2.14: Резултати метахеуристика на инстанцама $50 \leq n \leq 100$ где није познато оптимално решење

inst.	r	preth.	novo najb.	GA			EM		VNS	
				najb.	t (s)	$t_{tot}(s)$	najb.	t (s)	najb.	t (s)
050_1k	1	3099.083		**3099.083**	0.019	0.429	**3099.083**	0.080	**3099.083**	0.031
050_1k	2	3016.320	3037.537	**3037.537**	0.043	0.536	**3037.537**	0.159	**3037.537**	0.058
050_1k	3	2977.524	3006.541	**3006.541**	0.038	0.716	**3006.541**	0.185	**3006.541**	0.046
050_1k	4	2930.936		**2930.936**	0.05	0.795	**2930.936**	0.270	**2930.936**	0.070
050_1k	5	2915.139	2930.936	**2930.936**	0.072	0.947	**2930.936**	0.387	**2930.936**	0.072
050_1k	10	2890.945	2901.349	**2901.349**	0.102	1.452	**2901.349**	0.626	**2901.349**	0.118
050_1k	15	2853.382	2881.119	**2881.119**	0.187	1.925	**2881.119**	0.489	**2881.119**	0.123
050_1k	20	2840.953	2875.188	**2875.188**	0.374	2.358	**2875.188**	0.393	**2875.188**	0.194
100_500	10	1852.608		**1852.608**	0.351	1.985	**1852.608**	3.844	**1852.608**	0.515
100_500	15	1834.379		**1834.379**	0.132	2.240	**1834.379**	3.005	**1834.379**	0.587
100_500	20	1809.037		**1809.037**	0.617	3.116	**1809.037**	4.096	**1809.037**	0.498
100_3k	1	8874.360	9348.949	**9348.949**	0.034	1.638	**9348.949**	10.48	**9348.949**	0.261
100_3k	2	8718.983	9142.080	**9142.080**	0.087	2.069	**9142.080**	9.819	**9142.080**	0.750
100_3k	3	8718.983	9069.565	**9069.565**	0.263	3.504	**9069.565**	8.112	**9069.565**	1.113
100_3k	4	8709.104	9047.027	**9047.027**	0.200	3.398	**9047.027**	9.586	**9047.027**	1.124
100_3k	5	8698.801	8995.853	**8995.853**	0.358	4.345	**8995.853**	11.12	**8995.853**	1.846
100_3k	10	8709.173	8873.866	**8873.866**	0.729	7.382	8869.968	13.79	**8873.866**	2.452
100_3k	15	8402.444	8823.342	**8823.342**	0.652	10.82	**8823.342**	22.19	**8823.342**	3.350
100_3k	20	8521.629	8820.679	**8820.679**	1.063	13.80	**8820.679**	29.08	**8820.679**	3.720

Табела 2.15: Резултати метахеуристика на инстанцама $n = 300$ где није познато оптимално решење

inst.	r	preth.	novo najb.	GA			EM		VNS	
				najb.	t (s)	$t_{tot}(s)$	najb.	t (s)	najb.	t (s)
300_2k	1	7709.498	7887.541	**7887.541**	0.844	4.453	7852.239	0.512	**7887.541**	3.564
300_2k	2	7645.349	7794.294	**7794.294**	1.746	7.058	7775.851	0.960	7778.269	4.980
300_2k	3	7509.420	7757.624	**7757.624**	2.222	8.566	7706.137	0.968	**7757.624**	6.206
300_2k	4	7442.000	7742.078	**7742.078**	2.899	10.30	7673.744	1.085	**7740.887**	7.315
300_2k	5	7389.467	7709.929	**7709.929**	1.922	10.22	7657.365	1.223	7702.392	7.546
300_2k	10	7416.068	7642.630	**7642.630**	2.505	14.59	7593.994	1.893	**7642.630**	11.268
300_2k	15	7307.361	7590.892	7575.865	4.345	21.57	7544.598	2.478	**7590.892**	15.709
300_2k	20	7345.519	7550.243	7546.985	4.274	24.68	7539.583	3.079	**7550.243**	18.905
300_10k	1	30824.693	32460.882	**32460.882**	2.602	13.15	**32460.882**	1.880	**32460.882**	12.669
300_10k	2	29659.475	32232.752	**32232.752**	3.613	19.47	32197.892	3.653	**32232.752**	17.455
300_10k	3	30759.960	32146.765	**32146.765**	9.023	27.12	32111.377	4.331	32105.948	22.059
300_10k	4	30989.566	32141.179	**32141.179**	8.260	31.70	32123.699	4.678	32127.248	26.749
300_10k	5	30872.004	32008.414	**32008.414**	10.75	38.26	31970.801	5.508	31991.235	34.295
300_10k	10	30968.334	31865.995	**31865.995**	14.73	61.69	31800.963	8.610	31860.808	54.183
300_10k	15	30748.223	31813.305	**31813.305**	14.98	91.41	31726.461	15.71	31798.469	73.452
300_10k	20	30923.953	31801.412	**31801.412**	33.64	145.3	31724.882	21.08	31779.839	107.024
300_30k	1	85934.244	88870.472	**88870.472**	9.860	43.65	88839.255	5.322	88853.816	41.216
300_30k	2	83903.267	88454.815	**88454.815**	10.31	62.04	88430.781	11.67	88430.781	53.660
300_30k	3	84952.867	88273.593	**88273.593**	28.26	94.81	88216.253	14.13	88268.184	74.559
300_30k	4	85143.297	88135.112	**88135.112**	25.96	119.8	88062.439	16.56	88076.386	85.720
300_30k	5	85845.606	88052.251	88049.412	27.27	150.1	87928.584	21.75	**88052.251**	113.333
300_30k	10	85691.919	87803.363	87791.471	103.9	362.4	87705.325	35.50	**87803.363**	212.166
300_30k	15	85894.230	87731.150	**87731.150**	156.9	558.6	87710.310	54.75	87679.977	331.352
300_30k	20	85715.008	87651.003	**87651.003**	212.8	712.6	87453.229	72.95	87643.225	444.365

Табела 2.16: Резултати метахеуристика на инстанцама $n = 500$ где није познато оптимално решење

inst.	r	preth.	novo najb.	GA			EM		VNS	
				najb.	t (s)	$t_{tot}(s)$	najb.	t (s)	najb.	t (s)
500_1k	1	4744.994	4746.350	**4746.350**	5.785	11.11	4648.119	0.669	4727.554	9.109
500_1k	2	4739.982		**4722.977**	3.807	11.42	4658.193	1.082	4713.594	10.254
500_1k	3	4695.510	4699.366	**4699.366**	7.336	15.94	4604.884	1.444	4678.286	12.719
500_1k	4	4688.866		**4684.724**	9.776	19.56	4647.271	1.713	4662.030	14.866
500_1k	5	4687.445	4688.985	4684.161	11.26	22.19	4579.851	1.873	**4688.985**	16.595
500_1k	10	4636.442		**4630.468**	15.93	30.44	4540.539	3.023	4620.835	22.316
500_1k	15	4622.783	4639.590	**4639.590**	12.07	31.56	4552.096	3.935	4626.853	24.459
500_1k	20	4602.182	4608.352	**4608.352**	15.14	39.40	4542.346	4.983	4584.048	34.049
500_3k	1	11372.166	12027.040	**12027.040**	3.461	11.41	11916.205	1.751	11977.997	11.201
500_3k	2	11293.525	11862.723	**11862.723**	6.343	18.24	11784.635	2.702	11849.706	17.179
500_3k	3	11257.063	11850.243	**11850.243**	5.934	19.97	11725.473	2.702	11793.837	17.201
500_3k	4	11267.662	11765.877	**11765.877**	7.806	24.32	11639.063	3.378	11713.591	21.949
500_3k	5	11214.219	11723.852	**11723.852**	14.13	31.74	11646.231	3.939	11694.037	26.865
500_3k	10	10972.091	11712.643	**11712.643**	9.642	37.29	11617.508	5.863	11603.957	33.650
500_3k	15	11159.572	11578.937	**11578.937**	11.66	45.96	11473.366	7.645	11534.261	41.648
500_3k	20	11038.550	11561.387	**11561.387**	15.05	60.48	11440.231	9.990	11525.872	56.178
500_10k	1	31847.822	34528.000	**34528.000**	4.673	21.79	34409.012	4.514	34501.876	21.087
500_10k	2	32453.585	34260.188	**34260.188**	8.783	31.79	34131.424	7.701	34152.379	30.666
500_10k	3	32741.433	34154.737	**34154.737**	15.82	43.54	34028.891	8.400	34066.961	41.034
500_10k	4	32134.073	33912.136	**33912.136**	14.76	47.58	33803.204	9.124	33807.818	45.580
500_10k	5	32261.153	33911.097	**33911.097**	17.07	54.56	33788.798	12.31	33819.252	49.822
500_10k	10	32868.335	33742.713	**33742.713**	42.26	105.3	33624.379	17.50	33717.134	101.734
500_10k	15	32579.889	33635.495	**33635.495**	33.46	119.0	33479.666	27.78	33474.646	109.920
500_10k	20	32660.868	33617.122	**33617.122**	52.45	192.9	33468.009	36.07	33465.430	180.886
500_60k	1	173626.906	178734.277	**178734.277**	16.80	152.2	178550.864	27.72	178492.642	145.8
500_60k	2	169984.258	177599.746	**177599.746**	74.78	295.0	177263.292	81.03	177279.610	281.8
500_60k	3	171399.820	177238.640	**177238.640**	112.7	345.3	177021.657	85.35	177020.611	314.8
500_60k	4	166948.978	177078.523	**177078.523**	116.2	439.7	176834.418	96.04	176978.673	421.5
500_60k	5	172710.669	177017.653	**177017.653**	106.7	505.6	176629.617	100.5	176726.983	481.7
500_60k	10	165966.934	176618.265	**176618.265**	311.3	1056	176208.475	157.1	176297.561	889.3
500_60k	15	165512.248	176406.682	**176406.682**	436.9	1538	176021.311	252.6	176184.026	1273
500_60k	20	166051.696	176373.468	**176373.468**	668.8	2028	175924.068	295.8	176116.713	1896

Табела 2.17: Резултати метахеуристика на инстанцама $n = 1000$ где није познато оптимално решење

inst.	r	preth.	novo najb.	GA			EM		VNS	
				najb.	t (s)	$t_{tot}(s)$	najb.	t (s)	najb.	t (s)
1k_1.5k	1	7830.439		**7775.978**	37.98	51.86	7538.786	2.836	7730.085	51.325
1k_1.5k	2	7660.702		**7612.876**	51.87	70.30	7395.79	4.521	7547.915	65.546
1k_1.5k	3	7561.257		**7478.796**	50.89	72.12	7300.616	6.000	7466.919	71.088
1k_1.5k	4	7552.921		**7455.309**	63.55	89.76	7316.591	6.590	7453.238	87.372
1k_1.5k	5	7561.257		**7486.499**	81.02	102.5	7330.503	7.108	7460.453	100.537
1k_1.5k	10	7499.702		**7429.124**	74.73	121.8	7218.991	12.28	7369.720	118.701
1k_1.5k	15	7457.737		**7409.428**	114.2	170.1	7151.337	15.94	7325.439	165.461
1k_1.5k	20	7432.879		**7351.995**	136.4	192.2	7192.178	21.55	7272.166	186.898
1k_10k	1	34497.300	37642.910	**37642.910**	38.66	72.49	37340.775	18.99	37335.113	71.051
1k_10k	2	33431.473	37082.383	**37082.383**	57.40	108.75	36658.911	27.26	36851.114	106.235
1k_10k	3	35076.961	36811.521	**36811.521**	57.11	124.0	36487.822	30.98	36564.711	120.444
1k_10k	4	34828.276	36771.775	**36771.775**	63.48	143.6	36431.582	40.06	36338.953	137.109
1k_10k	5	35122.442	36682.944	**36682.944**	79.45	171.7	36435.325	39.64	36326.740	167.196
1k_10k	10	34552.295	36593.801	**36593.801**	72.59	205.3	36159.098	61.84	35990.061	196.652
1k_10k	15	34642.263	36429.494	**36429.494**	125.2	309.2	36075.022	96.69	36081.592	303.380
1k_10k	20	34789.795	36455.454	**36455.454**	196.9	424.8	36114.143	123.8	36058.370	419.638
1k_100k	1	294083.743	304705.500	**304705.500**	229.5	680.7	304143.577	237.5	303655.762	664.079
1k_100k	2	277883.845	303065.476	**303065.476**	356.8	1020	302752.951	481.8	302371.567	979.356
1k_100k	3	277770.801	302588.147	**302588.147**	587.0	1322	301729.009	465.5	301519.126	1287.860
1k_100k	4	287363.683	302061.731	**302061.731**	732.8	1667	301171.529	474.0	301111.823	1587.103
1k_100k	5	293827.978	301684.879	**301684.879**	950.	2038	300931.572	559.5	301026.029	1952.373
1k_100k	10	276657.467	301101.730	**301101.730**	2091	3679	300144.674	821.3	300249.889	3589.067
1k_100k	15	291638.645	300899.411	**300899.411**	1534	4286	300059.641	1263	300057.737	4019.348
1k_100k	20	294052.633	300608.129	**300608.129**	2322	5579	299723.170	1369	300298.373	5471.942
1k_350k	1	986248.932	1004141.663	**1004141.663**	1271	3164.523	1003349.956	560.8	1002271.215	3039.011
1k_350k	2	963299.917	998643.547	**998643.547**	2077	4184.105	998045.476	2293	997731.064	3886.649
1k_350k	3	962372.761	996287.655	**996287.655**	2170	4913.708	995344.707	2657	995357.745	4617.897
1k_350k	4	965016.683	995233.791	**995233.791**	2335	5982.324	994358.43	2576	994635.082	5736.208
1k_350k	5	963881.119	994544.743	**994544.743**	2614	7048.297	993655.168	2975	993896.013	6589.025
1k_350k	10	962136.163	993679.112	**993679.112**	5046	12618.262	992408.902	4107	993023.622	12101.822
1k_350k	15	962052.538	993116.857	**993116.857**	7073	17164.022	992155.293	5031	992579.670	14310.802
1k_350k	20	962008.567	992902.459	**992902.459**	7898	19608.342	991776.551	6951	992022.370	18644.139

Глава 3

Вишедимензионални проблем максималне бисекције графа на повезане подграфове

Вишедимензионални проблем максималне бисекције графа на повезане подграфове (енг. connected multidimensional maximum bisection problem - CMDMBP) је генерализација проблема максималне бисекције налик MDMBP разматраног у претходној глави, где се додатно захтева да подграфови индуковани партицијама буду повезани, а уведен је у [42].

Приступи решавању CMDMBP се могу разматрати на основу приступа решавању MBP и оних решавању MDMBP, као и проблема повезаних са партиционисањем графова на повезане подграфове. Као што се може видети MDMBP је једна релаксација CMDMBP (када се одбаце услови повезаности) те ограничења која важе MDMBP важе и за CMDMBP.

Вишедимензионални проблем максималне бисекције графа на повезане подграфове се јавља где год су релације између ентитета представљене са више нумеричких карактеристика уместо једног броја и где је повезаност подграфова од суштинске важности. Ово се може видети из конкретних проблема. Први проблем је везан за управљање људским ресурсима, где је један од најважнијих аспеката је компатибилност/некомпатибилност запослених. Компатибилност се може представити вектором који чине координате које одговарају карактеру, знању, искуству, итд. где су виши нивои некомпатибилности представљени већим бројевима. Запослени у овом случају су представљени чворовима и

чињеница да је одређени пар запослених радио заједно у ранијем периоду представљен је граном између датих чворова. Проблем је поделити групу запослених у два тима једнаких величина где највећи део некомпатибилности међу запосленима лежи између тимова. Повезаност подграфова (тимова) игра важну улогу јер се може сматрати да ће се људи који су раније радили заједно лакше уклопити у тим и у будућности. Други пример је сличан примеру везаном за дизајн електронских кола из претходног поглавља, само што се још додаје услов да мора да се одржи повезаност електронских компоненти на свакој плочи.

3.1 Дефиниција проблема

У овој секцији ће бити уведен вишедимензионални проблем максималне бисекције графа на повезане подграфове као генерализације проблема максималне бисекције.

Као што се може видети, повезаност подграфова може бити веома корисна у одређеним областима практичног и теоретског истраживања. Ово је главни разлог за формулисање нове генерализације MBP.

Проблем се може формулисати на следећи начин: нека је $G = (V, E)$ неоријентисани повезани граф, $S \subseteq V$ и w је функција која додељује свакој грани e неку r-торку ненегативних реалних бројева $(w_{e1}, w_{e2}, \ldots, w_{er})$. Пресек $C(S)$ одређен скупом S је дефинисан као

$$C(S) = \{(i_e, j_e) \in E | (i_e \in S \wedge j_e \notin S) \vee (j_e \in S \wedge i_e \notin S))\}.$$

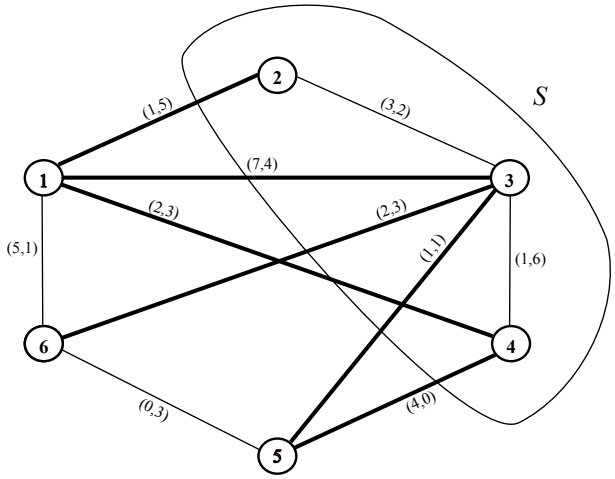
У даљем тексту ознака (i_e, j_e) не представља уређени пар чворова већ крајеве гране e. Очигледно је да важи једнакост $C(S) = C(V \setminus S)$. Тежина пресека је дефинисана као

$$w(C(S)) = \min_{1 \leq l \leq r} \sum_{(i,j) \in C(S)} w_{(i,j)l}.$$

Задатак генерализованог проблема максималне бисекције је пронаћи поделу скупа чворова на два скупа са једнаким бројем чворова где је тежина пресека максимална, и оба подграфа индукована партицијама су повезана. Овај проблем називамо вишедимензионални проблем максималне бисекције графа на

повезане подграфове.

Пример 3.1. *Вишедимензионални проблем максималне бисекције графа на повезане подграфове може бити илустрован примером графа датим на Слици 3.1, чије је оптимално решење дато скупом* $S = \{2, 3, 4\}$. *Скуп* S *генерише пресек* $C(S) = \{(1, 2), (1, 3), (1, 4), (3, 5), (3, 6), (4, 5)\}$ *где су суме по координатама* $(17, 16)$ *па је тежина пресека једнака* 16.



Слика 3.1: Вишедимензионална бисекција графа на повезане подграфове. Подебљане су гране које припадају пресеку.

Приметимо да је граф из претходног примера исти као граф из примера 2.1, и да је вредност оптималног решења за граф за MDMBP, једнако 18, а да је скуп $S = \{1, 3, 5\}$. У том случају S индукује повезани подграф, али $V \setminus S$ индукује подграф који није повезан.

3.2 Модел мешовитог целобројног линеарног програмирања

Као и у случају проблема из главе 2, и за овај проблем је формулисан модел мешовитог линеарног програмирања. Развој модела MILP-а, и у овом случају, може користити за одређивање егзактних решења за графове мањих димензија, као и за развој хеуристичких приступа [51] и [58].

У овом поглављу ће бити уведена формулација мешовитог линеарног програмирања за вишедимензионални проблем максималне бисекције графа на

повезане подграфове. Идеја моделовања повезаности подграфова прати принципе дате у [44].

Нека је $G = (V, E)$ где је $|V| = n$, n паран број и нека је $w_e = (w_{e1}, w_{e2}, \ldots, w_{er})$ вектор тежина гране $e \in E$. Уведимо нови чвор $0 \notin V$ и скуп нових грана $\partial E = \{(0, i) | i \in V\}$ и означимо $\overline{V} = V \cup \{0\}$ и $\overline{E} = E \cup \partial E$.

Задатак је наћи поделу (V_1, V_2) скупа чворова V где је $V_1 \cap V_2 = \emptyset$, $|V_1| = |V_2| = n/2$ тако да је $w(C(V_1))$ максимално, а да подграфови индуковани скуповима V_1 и V_2 буду повезани. Нека су $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ такви индуковани подграфови. Пошто су повезани, они садрже разапињућа стабла, $T_1 = (V_1, E_1')$ и $T_2 = (V_2, E_2')$. Нека су су чворови p и q произвољни чворови из, редом, V_1 и V_2. Означимо проширене скупове грана E_1' и E_2' са $\overline{E}_1' = E_1' \cup \{(0, p)\}$ и $\overline{E}_2' = E_2' \cup \{(0, q)\}$. Овим су одређена проширена разапињућа стабла $\overline{T}_1 = (V_1 \cup \{0\}, \overline{E}_1')$ и $\overline{T}_2 = (V_2 \cup \{0\}, \overline{E}_2')$. Нека су графови T и $\overline{T}$ дефинисани као $T = (V, E_1' \cup E_2')$ и $\overline{T} = (V \cup \{0\}, \overline{E}_1' \cup \overline{E}_2')$.

Да би се формулисао MILP модел за решавање CMDMBP уводе се следеће променљиве:

$$x_i = \begin{cases} 1, & i \in V_1 \\ 0, & i \in V_2, \end{cases} \tag{3.1}$$

$$t_e = \begin{cases} 1, & \text{ако грана } e \in C(V_1) \\ 0, & \text{у супротном,} \end{cases} \tag{3.2}$$

$$y_e = \begin{cases} 1, & \text{ако грана } e \in \overline{E}_1' \\ 0, & \text{у супротном,} \end{cases} \tag{3.3}$$

$$z_e = \begin{cases} 1, & \text{ако грана } e \in \overline{E}_2' \\ 0, & \text{у супротном,} \end{cases} \tag{3.4}$$

$$u_e \in [-n/2, n/2], e \in \overline{E} \tag{3.5}$$

Променљиве x_i одређују поделу скупа чворова, t_e одређују гране које припадају пресеку док y_e и z_e одређују гране које припадају одговарајућем проширеном разапињућем стаблу. Променљиве u_e представљају количину протока која пролази кроз одговарајућу грану. Да би се моделовала повезаност, користи се

врста протока код које у граф улази количина протока која је једнака броју чворова, а у сваком чвору се задржава количина протока 1. Треба напоменути да ова врста протока ни на који начин није слична са протоцима који се користе у класичним проблемима максималних протока и минималних резова.

Модел мешовитог линеарног програмирања за решавање CMDMBP можемо записати као:

$$\max U \tag{3.6}$$

тако да важи

$$U \leq \sum_{e \in E} w_{el} \cdot t_e, \quad 1 \leq l \leq r \tag{3.7}$$

$$x_{i_e} + x_{j_e} \geq t_e, \quad (i_e, j_e) = e \in E \tag{3.8}$$

$$x_{i_e} + x_{j_e} + t_e \leq 2, \quad (i_e, j_e) = e \in E \tag{3.9}$$

$$\sum_{i \in V} x_i = \frac{n}{2}, \tag{3.10}$$

$$y_e \leq \frac{1}{2} x_{i_e} + \frac{1}{2} x_{j_e}, \quad (i_e, j_e) = e \in E \tag{3.11}$$

$$z_e \leq 1 - \frac{1}{2} x_{i_e} - \frac{1}{2} x_{j_e}, \quad (i_e, j_e) = e \in E \tag{3.12}$$

$$y_e \leq x_{j_e}, \quad (0, j_e) = e \in \partial E \tag{3.13}$$

$$z_e \leq 1 - x_{j_e}, \quad (0, j_e) = e \in \partial E \tag{3.14}$$

$$u_e \leq \frac{n}{2} \cdot y_e + \frac{n}{2} \cdot z_e, \quad e \in \overline{E} \tag{3.15}$$

$$u_e \geq -\frac{n}{2} \cdot y_e - \frac{n}{2} \cdot z_e, \quad e \in \overline{E} \tag{3.16}$$

$$\sum_{e:j_e=i} u_e - \sum_{e:i_e=i} u_e = 1, \quad i \in V \tag{3.17}$$

$$\sum_{e:i_e=0} u_e = n \tag{3.18}$$

$$\sum_{e\in E} y_e = \frac{n}{2} - 1 \tag{3.19}$$

$$\sum_{e\in E} z_e = \frac{n}{2} - 1 \tag{3.20}$$

$$\sum_{e\in \partial E} y_e + \sum_{e\in \partial E} z_e = 2 \tag{3.21}$$

$$x_i, t_e, y_e, z_e \in \{0,1\}, \quad i \in V, e \in \overline{E} \tag{3.22}$$

$$u_e \in [-n/2, n/2], \quad e \in \overline{E} \tag{3.23}$$

Ограничењем (3.7) је одређена горња граница тежине пресека. За сваку грану која је у пресеку, тачно један чворова припада V_1, што је у моделу обезбеђено ограничењима (3.8) и (3.9). Ограничење (3.10) осигурава да партиције V_1 и V_2 имају подједнак број чворова. Повезаност индукованих подграфова се осигурава помоћу ограничења (3.11)-(3.20). Идеја дела модела којим се осигурава повезаност је заснована на приступу да се за подграфове индуковане чворовима из V_1, односно V_2, конструишу одговарајућа разапињућа стабла, која ће, пошто су стабла повезана, гарантовати повезаност читавих индукованих партиција. Ограничења (3.11) и (3.12) обезбеђују да грана чији су крајеви у различитим партицијама не може припадати ни T_1 ни T_2. Ограничења (3.13) обезбеђују да гране које повезују нулти чвор са чворовима из V_2 не могу бити у T_1. Аналогно, ограничења (3.14) обезбеђују да гране које повезују нулти чвор са чворовима из V_1 не могу бити у T_2. Ограничења (3.15) и (3.16) обезбеђују да је $u_e = 0$ за гране e које не припадају разапињућем стаблу $\overline{T}$. Ограничење (3.17) обезбеђује да је разлика улазних и излазних вредности промењљиве u_e кроз неки чвор једнака 1, тј. да је у чвору остао проток у количини 1. Ограничење (3.18) осигурава да је излаз из додатног чвора преко промењљиве u_e једнак броју чво-

рова у графу G, чиме се обезбеђује да у разапињућем стаблу има укупно $n+1$ чвор односно n грана. Пошто у подграфу индукованом са V_1, односно са V_2 има по $n/2$ чворова, одговарајућа покривајућа стабла садрже по $n/2-1$ грану. Ограничења (3.19) и (3.20) обезбеђују да разапињућа стабла садрже управо по толико грана. Ограничење (3.21) обезбеђује да постоје тачно две гране које излазе из додатног чвора 0 којима је $u_e \neq 0$.

Треба приметити да су ограничења (3.11)-(3.14), (3.17), (3.18) и (3.21) слична као у формулацији за решавање проблема налажења максималне балансиране повезане партиције уведене у Матићевом раду [44]. Идеја изложена у [44] обезбеђује повезаност партиција преко концепта протока.

Лема 1. *Из ограничења (3.11)-(3.21) следи да у оптималном решењу проблема (3.6)-(3.23) постоје чворови* $p \in V_1$ *и* $q \in V_2$*, такви да је* $u_{(0,p)} = |V_1|$*,* $u_{(0,q)} = |V_2|$ *и* $u_{(0,i)} = 0$ *за* $i \neq p, q$*.*

Доказ. Нека је e грана из $\overline{E}$. Грана e може али не мора бити укључена у разапињуће стабло $\overline{T}$. У случају да је грана e укључена у разапињуће стабло, тј. $e \in \overline{E}'_1 \cup \overline{E}'_2$, тада следи да је тачно једно од y_e или z_e једнако 1. У супротном, оба y_e и z_e су једнаки нули. Ограничењем (3.11), вредности y су ограничене са десном страном неједнакости вредностима x_{i_e} и x_{j_e} (слично, ограничења (3.12) ограничавају променљиве z). На пример, ако оба i_e и j_e припадају V_1, обе вредности x_{i_e} и x_{j_e} су истовремено једнаки 1, и у том случају, ограничење (3.11) дозвољава да грана e може бити укључена у разапињуће стабло T_1. Пошто су променљиве бинарног типа, постоје четири могућности за x_{i_e} и x_{j_e}:

(i) $x_{i_e} = 1$ и $x_{j_e} = 1$: То значи да оба чвора припадају V_1;

(ii) $x_{i_e} = 1$ и $x_{j_e} = 0$: чвор i припада V_1, а чвор j припада V_2;

(iii) $x_{i_e} = 0$ и $x_{j_e} = 1$: чвор i припада V_2 и j припада V_1;

(iv) $x_{i_e} = 0$ и $x_{j_e} = 0$: оба чвора припадају V_2;

Само у случајевима (i) и (iv), постоји могућност да је грана укључена у одговарајуће разапињуће стабло, а у случајевима (ii) и (iii), ограничења (3.11) и (3.12) гарантују да грана e неће бити укључена, те ће тада због (3.15) и (3.16) u_e бити једнако 0.

Из (3.21), директно следи да тачно две гране $(0,p)$ и $(0,q)$ имају вредност промењиве u различиту од нуле. Ове гране припадају ∂E и за њих због (3.13) и (3.14) или $y_e = 1$ или $z_e = 1$. Биће показано да p и q припадају различитим подскуповима V_1 и V_2. Претпоставимо да, без губитка општости, и p и q припадају V_1. Из (3.18) следи да $u_{(0,p)} + u_{(0,q)} = n$.

Сумирајмо ограничења (3.17), када је $i \in V_1$.

$$\begin{aligned}
|V_1| &= \sum_{i\in V_1}\left(\sum_{e:j_e=i} u_e - \sum_{e:i_e=i} u_e\right) = \sum_{e:j_e\in V_1} u_e - \sum_{e:i_e\in V_1} u_e = \\
&= \sum_{e:i_e\in V_1\wedge j_e\in V_1} u_e + \sum_{e:i_e=0\wedge j_e\in V_1} u_e + \sum_{e:i_e\in V_2\wedge j_e\in V_1} u_e - \\
&\quad \left(\sum_{e:i_e\in V_1\wedge j_e\in V_1} u_e + \sum_{e:i_e\in V_1\wedge j_e\in V_2} u_e\right) \\
&= \sum_{e:i_e=0\wedge j_e\in V_1} u_e = u_{(0,p)} + u_{(0,q)} = n
\end{aligned}$$

Прва и четврта сума се поништавају, док трећа и последња имају све сабирке једнаке нули. Тако је израз $|V_1| = n$ у контрадикцији са чињеницом да је $|V_1| = \frac{n}{2}$ која следи из ограничења (3.10). Закључујемо да p и q морају припадати различитим подскуповима V_1 и V_2. □

Теорема 2. *Нека је дат повезан граф $G = (V, E)$, где је $|V|$ паран број. Партиција скупа чворова $V = (V_1, V_2)$, и индуковани подграфови $G_1 = (V_1, E_1)$ и $G_1 = (V_2, E_2)$ чине решење CMDMBP ако и само ако проблем мешовитог линеарног програмирања (3.6)-(3.23) има оптимално решење.*

Доказ. ($\Rightarrow$) На основу партиционисања графа G, конструишимо промењиве (x, t, y, z, u) тако да функција циља (3.6) достиже оптималну вредност, а ограничења (3.7)-(3.23) буду задовољена. Нека су G_1, G_2, V_1, V_2, E_1, E_2, T_1, T_2, E'_1, E'_2, $\overline{E}'_1$, $\overline{E}'_2$, $\overline{T}_1$, $\overline{T}_2$, $\overline{T}$, T и ∂E одређени као на почетку поглавља 3.2.

Нека су промењиве дефинисане у складу са (3.1)-(3.4). Очигледно је да је ограничење (3.22) задовољено. За $e \in \partial E$, u_e дефинишимо као

$$u_e = \begin{cases} |V_1| & e_j = p \\ |V_2| & e_j = q \\ 0 & \text{иначе} \end{cases} \tag{3.24}$$

где је p произвољни елемент из V_1, а q произвољни елемент из V_1.

Како је $|V_1'| = \frac{n}{2}$ и $|V_2'| = \frac{n}{2}$, тада $u_e \in [-\frac{n}{2}, \frac{n}{2}]$ за све $e \in \partial E$.

Покажимо сада да и за остале гране e важи да је u_e из датог интервала $[-n/2, n/2]$. Посматрајмо два покривајућа стабла T_1 и T_2 за скупове чворова V_1 и V_2. Оваква разапињућа стабла се лако могу одредити неким алгоритмом претраживања, на пример, претраживањем по дубини или по ширини. Оба ова алгоритма обично полазе од произвољног чвора, који се може сматрати кореном стабла, док су сви други чворови уређени датим алгоритмом. У нашем случају полазни чворови или корени ће бити p и q. Формирајмо проток кроз граф $(V \cup \{0\}, \overline{E})$. Нека је $e = (i_e, j_e)$ из стабла T. Коришћењем поретка генерисаног алгоритмом претраге, у случају да је чвор i_e родитељ чвора j_e, вредност протока кроз грану e је дефинисана као број чворова у подстаблу са кореном у чвору j_e. У случају када је чвор j_e родитељ чвору i_e, вредност протока u_e дефинишемо као негативан број чворова у подстаблу са кореном у чвору i_e. За све друге гране, који не припадају ни E_1' ни E_2', количина протока је једнака нули тј. $u_e = 0$. Пошто је број чворова у сваком подстаблу мањи или једнак $\frac{n}{2}$, тада је $u_e \in [-\frac{n}{2}, \frac{n}{2}]$ за сваку грану $e \in \overline{E} \setminus \partial E$. Зато, (3.23) важи за свако $e \in \overline{E}$.

Покажимо да су ограничења (3.7)-(3.21) задовољена.

На основу дефиниције тежине пресека, ограничење (3.7) је задовољено, а пошто је партиционисање графа G оптимално, (3.6) ће, такође, бити задовољено.

Ако је $t_e = 0$ тада су (3.8) и (3.9) очигледно тачни. Ако је $t_e = 1$ тада одговарајућа грана $e = (i, j)$ припада пресеку, а тачно један чвор инцидентан грани e мора бити у скупу V_1, па је или $x_i = 1$ или $x_j = 1$ те (3.8) и (3.9) важе.

Ограничење (3.10) је очигледно испуњено пошто је захтевано да је скуп чворова подељен у два скупа са једнаким бројем чворова.

Да бисмо доказали неједнакости (3.11), разматрају се два случаја $y_e = 0$, или $y_e = 1$.

(i) $y_e = 0$: Неједнакости (3.11) су задовољене зато што је x_{i_e},$x_{j_e} \geq 0$ по

дефиницији.

(ii) $y_e = 1$ имплицира да је $e \in \overline{E}'_1 \Rightarrow x_{i_e} = x_{j_e} = 1 \Rightarrow y_e \leq \frac{1}{2}x_{i_e} + \frac{1}{2}x_{j_e}$.

Да докажемо једнакости (3.13), поново се разматрају два случаја:

(i) $y_e = 0$: Неједнакости (3.13) важе зато што је $x_{j_e} \geq 0$ по дефиницији.

(ii) $y_e = 1$ имплицира да је $e \in \overline{E}'_1$. Пошто је $e \in \partial E$, $e \in \overline{E}'_1 \cap \partial E = (0, p)$, јер је p корен стабла T_1, што даље имплицира да $j_e = p \in V_1$ па следи да је $x_{j_e} = 1$, а одатле неједнакости (3.13) важе.

Неједнакости (3.12) и (3.14) се доказују аналогно неједнакостима (3.11) и (3.13).

Да би доказали неједнакости (3.15) и (3.16) разматрају се два случаја:

(i) $e \in \overline{E}'_1 \cup \overline{E}'_2$. Вредности десних страна неједнакости (3.15) и (3.16) су редом $\frac{n}{2}$ и $-\frac{n}{2}$ јер је тачно једна од променљивих y_e и z_e једнака 1. Како је $u_e \in [-\frac{n}{2}, \frac{n}{2}]$, неједнакости (3.15) и (3.16) су задовољене.

(ii) $e \notin \overline{E}'_1 \cup \overline{E}'_2$. Тада је u_e једнако 0 по дефиницији. Неједнакости (3.15) и (3.16) су задовољене зато што су десна страна неједнакости (3.15) ненегативне, а десне стране неједнакости (3.16) су непозитивне.

Да би доказали неједнакост (3.17), без губитка општости, претпоставимо да је $i \in V_1$. Размотримо све гране из $\overline{E}'_1$, које почињу или завршавају чвором i. Једна од ових грана долази из родитељског чвора, а сви други чворови су наследници. За све друге гране инцидентне i, а које не припадају $\overline{E}'_1$, вредности u_e су једнаке 0 и не утичу на (3.17).

$$\sum_{e:j_e=i} u_e - \sum_{e:i_e=i} u_e = \sum_{e:j_e=i \wedge u_e>0} u_e + \sum_{e:j_e=i \wedge u_e<0} u_e - \sum_{e:i_e=i \wedge u_e>0} u_e - \sum_{e:i_e=i \wedge u_e<0} u_e =$$

$$= \sum_{e:j_e=i \wedge u_e>0} |u_e| + \sum_{e:j_e=i \wedge u_e<0} -|u_e| - \sum_{e:i_e=i \wedge u_e>0} |u_e| - \sum_{e:i_e=i \wedge u_e<0} -|u_e| =$$

$$= \sum_{e:j_e=i \wedge u_e>0} |u_e| + \sum_{e:i_e=i \wedge u_e<0} |u_e| - \left(\sum_{e:j_e=i \wedge u_e<0} |u_e| + \sum_{e:i_e=i \wedge u_e>0} |u_e| \right).$$

У трећој и четвртој суми у првом реду постоји само једна грана за коју је u_e различито од нуле и то је грана која долази у чвор i из његовог родитеља. За ту грану, $|u_e|$ је једнако броју чворова у подстаблу са кореном у i. У последње две

суме учествују све гране које полазе из чвора i ка наследницима у разапињућем стаблу $\overline{T}_1$. Укупна сума $|u_e|$ за те гране је једнака укупном броју чворова свих подстабала које за корен имају једног од наследника чвора i. Тих чворова има за 1 мање од чворова у стаблу са кореном у чвору i, па важи

$$\sum_{e:j_e=i\wedge u_e>0} |u_e| + \sum_{e:i_e=i\wedge u_e<0} |u_e| - \left(\sum_{e:j_e=i\wedge u_e<0} |u_e| + \sum_{e:i_e=i\wedge u_e>0} |u_e| \right) = 1.$$

Тиме је доказана неједнакост (3.17).

Искористимо дефиницију (3.24) промењљиве u_e, када је $e \in \partial E$ у доказу испуњености услова (3.18). Тада је очигледно

$\sum_{e:i_e=0} u_e = u(0,p) + u(0,q) = |V_1| + |V_2| = n.$

Како су T_1 и T_2 разапињућа стабла подграфова G_1 и G_2, тада је $\sum_{e\in E} y_e = |E_1'| = |V_1| - 1$ и $\sum_{e\in E} z_e = |E_2'| = |V_2| - 1$. То имплицира да су ограничења (3.19) и (3.20) испуњена.

За $e \in \partial E$, $y_e = 1$ за само једно e, и то у случају када је $e = (0,p)$. За све друге гране $e \in \partial E, y_e = 0$, што имплицира $\sum_{e\in\partial E} y_e = 1$ и слично, $\sum_{e\in\partial E} z_e = 1$, одакле следи да је ограничење (3.21) испуњено.

($\Leftarrow$) Претпоставимо да је $(x^*, t^*, y^*, z^*, u^*, U^*)$ оптимално решење проблема мешовитог линеарног програмирања (3.6)-(3.23). Конструишимо партиционисање графа G на подграфове G_1 и G_2 тако да оно представља решење CMDMBP.

Дефинишимо и

$V_1 = \{i \in V | x_i = 1\}$, $E_1 = \{e \in E | y_e = 1\}$,

$V_2 = \{i \in V | x_i = 0\}$, $E_2 = \{e \in E | z_e = 1\}$ и

$C(V_1) = \{e \in E | t_e = 1\}$. Одавде је очигледно да је $V_1 \cup V_2 = V$ и $V_1 \cap V_2 = \emptyset$.

Скуп $C(V_1)$ представља пресек генерисан скупом чворова V_1.

Из ограничења (3.22) t_e је или 0 или 1.

Ако је $t_e = 1$ тада из ограничења (3.8) и (3.9) следи да инцидентни чворови гране e припадају различитим скуповима V_1 и V_2.

Нека је e' грана таква да је $t_{e'} = 0$. Покажимо да из услова (3.8)-(3.9) $i_{e'}$ и $j_{e'}$ морају да буду у истој партицији (или V_1 или V_2). Уколико би припадали различитим партицијама, сума у десној страни услова (3.7) би била мања од суме у којој би за ову грану e' вредност од $t_{e'}$ било једнако 1. То јест, $U =$

$\sum_{e\in E,e':t(e')=1} w_{el}\cdot t_e, 1\leq l\leq r$ би било веће од U^*, што је у контрадикцији да је U^* оптимално решење проблема (3.6)-(3.23). Дакле, када је $t_e=0$ оба чвора инцидентна са e припадају истој партицији.

Из ограничења (3.10) следи да је $|V_1|=n/2=|V_2|$ што значи да је скуп чворова подељен у два скупа са једнаким бројем чворова.

Дефинишимо скупове $\overline{E}_1'$ и $\overline{E}_2'$ на следећи начин: $\overline{E}_1'=\{e\in\overline{E}|y_e=1\}$, $\overline{E}_2'=\{e\in\overline{E}|z_e=1\}$, $E_1'=\{e\in E|y_e=1\}$ и $E_2'=\{e\in E|z_e=1\}$.

Ограничења (3.11) - (3.14) осигуравају да су скупови $\overline{E}_1'$ и $\overline{E}_2'$ добро дефинисани, тј. све гране из $\overline{E}_1'$ имају крајње тачке из $V_1\cup\{0\}$, и све гране из $\overline{E}_2'$ имају крајње тачке из $V_2\cup\{0\}$:

(i) ако је $e\in E_1'$ из дефиниције скупа E_1' следи да је $y_e=1$. Из ограничења (3.11) и бинарне природе променљивих x_{i_e} и x_{j_e}, следи да $x_{i_e}=1$ и $x_{j_e}=1$, што имплицира да $i_e,j_e\in V_1$.

(ii) ако је $e\in\overline{E}_1'$ и $i_e=0$ (реч је о чвору 0 који смо додали) такође следи да је $y_e=1$, а из ограничења (3.13) следи $x_{j_e}=1$, што имплицира да $j_e\in V_1$.

Слично, ограничења (3.12) и (3.14) осигуравају да све гране из $\overline{E}_2'$ имају инцидентне чворове из $V_2\cup\{0\}$. Ограничења (3.19) и (3.20) осигуравају да је број грана у E_1' и E_2' једнак $n/2-1$, што обезбеђује да су у подстаблима укључени сви чворови из V_1 односно V_2.

Из неједнакости (3.15) и (3.16), следи да $u_e=0$, за свако $e\notin\overline{E}_1'\cup\overline{E}_2'$.

Докажимо сада повезаност графа (V_1,E_1'), а повезаност графа (V_2,E_2') се доказује на аналоган начин.

Повезаност скупа V_1 ћемо показати тако што ћемо доказати да за свако партиционисање скупа V_1 на подскупове S' и S'' важи да постоји грана чији је један крај у S', а други у S''. Претпоставимо да су S' и S'' произвољни подскупови скупа V_1, такви да је $S'\cup S''=V_1$ и $S'\cap S''=\emptyset$, $S',S''\neq\emptyset$. Докажимо да $(\exists e\in E_1')i_e\in S'\wedge j_e\in S''$.

Сумирајмо ограничења дата у (3.17), за свако $i\in S'$. Добијамо

$$\sum_{e:j_e\in S'} u_e-\sum_{e:i_e\in S'} u_e=|S'|,$$

пошто су са десне стране у условима (3.17) једнаке 1.

Размотримо сада суму са леве стране:

$$\sum_{e:j_e\in S'} u_e - \sum_{e:i_e\in S'} u_e =$$

$$\sum_{e:j_e\in S'\wedge i_e\in S'} u_e + \sum_{e:j_e\in S'\wedge i_e\in S''} u_e + \sum_{e:j_e\in S'\wedge i_e=0} u_e + \sum_{e:j_e\in S'\wedge i_e\in V_2} u_e -$$

$$-\left(\sum_{e:i_e\in S'\wedge j_e\in S'} u_e + \sum_{e:i_e\in S'\wedge j_e\in S''} u_e + \sum_{e:i_e\in S'\wedge j_e\in V_2} u_e\right).$$

Означимо сабирке у последњој једначини са A, B, C, D, E, F и G. Тада следи $\sum_{e:j_e\in S'} u_e - \sum_{e:i_e\in S'} u_e = A + B + C + D - (E + F + G)$.

Очигледно је да је $A = E$ и $D = G = 0$ (у $E' = E'_1 \cup E'_2$ нема грана између чворова у V_1 и V_2). Користећи Лему 1, која тврди да је тачно један чвор (p у овом случају) из V_1 повезан са чвором 0, закључујемо да је $C = 0$ или $C = |V_1|$ зависно од тога да ли је $p \in S'$ или $p \notin S'$. Одатле важи

$$\sum_{e:j_e\in S'\wedge i_e\in S''} u_e \neq \sum_{e:i_e\in S'\wedge j_e\in S''} u_e (B \neq F).$$

Последњи израз имплицира да постоји грана e таква да има крајеве у различитим скуповима S' и S'', за коју је $u_e \neq 0$. Према (3.15) и (3.16) $u_e \neq 0 \Rightarrow y_e = 1$ а одатле је $e \in E'_1$ грана која повезује S' и S''. Дакле, (V_1, E'_1) повезан.

Пошто је $|V_1| = n/2$, а кардиналност од $|E'_1|$ једнака $n/2-1$ следи да је (V_1, E'_1) је стабло. Слично томе, (V_2, E'_2) је такође стабло. Ова два стабла, назовимо их T_1 и T_2, ћемо искористити за формирање подграфова индукованих скуповима чворова V_1 и V_2 на следећи начин. Скуп E_1 ће бити све гране из стабла T_1 на који ћемо додати све гране из графа G које међусобно повезују чворове из V_1, а нису из E'_1. На сличан начин добијамо и скуп E_2. За ове придодате гране ће одговарајуће y_e и z_e бити 0 тако да је граф G партиционисан у два подграфа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ којима ће T_1 и T_2 бити разапињућа стабла. Као што смо видели, једине гране из E којима ће t бити једнако 1, су оне које повезују подграфове G_1 и G_2. Дакле, t_e ће бити једнако 1 само за гране које припадају пресеку $C(V_1)$.

Размотримо сада ограничење

$$U \leq \sum_{e\in E} w_{el} \cdot t_e, \quad 1 \leq l \leq r.$$

Из раније реченог $t_e = 1$ једино уколико је грана у пресеку насталом пар-

тиционисањем графа G на два повезана, једнака по кардиналости чворова, подграфа. Одавде следи да је $U \leq \sum_{\substack{(i,j)\in E \\ i\in S, j\notin S}} w_{(i,j)l}$ за $1 \leq l \leq r$. што значи $U \leq w(C(S))$ за свако S, тако да је $|S| = n/2$ и графови индуковани S и $V \setminus S$ повезани.

Како је U^* оптимално решење проблема (3.6)-(3.23) то ће и подграфови генерисани на горе описани начин за променљиве x^*, y^*, z^*, u^*, t^* бити повезани, имати кардиналност једнаку $n/2$, а вредност њиховог пресека бити максимална, те ће представљати оптимално решење CMDMBP за граф G. □

Да бисмо илустровали предложену математичку формулацију, следећи пример садржи вредности свих променљивих.

Пример 3.2. *Размотримо граф из Примера 3.1. Решавањем MILP модела добија се да је вредност оптималног решења 16, где је скуп* $S = \{2, 3, 4\}$*, и где је пресек* $C(S) = \{(1,2), (1,3), (1,4), (3,5), (3,6), (4,5)\}$*. Ненула вредности променљивих су:*

$x_1 = x_5 = x_6 = 1$, $t_{(1,2)} = t_{(1,3)} = t_{(1,4)} = t_{(3,5)} = t_{(3,6)} = t_{(4,5)} = 1$, $y_{(0,1)} = y_{(1,6)} = y_{(5,6)} = 1$, $z_{(0,3)} = z_{(2,3)} = z_{(3,4)} = 1$, $u_{(0,1)} = 3, u_{(1,6)} = 2, u_{(5,6)} = -1$, $u_{(0,3)} = 3, u_{(2,3)} = -1, u_{(3,4)} = 1$.

3.3 Генетски алгоритам

У овој секцији ћемо описати само разлике у GA за решавање CMDMBP у односу на MDMBP. Кодирање, генетски оператори, политика замене генерација и кеширање GA су потпуно исти и већ описани у претходном поглављу, те стога неће бити поново описивани.

Предложени GA за решавање CMDMBP је генерално сличан алгоритму 1, али уз неке измене функције циља и позивања процедуре локалне претраге. Због тога је целокупно представљен алгоритмом 7.

3.3.1 Функција циља и казнена функција

За разлику од функције циља из GA за претходни проблем, у случају CMDMBP се мора водити рачуна о томе да добијене партиције не морају бити повезане,

Algorithm 7: GA псеудо код

```
1  Input_Data() ;
2  Random_Init() ;
   while not Stopping_Criterion() do
       foreach ind from (N_elite + 1) to N_pop do
           if Exist_in_Cache(ind) then
3              obj_ind ← Get_Value_From_Cache(ind);
           end
           else
4              obj_ind ← Objective_Function(ind) ;
5              Local_Search(ind,obj_ind) ;
6              Put_Into_the_Cache_Memory(ind,obj_ind) ;
               if Full_Cache_Memory() then
                   Remove_LRU_Block_From_Cache_Memory();
               end
           end
7      end
8      Fitness_Function();
9      Selection();
10     Crossover();
11     Mutation();
   end
12 Output_Data();
```

што доводи до појаве недопустивих јединки. Стога се принцип рачунања функције циља мора унапредити, како би се решио проблем појаве некоректних решења. Један од често коришћених начина који ефективно разрешава дати недостатак је примена казнених функција. Уопштено, идеја казнених функција је свођење проблема оптимизације са ограничењима на проблем оптимизације без ограничења, чија решења конвергирају решењу проблема са ограничењима. Рецимо да је задатак решити следећи проблем са једним ограничењам

$$\max f(\mathbf{x})$$

тако да

$$c(\mathbf{x}) \geq 0.$$

Овај проблем са једним ограничењем се своди на следећи проблем максимизације

$$\max F_i(\mathbf{x}) = f(\mathbf{x}) - p_i(c(\mathbf{x})),$$

где је $p_i(c(\mathbf{x}))$ функција казне у i-тој итерацији. Ако је казнена функција иста у свим итерацијама онда имамо статичну, а у супротном имамо динамичку казнену функцију. Казнена функција може бити неодговарајућа по два основа. Са једне стране може да производи решења са већим локалним оптимумом али при том дозволивши недопустива решења. Са друге стране може да производи решења која су допустива, али је њихова вредност далеко од оптималне. Зато је неопходно наћи добар компромис између допустивости решења и квалитета локалног оптимума.

За решавање CMDMBP, као и за већину осталих проблема оптимизације са ограничењима, је неопходно обезбедити да се у каснијим фазама претраге, добијају (углавном) допустива решења. У наставку ћемо дефинисати једну казнену функцију која ће покушати да оствари зацртано.

Нека је

$$pen(i, k_1, k_2) = (k_1 + k_2 - 2) \cdot a_d \cdot maxw_i,$$

казнена функција, где су k_1 и k_2 бројеви повезаних компоненти S и $V \setminus S$, a_d је просечан степен чворова графа, и $maxw_i$ је максимална тежина за i-ту

координату вектора тежина. Очигледно је да је први фактор производа једнак нули ако оба скупа S и $V \setminus S$ индукују повезане графове. Фактор $a_d \cdot maxw_i$ у казненој функцији је изабран зато што је неопходно узети у обзир и природу самог графа. Тако је са повећавањем вредности функције циља потребно повећавати и вредност функције казне. Вредност функције циља се повећава у основи из два разлога: повећањем броја грана у односу на број чворова и повећањем вредности координата у вектору тежина на гранама. То значи да је за густе графове потребно повећати вредност казнене функције да би она имала ефекта, што се постиже фактором a_d, који представља просечан степен чворова датог графа. Наиме, повећање просечног степена чвора графа доводи до повећања броја грана у графу што доводи до повећања вредност функције циља. На сличан начин, $maxw_i$ одражава природу вектора тежина на датој координати. Повећање $maxw_i$, доводи до повећања вредности функције циља. Када је неки од индукованих графова неповезан, предложена казна се узима у разматрање, тј. казнена функција враћа вредност већу од нуле.

Као што се може видети из експерименталних резултата у поглављу 3.6, предложена функција казне представља добар компромис између допустивости решења и квалитета локалног оптимума. За сваку инстанцу, најмање једно (допустиво) решење локалног оптимума је пронађено. Такође, ова решења су одличног квалитета, тј. сва претходно позната глобално оптимална решења су достигнута, осим у једном случају за GA ($|V| = 30$, $|E| = 400$, $r = 15$).

3.3.2 Локална претрага

За разлику од две процедуре локалне претраге за решавање MDMBP описане у претходном поглављу, у овом случају (за решавање CMDMBP), биће примењена само једна, заснована на 1-замени, код које један чвор из S и један чвор из $V \setminus S$ мењају места. То је због тога што се казнена функција не може израчунавати преко добитака и ефикасних ажурирања, пошто се број компоненти повезаности партиције из које се чвор пребацује мора поново израчунати. Због тога, примена локалног претраживања заснованог на 1-пребацивању, у овом случају, уопште није сврсисходна, па није примењивана. Приметимо да је и процедура локалног претраживања заснована на 1-замени вишеструко спорија него у случају MDMBP због израчунавања казнене функције која захтева поновно израчунавање броја

компоненти повезаности. Да време извршавања овакве локалне претраге не би било предуго, уобичајена је пракса да се оно ограничи. Постоји неколико начина да се то постигне: ограничавањем броја решења која су тестирана у суседству, ограничавањем времена проведеном у локалној претрази итд. У предложеном GA је ограничен број итерација у локалној претрази на $lsMax$.

У свакој итерацији локалне претраге, алгоритам покушава да унапреди решење мењањем једног елемента $v_1 \in S$ једним елементом $v_2 \in V \setminus S$. Она је примењена на сваки пар $(v_1, v_2) \in S \times (V \setminus S)$ са стратегијом првог побољшања, што значи да када је прво побољшање постигнуто, оно је одмах примењено, и локална претрага почиње од почетка са побољшаним решењем као основним. Ако за сваки пар (v_1, v_2) замена производи вредности функције циља мањих или једнаких од оригиналне или је достигнуто $lsMax$ итерација, локална претрага завршава.

Algorithm 8: Псеудо код локалне претраге

```
   Algorithm Local search ;
1  lsCount ← 0 ;
   impr ← true;
   oldFunctionValue ← FuctionValue(Solution);
   while impr do
       lsCount ← lsCount + 1;
       if lsCount > lsMax then
           break;
       end
       impr ← false;
       foreach (v1, v2) ∈ S × (V \ S) do
           newFunctionValue ← lsF(v1, v2);
           if newFunctionValue > oldFunctionValue then
               oldFunctionFalue ← newFunctionValue;
               Solution ← SwapVertices(v1, v2);
               impr ← true;
           end
2      end
   end
```

У претходном алгоритму, $Solution$ представља текуће решење, $lsMax$ представља максимални број итерација локалне претраге, и $lsF(v_1, v_2)$ је функција циља са замењеним решењима, тј. $v_1 \in V \setminus S$ и $v_2 \in S$. Максимални број итерација локалне претраге је постављен на $lsMax = 20$.

Пошто је оваква процедура локалне претраге временски захтевна, њена примена на све јединке у свакој генерацији би могла да значајно успори извршење читавог оптимизационог процеса. Због тога се у предложеном GA она примењује само на најбољу јединку, само онда када се она променила (побољшала) и то не у првих 20 генерација. Последњи услов се примењује да се не би непотребно трошило рачунарско време на побољшавање јединки лошијег квалитета које се добијају у почетним фазама GA претраге.

3.4 Метода променљивих околина

Метода променљивих околина за решавање CMDMBP је састављена из три постојеће (већ описане) компоненте:

- Систем околина је потпуно исти као и за решавање MDMBP, а који је већ детаљно описан у претходном поглављу;
- Функција циља је потпуно иста као у GA за решавање CMDMBP, а која је већ детаљно описана у претходној секцији;
- Локална претрага је такође скоро потпуно иста као у GA за решавање CMDMBP, осим што је $lsMax = \infty$.

Приметимо да је детаљна шема VNS методе скоро потпуно иста као при решавању MDMBP, па је нећемо поново наводити. Функција циља и локална претрага су додуше другачије осмишљени, али су они исти као и у GA за CMDMBP.

Потребно је напоменути да иако је процедура локалне претраге потпуно иста као за остале две метахеуристике, учестаност њеног позивања је донекле различита. Наиме, код GA се позивала специфично (само за најбољу јединку и не у свакој генерацији), док се код VNS методе позива у сваком кораку (итерацији).

3.5 Метахеуристика заснована на електромагнетизму

И метода EM за решавање CMDMBP, потпуно исто као и VNS у претходној секцији, је такође састављена из три постојеће (већ описане) компоненте, па

их нема потребе понављати. Механизам привлачења/одбијања и процедура скалирања су потпуно исте као и у EM за решавање MDMBP. Пошто су оне већ потпуно описане у претходном поглављу, нема потребе да се поново описују. Као и у случају VNS, локална претрага је потпуно иста као у GA за решавање CMDMBP, али се такође позива нешто другачије. Наиме, процедура локалне претраге се у EM методи позива у свакој итерацији и то за сваку јединку.

3.6 Експериментални резултати

Сви експериментални резултати у овом раду су добијени извршавањем на AMD FX8300, 3.4GHz PC рачунару са 4GB RAM коришћењем само једног језгра. Сви алгоритми су имплементирани у програмском језику C, док су резултати извршавања комерцијалних решавача добијени коришћењем CPLEX 12.6 и Gurobi 5.6. Резултати су приказани на сличан начин као и у претходној глави. Због већег времена извршавања, приказани су само резултати на MDMBP инстанцама до 500 чворова и 60000 грана.

3.6.1 Експериментални резултати егзактних решавача

Табеле 3.1-3.7 садрже податке о извршавању егзактних решавача за инстанце са редом 10, 20, 30, 50, 100, 300 и 500 чворова. Прве две колоне за све табеле имају исто значење као и у претходној глави. Трећа колона у табелама 3.1-3.3 садржи оптималну вредност решења која је верификована бар једним од егзактних решавача, Наредних шест колона табеле 3.1 садрже податке о укупном времену извршавања (t_{tot}) и времену потребно да се достигне оптимално решење (t) за тоталну енумерацију, CPLEX и Gurobi, респективно.

Како CPLEX и Gurobi нису успели да верификују ниједно оптимално решење за $n \geq 20$, табеле 3.2 и 3.3 су организоване мало другачије. Првих пет колона имају исто значење као и код табеле 3.1, док наредне четири колоне садрже најбоље решење, односно најбоље горње ограничење које су достигли CPLEX односно Gurobi. Ако је неко од њих оптимално дата је ознака *opt*, а ако нека вредност није уопште достигнута, стоји ознака "-".

За инстанце већих димензија $n \geq 50$ ниједан од егзактних решавача није дошао до оптималног решења, па у табели имамо податке извршавања само за

Табела 3.1: Егзактни резултати решавача на инстанцама $n = 10$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t_{tot} (s)	t (s)	t_{tot} (s)	t (s)	t_{tot} (s)	t (s)
010_015	1	34.750	0.002	0.001	2.074	1.254	1.140	1
010_015	2	30.884	0.001	0.001	2.605	2.228	0.690	0
010_015	3	30.803	0.001	0.001	2.043	1.496	0.560	0
010_015	4	29.621	0.001	0.001	2.714	2.013	0.780	0
010_015	5	29.621	0.001	0.001	1.653	1.443	1.040	1
010_015	10	24.870	0.001	0.001	2.932	2.610	0.820	0
010_015	15	24.870	0.001	0.001	2.246	0.748	0.997	0
010_015	20	24.870	0.001	0.001	2.324	2.210	1.010	0
010_025	1	80.939	0.001	0.001	4.143	3.967	1.850	‘1
010_025	2	80.939	0.001	0.001	2.341	2.118	1.790	1
010_025	3	79.982	0.001	0.001	0.544	0.528	0.980	0
010_025	4	73.430	0.001	0.001	3.847	3.779	3.144	3
010_025	5	73.430	0.001	0.001	1.887	0.367	3.870	3
010_025	10	73.430	0.001	0.001	1.124	1.029	1.405	1
010_025	15	70.241	0.001	0.001	1.566	1.443	0.974	0
010_025	20	70.241	0.001	0.001	3.488	3.248	1.188	1
010_040	1	109.743	0.001	0.001	4.056	4.010	1.500	1
010_040	2	109.743	0.001	0.001	0.670	0.517	2.200	1
010_040	3	109.743	0.001	0.001	2.854	2.370	1.390	1
010_040	4	109.743	0.001	0.001	0.795	0.450	2.120	2
010_040	5	109.743	0.001	0.001	3.712	2.930	2.950	1
010_040	10	109.743	0.001	0.001	5.803	5.371	2.360	1
010_040	15	109.743	0.001	0.001	5.350	4.796	4.210	1
010_040	20	108.651	0.001	0.001	0.592	0.574	2.870	2

CPLEX и Gurobi. Зато у табелама 3.4-3.7 колоне 3–6 имају исто значење као колоне 6–9 табеле 3.3.

Уз помоћ егзактних решавача добијене су и верификоване оптималне вредности само за инстанце $n \leq 30$. Оптималне вредности су добила сва три решавача за $n = 10$, а за $n = 20$ и $n = 30$ само енумерација. CPLEX и Gurobi су за $n \geq 20$ добили (оптимална) решења у појединим случајевима и горње границе у свим случајевима.

Из табела 3.2 и 3.3 се може видети да је CPLEX достигао 7 оптималних решења, али их није верификовао, док је Gurobi достигао 3 неверификована оптимална решења. CPLEX је достигао 10 субоптималних решења, а Gurobi 12.

Решавачи CPLEX и Gurobi су добили само горња ограничења оптималних вредности решења проблема CMDMBP за инстанце за које је $n \geq 50$ и ти подаци су представљени у табелама 3.4 – 3.7.

Табела 3.2: Егзактни резултати решавача на инстанцама $n = 20$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t_{tot} (s)	t (s)	*Obj*	*UB*	*Obj*	*UB*
020_030	1	71.285	0.499	0.141	opt	85.854	57.858	100.206
020_030	2	57.952	0.483	0.187	opt	72.984	55.877	91.892
020_030	3	57.952	0.468	0.187	57.526	74.467	opt	81.363
020_030	4	54.856	0.483	0.202	opt	65.467	opt	75.929
020_030	5	53.691	0.485	0.109	opt	60.235	49.776	75.316
020_030	10	53.691	0.514	0.109	opt	58.095	opt	74.195
020_030	15	53.691	0.468	0.109	opt	60.103	43.670	76.294
020_030	20	49.555	0.483	0.156	40.588	67.073	44.443	73.197
020_070	1	238.063	1.224	0.499	223.865	250.973	-	250.973
020_070	2	238.063	1.232	0.514	212.575	250.973	236.318	250.973
020_070	3	238.063	1.263	0.514	202.731	246.420	220.170	246.420
020_070	4	238.063	1.326	0.562	192.838	246.420	-	246.420
020_070	5	238.063	1.295	0.531	-	246.420	220.051	246.420
020_070	10	238.063	1.438	0.592	-	244.988	211.309	244.988
020_070	15	230.212	1.591	0.632	-	244.988	217.061	244.988
020_070	20	230.212	1.761	0.686	-	244.988	216.163	244.988
020_150	1	502.411	2.465	0.187	-	opt	-	opt
020_150	2	476.472	2.592	1.263	-	opt	463.085	opt
020_150	3	466.079	2.652	1.232	-	opt	-	opt
020_150	4	466.079	2.792	1.326	-	opt	-	opt
020_150	5	455.679	2.808	1.108	405.347	opt	-	opt
020_150	10	442.486	3.088	1.076	-	opt	-	opt
020_150	15	440.870	3.432	1.170	-	442.004	-	opt
020_150	20	440.870	3.172	1.279	opt	opt	-	opt

3.6.2 Експериментални резултати добијени метахеуристикама

Приликом тестирања метахеуристика коришћени су исти параметри као и за метахеуристике у претходном поглављу, изузев за GA где је нешто промењен критеријум завршетка GA. Наиме, за решавање CMDMBP је максимални број генерације био постављен на 10000, а максимални број генерација без понављања на 4000. Критеријуми заустављања VNS и EM су постављени као средње максимално извршавање GA за дату инстанцу. Напоменимо да је време извршавања локалне претраге значајно веће него за MDMBP. Пошто се у овој имплементацији локална претрага не може прекинути, у неким случајевима t може бити нешто веће од t_{tot} за GA. У скоро свим случајевима t не прелази t_{tot} за више од 2.5% осим за инстанцу 300_500 и $r = 20$, где је t_{tot} за GA 33.279 секунди, а t за EM је 35.549 секунди. Колона nf представља колико пута од 20 извршавања је добијено допустиво решење.

Табела 3.3: Егзактни резултати решавача на инстанцама $n = 30$

inst.	r	opt.	enum.		CPLEX		Gurobi	
			t_{tot} (s)	t (s)	*Obj*	*UB*	*Obj*	*UB*
030_050	1	137.059	172.2	22.2	-	193.318	-	191.715
030_050	2	137.059	172.3	22.0	-	193.318	-	191.299
030_050	3	137.059	172.0	22.03	-	188.292	-	185.953
030_050	4	129.174	175.0	35.13	-	188.128	-	186.164
030_050	5	125.434	174.5	47.50	-	186.818	-	184.287
030_050	10	115.110	178.6	14.57	96.902	175.547	-	170.811
030_050	15	112.075	179.2	14.20	-	165.758	-	163.513
030_050	20	111.072	181.3	61.24	84.974	165.680	-	161.539
030_150	1	589.593	513.7	178.8	-	opt	-	opt
030_150	2	532.589	523.3	183.2	420.211	536.473	-	536.473
030_150	3	530.556	542.5	188.8	-	533.635	-	533.635
030_150	4	530.556	574.6	198.6	-	533.635	-	533.635
030_150	5	530.556	581.2	202.3	-	533.635	-	533.635
030_150	10	519.005	698.5	246.9	-	525.300	-	525.300
030_150	15	519.005	791.6	279.9	-	519.586	-	519.586
030_150	20	518.672	903.2	315.6	-	519.586	-	519.586
030_400	1	1331.773	1325	424.8	-	1565.534	-	1391.360
030_400	2	1230.856	1377	480.2	-	1502.537	-	1313.595
030_400	3	1208.703	1421	126.8	-	1486.413	-	1301.961
030_400	4	1187.753	1473	513.5	-	1486.214	-	1300.283
030_400	5	1181.987	1500	42.26	-	1469.257	-	1296.115
030_400	10	1154.353	1774	495.9	-	1483.129	-	1294.961
030_400	15	1142.317	2122	428.0	-	1455.621	-	1268.590
030_400	20	1140.635	2394	465.0	-	1461.820	-	1268.646

3.7 Анализа резултата и завршна разматрања

Из табела 3.1-3.7 се може видети да егзактне методе показују још лошије резултате него за MDMBP разматран у претходној глави. Егзактне методе дају оптимална решења за $n = 10$ док за $n \geq 20$ решавачи CPLEX и Gurobi не дају ни једно верификовано оптимално решење. За $n \geq 50$ ниједан од решавача не даје уопште решења у оквиру датог временског ограничења од 7200 секунди, а решавачи CPLEX и Gurobi дају само горња ограничења вредности оптималних решења. Из тог разлога је развој метахеуристичких приступа решавању CMDMBP веома значајан.

Као што се може видети у Табелама 3.8 - 3.10, све три предложене метахеуристичке методе су достигле сва позната оптимална решења на графовима до 30 чворова, у релативно кратком временском периоду мањем од једне секунде, осим GA на инстанци са 30 чворова и 400 грана.

Као што се може видети из табела 3.11 - 3.14, од укупно 120 инстанци где није познато оптимално решење, метахеуристике су у свим случајевима дошле до бар једног допустивог решења. Од тога је GA добио 89 најбољих решења, VNS 46, а EM 38. Напоменимо да су у 19 случајева све три метахеуристике

Табела 3.4: Приближни резултати решавача за $n = 50$

inst.	r	CPLEX		Gurobi	
		Obj	*UB*	*Obj*	*UB*
050_080	1	-	306.736	-	306.529
050_080	2	-	285.214	-	285.158
050_080	3	-	285.214	-	285.212
050_080	4	-	276.616	-	276.578
050_080	5	-	275.155	-	275.155
050_080	10	-	273.597	-	272.763
050_080	15	-	269.578	-	268.156
050_080	20	-	269.318	-	268.156
050_300	1	-	1239.456	-	1127.218
050_300	2	-	1225.140	-	1098.891
050_300	3	-	1217.340	-	1098.891
050_300	4	-	1214.856	-	1094.621
050_300	5	-	1231.340	-	1100.505
050_300	10	-	1205.792	-	1076.105
050_300	15	-	1203.418	-	1067.490
050_300	20	-	1206.485	-	1062.553
050_1k	1	-	4276.988	-	4035.271
050_1k	2	-	4204.742	-	4111.562
050_1k	3	-	4196.904	-	4033.964
050_1k	4	-	4057.247	-	3936.653
050_1k	5	-	4036.394	-	3957.950
050_1k	10	-	4062.104	-	3923.557
050_1k	15	-	4004.614	-	3967.787
050_1k	20	-	4046.152	-	3955.236

добиле иста решења.

Време извршавања метахеуристика је углавном достижно, најчешће мање од 10 минута, осим инстанци 300_30k и 500_60k, који представљају врло густе графове велике димензије.

Представљени експериментални резултати јасно демонстрирају корисност предложеног MILP модела, као и GA, EM и VNS метахеуристика за решавање проблема CMDMBP. Предложени MILP модел, за који је доказана коректност, представља вредан допринос за егзактно и приближно решавање CMDMBP. Свака од предложених метахеуристичких метода има неких својих предности, па тако VNS даје најбоља достигнута решења у 22 од 32 случаја за графове са 100 чворова, GA даје најбоља достигнута решења за 63 од 64 случаја за графове са 300 и 500 чворова, док за графове са 30 чворова EM најбрже достиже оптимална решења у 14 од 24 случаја. Ипак, укупни резултати јасно показују да GA даје значајно боље резултата у односу на EM и VNS.

Табела 3.5: Приближни резултати решавача за $n = 100$

inst.	r	CPLEX		Gurobi	
		Obj	*UB*	*Obj*	*UB*
100_150	1	-	568.223	-	568.223
100_150	2	-	535.212	-	534.635
100_150	3	-	532.700	-	532.247
100_150	4	-	532.816	-	532.170
100_150	5	-	531.625	-	531.379
100_150	10	-	524.281	-	522.605
100_150	15	-	515.820	-	514.682
100_150	20	-	500.644	-	499.331
100_500	1	-	2215.634	-	2078.733
100_500	2	-	2148.910	-	2057.091
100_500	3	-	2123.734	-	2040.489
100_500	4	-	2134.294	-	2060.152
100_500	5	-	2111.573	-	1996.171
100_500	10	-	2092.021	-	2000.919
100_500	15	-	2068.531	-	1975.995
100_500	20	-	2042.519	-	1984.093
100_1.5k	1	-	6456.068	-	6511.752
100_1.5k	2	-	6508.191	-	6483.175
100_1.5k	3	-	6436.800	-	6448.598
100_1.5k	4	-	6434.128	-	6382.496
100_1.5k	5	-	6302.594	-	6409.240
100_1.5k	10	-	6262.907	-	6355.897
100_1.5k	15	-	6252.872	-	6397.839
100_1.5k	20	-	6295.912	-	6394.270
100_3k	1	-	12890.147	-	12960.772
100_3k	2	-	12873.541	-	13057.957
100_3k	3	-	12857.005	-	13062.244
100_3k	4	-	12835.700	-	13065.684
100_3k	5	-	12720.750	-	12879.971
100_3k	10	-	12679.701	-	12851.004
100_3k	15	-	12618.178	-	12905.474
100_3k	20	-	13132.399	-	12871.523

Табела 3.6: Приближни резултати решавача за $n = 300$

inst.	r	CPLEX		Gurobi	
		Obj	*UB*	*Obj*	*UB*
300_500	1	-	2079.573	-	2079.462
300_500	2	-	2010.678	-	2010.463
300_500	3	-	1963.331	-	1963.240
300_500	4	-	1955.244	-	1955.572
300_500	5	-	1945.354	-	1945.337
300_500	10	-	1901.643	-	1901.500
300_500	15	-	1898.271	-	1898.195
300_500	20	-	1885.694	-	1885.180
300_2k	1	-	9662.085	-	9942.106
300_2k	2	-	9583.047	-	9841.172
300_2k	3	-	9537.538	-	9929.543
300_2k	4	-	9554.433	-	9895.460
300_2k	5	-	9491.205	-	9918.199
300_2k	10	-	9498.908	-	9855.687
300_2k	15	-	9386.662	-	9772.943
300_2k	20	-	9409.676	-	9737.626
300_10k	1	-	54841.066	-	50346.727
300_10k	2	-	54841.066	-	50273.724
300_10k	3	-	54841.066	-	50303.294
300_10k	4	-	54841.066	-	50252.176
300_10k	5	-	54499.485	-	50059.626
300_10k	10	-	54499.485	-	50155.933
300_10k	15	-	54499.485	-	50250.613
300_10k	20	-	54499.485	-	50000.338
300_30k	1	-	164263.369	-	162269.04
300_30k	2	-	164263.369	-	162639.00
300_30k	3	-	164263.369	-	163445.16
300_30k	4	-	164263.369	-	162628.15
300_30k	5	-	164263.369	-	162995.23
300_30k	10	-	164263.369	-	162650.54
300_30k	15	-	164263.369	-	162591.17
300_30k	20	-	164263.369	-	162650.83

Табела 3.7: Приближни резултати решавача за $n = 500$

inst.	r	CPLEX		Gurobi	
		Obj	*UB*	*Obj*	*UB*
500_1k	1	-	4285.084	-	4314.408
500_1k	2	-	4199.150	-	4229.441
500_1k	3	-	4148.080	-	4184.994
500_1k	4	-	4122.734	-	4133.956
500_1k	5	-	4110.383	-	4131.120
500_1k	10	-	4056.694	-	4075.582
500_1k	15	-	4056.358	-	4090.711
500_1k	20	-	4028.691	-	4049.712
500_3k	1	-	14595.865	-	15219.802
500_3k	2	-	14521.286	-	15209.127
500_3k	3	-	14503.344	-	15216.722
500_3k	4	-	14445.535	-	15191.969
500_3k	5	-	14410.714	-	15144.083
500_3k	10	-	14334.349	-	15179.324
500_3k	15	-	14236.182	-	15020.860
500_3k	20	-	14247.541	-	15032.847
500_10k	1	-	55093.070	-	52561.925
500_10k	2	-	55048.738	-	52102.044
500_10k	3	-	55048.738	-	52158.544
500_10k	4	-	54501.496	-	51466.573
500_10k	5	-	54501.496	-	51889.531
500_10k	10	-	54501.496	-	51573.422
500_10k	15	-	54501.496	-	51812.270
500_10k	20	-	54501.496	-	51480.227
500_60k	1	-	329331.852	-	329331.85
500_60k	2	-	329255.253	-	329255.25
500_60k	3	-	329255.253	-	329255.25
500_60k	4	-	329255.253	-	329255.25
500_60k	5	-	329255.253	-	329255.25
500_60k	10	-	328970.410	-	328970.41
500_60k	15	-	328970.410	-	328970.41
500_60k	20	-	328694.683	-	328694.68

Табела 3.8: Резултати метахеуристика на инстанцама са познатим оптималним решењем $n = 10$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
010_015	1	34.750	opt	20	0.578	0.001	opt	20	0.00270	opt	20	0.002
010_015	2	30.884	opt	20	0.653	0.001	opt	20	0.00270	opt	20	0.002
010_015	3	30.803	opt	20	0.677	0.001	opt	20	0.00135	opt	20	0.001
010_015	4	29.621	opt	20	0.734	0.001	opt	20	0.00145	opt	20	0.001
010_015	5	29.621	opt	20	0.729	0.001	opt	20	0.00145	opt	20	0.001
010_015	10	24.870	opt	20	0.784	0.001	opt	20	0.00180	opt	20	0.001
010_015	15	24.870	opt	20	0.806	0.001	opt	20	0.00205	opt	20	0.001
010_015	20	24.870	opt	20	0.811	0.001	opt	20	0.00155	opt	20	0.002
010_025	1	80.939	opt	20	0.795	0.001	opt	20	0.00175	opt	20	0.001
010_025	2	80.939	opt	20	0.799	0.001	opt	20	0.00215	opt	20	0.002
010_025	3	79.982	opt	20	0.785	0.001	opt	20	0.00175	opt	20	0.002
010_025	4	73.430	opt	20	0.818	0.001	opt	20	0.00180	opt	20	0.002
010_025	5	73.430	opt	20	0.797	0.001	opt	20	0.00165	opt	20	0.002
010_025	10	73.430	opt	20	0.835	0.001	opt	20	0.00175	opt	20	0.003
010_025	15	70.241	opt	20	0.866	0.001	opt	20	0.00185	opt	20	0.003
010_025	20	70.241	opt	20	0.910	0.001	opt	20	0.00165	opt	20	0.002
010_040	1	109.743	opt	20	0.859	0.001	opt	20	0.00195	opt	20	0.002
010_040	2	109.743	opt	20	0.836	0.001	opt	20	0.00185	opt	20	0.002
010_040	3	109.743	opt	20	0.870	0.001	opt	20	0.00225	opt	20	0.003
010_040	4	109.743	opt	20	0.871	0.001	opt	20	0.00220	opt	20	0.002
010_040	5	109.743	opt	20	0.878	0.001	opt	20	0.00230	opt	20	0.002
010_040	10	109.743	opt	20	0.936	0.001	opt	20	0.00205	opt	20	0.003
010_040	15	109.743	opt	20	0.988	0.001	opt	20	0.00195	opt	20	0.003
010_040	20	108.651	opt	20	1.019	0.001	opt	20	0.00260	opt	20	0.002

Табела 3.9: Резултати метахеуристика на инстанцама са познатим оптималним решењем $n = 20$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
020_030	1	71.285	opt	20	1.421	0.006	opt	20	0.00305	opt	20	0.007
020_030	2	57.952	opt	20	1.473	0.058	opt	20	0.00345	opt	20	0.009
020_030	3	57.952	opt	20	1.635	0.158	opt	20	0.00505	opt	20	0.007
020_030	4	54.856	opt	20	1.527	0.044	opt	20	0.06380	opt	20	0.010
020_030	5	53.691	opt	20	1.837	0.312	opt	20	0.03980	opt	20	0.016
020_030	10	53.691	opt	20	2.049	0.416	opt	20	0.04720	opt	20	0.015
020_030	15	53.691	opt	20	1.826	0.133	opt	20	0.01550	opt	20	0.010
020_030	20	49.555	opt	20	2.200	0.363	opt	20	0.00760	opt	20	0.007
020_070	1	238.063	opt	20	2.059	0.184	opt	20	0.00880	opt	20	0.007
020_070	2	238.063	opt	20	2.120	0.182	opt	20	0.00660	opt	20	0.005
020_070	3	238.063	opt	20	1.869	0.029	opt	20	0.00595	opt	20	0.003
020_070	4	238.063	opt	20	2.328	0.365	opt	20	0.00595	opt	20	0.004
020_070	5	238.063	opt	20	2.381	0.304	opt	20	0.00590	opt	20	0.003
020_070	10	238.063	opt	20	2.418	0.149	opt	20	0.00575	opt	20	0.005
020_070	15	230.212	opt	20	2.722	0.276	opt	20	0.00765	opt	20	0.007
020_070	20	230.212	opt	20	2.685	0.154	opt	20	0.00775	opt	20	0.008
020_150	1	502.411	opt	20	3.208	0.012	opt	20	0.00925	opt	20	0.002
020_150	2	476.472	opt	20	3.349	0.010	opt	20	0.00860	opt	20	0.006
020_150	3	466.079	opt	20	3.235	0.012	opt	20	0.00715	opt	20	0.005
020_150	4	466.079	opt	20	3.140	0.010	opt	20	0.00745	opt	20	0.005
020_150	5	455.679	opt	20	3.438	0.173	opt	20	0.01500	opt	20	0.015
020_150	10	442.486	opt	20	4.046	0.310	opt	20	0.01415	opt	20	0.019
020_150	15	440.870	opt	20	5.431	0.643	opt	20	0.01520	opt	20	0.020
020_150	20	440.870	opt	20	5.927	0.441	opt	20	0.01790	opt	20	0.022

Табела 3.10: Резултати метахеуристика на инстанцама са познатим оптималним решењем $n = 30$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
030_050	1	137.059	opt	20	2.960	0.378	opt	20	0.2529	opt	20	0.05370
030_050	2	137.059	opt	20	3.078	0.519	opt	20	0.4182	opt	20	0.05135
030_050	3	137.059	opt	20	2.857	0.524	opt	20	0.1078	opt	20	0.11305
030_050	4	129.174	opt	20	3.208	0.475	opt	20	0.2510	opt	20	0.10370
030_050	5	125.434	opt	20	3.422	0.514	opt	20	0.4175	opt	20	0.63630
030_050	10	115.110	opt	20	3.710	0.771	opt	20	0.6071	opt	20	0.17920
030_050	15	112.075	opt	20	3.926	0.546	opt	20	0.3196	opt	20	0.77030
030_050	20	111.072	opt	20	3.681	0.577	opt	20	0.3452	opt	20	0.68685
030_150	1	589.593	opt	20	3.331	0.061	opt	20	0.0176	opt	20	0.02485
030_150	2	532.589	opt	20	5.404	0.821	opt	20	0.0829	opt	20	1.19475
030_150	3	530.556	opt	20	3.996	0.045	opt	20	0.0237	opt	20	0.09185
030_150	4	530.556	opt	20	4.257	0.059	opt	20	0.0268	opt	20	0.14045
030_150	5	530.556	opt	20	4.544	0.019	opt	20	0.0346	opt	20	0.18950
030_150	10	519.005	opt	20	6.117	0.058	opt	20	0.0336	opt	20	0.04420
030_150	15	519.005	opt	20	6.091	0.029	opt	20	0.0629	opt	20	0.35085
030_150	20	518.672	opt	20	6.798	0.204	opt	20	0.0305	opt	20	0.52795
030_400	1	1331.773	opt	20	10.139	0.054	opt	20	0.0379	opt	20	0.04435
030_400	2	1230.856	opt	20	8.994	0.104	opt	20	0.0310	opt	20	0.09665
030_400	3	1208.703	opt	20	9.552	0.162	opt	20	0.6435	opt	20	0.58085
030_400	4	1187.753	opt	20	10.955	0.548	opt	20	0.1295	opt	20	0.21130
030_400	5	1181.987	opt	20	11.199	0.736	opt	20	0.0702	opt	20	0.18235
030_400	10	1154.353	opt	20	10.329	0.518	opt	20	0.0509	opt	20	0.18880
030_400	15	1142.317	1142.251	20	11.157	1.226	opt	20	1.7548	opt	20	0.73100
030_400	20	1140.635	opt	20	15.860	0.486	opt	20	0.0652	opt	20	0.10755

Табела 3.11: Резултати метахеуристика на инстанцама са непознатим оптималним решењем $n = 50$

inst.	r	najb.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
050_080	1	213.346	213.298	20	4.334	0.686	najb.	20	0.785	najb.	20	0.79060
050_080	2	187.251	186.197	20	3.952	0.417	186.756	20	0.877	najb.	20	1.26560
050_080	3	187.362	186.197	20	3.632	0.618	najb.	20	0.910	186.852	20	1.32825
050_080	4	184.064	182.090	20	5.372	1.130	najb.	20	1.593	najb.	20	1.95825
050_080	5	174.214	171.281	20	6.960	0.957	najb.	20	1.940	najb.	20	1.80395
050_080	10	172.162	167.405	20	5.496	0.486	najb.	20	1.406	najb.	20	1.45830
050_080	15	164.649	162.964	20	4.430	0.209	164.459	20	0.897	najb.	20	1.47640
050_080	20	162.555	159.925	20	4.407	0.588	162.246	20	1.260	najb.	20	1.30710
050_300	1	1124.331	najb.	20	6.057	0.434	najb.	20	0.690	najb.	20	0.20970
050_300	2	1098.891	najb.	20	7.639	1.083	najb.	20	0.861	najb.	20	0.41490
050_300	3	1098.891	najb.	20	4.038	0.393	najb.	20	0.693	najb.	20	0.48490
050_300	4	1094.621	najb.	20	7.271	1.071	najb.	20	0.898	najb.	20	0.42180
050_300	5	1094.621	najb.	20	5.707	0.286	najb.	20	0.729	najb.	20	0.48340
050_300	10	1076.105	najb.	20	4.115	0.19	najb.	20	0.704	najb.	20	0.50305
050_300	15	1062.553	najb.	20	11.027	0.685	najb.	20	1.111	najb.	20	0.92425
050_300	20	1062.553	najb.	20	12.419	0.975	najb.	20	0.993	najb.	20	1.78615
050_1k	1	3099.083	najb.	20	21.005	1.083	najb.	20	2.112	najb.	20	2.37280
050_1k	2	3037.537	najb.	20	17.524	4.158	najb.	20	4.592	najb.	20	4.28600
050_1k	3	3006.541	najb.	20	24.353	0.804	najb.	20	2.989	najb.	20	2.54515
050_1k	4	2930.936	najb.	20	21.899	2.041	najb.	20	3.722	najb.	20	4.34140
050_1k	5	2930.936	najb.	20	24.847	1.797	najb.	20	5.922	najb.	20	3.97090
050_1k	10	2901.349	najb.	20	26.741	6.101	najb.	20	3.821	najb.	20	5.19155
050_1k	15	2881.119	najb.	20	43.279	7.923	najb.	20	4.496	najb.	20	4.12530
050_1k	20	2875.188	najb.	20	28.495	2.189	najb.	20	3.953	najb.	20	4.67845

Табела 3.12: Резултати метахеуристика на инстанцама са непознатим оптималним решењем $n = 100$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
100_150	1	338.888	314.618	15	4.902	0.552	327.774	20	3.784	najb.	20	3.5723
100_150	2	313.987	313.312	17	5.47	0.613	312.417	20	4.467	najb.	20	3.1332
100_150	3	298.090	287.949	17	5.088	0.72	293.252	20	3.786	najb.	20	3.3181
100_150	4	298.090	290.206	18	5.027	0.504	292.901	20	3.398	najb.	20	3.4342
100_150	5	293.550	290.242	17	5.242	0.599	290.541	20	2.936	najb.	20	3.5598
100_150	10	282.111	274.365	17	5.724	0.405	280.661	20	4.485	najb.	20	3.1534
100_150	15	271.935	najb.	17	5.393	0.54	268.23	20	4.024	268.751	20	3.5357
100_150	20	265.247	262.701	16	5.68	0.822	najb.	20	5.087	265.189	20	3.3694
100_500	1	1973.881	najb.	20	6.008	0.246	najb.	20	4.989	najb.	20	2.97265
100_500	2	1902.448	1896.735	20	5.673	0.326	najb.	20	3.890	najb.	20	3.65115
100_500	3	1877.486	1869.455	20	6.133	0.355	najb.	20	5.290	1872.815	20	4.24865
100_500	4	1877.486	1873.909	20	6.069	0.544	najb.	20	4.881	1863.479	20	3.30255
100_500	5	1852.652	1850.259	20	6.760	0.676	najb.	20	5.781	najb.	20	4.85545
100_500	10	1833.645	1814.887	20	6.313	0.314	1830.954	20	5.554	najb.	20	4.10280
100_500	15	1812.805	1794.856	20	8.049	1.669	1812.080	20	6.081	najb.	20	5.01160
100_500	20	1788.899	1780.249	20	8.032	0.918	najb.	20	7.147	1788.378	20	4.47330
100_1.5k	1	5082.071	najb.	20	13.248	1.514	najb.	20	10.632	najb.	20	7.36640
100_1.5k	2	5002.550	4989.542	20	14.254	2.257	najb.	20	11.156	najb.	20	8.08465
100_1.5k	3	4975.478	4963.852	20	14.925	2.187	najb.	20	12.318	najb.	20	8.56855
100_1.5k	4	4898.990	najb.	20	15.206	2.327	4897.422	20	13.166	4897.422	20	9.40765
100_1.5k	5	4897.100	4889.522	20	14.009	1.202	najb.	20	12.270	4895.688	20	7.46080
100_1.5k	10	4871.164	4851.757	20	16.983	2.32	najb.	20	14.074	najb.	20	7.74320
100_1.5k	15	4851.995	najb.	20	16.797	1.387	najb.	20	12.934	najb.	20	9.38565
100_1.5k	20	4851.995	4824.689	20	18.416	1.222	najb.	20	15.442	najb.	20	12.14720
100_3k	1	9348.949	najb.	20	21.434	3.064	najb.	20	19.552	najb.	20	14.41885
100_3k	2	9142.080	najb.	20	21.340	2.519	najb.	20	18.812	najb.	20	16.75895
100_3k	3	9059.661	najb.	20	20.781	1.588	9039.517	20	19.055	najb.	20	13.85800
100_3k	4	9046.557	9029.164	20	47.031	1.814	najb.	20	34.04	9041.415	20	27.16105
100_3k	5	8995.853	8994.863	20	21.377	1.734	najb.	20	17.129	najb.	20	13.16150
100_3k	10	8873.866	najb.	20	24.171	1.702	8848.31	20	22.650	8866.421	20	14.62410
100_3k	15	8820.340	8790.527	20	27.477	3.272	8816.297	20	25.060	najb.	20	16.52020
100_3k	20	8794.929	najb.	20	29.878	3.921	8780.825	20	25.666	8792.409	20	23.51835

Табела 3.13: Резултати метахеуристика на инстанцама са непознатим оптималним решењем $n = 300$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
. 300_500	1	1324.418	najb.	14	33.551	9.303	1194.938	12	34.337	1240.124	13	30.60023
300_500	2	1274.133	najb.	17	31.859	7.888	1135.834	12	30.590	1199.421	13	22.96569
300_500	3	1183.050	najb.	13	29.339	5.565	1116.812	11	27.693	1167.628	11	25.48891
300_500	4	1177.683	najb.	15	30.990	4.837	1124.730	10	31.610	1163.140	13	25.78100
300_500	5	1169.595	najb.	18	30.734	5.39	1104.850	9	29.420	1130.179	9	26.43511
300_500	10	1110.863	najb.	14	31.604	4.832	1043.709	6	30.262	1103.358	10	26.91010
300_500	15	1100.779	najb.	16	31.372	4.524	1097.490	4	30.787	1080.824	13	26.50569
300_500	20	1098.828	najb.	13	33.279	6.156	1032.913	3	35.549	1075.401	14	29.88843
300_2k	1	7773.632	najb.	20	53.989	20.854	7489.231	20	53.724	7695.474	20	46.74635
300_2k	2	7675.004	najb.	20	48.316	17.023	7213.019	20	48.102	7500.574	20	44.74890
300_2k	3	7609.954	najb.	20	48.14	14.533	7259.322	20	47.176	7455.742	20	43.35725
300_2k	4	7600.075	najb.	20	50.111	16.514	7266.144	20	49.425	7421.144	20	47.35235
300_2k	5	7567.449	najb.	20	51.252	18.417	7187.469	20	50.248	7392.934	20	44.86815
300_2k	10	7462.868	najb.	20	59.411	23.697	7160.807	20	58.515	7341.555	20	54.15335
300_2k	15	7496.399	najb.	20	64.006	25.476	7057.088	20	63.689	7327.285	20	56.40665
300_2k	20	7419.318	najb.	20	72.483	31.264	7017.887	20	70.810	7248.907	20	62.54130
300_10k	1	32395.438	najb.	20	154.478	78.029	31531.789	20	53.724	32212.270	20	146.44915
300_10k	2	31980.797	najb.	20	162.23	81.213	31159.597	20	48.102	31674.133	20	154.44650
300_10k	3	31875.891	najb.	20	173.408	88.26	31099.041	20	47.176	31492.466	20	158.36600
300_10k	4	31857.961	najb.	20	160.297	74.808	31043.681	20	49.425	31520.111	20	142.27455
300_10k	5	31797.794	najb.	20	163.371	73.747	30724.307	20	50.248	31343.375	20	154.84115
300_10k	10	31613.164	najb.	20	198.737	95.781	30601.736	20	58.515	31347.247	20	187.81280
300_10k	15	31620.138	najb.	20	205.875	89.715	30427.956	20	63.689	31072.350	20	186.73130
300_10k	20	31551.037	najb.	20	250.351	116.643	30525.569	20	70.810	31064.377	20	237.75435
300_30k	1	88839.192	najb.	20	401.954	195.968	87256.816	20	402.877	88105.08	20	379.45595
300_30k	2	88330.651	najb.	20	435.061	222.295	86701.548	20	421.220	87574.98	20	400.79350
300_30k	3	88217.846	najb.	20	413.405	182.765	86676.523	20	407.370	87449.47	20	376.38895
300_30k	4	87872.911	najb.	20	442.317	211.966	86533.205	20	427.806	87425.58	20	407.32770
300_30k	5	87892.157	najb.	20	446.153	203.368	86404.757	20	433.422	87165.92	20	418.88040
300_30k	10	87537.174	najb.	20	598.562	287.118	86318.197	20	593.914	87105.67	20	554.33690
300_30k	15	87399.836	najb.	20	710.418	329.221	86070.883	20	683.943	86729.44	20	660.34465
300_30k	20	87413.270	najb.	20	920.835	460.381	85911.417	20	876.596	86766.54	20	832.56175

Табела 3.14: Резултати метахеуристика на инстанцама са непознатим оптималним решењем $n = 500$

inst.	r	opt.	GA				EM			VNS		
			vred.	nf	t_{tot}	t	*vred.*	nf	t	*vred.*	nf	t
500_1k	1	2992.699	najb.	19	99.763	48.551	2850.224	19	96.844	2976.415	20	94.10360
500_1k	2	2916.457	2894.228	20	88.949	33.893	2790.726	19	86.213	najb.	20	80.80705
500_1k	3	2851.493	najb.	19	92.957	40.858	2752.998	17	89.535	2821.687	20	86.71245
500_1k	4	2834.832	najb.	19	93.158	37.723	2752.695	18	90.148	2798.717	18	80.43689
500_1k	5	2795.359	najb.	19	87.841	35.826	2712.305	17	83.209	2773.204	19	81.62968
500_1k	10	2753.008	najb.	18	88.926	32.487	2608.400	10	86.417	2701.594	17	80.76065
500_1k	15	2756.598	najb.	19	88.783	34.403	1124.730	10	31.610	2697.272	18	76.65367
500_1k	20	2732.181	najb.	19	92.758	37.563	2594.619	10	91.080	2678.964	17	80.00024
500_3k	1	11737.433	najb.	20	178.537	119.353	11020.996	20	177.320	11482.514	20	165.50290
500_3k	2	11577.545	najb.	20	152.292	92.03	10827.773	20	151.134	11212.122	20	142.96080
500_3k	3	11550.756	najb.	20	167.264	107.057	10705.749	20	164.613	11201.028	20	160.03350
500_3k	4	11438.581	najb.	20	152.809	92.809	10633.852	20	151.976	11082.204	20	140.48900
500_3k	5	11460.073	najb.	20	155.986	94.525	10656.311	20	153.824	11074.247	20	147.15965
500_3k	10	11338.350	najb.	20	178.348	113.773	10485.542	20	173.920	11021.250	20	165.95000
500_3k	15	11325.421	najb.	20	186.358	117.316	10329.541	20	184.500	10780.427	20	173.23805
500_3k	20	11343.767	najb.	20	224.537	153.41	10384.989	20	223.328	10928.310	20	204.22365
500_10k	1	34291.616	najb.	20	390.031	294.212	32719.908	20	389.155	33713.942	20	371.96925
500_10k	2	33920.113	najb.	20	377.236	276.401	32092.809	20	373.194	33257.534	20	345.60920
500_10k	3	33718.914	najb.	20	355.452	252.386	32225.738	20	347.424	32918.884	20	337.05260
500_10k	4	33551.955	najb.	20	331.963	228.406	31410.895	20	330.815	32505.223	20	312.61495
500_10k	5	33540.110	najb.	20	334.847	226.348	31469.935	20	331.408	32381.445	20	309.81965
500_10k	10	33265.429	najb.	20	444.867	322.158	31409.739	20	436.970	32434.728	20	420.96440
500_10k	15	33362.961	najb.	20	547.279	408.672	31312.944	20	538.762	32462.614	20	515.48810
500_10k	20	33446.111	najb.	20	569.966	413.313	31471.831	20	554.388	32437.646	20	547.02810
500_60k	1	178038.795	najb.	20	2501.887	2089.949	174558.871	20	2490.037	176760.704	20	2340.16380
500_60k	2	176789.554	najb.	20	2606.552	2128.514	173654.159	20	2566.521	175450.818	20	2487.39050
500_60k	3	176502.062	najb.	20	2429.732	1920.088	174175.611	20	2388.965	174823.531	20	2261.98180
500_60k	4	176526.991	najb.	20	2782.215	2193.436	172653.829	20	2768.300	174591.743	20	2620.08145
500_60k	5	176203.278	najb.	20	3158.016	2536.732	173184.553	20	3102.322	174428.498	20	2963.04555
500_60k	10	175823.936	najb.	20	3264.921	2470.392	171939.859	20	3162.962	174259.165	20	3029.72375
500_60k	15	175662.787	najb.	20	3874.2	3001.442	172286.55	20	3796.903	174054.871	20	3684.49270
500_60k	20	175501.649	najb.	20	4371.987	3402.462	171553.886	20	4253.287	173932.268	20	4163.82765

Глава 4

Проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена

4.1 Дефиниција проблема

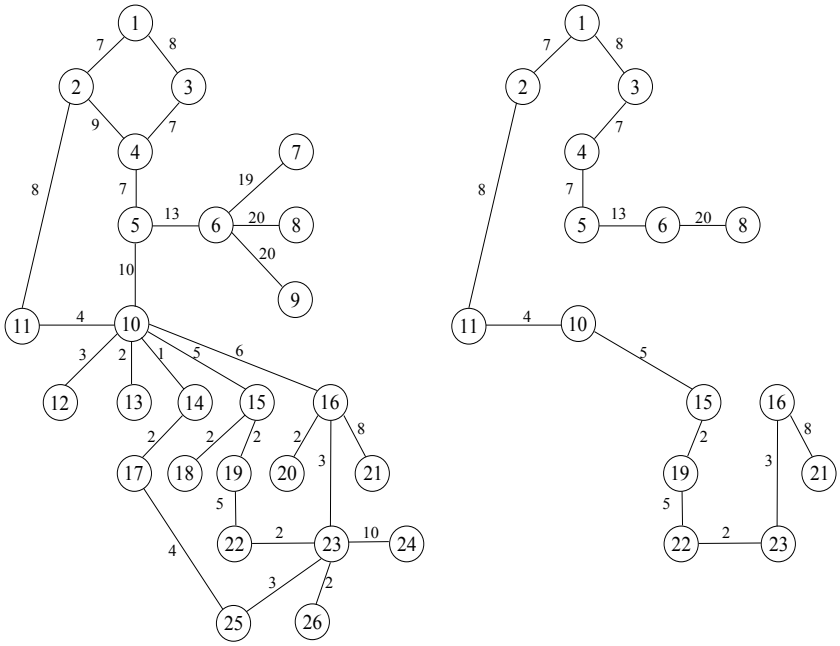
Нека је $G = (V, E)$ прост граф, $d \geq 2$ природан број и $w : E \to \mathbb{R}^+$ тежинска функција. Проблем проналажења повезаног подграфа највеће тежине са чворовима ограниченог степена (енг. Maximum Degree Bounded Connected Subgraph Problem - MDBCSP) захтева проналажење подграфа $G' = (V', E')$, где је $V' \subseteq V$ и $E' \subseteq E$, тако да је подграф G' повезан, степен чворова не прелази d и $\sum_{e \in E'} w(e)$ има максималну вредност. MDBCSP може бити илустрован следећим примером.

Пример 4.1. *Нека је* $|V| = 26$, $|E| = 29$ *и граф* $G = (V, E)$ *са тежинама приказан на левој страни слике 4.1.*

Оптимално решење за $d = 2$ је приказано на десној страни слике 4.1, чија је вредност 99. Овај резултат је добијен коришћењем нове MILP формулације за MDBCSP која је уведена у следећем одељку.

4.2 Преглед постојећих резултата

MDBCSP је један од класичних NP-тешких проблема приказаних у монографији Гарија и Џонсона [21] (као проблем GT26), за који не постоји полиноми-



Слика 4.1: Граф G и одговарајуће оптимално решење

јални апроксимативни алгоритам са константним фактором [2]. Ипак, у [11, 12] је показано да када тежине задовољавају неједнакости троугла, постоји апроксимативни алгоритам са константним фактором $2 + \frac{k-1}{n} + \frac{1}{k}$. Временска сложеност неких сличних проблема је дата у [1]. Треба приметити да је захтев повезаности круцијалан, јер се без њега одговарајући проблем може решити у полиномијалном времену коришћењем техника упаривања (matching techniques).

У [34] је показано да сваки 3-повезан $K_{1,d}$-слободан граф са n чворова садржи разапињујући 3-повезан подграф чији је највећи степен највише $2d - 1$. $K_{1,d}$-слободан граф је граф који не садржи индуковану бипартитну клику која повезује 1 чвор са d чворова. 3-повезан граф је онај који избацивањем било која два чвора остаје и даље повезан.

У [39] је показано слично тврђење о 2-повезаним графовима. Показано је да сваки 2-повезан $K_{1,d}$-слободан граф има 2-повезано разапињуће стабло чији је највећи степен највише d.

У [61, 62] су дата истраживања у вези примене MDBCSP за хемијска једињења. Иако је проблем NP-тежак у општем случају, специјални графови који одговарају хемијским једињењима имају полиномијалан број разапињућих стабала, па је могуће применити и предложен алгоритам који ради у полино-

мијалном времену.

Субекспоненцијални параметризовани алгоритми над планарним графом за проналажење повезаног подграфа са ограниченим максималним степеном и максималним бројем грана (или чворова) су приказани у [54, 55]. Аутори користе планарност комбиновану са техникама динамичког програмирања над декомпозицијом гранања улазног графа. Међутим, пошто је предложени алгоритам временски изузетно захтеван, применљив је само за графове мањих димензија.

Роман и Кајкобад су у [50] нашли везу између проблема MDBCSP и проблема налажења максималног индукованог подграфа са изолованим чворовима (Maximal Independence Problem). Они су такође приказали алгоритам сложености $O(n^2)$ за конструкцију разапињућег стабла, али је проблем налажења максималног индукованог подграфа са изолованим чворовима такође NP-тежак проблем, што ограничава његову применљивост.

У [10] је приказан алгоритам који ради у $O(n)$-времену који конструише повезани подграф максималног степена $14 + \lceil 2\pi/\gamma \rceil$, за дати реални број γ, где је $0 < \gamma < \pi$. Довољни услови да повезани граф има стабло са ограниченим степеном су приказани у [46, 45].

Сличан проблем, у коме се за дати граф тражи подграф са ограниченим степеном чворова и ограниченим дијаметром, је разматран у [14]. Дати проблем има широку примену у: безбедности електронских комуникација, дизајну мрежа и паралелном процесирању. У датом раду су изведене неке границе за степен највећег подграфа у неким графовима од практичног значаја: хиперкоцке (енг. hypercube) и мреже (енг. mesh). Такође је предложен хеуристички приступ са константним фактором за решавање овог проблема. Експериментални резултати са разним мрежама су показали да алгоритам у пракси добија боља решења у односу на горњу границу решења предвиђену теоретским константним фактором.

У [8] је приказана једна формулација целобројног линеарног програмирања. Иако је ова формулација са теоријског аспекта веома значајна, број ограничења експоненцијално расте са димензијом проблема, те стога није применљива на инстанце већих димензија. Поред предложеног модела, исти аутор у радовима [8, 9] предлаже и генетски алгоритам који се може примењивати и на инстанце

већих димензија. Генетски алгоритам користи бинарну репрезентацију, фино градирану турнирску селекцију, укрштање у једној тачки, просту мутацију са замрзнутим генима и технике кеширања. Експериментални резултати су садржали приближна решења за више графова са до 212 чворова и 229 грана.

4.2.1 Постојећа ILP формулација

Прва формулација целобројног линеарног програмирања за дати проблем је дата у [8]. Значај овог модела је у томе што је то прва формулација целобројног линеарног програмирања за овај проблем. Са друге стране, пошто модел садржи експоненцијални број ограничења у односу на димензију проблема, његов значај је пре свега теоретске природе, док се при решавању MDBCSP може користити само на проблеме малих димензија.

Нека је дат граф $G = (V, E)$ и нека је $G' = (V', E')$ највећи повезани подграф графа G са чворовима ограниченог степена. И нека је (V', T') разапињуће стабло за подграф G'. Тада можемо увести променљиве

$$x_e = \begin{cases} 1, & e \in E' \\ 0, & e \notin E' \end{cases} \tag{4.1}$$

$$y_e = \begin{cases} 1, & e \in T' \\ 0, & e \notin T' \end{cases} \tag{4.2}$$

$$z_i = \begin{cases} 1, & i \in V' \\ 0, & i \notin V' \end{cases} \tag{4.3}$$

где је $e \in E$.

Тада се MDBCSP може формулисати на следећи начин:

$$\max \sum_{e \in E} w(e) \cdot x_e \tag{4.4}$$

тако да важи

$$\sum_{e : e \ni i} x_e \leq d \qquad\qquad i \in V, \tag{4.5}$$

$$y_e \leq x_e \qquad e \in E, \tag{4.6}$$

$$\sum_{e \in E} y_e = -1 + \sum_{i \in V} z_i \tag{4.7}$$

$$x_e \leq z_{i_e} \qquad e \in E, \tag{4.8}$$

$$x_e \leq z_{j_e} \qquad e \in E, \tag{4.9}$$

$$\sum_{i_e, j_e \in S} y_e \leq |S| - 1 \qquad S \subseteq V, |S| \geq 3 \tag{4.10}$$

$$x_e, y_e \in \{0,1\}, z_i \in \{0,1\}, \qquad e \in E, i \in V \tag{4.11}$$

где користимо ознаку $e : e \ni i$ која значи да сумирамо по гранама које садрже чвор i.

За чворове инцидентне грани e користимо ознаке i_e и j_e. Као што се може видети, функција циља (4.4) максимизује суму укупних тежина, ограничења (4.5) ограничавају степене чворова, (4.6) значи да је подграф G' повезан пошто садржи разапињуће стабло дато ограничењима (4.7)-(4.10). Ограничења (4.11) приказују бинарну природу променљивих. Очигледно је да број ограничења (4.10) расте експоненцијално са повећањем броја чворова, пошто је број услова једнак броју свих подскупова са бар три елемента датог скупа чворова.

У следећем поглављу представљен је MILP модел који садржи полиномијалан број ограничења. Очигледно је да, у односу на постојећи модел презентован у [8], највећи изазов представља реализовање повезаности подграфа. Идеја је да у новом моделу повезаност буде заснована на протоку. Да би се то реализовало поменута ограничења (4.10) морају бити замењена, зато што су то једина ограничења из (4.4)-(4.11) чији број расте експоненцијално.

Зато, бинарне променљиве y које учествују у (4.10), ће бити изостављене заједно са свим ограничењима која користе дате променљиве.

4.3 Нови модел мешовитог целобројног линеарног програмирања

Конструишимо сада нову ILP формулацију засновану на протоку у мрежи. Као и у случају са проблемима описаним у другој и трећој глави, и овде ћемо користити принцип увођења тока, тако да сваки чвор конзумира количину протока једнаку 1, и да проток може тећи само дуж грана из покривајућег стабла. Ова идеја је различита од добро познатог проблема протока кроз мрежу, зато што скупови V' и E' подграфа $G' = (V', E')$, и извор тока нису познати унапред, већ је проток кроз мрежу коришћен као идеја за опис модела. Даље, коришћен је принцип одржања тока у мрежи, имајући на уму да сваки чвор конзумира количину тока која износи 1, па је спољни проток једнак кардиналности V'. Убацивањем спољног тока кроз један чвор и даљим пропагирањем тока кроз граф добија се стабло које има онолико чворова колики је био спољни ток. Увећавањем спољног тока добијају стабла са све више и више чворова.

Да се постигне овај циљ уводе се додатне променљиве. Бинарна променљива t представља извор спољног тока, нпр. $t_i = 1, i \in V$ ако је чвор i извор спољног протока, а 0 у супротном. Износ спољног протока је представљен променљивама u. Приметимо да је $u_i = 0$ за све i који нису извори спољног тока. Непрекидне променљиве v_e означавају износ протока који пролази кроз грану e. Пошто је граф неоријентисан, а проток кроз мрежу се дефинише на оријентисаним мрежама, додатно ће смер сваке гране бити дефинисан. На пример, један начин да се оријентише граф је да се користи лексикографски поредак ($i_e < j_e$). Ако је проток мреже у правцу гране e тада је v_e позитиван, док је у супротном негативан. Мрежни проток може тећи само кроз гране из E', односно $v_e = 0, e \in E \setminus E'$. Пошто је овако дефинисан проток кроз мрежу потпуно независан од тежина на гранама, (његова улога је само да гарантује повезаност G') тада v_e може бити једнако 0 чак и за неке гране e из E'.

Нека је дат граф $G = (V, E)$ и нека је $G' = (V', E')$ повезани подграф графа G највеће тежине са чворовима ограниченог степена, и нека је (V', T') разапињуће стабло за подграф G'. Тада можемо увести променљиве

$$x_e = \begin{cases} 1, & e \in E' \\ 0, & e \notin E' \end{cases} \tag{4.12}$$

$$z_i = \begin{cases} 1, & i \in V' \\ 0, & i \notin V' \end{cases} \tag{4.13}$$

где је $e \in E$, и тада се модел мешовитог целобројног линеарног програмирања за решавање MDBCSP се формулише на следећи начин:

$$\max \sum_{e \in E} w(e) \cdot x_e \tag{4.14}$$

тако да важи

$$\sum_{e:e \ni i} x_e \leq d \qquad i \in V, \tag{4.15}$$

$$x_e \leq \frac{1}{2} z_{i_e} + \frac{1}{2} z_{j_e} \qquad e \in E \tag{4.16}$$

$$\sum_{i=1}^{n} t_i = 1 \tag{4.17}$$

$$u_i \leq n \cdot t_i \qquad i \in V \tag{4.18}$$

$$v_e \leq n \cdot x_e \qquad e \in E \tag{4.19}$$

$$v_e \geq -n \cdot x_e \qquad e \in E \tag{4.20}$$

$$u_i + \sum_{e:j_e = i} v_e - \sum_{e:i_e = i} v_e = z_i \qquad i \in V \tag{4.21}$$

$$z_i \leq \sum_{e:e \ni i} x_e \qquad i \in V \tag{4.22}$$

$$\begin{aligned} x_e \in \{0,1\}, z_i \in \{0,1\} \qquad & e \in E, i \in V \\ t_i \in \{0,1\}, u_i \in \mathbb{N} \cup \{0\}, v_e \in [-n,n] \qquad & e \in E, i \in V \end{aligned} \tag{4.23}$$

Ограничења (4.17) и (4.18) обезбеђују да се спољни проток у G' добија из једног извора, са количином која не прелази n. Ограничења (4.19) и (4.20) обезбеђују да се проток дистрибуира само кроз гране из G'. Принцип одржања протока у мрежи је дат ограничењима (4.21). Сваки чвор из V' мора буде инцидентан бар једној грани, што се обезбеђује ограничењем (4.22).

Као што се види из предложених модела: модела из [8] приказаног у поглављу 4.2 и новог модела из овог поглавља, функција циља је остала иста, као и ограничење (4.15), док су остала ограничења нова. Напоменимо да се ограничења (4.8) и (4.9) мењају ограничењем (4.16).

Докажимо сада коректност нове формулације.

Теорема 3. *Ако је подграф $G' = (V', E')$ датог графа $G = (V, E)$ допустиво решење проблема повезаног подграфа највеће тежине са чворовима ограниченог степена, онда одговарајуће вредности променљивих x, z, t, u, v дају оптимално решење MILP формулације (4.12) − (4.21) и вредност функције циља (4.12) није мања од вредности допустивог решења проблема MDBCSP.*

Доказ. Нека је $G' = (V', E')$ повезан подграф са чворовима чији степен не прелази d. Нека су променљиве x и z редом дефинисане као у (4.12) и (4.13). Изаберимо произвољан чвор k из V' и прогласимо га извором кроз који спољни проток улази у мрежу. Дакле, $t_k = 1$ и $u_k = |V'|$, док је $t_i = 0$ и $u_i = 0$ за све $i \in V$ и $i \neq k$. Да би се добиле вредности v_e потребно је извршити уређење чворова графа G' почев од чвора k (на пример може се користити претрага у дубину или ширину). Нека је $s(v)$ енумерација чворова из V' генерисана таквим процесом. На пример, $s(k) = 1$, за последњи нумерисани чвор w важи да је $s(w) = |V'|$, док је $s(v) = 0$ за све чворове из $v \in V \setminus V'$. За сваку грану $e \in E'$, вредност променљиве v_e је дефинисана на следећи начин. Ако је граф претраживан од чвора i_e до j_e кроз грану e ($s(j_e) = s(i_e) + 1$) тада је v_e једнако броју чворова у подстаблу претраге са кореном у i_e, Иначе, v_e је дефинисано као број чворова у подстаблу претраге са кореном у i_e помножен са -1. За све друге гране из E' и све гране из $E \setminus E'$ важи да је $v_e = 0$.

Како је подграф G' максималног степена d, очигледно да важи (4.15). Ограничења (4.17) и (4.18) су директно задовољена из дефиниције променљивих t и u.

Ако је $e \in E \setminus E'$, тада из дефиниције променљивих x и v, важи да је $x_e = 0$ и $v_e = 0$ те су ограничења (4.19) и (4.20) задовољена. У супротном, када је $e \in E'$, имплицира $x_e = 1$. Пошто је v_e дефинисано као број чворова неког подстабла (евентуално помножен са -1), тада мора да буде $v \in [-n, n]$, и ограничења (4.19) и (4.20) су задовољена.

Ако је $i \in V \setminus V'$, тада све гране e тако да $i_e = i$ или $j_e = i$ су у $E \setminus E'$. Одакле, $u_i = 0, v_e = 0$ и $z_i = 0$ те су ограничења (4.21) задовољена. Ако је $i = k$ тада $u_i = |V'|$ и $z_i = 1$. Укупни број чворова у подстаблима следбеника чвора k је једнак $|V'| - 1$ (сви чворови из V' су укључени осим чвора k). Одатле, по дефиницији променљиве v, важи да је

$$\sum_{e:i_e=i} v_e - \sum_{e:j_e=i} v_e = \sum_{\substack{e:i_e=k \\ s(j_e)<s(k)}} v_e + \sum_{\substack{e:i_e=k \\ s(j_e)>s(k)}} v_e - \sum_{\substack{e:j_e=k \\ s(i_e)<s(k)}} v_e - \sum_{\substack{e:j_e=k \\ s(i_e)>s(k)}} v_e$$

Пошто је $s(k) = 1$, $s(i_e) > 1$ и $s(j_e) > 1$ те су прва и трећа сума једнаке нули. Како су сви елементи друге суме позитивни, и сви елементи четврте суме негативни, укупна сума је једнака укупном броју чворова у подстаблима који су следбеници од k тј. $|V'| - 1$.

Претпоставимо да је $i \neq k$.

$$\sum_{e:i_e=i} v_e - \sum_{e:j_e=i} v_e = \sum_{\substack{e:i_e=i \\ s(j_e)<s(i)}} v_e + \sum_{\substack{e:i_e=i \\ s(j_e)>s(i)}} v_e - \sum_{\substack{e:j_e=i \\ s(i_e)<s(i)}} v_e - \sum_{\substack{e:j_e=i \\ s(i_e)>s(i)}} v_e$$

Разлика прве и треће суме је једнака броју чворова у подстаблима чији је корен у i. Пошто је разлика друге и четврте суме једнака броју чворова у подстаблима чији су корени следбеници од i, то имплицира да је лева страна ограничења (4.21) једнака 1. За $i \in V'$ $z_i = 1$ што је десна страна ограничења (4.21).

За $i \in V \setminus V'$, $e \in E \setminus E'$ имплицира да је $z_i = 0$ и $\sum_{e:e \ni i} 0 = 0$ па је ограничење (4.22) задовољено. Обрнуто, када је $i \in V'$, најмање једна грана $e \in E'$ је инцидентна чвору i, тј. $x_e = 1$, те је десна страна већа или једнака од

1, колика је вредност леве стране ($z_e = 1$).

Ограничења (4.23) су задовољена на основу дефиниције и природе свих променљивих.

За $e \in E \setminus E'$ ограничења (4.16) су тривијално задовољена пошто је $x_e = 0$. Супротно, када је $e \in E'$, тада су $i_e, j_e \in V'$ те су $x_e = z_{i_e} = z_{j_e} = 1$ чиме су ограничења (4.16) задовољена.

Из функције циља (4.12), $\sum_{e \in E} w(e) \cdot x_e = \sum_{e \in E'} w(e)$, одакле следи $\max \sum_{e \in E} w(e) \cdot x_e \geq \sum_{e \in E'} w(e)$.

□

Теорема 4. *Ако је (x, z, t, u, v) допустиво решење новог MILP модела, онда је подграф $G' = (V', E')$ где је $V' = \{v \in V | z_v = 1\}$ и $E' = \{e \in E | x_e = 1\}$, допустиво решење MDBCSP проблема, са вредношћу циља не мањом од вредности функције циља (4.12) за дато (x, z, t, v, u).*

Доказ. Нека су вредности (x, z, t, v, u) решења MILP модела датог са (4.14) и ограничењима (4.15)-(4.23). Дефинишимо граф $G' = (V', E')$, где је $V' = \{v \in V | z_v = 1\}$ и $E' = \{e \in E | x_e = 1\}$. У наставку ће бити показано да је G' добро дефинисан, повезан и да су му чворови степена не већег од d.

- Ако је $e \in E'$ тада је $x_e = 1$. Из ограничења (4.16) следи да $z_{i_e} = z_{j_e} = 1$ што значи да важи да $i_e, j_e \in V'$. Одатле, граф G' је добро дефинисан.

- Подграф G' нема изолованих чворова, јер из (4.22), за сваки чвор $v \in V$, $(\forall e \in E')(v \notin e) \Rightarrow \sum_{e: e \ni v} x_e = \sum_{e: e \ni v, e \in E \setminus E'} x_e = 0 \Rightarrow z_v = 0 \Rightarrow v \in V \setminus V'$. Ово директно значи да ако $v \in V'$ тада мора бити најмање једна грана $e \in E'$ инцидентна чвору v.

- Из ограничења (4.15) следи да G' нема чворове са степеном већем од d;

- Сада можемо доказати да је V' повезан. Довољно је доказати да за сваку партицију V' у два дисјунктна подскупа V'_1, V'_2 са $|V'_1|, |V'_2| \geq 2$ постоји најмање једна грана $e \in E$ којој су инцидентни чворови из сваког од подскупова. Нека је k чвор такав да је $t_k = 1$. Претпоставимо, без губитка општости, да $k \notin V'_1$ и пошто важи (4.19), тада важи и

$$\sum_{i \in V'_1} \left(u_i + \sum_{e: j_e = i} v_e - \sum_{e: i_e = i} v_e \right) = \sum_{i \in V'_1} z_i \tag{4.24}$$

Десна страна R је једнака $|V_1'|$. Лева страна је једнака

$$\sum_{i\in V_1'} u_i + \sum_{i\in V_1'}\sum_{e:j_e=i} v_e - \sum_{i\in V_1'}\sum_{e:i_e=i} v_e =$$

$$= 0 + \sum_{e:i_e,j_e\in V_1'} v_e + \sum_{e:i_e\in V_2',j_e\in V_1'} v_e + \sum_{e:j_e\in V_1',i_e\in V\setminus V'} v_e -$$

$$- \sum_{e:i_e,j_e\in V_1'} v_e - \sum_{e:i_e\in V_1',j_e\in V_2'} v_e - \sum_{e:i_e\in V_1',j_e\in V\setminus V'} v_e =$$

$$= \sum_{e:i_e\in V_2',j_e\in V_1'} v_e - \sum_{e:i_e\in V_1',j_e\in V_2'} v_e$$

Пошто је $L = R$

$$\sum_{e:i_e\in V_2',j_e\in V_1'} v_e - \sum_{e:i_e\in V_1',j_e\in V_2'} v_e = |V_1'|,$$

а одавде да је бар једна сума различита од нуле, и

$$(\exists e)(((i_e \in V_2' \wedge j_e \in V_1') \vee (i_e \in V_1' \wedge j_e \in V_2'))v_e \neq 0.$$

Сада,

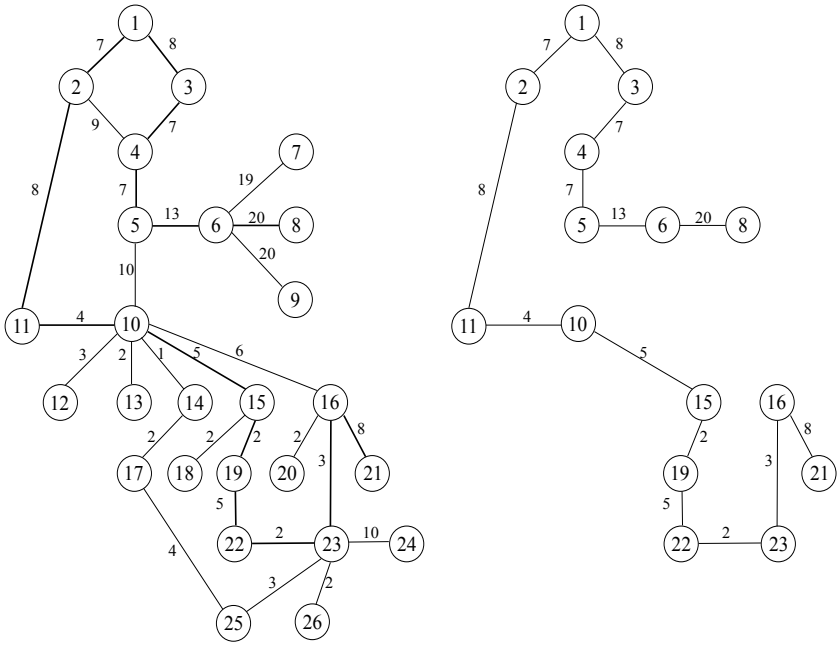
$$(\exists e)(((i_e \in V_2' \wedge j_e \in V_1') \vee (i_e \in V_1' \wedge j_e \in V_2'))x_e = 1.$$

Ако $v_e > 0$ тада из (4.19) следи да је $x_e = 1$. Ако је $v_e < 0$ тада из (4.20) следи да је $x_e = 1$. Одатле следи да постоји најмање једна грана $e \in E$, чија оба инцидентна чвора не припадају једном од подскупова V_1' и V_2'.

- За претходно дефинисано E' важи да

$$\sum_{e\in E'} w(e) = \sum_{e\in E} w(e)\cdot x_e = obj_{MILP}(x,z,t,u,v)$$

Како је оптимално решење MDBCSP максимум $w(G'')$ по свим повезаним подграфовима G'' чији чворови имају степене не веће од d, а горе дефинисани подграф G' задовољава све MDBCSP услове и чија је функција циља једнака вредности (4.12) у (x,z,t,u,v) (они одређују допустиво решење.) тада оптимална вредност MDBCSP за дати граф G је једнака или већа од (4.12) у (x,z,t,u,v).



Слика 4.2: Граф G и одговарајуће оптимално решење проблема MDBCSP

□

Јасно је да уколико тражимо максималне вредности по свим допустивим решењима, како MDBCSP тако и MILP формулације (4.12)-(4.21) из изведених тврђења претходне две теореме оптимална вредност MDBCSP и оптимална вредност проблема MILP формулације (4.12)-(4.21) ће бити једнаке.

Природа промењљивих представљеног MILP модела се може илустровати примером 4.1. Размотримо граф са леве стране слике 4.2. Гране које су укључене у повезани подграф чији су чворови највише 2, су на слици приказане подебљано.

Као што је претходно речено, за $d = 2$, оптимална вредност је 99. Једно оптимално решење је представљено на следећи начин.

Вредност $t_2 = 1$ и одговарајући $u_2 = 15$ док су све друге вредности $t_i = 0$ и $u_i = 0$ за $i \neq 2$.

Вредности z-промењљивих су $z_1 = z_2 = z_3 = z_4 = z_5 = z_6 = z_8 = z_{10} = z_{11} = z_{15} = z_{16} = z_{19} = z_{21} = z_{22} = z_{23} = 1$, док су $z_7 = z_9 = z_{12} = z_{13} = z_{14} = z_{17} = z_{18} = z_{20} = z_{24} = z_{25} = z_{26} = 0$.

Следеће x-промењљиве имају нула вредности: $x_{(2,4)} = x_{(5,10)} = x_{(6,7)} = x_{(6,9)} = x_{(10,12)} = x_{(10,13)} = x_{(10,14)} = x_{(10,16)} = x_{(14,17)} = x_{(15,18)} = x_{(16,20)} = x_{(17,25)} = x_{(23,24)} = x_{(23,25)} = x_{(23,26)}$. Одговарајуће v-вредности имају такође нула вредности. Све друге x-промењљиве узимају вредност 1, и одговарајуће v-промењљиве

имају вредности: $v_{(1,2)} = -6$, $v_{(1,3)} = 5$, $v_{(3,4)} = 4$, $v_{(4,5)} = 3$, $v_{(5,6)} = 2$, $v_{(6,8)} = 1$, $v_{(10,11)} = -7$, $v_{(10,15)} = 6$, $v_{(15,19)} = 5$, $v_{(16,21)} = 1$, $v_{(16,23)} = -2$, $v_{(19,22)} = 4$, $v_{(22,23)} = 3$ и $v_{(2,11)} = 8$.

Предложена MILP формулација, коришћењем CPLEX 10.1 решавача, даје резултат за 1.119 секунди при чему има 192 ограничења, док CPLEX са ранијом ILP формулацијом не успева да дође до оптималног решења за 3600 секунди при чему има 643568 ограничења.

У овом поглављу је уведена формулација мешовитог целобројног линеарног програмирања са полиномијалним бројем ограничења, са доказом коректности. На тај начин, проблем се може решавати коришћењем стандардних решавача за инстанце већих димензија. Будући рад се може усмерити на неколико начина. Било би пожељно испитати примену егзактних метода коришћењем предложене MILP формулације.

Глава 5

Закључак

5.1 Преглед нових резултата

У овој дисертацији су решавана два уопштења проблема максималне бисекције: вишедимензионални проблем максималне бисекције и вишедимензионални проблем максималне бисекције на повезане подграфове. Оба проблема су NP-тешка, као што је и основни проблем. На почетку је дат преглед постојеће литературе за основни проблем. За оба проблема су дате формулације мешовитог целобројног линеарног програмирања са доказом коректности. Предложени модели омогућују егзактно решавање инстанци мањих димензија. Поред формулације MILP-а, представљене су три метахеуристичке методе за решавање оба проблема на инстанцама средњих и великих димензија: генетски алгоритам, метода променљивих околина и метахеуристика заснована на електромагнетизму.

Употребљивост предложеног MILP модела тестирана је помоћу два стандардна решавача CPLEX и Gurobi-ja. Оптимална решења су добијена у свим случајевима за инстанце до 30 чворова, а у неким случајевима и за ретке графове до 300 чворова. Оба решавача су примењивана и на веће инстанце, уз ограничење времена извршавања на 7200 секунди. За добијање приближних решења на инстанцама средњих и великих димензија, за које нису добијена оптимална решења, развијене су и три метахеуристике. Генетски алгоритам користи специфично целобројно кодирање где су сва решења допустива, што је омогућило примену стандардних генетских оператора једнопозиционог укрштања и просте мутације са залеђеним генима. Између осталог, коришћена је

и фино градирана турнирска селекција, кеширање ГА и стационарни ГА са елитистичком стратегијом. Метода променљивих околина је примењена коришћењем одговарајућег система околина заснованог на k-замени, где тачно k чворова мења своју партицију. Метахеуристика заснована на електромагнетизму комбинује механизам привлачења/одбијања са процедуром скалирања. У оквиру све три метахеуристике користе се две процедуре локалне претраге. Прва процедура је заснована на 1-пребацивању где један (најповољнији) чвор прелази из једне партиције у супротну. Да би се очувала кардиналност, мора постојати и инверзан корак, где се из супротне партиције премешта најповољнији чвор. Дата процедура користи врло ефикасно имплементирану технику где се памти матрица свих повољности (за одређени пар чвор-координата у вектору тежина). Друга процедура локалне претраге заснована на 1-заменама је садржајнија, али и знатно мање ефикасна, па се примењује само када прва процедура заврши свој рад.

При решавању CMDMBP, такође су прво коришћени CPLEX и Gurobi стандардни решавачи који користе предложену MILP формулацију. Оптимална решења су добијена у свим случајевима за инстанце до 10 чворова. Оба решавача су примењивана и на веће инстанце, уз ограничење времена извршавања на 7200 секунди. Као и за MDMBP, за добијање приближних решења на инстанцама средњих и великих димензија, развијене су исте три метахеуристике. С обзиром на сличност проблема MDMBP и CMDMBP, основне конструкције све три развијене метахеуристике су аналогне као у случају MDMBP: целобројно кодирање ГА, стандардни генетски оператори, систем околина VNS, механизам привлачења/одбијања ЕМ и процедура скалирања ЕМ. Међутим, због захтева повезаности, појављује се проблем недопустивих решења, који је разрешен коришћењем посебно осмишљене казнене функције. Због тога је процедура локалног претраживања заснована на 1-пребацивању изгубила свој смисао, па је не примењујемо, а функција циља и процедура локалног претраживања заснована на 1-заменама су потпуно редефинисане.

При решавању MDBCSP је у литератури већ био познат генетски алгоритам за решавање, па је због тога предложена потпуно нова MILP формулација за његово решавање са полиномијалним бројем услова. У раду је дат и математички доказ коректности дате формулације. Тиме је омогућено и егзактно

решавање датог проблема на инстанцама нешто веће димензије.

Резултати приказани у овој тези могу се проширити или унапредити на неколико начина:

- Могуће коришћење предложених модела мешовитог целобројног линеарног програмирања за развој егзактних метода;
- Модификација описаних метода за решавање сличних проблема из теорије графова;
- Хибридизација предложених метода са неким другим хеуристикама и/или егзактним методама;
- Добијени резултати се можда могу користити за генерисање неких теоријских хипотеза за предложене проблеме.

5.2 Научни допринос

Најважнији резултати који представљају научни допринос ове дисертације су:

- формулисање уопштења проблема максималне бисекције графа увођењем вишедимензионалних тежина и/или наметање услова повезаности;
- формулација мешовитог целобројног линеарног програмирања за вишедимензионални проблем бисекције графа;
- формулација мешовитог целобројног линеарног програмирања за вишедимензионални проблем бисекције графа на повезане подграфове;
- побољшана формулација мешовитог целобројног линеарног програмирања за проблем одређивања повезаног подграфа највеће тежине са чворовима ограниченог степена;
- конструисање казнених функција које се примењују на функцију циља у случају некоректних решења, које користе све предложене методе за решавање вишедимензионалног проблема бисекције графа на повезане подграфове;

- осмишљавање начина кодирања и одговарајуће функције циља у генетском алгоритму, без појаве некоректних јединки, за решавање вишедимензионалног проблема бисекције графа;
- имплементација ефикасних метода локалног претраживања за све предложене методе за решавање вишедимензионалног проблема бисекције графа;
- конструисање система околина у методи променљивих околина који омогућава достизање обећавајућих региона претраге;
- прилагођавање технике скалирања у оквиру метахеуристике засноване на електромагнетизму за решавање оба предложена проблема бисекције графа.

Истраживање приказано у овом раду је од изузетног значаја јер врло успешно решава неколико значајних проблема оптимизације на графовима. Оно представља допринос у областима дискретних проблема на графовима, комбинаторне оптимизације, математичког моделирања као и апроксимативних метода и хеуристика. Неки од резултата ове дисертације су већ објављени у међународним часописима, а један део је на рецензији.

Литература

[1] O. Amini, I. Sau, and S. Saurabh. "Parameterized complexity of finding small degree-constrained subgraphs". *J. Discrete Algorithms* 10 (2012), pp. 70–83.

[2] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh. "Degree-constrained subgraph problems: Hardness and approximation results". *Lect. Notes Comput. Sci.* 5426 (2009), pp. 29–42.

[3] O. Amini, I. Sau, and S. Saurabh. "Parameterized complexity of the smallest degree-constrained subgraph problem" in *Parameterized and Exact Computation*. Springer, 2008, pp. 13–29.

[4] M. Armbruster, M. Fügenschuh, C. Helmberg, and A. Martin. "A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem" in *Integer Programming and Combinatorial Optimization*. Springer, 2008, pp. 112–124.

[5] C. Avanthay, A. Hertz, and N. Zufferey. "A variable neighborhood search for graph coloring". *European Journal of Operational Research* 151(2) (2003), pp. 379–388.

[6] Ş. İ. Birbil and S.-C. Fang. "An electromagnetism-like mechanism for global optimization". *Journal of global optimization* 25(3) (2003), pp. 263–282.

[7] C. Blum and A. Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". *ACM Computing Surveys* 35(3) (2003), pp. 268–308.

[8] M. Bogdanović. "An ILP formulation and genetic algorithm for the maximum degree-bounded connected subgraph problem". *Comput. Math. Appl.* 59(9) (2010), pp. 3029–3038.

[9] M. Bogdanović. "Solving the MDBCS problem using the metaheuristic genetic algorithm". *Int. J. Adv. Comput. Sci. Appl.* 2(12) (2011), pp. 161–167.

[10] P. Bose, M. Smid, and D. Xu. "Delaunay and diamond triangulations contain spanners of bounded degree". *Int. J. Comput. Geom. Appl.* 2(19) (2009), pp. 119–140.

[11] Y. Chan, W. Fung, L. Lau, and C. Yung. "Degree bounded network design with metric costs" in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science - FOCS, art. no. 4690947*. 2008, pp. 125–134.

[12] Y. Chan, W. Fung, L. Lau, and C. Yung. "Degree bounded network design with metric costs". *SIAM J. Comput.* 40(4) (2011), pp. 953–980.

[13] C. Dang, L. He, and I. K. Hui. "A deterministic annealing algorithm for approximating a solution of the max-bisection problem". *Neural networks* 15(3) (2002), pp. 441–458.

[14] A. Dekker, H. Pérez-Rosés, G. Pineda-Villavicencio, and P. Watters. "The Maximum Degree & Diameter-Bounded Subgraph and its Applications". *J. Math. Model. Algorithms* 11(3) (2012), pp. 249–268.

[15] V. Filipović. "Fine-grained tournament selection operator in genetic algorithms". *Computing and Informatics* 22 (2003), pp. 143–161.

[16] V. Filipović. *Operatori selekcije i migracije i web servisi kod paralelnih evolutivnih algoritama, doktorska disertacija*. 2006.

[17] V. Filipović. "An electromagnetism metaheuristic for the uncapacitated multiple allocation hub location problem". *Serdica Journal of Computing* 5(3) (2011), pp. 261–272.

[18] A. Frieze and M. Jerrum. "Improved approximation algorithms for maxk-cut and max bisection". *Algorithmica* 18(1) (1997), pp. 67–81.

[19] M. Furer and B. Raghavachari. "Approximating the minimum degree spanning tree to within one from the optimal degree" in *SODA'92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*. 1992, pp. 317–324.

[20] M. Furer and B. Raghavachari. "Approximating the minimum-degree Steiner tree to within one of optimal". *Journal of Algorithms* 17(3) (1994), pp. 409–423.

[21] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completness*. New York: Freeman W.H., 1979.

[22] M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problems". *Theoretical computer science* 1(3) (1976), pp. 237–267.

[23] M. Gendreau and J. Potvin. "Metaheuristics in Combinatorial Optimization". *Annals of Operations Research* 140 (1 2005), pp. 189–213.

[24] M. X. Goemans and D. P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". *Journal of the ACM (JACM)* 42(6) (1995), pp. 1115–1145.

[25] I. Gurobi Optimization. *GUROBI OPTIMIZER REFERENCE MANUAL*. https://www.gurobi.com/documentation/5.6/refman.pdf. [Online; accessed 14-May-2016]. 2013.

[26] E. Halperin and U. Zwick. "A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems". *Random Structures & Algorithms* 20(3) (2002), pp. 382–402.

[27] P. Hansen and N. Mladenović. "First vs. best improvement: An empirical study". *Discrete Applied Mathematics* 154(5) (2006), pp. 802–817.

[28] P. Hansen, N. Mladenović, and J. A. M. Pérez. "Variable neighbourhood search: methods and applications". *Annals of Operations Research* 175(1) (2010), pp. 367–407.

[29] P. Hansen, J. Brimberg, D. Uroševic, and N. Mladenovic. "Primal-dual variable neighborhood search for the simple plant-location problem". *INFORMS Journal on Computing* 19(4) (2007), pp. 552–564.

[30] J Hastad. "Getting optimal in-approximability results" in *29th STOC*. 1997, pp. 1–10.

[31] B. Hendrickson and R. Leland. "An improved spectral graph partitioning algorithm for mapping parallel computations". *SIAM Journal on Scientific Computing* 16(2) (1995), pp. 452–469.

[32] J. H. Holland. "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence." (1975).

[33] IBM. *CPLEX User's Manual*. `http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.1/ilog.odms.studio.help/pdf/usrcplex.pdf`. [Online; accessed 14-May-2016]. 2014.

[34] B. Jackson and N. Wormald. "Long Cycles and 3-Connected Spanning Subgraphs of Bounded Degree in 3-Connected $K_{1,d}$-Free Graphs". *J. Comb. Theory, Ser. B* 63(2) (1995), pp. 163–169.

[35] L. Kallel, B. Naudts, and A. Rogers. *Theoretical aspects of evolutionary computing*. Springer Science & Business Media, 2013.

[36] A. Kartelj. "Electromagnetism metaheuristic algorithm for solving the strong minimum energy topology problem". *Yugoslav Journal of Operations Research* 23(1) (2013), pp. 43–57.

[37] J. Kratica. "Improving performances of the genetic algorithm by caching". *Computers and Artificial Intelligence* 18 (1999), pp. 271–283.

[38] J. Kratica. "An electromagnetism-like approach for solving the low autocorrelation binary sequence problem". *International Journal of Computers Communications & Control* 7(4) (2012), pp. 688–695.

[39] R. Kužel and J. Teska. "On 2-Connected Spanning Subgraphs with Bounded Degree in $K_{1,r}$-Free Graphs". *Graphs Comb.* 27(2) (2011), pp. 199–206.

[40] J. Lazić, S. Hanafi, N. Mladenović, and D. Urošević. "Variable neighbourhood decomposition search for 0–1 mixed integer programs". *Computers & Operations Research* 37(6) (2010), pp. 1055–1067.

[41] A.-f. Ling, C.-x. Xu, and L. Tang. "A modified VNS metaheuristic for max-bisection problems". *Journal of Computational and Applied Mathematics* 220(1) (2008), pp. 413–421.

[42] Z. Maksimović. "A connected multidimensional maximum bisection problem, arXiv". *arXiv preprint: 1512.00614* (2015).

[43] Z. Maksimović. "A multidimensional maximum bisection problem". *arXiv preprint: 1506.07731* (2015).

[44] D. Matić. "A mixed integer linear programming model and variable neighborhood search for Maximally Balanced Connected Partition Problem". *Applied Mathematics and Computation* 237 (2014), pp. 85–97.

[45] H. Matsuda and H. Matsumura. "Degree conditions and degree bounded trees". *Discrete Math.* 309(11) (2009), pp. 3653–3658.

[46] H. Matsumura. "Degree Conditions and Degree Bounded Trees". *Electron. Notes Discrete Math.* 22 (2005), pp. 295–298.

[47] R. M'Hallah, A. Alkandari, and N. Mladenovic. "Packing unit spheres into the smallest sphere using VNS and NLP". *Computers & Operations Research* 40(2) (2013), pp. 603–615.

[48] N. Mladenović and P. Hansen. "Variable neighborhood search". *Computers & Operations Research* 24(11) (1997), pp. 1097–1100.

[49] I. H. Osman and G. Laporte. "Metaheuristics: A bibliography". *Annals of Operations research* 63(5) (1996), pp. 511–623.

[50] M. Rahman and M. Kaykobad. "Independence number and degree bounded spanning tree". *Appl. Math. E-Notes* 4 (2004), pp. 122–124.

[51] G. Raidl. "Decomposition Based Hybrid Metaheuristics". *European Journal of Operational Research* 244(1) ((2015)), 66––76.

[52] F. Rendl, G. Rinaldi, and A. Wiegele. "Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations". *Mathematical Programming* 121(2) (2010), pp. 307–335.

[53] K. J. Rossi C. Marchiori E. "An adaptive evolutionary algorithm for the satisfiability problem". *In Proceedings of the 2000 ACM symposium on Applied Computing* (2000), pp. 463–469.

[54] I. Sau and D. Thilikos. "Subexponential parameterized algorithms for bounded-degree connected subgraph problems on planar graphs". *Electron. Notes Discrete Math.* 32 (2009), pp. 59–66.

[55] I. Sau and D. Thilikos. "Subexponential parameterized algorithms for degree-constrained subgraph problems on planar graphs". *J. Discrete Algorithms* 8(3) (2010), pp. 330–338.

[56] J. Shi and J. Malik. "Normalized cuts and image segmentation". *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8) (2000), pp. 888–905.

[57] A. Słowik and M. Białko. "Partitioning of VLSI circuits on subcircuits with minimal number of connections using evolutionary algorithm" in *Artificial Intelligence and Soft Computing–ICAISC 2006*. Springer, 2006, pp. 470–478.

[58] R. Todosijević. "Variable and Single Neighbourhood Diving for MIP Feasibility". *YUJOR in press* DOI:10.2298/YJOR140417027L (2016), 66––76.

[59] Q. Wu and J.-K. Hao. "Memetic search for the max-bisection problem". *Computers & Operations Research* 40(1) (2013), pp. 166–179.

[60] F. Xu, X. Ma, and B. Chen. "A new Lagrangian net algorithm for solving max-bisection problems". *Journal of computational and applied mathematics* 235(13) (2011), pp. 3718–3723.

[61] A. Yamaguchi, K. Aoki, and H. Mamitsuka. "Finding the maximum common subgraph of a partial k-tree and a graph with a polynomially bounded number of spanning trees". *Inf. Process. Lett.* 92(2) (2004), pp. 57–63.

[62] A. Yamaguchi and H. Mamitsuka. "Finding the maximum common subgraph of a partial k-tree and a graph with a polynomially bounded number of spanning trees". *Lect. Notes Comput. Sci.* 2906 (2003), pp. 58–67.

[63] Y. Ye. "A. 699-approximation algorithm for Max-Bisection". *Mathematical Programming* 90(1) (2001), pp. 101–111.

Биографија

Зоран Максимовић је рођен 19. јула 1967. године у Ужицу. Основну школу и Математичку гимназију је завршио у Београду са одличним успехом. Редовне студије на Математичком факултету у Београду је почео 1987 године (после одслуженог војног рока), а дипломирао 28. фебруара 1994. године. Последипломске (магистарске) студије на Математичком факултету је уписао 1995 године и положио је све испите предвиђене планом и програмом са просечном оценом 10. Магистарску тезу под називом *„О синтези оптималних аутомата у неким класама лавиринатa“* одбранио је на Математичком факултету 21. марта 2001. године под руководством проф. Жарка Мијајловића, и тиме стекао звање магистра математичких наука.

Од 1994. до 2012 године је изводио вежбе на Технолошко-металуршком факултету Универзитета у Београду, прво као асистент-приправник а затим као асистент. Од 2012 године запослен је на Војној академији Универзитета одбране као асистент. На Технолошко-металуршком факултету је изводио вежбе из предмета Математика 1 и Математика 2 на основним студијама. На Војној академији је изводио вежбе из предмета Математика за економисте, Математика 3, Теорија вероватноће и статистика, Математика 2 на основним студијама и Нумеричке методе на мастер студијама.

У раду је користио разне програмске језике: C++, Pascal, SQL, Fortran, Prolog. Учествовао је у реализацији информационог система Лексикон иконографије класичне митологије (LIMC) САНУ, као и у развоју бројних апликација за подршку пословања пословних субјеката.

Научни радови кандидата

[1] Maksimović Z. „A new mixed integer linear programming formulation for the maximum degree bounded connected subgraph problem“, Publications de l'Institut Mathematique, у штампи, DOI: 10.2298/PIM1613099M (M23)

[2] Maksimović Z., Kratica J., Savić A.„Two metaheuristics for solving the connected multi-dimensional maximum bisection problem“, Soft Computing, DOI 10.1007/ s00500-016-2203-1, ISSN:1432-7643, IF2014=1.271, Computer Science, Artificial Intelligence (65/123), Computer Science, Interdisciplinary Applications (58/102) (M23)

[3] Bogdanović M., Maksimović Z., Simić A., Milošević J. „A mixed integer linear programming formulation for low discrepancy consecutive k-sums permutation problem“, Yugoslav Journal of Operations Research, у штампи, DOI:10.2298/YJOR160104005B (M51)

[4] Maksimovič Z. „Ob ekvivalentnosti častičnih avtomatov“, Intelektualnie sistemi, Tom 7, vip. 1-4, 2003, 337-352 . (M53)

[5] Максимовић З. „О синтези аутомата у лавиринтима“, XI конгрес математичара, Петровац, 2004. (M64)

PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série, tome 99(113) (2016), 99–108 DOI: 10.2298/PIM1613099M

A NEW MIXED INTEGER LINEAR PROGRAMMING FORMULATION FOR THE MAXIMUM DEGREE BOUNDED CONNECTED SUBGRAPH PROBLEM

Zoran Maksimović

Abstract. We give a new mixed integer linear programming (MILP) formulation for Maximum Degree Bounded Connected Subgraph Problem (MDBCSP). The proposed MILP formulation is the first in literature with polynomial number of constraints. Therefore, it will be possible to solve optimally much more instances before in a reasonable time.

1. Introduction

Let $G = (V, E)$ be an undirected graph, $d \geqslant 2$ an integer, and $w : E \to \mathbb{R}^+$ a weight function. The Maximum Degree Bounded Connected Subgraph (MDBCS) problem is to find a subgraph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, such that the subgraph G' is connected, has no vertex with degree exceeding d and $\sum_{e \in E'} w(e)$ has a maximum value. MDBCSP can be illustrated with the following small example.

Example 1.1. Let $|V| = 26$, $|E| = 29$ and graph $G = (V, E)$ with weights is presented on the left side in Fig. 1.

One optimal solution for $d = 2$ is presented on the right-hand side in Fig. 1, with value 99. This result is obtained by using new formulation which is introduced in Section 4.

It is useful to represent discrete optimization problems as integer programming problems in order to use different well-known optimization techniques for their exact solving [**8, 11, 22**]. The main effort is to design integer linear programming models with polynomial number of constraints.

2. Related work

MDBCSP is one of the classical NP-hard problems listed in Garey and Johnson's monograph [**10**] (as problem GT26), and there is no polynomial factor approximation algorithm for this problem [**1**]. However, in [**6, 7**] it is shown that

2010 *Mathematics Subject Classification:* 90C11; 90C27.
Key words and phrases: integer linear programming, degree-constrained subgraph, combinatorial optimization.
Communicated by Slobodan K. Simić.

Soft Comput
DOI 10.1007/s00500-016-2203-1

METHODOLOGIES AND APPLICATION

Two metaheuristics for solving the connected multidimensional maximum bisection problem

Zoran Maksimović[1] · **Jozef Kratica**[2] · **Aleksandar Savić**[3]

Abstract In this paper, a connected multidimensional maximum bisection problem is considered. This problem is a generalization of a standard NP-hard maximum bisection problem, where each graph edge has a vector of weights and induced subgraphs must be connected. We propose two metaheuristic approaches, a genetic algorithm (GA) and an electromagnetism-like metaheuristic (EM). The GA uses modified integer encoding of individuals, which enhances the search process and enables usage of standard genetic operators. The EM, besides standard attraction–repulsion mechanism, is extended with a scaling procedure, which additionally moves EM points closer to local optima. A specially constructed penalty function, used for both approaches, is performed as a practical technique for temporarily including infeasible solutions into the search process. Both GA and EM use the same local search procedure based on 1-swap improvements. Computational results were obtained on instances from literature with up to 500 vertices and 60,000 edges. EM reaches all known optimal solutions on small-size instances, while GA reaches all known optimal solutions except for one case. Both proposed methods give results on medium-size and large-scale instances, which are out of reach for exact methods.

Keywords Genetic algorithms · Electromagnetism-like approach · Evolutionary computation · Graph bisection · Combinatorial optimization

Communicated by V. Loia.

This research was partially supported by Serbian Ministry of Education, Science and Technological Development under the Grant Nos. 174010 and 174033.

✉ Zoran Maksimović
zmaksimovic@beotel.net

Jozef Kratica
jkratica@mi.sanu.ac.rs

Aleksandar Savić
asavic@matf.bg.ac.rs

[1] University of Defence, Military Academy, Generala Pavla Jurišića Šturma 33, Belgrade 11000, Serbia

[2] Mathematical Institute, Serbian Academy of Sciences and Arts, Kneza Mihaila 36/III, Belgrade 11000, Serbia

[3] Faculty of Mathematics, University of Belgrade, Studentski trg 16/IV, 11000 Belgrade, Serbia

1 Introduction

The maximum bisection problem (MBP) is a well-known combinatorial optimization problem. For a weighted graph $G = (V, E)$ with nonnegative weights on the edges, and where the set V is partitioned into two sets, the weight of the cut is the sum of weights of the edges between those sets. For a weighted graph $G = (V, E)$ with nonnegative weights on the edges and where $|V|$ is an even number, the maximum bisection problem consists in finding a partition of the set of vertices V in two subsets with the equal cardinality where the weight of the cut is maximal. The maximum bisection can be applied in different fields such as VLSI design (Słowik and Białko 2006), image processing (Shi and Malik 2000) and compiler optimization (Hendrickson and Leland 1995).

The maximum bisection problem is NP-hard as shown in Garey et al. (1976). The complexity of finding optimal and near-optimal solutions of the maximum bisection problem has given raise to various solution approaches ranging from approximation algorithms, exact methods to metaheuristics.

A multidimensional generalization of the maximum bisection problem is proposed in the paper (Maksimović 2015b), where weights on edges, instead of numbers, are n-tuples of

Published online: 31 May 2016

ОБРАЗАЦ 1.

Изјава о ауторству

Потписани-а: Зоран Максимовић
Број уписа:

Изјављујем

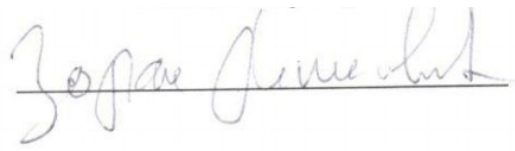
Да је докторска дисертација под насловом:

Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица

Потпис аутора

У Крагујевцу, 20.06.2016.



ОБРАЗАЦ 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора : Зоран Максимовић
Број уписа:
Студијски програм : студије по старом програму
Наслов рада: Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова
Ментор: др Јозеф Кратица
Потписани: Зоран Максимовић

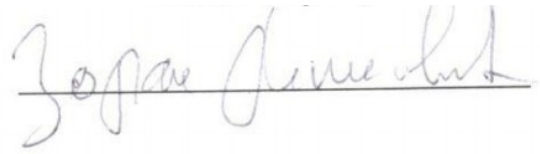
Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Крагујевцу**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Крагујевцу.

Потпис аутора

У Крагујевцу, 20.06.2016.



ОБРАЗАЦ 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку да у Дигитални репозиторијум Универзитета у Крагујевцу унесе моју докторску дисертацију под насловом:

Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова

која је моје ауторско дело.

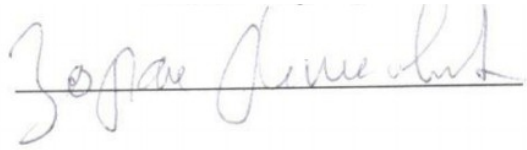
Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју Докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Крагујевцу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство
2. Ауторство – некомерцијално
3. **Ауторство – некомерцијално – без прераде**
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

Потпис аутора

У Крагујевцу, 20.06.2016.



УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ
ИНСТИТУТ ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ

Докторска дисертација под називом

Неки оптимизациони проблеми уопштења бисекције графова и повезаности подграфова

одбрањена је _____.

МЕНТОР:

_____
др Јозеф Кратица, научни саветник,
Математички институ САНУ

ЧЛАНОВИ КОМИСИЈЕ:

_____
др Љиљана Павловић, редовни професор,
Природно-математички факултет у Крагујевцу

_____
др Бобан Стојановић, ванредни професор,
Природно-математички факултет у Крагујевцу

_____
др Александар Савић, доцент,
Математички факултет у Београду

_____
др Драган Матић, доцент,
Природно-математички факултет у Бања Луци