УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ

Драган Матић

РЈЕШАВАЊЕ НЕКИХ ПРОБЛЕМА У НАСТАВИ ПРИМЈЕНОМ МЕТОДА КОМБИНАТОРНЕ ОПТИМИЗАЦИЈЕ

докторска дисертација

Београд, 2013

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Dragan Matić

SOLVING SOME PROBLEMS IN TEACHING BY USING COMBINATORIAL OPTIMIZATION METHODS

Doctoral Dissertation

Belgrade, 2013

Подаци о ментору и члановима комисије

Ментор

др Владимир Филиповић, доцент, Математички факултет, Универзитет у Београду

Чланови комисије

др Душан Тошић, редовни професор, Математички факултет, Универзитет у Београду

др Милан Божић, ванредни професор, Математички факултет, Универзитет у Београду

др Илија Лаловић, доцент, Природно математички факултет, Универзитет у Бањој Луци

др Владимир Филиповић, доцент, Математички факултет, Универзитет у Београду

др Александар Савић, доцент, Математички факултет, Универзитет у Београду

Датум одбране:

Подаци о докторској дисертацији

Наслов докторске дисертације

Рјешавање неких проблема у настави примјеном метода комбинаторне оптимизације.

Резиме

У овом раду се истражују неки актуелни проблеми комбинаторне оптимизације. Анализиране су и представљене различите методе рјешавања сљедећих NP тешких проблема: проблем проналажења максималне повезане партиције, уопштени проблем проналажења максимално балансиране повезане партиције у графу са q партиција ($q \geq 2$), проблем проналажења подјеле скупа на двије партиције и проблем проналажења p-арне транзитивне редукције у диграфу. Заједно са истраживањем метода комбинаторне оптимизације, којима се рјешавају наведени проблеми, у дисертацији се истражује и могућност примјене неких од наведених проблема комбинаторне оптимизације у организацији наставе.

За сваки од ових проблема приказане су метахеуристике за њихово рјешавање: метод промјењивих околина је развијен за сва четири проблема, док је за проблем транзитивне редукције у диграфу развијен и генетски алгоритам. За проблем максимално балансиране повезане партиције у графу је развијен и модел мјешовитог цјелобројног линеарног програмирања, који омогућава проналажење тачног рјешења за инстанце мањих димензија. Добијени експериментални резултати указују на високу употребну вриједност свих развијених метода.

Проблеми који су рјешавани у овом раду су од великог теоријског и практичног значаја. Користе се у производњи, областима рачунарских мрежа, инжењерству, обради слика, биологији, друштвеним наукама, а такође и у областима примијењене математике и рачунарства.

У раду је разматрана примјена неких од наведених проблема у организацији наставе. Показало се да се проналажења максимално балансиране повезане партиције у графу и проблем проналажења подјеле скупа на двије партиције успјешно могу примијенити у огранизацији планова и програма, како је то приказано и на конкретним примјерима. Развијене су технике за начине повези-

вања лекција, као и за одређивање њихових тежина, заснованих на објективним показатељима и субјективним процјенама професора. Тиме је постигнуто да се читав курс представи као повезан тежински граф, што пружа могућност да се проблем подјеле лекција унутар курса посматра и рјешава као математички проблем.

Придруживањем лекција одговарајућим категоријама (тематским цјелинама) унутар једног курса, креира се фамилија подскупова (тематских цјелина) читавог скупа лекција. Ако претпоставимо да лекције курса треба разбити у два дисјунктна подскупа (на примјер на зимски и љетњи семестар), тако да што више тематских цјелина буде "покривено" у оба та подскупа, тада се наведени проблем своди на рјешавање проблема максималне подјеле скупа.

Развијеним моделима у организацији наставе, из којих настају NP тешки проблеми, овом раду је, поред научног доприноса у пољу математичког програмирања и операционих истраживања, придодат и допринос из области методологије наставног процеса, са нагласком на методологију наставе математике и рачунарства.

Кључне ријечи

комбинаторна оптимизација, цјелобројно линеарно програмирање, метахеуристике, метода промјенљивих околина, генетски алгоритми, балансирани графови, транзитивна редукција у графу, проблем подјеле скупа, примјена рачунара у настави

Научна област

Математика

Ужа научна област

Методика наставе математике и рачунарства

УДК број

[519.1:519.863]:371.3(043.3)

Dissertation Data

Doctoral dissertation title

Solving some problems in teaching by using combinatorial optimization methods

Abstract

In this work some actual combinatorial optimization problems are investigated. Several different methods are suggested for solving the following NP hard problems: maximally balanced connected partition problem in graph, general maximally balanced problem with q partitions ($q \geq 2$), maximum set splitting problem and p-ary transitive reduction problem in digraphs. Together with investigation of combinatorial optimization methods for solving these problems, the applying of these problems in education is also considered in the dissertation.

For solving each of these problems, metaheuristics are developed: variable neighborhood search is developed for each problem and genetic algorithm is used for solving p-ary transitive reduction problem in digraphs. For maximally balanced connected partition problem a mixed linear programming model is established, which enables to solve the problem exactly for the instances of lower dimensions.

Achieved numerical results indicate the high level of reliability and usability of the proposed methods.

Problems solved in this research are of a great interest both in theoretical and practical points of view. They are used in production, computer networks, engineering, image processing, biology, social sciences and also in various fields of applied mathematics and computer science.

In this work the applying of some problems in educational issues is also considered. It is shown that approaches of finding maximally balanced connected partition in graph and finding maximum splitting of the set can be successfully used in course organization, which is verified on the concrete examples. Based on the objective indicators and professor's assessment, the techniques for the identifying the connections between the lessons, as well as the weights of the lessons are developed. Thus, whole course can be represented as a connected weighted graph, enabling the resolving of the lesson partition problem by mathematical approaches.

By assigning the lessons into the appropriate categories (topics area) inside a

course, a collection of subsets (corresponding to the topics) of the set of lessons is created. If we set the requirement that lessons should be split into two disjoint subsets (e.g. into the winter and summer semesters), in a way that corresponding topics are processed in both subsets, then the mathematical model of the requirement and its solution corresponds to the set splitting problem.

By the developed models of course organization, from which the NP hard problems arise, in addition to the scientific contributions in the fields of mathematical programming and operational research, contributions in educational aspects are added, especially in the methodology of teaching mathematics and computer science.

Keywords

combinatorial optimization, mixed integer linear programming, metaheuristics, variable neighborhood search, genetic algorithms, balanced graphs, transitive reduction in graphs, set splitting problem, computers in education

Scientific field

Mathematics

Scientific discipline

Mathematics and computer science teaching methodology

UDC number

[519.1:519.863]:371.3(043.3)

Предговор

У овом раду се истражују неки актуелни проблеми комбинаторне оптимизације и њихова примјена у организовању наставе. У времену интензивних реформи у образовању јављају се специфични организациони проблеми који иницирају развој сложених и напредних метода за њихово рјешавање. Проблеми присутни у организацији наставе често се срећу и у другим областима и предмет су изучавања не само математичара, већ и истраживача из других научних области, као што су економија, менаџмент и инжењерство. С обзиром на сложеност и велике димензије проблема, пред особе укључене у управљање организационим процесима постављају се задаци чије рјешавање подразумијева проналажење ако не најбољег, онда бар довољно доброг рјешења.

У већем броју случајева, показује се да организациони проблеми у наставном процесу у свом изворном облику заправо јесу проблеми математичке природе и као такви се јављају и у другим областима. Стога се провјерене и доказане методе рјешавања сличних проблема, као и искуства стечена у теоријским анализама и пракси, (као што су управљање производњом, организовање транспорта, управљање рачунарским мрежама итд.), могу примијенити и у организовању наставног процеса.

Циљ овог истраживања је да се детаљно анализирају неке специфичне потребе у организовању наставе, те да се за посматране практичне проблеме конструишу одговарајући математички модели. Математички модели који одговарају проблемима анализираним у овом раду имају своју ширу примјену и спадају у класу тзв. NP тешких проблема, за које не постоји алгоритам полиномске сложености који рјешава општи случај. Стога су оправдани развој и примјена приближних, метахеуристичких метода за њихово рјешавање.

Показало се да подјела лекција једног курса на два дијела, тако да тежина тих дијелова буде колико је год могуће више уједначена, може бити од практич-

ног значаја. Ако се овом приступу придода и захтјев којим се дефинише повезаност лекција, онда се за тај приступ може формирати еквивалентан математички проблем, у теорији познат као проблем проналажења максималне повезане партиције у графу. Слично, приступ у којем се лекције курса групишу у тематске цјелине, док се као задатак поставља подјела лекција на два дијела, тако да што више тематских цјелина буде присутно у оба дијела, своди се на проблем проналажења подјеле скупа на двије партиције. Овај приступ се такође успјешно може примијенити у огранизацији наставних планова и програма, како је то приказано и на конкретним примјерима.

У дисертацији су стога анализиране и представљене различите методе рјешавања сљедећих NP тешких проблема: проблем проналажења максималне повезане партиције и уопштење овог проблема, проблем проналажења максимално балансиране повезане партиције у графу са q партиција ($q \geq 2$) и проблем проналажења подјеле скупа.

Развојем метода за рјешавање ових проблема, отвара се могућност примјене тих метода и за рјешавање неких других сродних NP тешких проблема, који проналазе примјену у другим организационим питањима, као што су проблеми у биологији, инжењерству или управљању у привреди. Тако је, уз поменуте проблеме, у дисертацији проучаван и проблем проналажења p-арне транзитивне редукције у диграфу.

Резултати постигнути у дисертацији су од теоријског и практичног значаја. За дате проблеме развијене су ефикасне методе за рјешавање, чија је ефикасност и поузданост показана извршавањем алгоритама на одговарајућим тестним подацима. Илустрацијом употребе неких од анализираних проблема на конкретним курсевима, дефинисан је правац кориштења датих приступа у организацији наставе.

Желим да се захвалим ментору, доц. др Владимиру Филиповићу, који је руководио израдом моје докторске дисертације, као и члановима Комисије: проф. др Душану Тошићу, проф. др Милану Божићу, доц. др Илији Лаловићу, доц. др Александру Савићу на пажљивом читању и корисним сугестијама које су побољшале квалитет овог рада.

Током докторских студија имао сам подршку и у професорима др Ђорђу Дугошији, др Александру Липковском и др Зорици Станимировић са Матема-

тичког факултета у Београду, проф. др Ненаду Младеновићу и др Јозефу Кратици, научним савјетницима Математичког института у Београду, професорима др Милану Јовановићу и др Душку Јојићу са Природно математичког факултета у Бањалуци, те професорици академику др Олги Хаџић и професорима др Ђури Паунићу и др Синиши Црвенковићу са Природно математичког факултета у Новом Саду. Надам се да ћу у својој будућој научној и наставној каријери оправдати њихово повјерење и подршку.

Захвалност за подршку у писању дисертације дугујем колегиници Марији Милановић и колеги Александру Картељу, асистентима на Математичком факултету у Београду, као и колеги Владимиру Телебаку са Природно математичког факултета у Бањалуци.

Снагу, вољу и стрпљење за дуг и посвећен рад какав оваква дисертација захтијева и заслужује, пронашао сам у својој породици, супрузи Драгани и дјеци Наташи и Милану, мојим највећим животним успјесима.

Садржај

Глава 1

Увод

У времену интензивних реформи у образовању јављају се организациони проблеми који иницирају развој сложених и напредних метода за њихово рјешавање. Како би се постигла боља ефикасност, већа уштеда ресурса или смањење трошкова, пред особе укључене у управљање и организовање наставног процеса се стављају задаци чије рјешавање подразумијева проналажење најбољег или барем довољно доброг рјешења. Овакви проблеми се у пракси често срећу и предмет су изучавања не само математичара, већ и истраживача из других научних области, као што су економија, менаџмент и инжењерство.

У већем броју случајева, показује се да организациони проблеми у наставном процесу у свом изворном облику заправо јесу проблеми математичке природе и као такви се јављају и у другим областима. Стога се провјерене и доказане методе рјешавања сличних проблема, као и искуства стечена у другим областима науке и привреде, (као што су управљање производњом, организовање транспорта, управљање рачунарским мрежама итд.), могу примијенити и у организовању наставног процеса. Тиме се отвара широка палета алата и система за рјешавање, што генерално може да допринесе проналажењу добрих рјешења и код проблема организовања наставе.

Као што се може очекивати, овакви проблеми су веома сложени. Иако у данашње вријеме употреба рачунара у значајној мјери олакшава њихово рјешавање, показује се да се за ове проблеме морају развијати специфичне методе рјешавања, које су засноване на развоју нових математичких модела и тзв. метахеуристичком приступу, који подразумијева усмјеравање процеса проналаска рјешења ка оним областима простора претраживања за које се претпоставља

да садрже боља рјешења.

Да би се направио математички модел за дати проблем, сваки елемент проблема мора бити представљен одговарајућом математичком структуром, док односи између елемената индукују одговарајуће релације између тих математичких структура. За рјешавање ових проблема истраживачи теже ка употреби ефикасних и функционалних математичких структура за које већ постоје теоријски резултати и за које су већ развијени алати за рјешавање. За рјешавање многих проблема, који су комбинаторне природе, користе се математичке структуре из области комбинаторике и теорије графова, те се за рјешавање одговарајућих графовских проблема користе постојеће или развијају нове технике из те области[80].

1.1 Проблеми комбинаторне оптимизације

Приликом рјешавања оптимизационих проблема, било да је ријеч о теоријском или практичном приступу, једно од основних и полазних питања са којем се сусрећемо је избор најбоље конфигурације за представљање скупа промјенљивих којима се описују елементи проблема. Оптимизациони проблеми се стога, према природи промјенљивих које се користе, дијеле у двије класе: оне код којих се рјешења представљају реалним промјенљивима и оне код којих се за представљање рјешења користе дискретне структуре. Оптимизациони проблеми који користе дискретне структуре формирају класу проблема комбинаторне оптимизације. У проблемима комбинаторне оптимизације, објекат који представља рјешење тражимо у неком коначном, или пребројивом скупу. Тај објекат је најчешће цио број, подскуп, пермутација, граф и слично [90].

Дефиниција 1. *Проблем комбинаторне оптимизације $P = (S, f)$ се може дефинисати на сљедећи начин [10]: нека је $X = \{x_1, x_2, ..., x_n\}$ скуп промјенљивих и нека се знају:*

- *домени за промјенљиве $D_1, D_2, ..., D_n$;*
- *ограничења на промјенљивима;*
- *функција циља f која се минимизује: $f : D_1 \times D_2 \times ... \times D_n \to \boldsymbol{R}^+$.*

Скуп

$$S = \{s = (v_1, v_2, ..., v_n) | v_i \in D_i, s \text{ задовољава сва ограничења}\}$$

назива се скуп допустивих тачака или простор рјешења и сваки елемент скупа S представља потенцијално рјешење проблема. Циљ оптимизационог процеса је проналазак оне допустиве тачке $s^* \in S$ за коју функција f достиже минималну вриједност, тј. важи $(\forall s \in S)\ f(s^*) \leq f(s)$. s^* се зове глобално оптимално рјешење проблема (S, f).

Многи познати проблеми комбинаторне оптимизације су такви да садрже релативно једноставну формулацију, док је њихово рјешавање прилично тешко. На примјер, проблем трговачког путника или проблем бојења графа се веома лако дефинишу и разумљиви су и математичким лаицима, док за рјешавање тих проблема не постоји алгоритам који општи случај рјешава у разумном времену. Ови проблеми припадају широком скупу проблема комбинаторне оптимизације за које није познат алгоритам полиномске сложености који рјешава сам проблем, док се верификација рјешења може реализовати у полиномском времену. У теорији комплексности је доказано да ови проблеми припадају класи NP тешких проблема. С обзиром на велики значај ових проблема са једне стране и чињенице да за велике димензије проблема чак ни најсавременији рачунари не могу да пронађу тачно рјешење у разумном времену, јавља се потреба за развојем метода за приближно рјешавање ових проблема, са циљем да та приближна рјешења буду добра у некој задовољавајућој мјери.

Алгоритми који проналазе приближно рјешење се називају приближни (апроксимативни) алгоритми. За рјешења добијена овим алгоритмима се зна колико максимално могу да одступају од оптималног, тј. зна се тзв. фактор приближног алгоритма (фактор апроксимације). За проблеме максимизације, фактор апроксимације x, $x < 1$ је она вриједност за коју важи да рјешење добијено датим алгоритмом сигурно није лошије од вриједности $x \cdot OPT$, гдје је OPT оптимално рјешење проблема. За приближне алгоритме којима се рјешавају проблеми минимизације, за фактор апроксимације се узима она вриједност x, $x > 1$, таква да рјешење добијено датим алгоритмом сигурно није веће од $x \cdot OPT$ (OPT је оптимално рјешење).

Широку класу хеуристичких метода чине и тзв. "универзалне" хеуристичке

методе које се могу примјењивати на више проблема. Такве методе називамо метахеуристичким методима, или метахеуристикама. Метахеуристикама је посвећен одјељак који слиједи.

1.2 Метахеуристике

Метахеуристика се формално дефинише као итеративни процес који користи приближне методе, комбинујући на интелигентан начин различите концепте за претраживање и искориштење читавог простора рјешења, да би се у разумном времену, на ефикасан начин, одредило рјешење што ближе оптималном [89].

Са практичног, употребног аспекта, циљ метахеуристика је претраживање скупа допустивих рјешења, при чему је (уколико је потребно) дозвољено и проширење тог скупа и елементима који нису допустиви, прелазак у рјешења слабијег квалитета како би се избјегли локални оптимуми који нису и глобални, комбиновање са другим хеуристикама итд. Према [10], сљедеће особине метахеуристика се могу сматрати фундаменталним:

- Метахеуристике управљају процесом претраживања;
- Циљ је да се ефикасно претражује читав простор да би се пронашло што боље, а у најбољем случају оптимално рјешење;
- Технике које се користе унутар метахеуристика су у распону од једноставних процедура локалне претраге, па до сложених процеса заснованих на техникама учења;
- Метахеуристике су приближне методе и обично недетерминистичке;
- Могу садржавати механизме за избјегавање упадања у локална субоптимална рјешења;
- Концепт метахеуристика омогућава дефинисање процедура на апстрактном нивоу;
- Метахеуристике се не везују за специфичне проблеме;
- Метахеуристике могу да користе сазнања специфична за конкретан домен у форми хеуристика којсе се контролишу неком стратегијом са виших нивоа;

- Савремене метахуеристике памте међурјешења добијена у претходним итерацијама, како би убрзале или боље усмјериле претрагу ка обећавајућим областима претраге.

1.2.1 Класификација метахеуристика

С обзиром на велики број разних метахеуристика које се развијају у посљедње вријеме, природно је очекивати да се за њихов развој користе врло различити приступи, почев од полазне мотивације, преко стратегије претраживања, различитих намјена и класа проблема на којима се примјењују. У савременој литератури, среће се неколико критеријума за класификацију метахеуристика.

Класификација метахеуристика према полазној мотивацији идентификује тзв. метахеуристике засноване на симулацији природних појава (енг. nature based metaheuristics), као што су на примјер генетски алгоритми, мрављe колоније, или електромагнетизам. Насупрот њима, велики број метахеуристика није мотивисан природним процесима (на примјер, табу претраживање, итеративна локална претрага итд.)

Према броју тачака којима се располаже унутар једне итерације, метахеуристике се дијеле на двије класе: метахеуристике засноване на популацији (енг. population based) и метахеуристике које раде са једном тачком унутар једне итерације (енг. single point search). У првој класи, у истом тренутку алгоритам располаже са више тачака. Претраживање простора је засновано на примјени различитих оператора рекомбиновања, помјерања, измјена и одабира појединих тачака, како би се у свакој новој итерацији формирала нова популација, са потенцијално бољим јединкама. Такве метахеуристике су: генетски алгоритми, мрављe колоније, електромагнетизам, јата птица итд. У метахеуристикама које располажу једном тачком, примјењује се стратегија заснована на праћењу трајекторије (енг. trajectory methods), који подразумијева да се кроз итерације формира низ рјешења, од којих је свако наредно (по правилу, али не и обавезно) боље од претходног. Ови алгоритми најчешће, (али опет не и обавезно) користе и неке методе локалног претраживања, гдје се ново, потенцијално боље рјешење бира унутар неке околине тренутног рјешења. Неке од метода које припадају овој класи су метода промјењивих околина, табу претраживање, симулирано каљење итд.

Према типу функције циља, може се направити класификација на оне метахеуристике које користе статичку, те оне које користе динамичку функцију циља.

Неке од најчешће кориштених метахеуристика су:

- похлепни алгоритам са прилагођеном случајном претрагом (енг. Greedy randomized adaptive search procedure - GRASP);
- табу претраживање (енг. Tabu search - TS);
- генетски алгоритам (енг. Genetic algorithm - GA);
- метод промјењивих околина (енг. Variable neighborhood search – VNS);
- Лагранжова релаксација (енг. Lagrangian relaxation - LR);
- неуронске мреже (енг. Neural network algorithm - NNA);
- симулирано каљење (енг. Simulated annealing - SA);
- мрављe колоније (енг. Ant colony optimization - ACO);
- пчелиње колоније (енг. Artificial Bee Colony - ABC);
- електромагнетизам (енг. Electromagnetism - EM);
- управљано локално претраживање (енг. Guided Local Search - GLS).

С обзиром на то да се за рјешавање проблема у овом раду користе двије метахеурисичке методе: метода промјењивих околина (VNS) и генетски алгоритми (GA), тим методама су посвећене наредне двије секције. Детаљнији описи осталих метода превазилази обим овог рада и налазе се, на примјер, у [10, 89, 42].

1.2.2 Метода промјењивих околина

Метода промјењивих околина је метахеуристика уведена од стране Ненада Младеновића и Пјера Хансена у [87]. Претрага је заснована на систематској промјени околина да би се избјегле ситуације када алгоритам "упада" у субоптимална рјешења. Ефикасност VNS алгоритма је заснована на разматрању да у многим практичним оптимизационим проблемима постоји веза између локалних минимума. Стога метод промјењивих околина и не прати унапријед задату

путању, већ "скаче" на потенцијално боља рјешења која се налазе у некој околини тренутно најбољег. На овај се начин очувавају добре особине тренутног рјешења (ако су неке промјенљиве већ достигле оптималне вриједности), док се у новим околинама покушавају побољшати преостале промјенљиве. Овај систем је додатно ојачан системом локалног претраживања, којим се прелази из тренутног рјешења у најквалитетније рјешење у некој његовој околини.

Тако, идеја да се претрага усмјери са једног на (могуће бољи) други минимум који је смјештен у некој околини полазног минимума представља разуман и, као што је то случај у великом броју разних оптимизационих проблема, оправдан приступ.

Да би се проширила претрага, VNS у потрази за бољим локалним минимумом обично користи околине растуће кардиналности. Да би се дефинисао скуп околина, претпоставимо да је x произвољно рјешење и N_k, за $k = k_{min}, \cdots, k_{max}$, коначан скуп структура околина. Тада се $N_k(x)$ дефинише као скуп рјешења у k-тој околини тачке x. Критеријум за прекид итеративног процеса је обично достизање максималног броја итерација, максималног броја итерација без унапређења рјешења, или максимално дозвољеног времена извршења.

Широка примјена VNS методе, као и разне имплементације дозвољавају и различите стратегије. Детаљнији опис разних варијанти VNS проблема може се пронаћи у [47, 48], док је основна шема за VNS алгоритам приказана на Слици 1.1.

Читав VNS алгоритам се реализује преко двије угнијеждене петље. Спољашња петља контролише читав процес претраживања, док се главна претрага одвија у унутрашњој петљи, преко двије најважније процедуре: "размрдавања" (енг. shaking) и локалне претраге. Процедура размрдавања управља околинама и у сваком кораку, предлаже нову тачку из неке околине тренутног рјешења као потенцијално ново рјешење. Локална претрага покушава да унаприједи предложено ново рјешење, анализирајући и тражећи боља рјешења у његовој околини. Унутрашња петља се понавља све док се врши унапређење рјешења. Када се она заврши, спољашња петља улази у нову итерацију, све док се не испуни критеријум за завршетак.

```
1. UcitajUlaznePodatake();
2. x = InicijalnoResenje();
3. N = GenerisiOkoline(k_min, k_max);
4. OdaberiKriterijumZaustavljanja();
5. While(!IspunjenKriterijumZaustavljanja()) do
        6. k = k_min;
        7. While (k <= k_max) do
                8. x' = Razmrdavanje(x,N_k);
                9. x'' = LokalnaPretraga(x');
                10. If (FunkcijaCilja(x'')<FunkcijaCilja(x)) then
                        11. x = x'';
                        12. Goto 6;
                else
                        13. k = k + 1;
14. StampajRjesenje();
```

Слика 1.1: Основна шема VNS алгоритма

1.2.3 Генетски алгоритми

Генетски алгоритми спадају у класу еволуцијских алгоритама, чије понашање потиче из процеса еволуције који се одвија у природи. У основи се подразумијева рад са популацијом јединки, гдје свака јединка представља потенцијално рјешење, а свака популација је подскуп укупног простора претраживања. Популација се у итеративном поступку мијења тако што се старе јединке мијењају новим, потенцијално бољим. Основни оквир за функционисање генеских алгоритама је дат од стране Џона Холанда, у раду „Adaptation in natural and artificial systems“ [51]. Свакој јединки се додјељује вриједност, тзв. прилагођеност (енг. fitness), која оцјењује квалитет посматране јединке. Циљ генетског алгоритма је да се из итерације у итерацију проналазе јединке са све бољом прилагођеношћу, под чиме се сматра и појединачно побољшање сваке јединке, али и просјечна прилагођеност комплетне популације, што се постиже стандардним генетским операторима: селекцијом, укрштањем и мутацијом.

Популација је централна структура код генетског алгоритма и она обично броји од неколико, па до неколико стотина јединки (у ријетким ситуацијама и до хиљаду). Свака јединка се представља генетским кодом, који се записује у некој коначној азбуци (најчешће бинарној или пак цјелобројној).

Почетна популација се обично креира на случајан начин, чиме се омогућава

креирање разноврсног генетског материјала, корисног за касније фазе алгоритма и разноврсност самих јединки. У неким случајевима се за генерисање полазних јединки користи и нека друга хеуристика, чиме се претрага од почетка усмјерава ка оним областима претраживања за које се претпоставља да садрже квалитетнија рјешења.

Током рада алгоритма, на популацију се примјењују генетски оператори селекције, мутације и укрштања.

Оператор селекције бира јединке које преживљавају у процесу еволуције. По правилу, бољим јединкама (јединкама са бољом прилагођеношћу) се оператором селекције даје већа вјероватноћа да ће преживјети у наредну генерацију, док слабије јединке селекцијом добијају мању шансу да преживе, те се тако избацују из популације. Неке од популарнијих техника селекције су једноставна (или рулет) селекција, турнирска, елиминацијска селекција и елитизам. Рулет селекција диктира да је вјероватноћа да ће јединка преживјети у наредну генерацију пропорционална прилагођености. Код турнирске селекције се формирају скупови јединки, од којих само она са најбољом прилагођеношћу преживљава у сљедећу генерацију. Елиминацијска селекција је примарно заснована на избацивању лоших јединки прије него на одабиру бољих за наредну генерацију. Елитизам подразумијева да ће неколико најбољих (елитних) јединки директно прећи у наредну генерацију, чиме се даје поуздана претпоставка да алгоритам тежи глобалном оптимуму.

Укрштањем се рекомбинују гени јединки. Резултат укрштања јединки су нове јединке, које потенцијално садрже "добре" гене родитеља од којих су настале. Овим механизмом и јединке слабије прилагођености, али које садрже добре гене, добијају шансу да учествују у репродукцији и пренесу "добар" генетски материјал на сљедећу генерацију. Најчешће кориштени оператори укрштања су тзв. једнопозиционо, односно двопозиционо укрштање, код којих се одређује једна, односно двије тачке прекида на којима се одвија размјена генетских кодова родитеља, респективно.

Мутацијом се врши случајна промјена одређеног гена (са неком малом вјероватноћом да ће се мутација десити), чиме се постиже могућност враћања доброг генетског материјала, уколико је он изгубљен примјеном селекције и укрштања. Тиме се процес претраге извлачи из преурањене конвергенције и

усмјерава ка оним областима претраживања које садрже потенцијално боље јединке.

Након примјене генетских оператора, формира се наредна генерација. Популација се тако састоји од потомака и неких старијих јединки које су преживјеле с обзиром на њихов квалитет (елитне јединке). Продукција нових генерација се наставља све док не буду испуњени услови за завршетак алгоритма.

Основна шема функционисања генетског алторима је приказана на слици 1.2.

N_{pop} означава укупан број јединки у популацији. N_{elite} је број елитних јединки, док су промјенљивима i и obj_i представљене јединке и одговарајућа вриједност функције циља.

```
1. UcitajUlaznePodatake();
2. P = GenerisiInicijalnuPopulaciju(N_pop);
3. OdaberiKriterijumZaustavljanja();
4. While(!IspunjenKriterijumZaustavljanja()) do
     5. For (i = N_elite + 1; i < N_pop; i++) do
          6. Obj_i = RacunajFunkcijuCilja(P_i);
     7. RacunajFunkcijuPrilagodjenosti(P);
     8. IzvrsiSelekciju(P);
     9. IzvrsiMutaciju(P);
     10. IzvrsiUkrstanje(P);
11. StampajRjesenje();
```

Слика 1.2: Основна шема GA алгоритма

Као и у случају методе промјенљивих околина, детаљнији опис генетских алгоритама превазилази обим овог рада и може се пронаћи, на примјер, у [7, 36, 99]. У наредна три одјељка су мало детаљније објашњени специфични аспекти генетског алгоритма кориштеног за рјешавање проблема из овог рада. То су фино градирана турнирска селекција, концепт смрзнутих гена и кеширање.

Фино градирана турнирска селекција

У ситуацијама када је пожељно да просјечна величина турнира буде рационалан број, може се користити тзв. фино градирана турнирска селекција (енг. fine grained tournament selection - FGTS) [35] која је унапређење стандардне турнирске селекције. Као и у стандардној турнирској селекцији, за даљу репродукцију

се из посматраног турнира бира она јединка са најбољим фитнесом. Међутим, код стандардне турнирске селекције, сви турнири су исте величине, док су код FGTS-а присутне двије величине. За сваку генерацију, први тип турнира се спроводи k_1 пута и његова величина је $\lceil F_{tour} \rceil$, док се други тип турнира, у којем учествује $\lfloor F_{tour} \rfloor$ јединки реализује k_2 пута. Бројеви k_1 и k_2 се одређују тако да задовољавају сљедеће услове: $k_1 + k_2 = N_{pop} - N_{elite}$, а просјечна величина турнира је што ближа вриједности F_{tour}.

У [37, 35, 36] је приказано мноштво нумеричких експеримената за разне оптимизационе проблеме. Резултати ових експеримената указују да FGTS даје најбоље резултате за вриједност $F_{tour} = 5.4$, која је такође кориштена у GA имплементацијама за рјешавање проблема у овом раду. На примјер, ако се FGTS примјењује на $N_{nonel} = 50$ неелитистичких јединки, укупно се одржава $k_1 = 20$ и $k_2 = 30$ турнира величина 6 и 5 респективно.

Вријеме извршења FGTS оператора је $O(N_{nonel} * F_{tour})$. У случајевима када се F_{tour} и N_{nonel} сматрају константама (не зависе од димензије проблема), тада оператор селекције има константну временску сложеност.

Концепт смрзнутих гена

Током извршења GA може се десити да на некој позицији све јединке садрже бит исте вриједности. Такав ген се назива смрзнути ген. Појава смрзнутих гена може драстично смањити простор претраживања, што повећава вјероватноћу појаве преурањене конвергенције [68]. На примјер, у случају бинарног кодирања, ако је број смрзнутих битова n_f, тада простор претраживања постаје 2^{n_f} пута мањи. Оператори селекције и укрштања не могу промијенити вриједност смрзнутих гена, док је вјероватноћа да ће се баш на тај бит примијенити основна мутација прилично мала. Као посљедица, јавља се појава да се изгубљени подрегиони претраге више не могу вратити. Механизам којим се ова појава елиминише је заснован на значајном повећању вјероватноће да ће на смрзнутим генима бити примијењена мутација. Стога се за смрзнуте гене у генетском алгоритму основни ниво мутације множи одговарајућим бројем, фактором мутације за смрзнуте гене (енг. frozen factor).

Кеширање

За вријеме извршења генетског алгоритма, могућа је појава идентичних јединки. С обзиром да је рачунање функције циља временски често захтјевна операција, пожељно је примијенити технику којом се памте генетски кодови и њихове вриједности, па их касније користити, него их увијек изнова рачунати. Ова техника се назива техника кеширања, која је уведена у [66]. Кеш је имплементиран као хеш-ред (енг. hash-queue) структура података, унапријед заданог капацитета. Када треба да се рачуна вриједност функције циља за дату јединку, прво се провјерава да ли је та вриједност већ израчуната и похрањена у кешу. У случају да вриједност не постоји у кеш структури, она се тада по први пут рачуна и похрањује у кеш. У случају да је вриједност пронађена, она се не рачуна, већ се само прочита из кеша. У случају да се кеш напуни, бришу се оне вриједности које најдуже нису кориштене. Треба истаћи да кеширање ни у ком случају не утиче на квалитет рјешења, већ само служи за убрзање извршења GA.

1.3 Неки проблеми комбинаторне оптимизације који се примјењују у настави

Вјероватно најпознатија и највише изучавана класа проблема комбинаторне оптимизације који се односе на наставни процес је проблем прављења распореда (енг. timetabling problems). Ова широка класа проблема се према [96] дијели на три поткласе: проблем организовања распореда часова у школама (енг. school timetabling), те проблеме који се односе на универзитет: проблем организовања распореда курсева (енг. course timetabling) и проблем организовања испита (енг. examination timetabling).

Проблем распореда часова у школама се односи на креирање седмичног распореда за све часове у једној школи. Проблем се састоји од скупа наставника, учионица, предмета, разреда и предефинисаних седмичних периода. Рјешавање проблема се своди на одређивање у ком периоду који наставник држи час у ком разреду, уз поштовање прописаних ограничења. Основни услов који разликује овај проблем од проблема универзитетског распореда је да наставник у исто вријеме може предавати само једном разреду, односно разред у исто вријеме

може слушати наставу из само једног предмета.

Проблем оранизовања курсева на универзитету подразумијева израду седмичног распореда за све предмете који се нуде у наставном програму. За разлику од проблема прављења школског распореда, у универзитетском проблему групе студената нису фиксне, већ сваки студент посједује сопствену листу курсева. Прављење распореда на факултету узима у обзир и доступност и капацитете просторија. У идеалном случају, распоред изгледа тако да ни за једног студента не постоји колизија, али се у стварним проблемима, усљед ограничених ресурса (просторија, термина) и често великог броја студената, проблем своди на минимизовање броја студената код којих постоји колизија у распореду.

Проблем креирања распореда испита у испитном року се састоји од распоређивања испита у одговарајуће просторије и термине, уз основни услов да испити, који имају заједничке студенте, не смију бити организовани у истом тренутку. За разлику од прављења распореда наставе, у овом проблему је дозвољено да у истом тренутку у истој просторији буду одржани различити испити. У општем случају, тежи се изради оних распореда код којих су термини одржавња испита, који имају заједничке студенте, међусобно удаљени што је више могуће.

Познато је да је проблем креирања распореда у блиској вези са проблемом бојења графа. На примјер, код проблема прављења распореда испита, испите представљамо чворовима графа. Два чвора (два испита) су сусједи ако постоји барем један студент који полаже оба испита. Боје, којима се боје чворови, везују се за термине испита. Проблем прављења распореда испита се тако своди на проблем бојења чворова графа: два чвора могу бити обојени истом бојом ако нису повезани граном, тј. два испита могу бити одржана у истом термину ако нема студената који полажу оба та испита.

Детаљнија анализа наведених проблема прављења распореда није предмет овог рада. Прегледни радови новијег датума о приступима за рјешавање проблема распореда могу се наћи у [96, 91, 72, 85].

Проблем прављења распореда испита се може даље класификовати на сљедећих пет проблема: распоређивање курсева (енг. course scheduling), прављење распореда разред - наставник (class-teacher timetabling), прављење распореда за

студенте (student scheduling), проблем распоређивања наставника (енг. teacher assignment) и проблем распоређивање учионица (енг. classroom assignment) [12] и сваки од ових проблема се може посматрати независно од осталих и фокусира претрагу и процес оптимизације на засебан начин.

На примјер, код проблема распоређивања наставника (енг. teacher assignment problem) фокус се ставља на распоређивање наставника на одговарајуће предмете, уз разна ограничења која се односе на заузетост професора, професорово професионално знање, школске ресурсе, префериране предмете и слично. Нека од скоријих истраживања овог проблема су приказана у [102, 44, 43], гдје су развијани генетски алгоритми, табу претраживање и алгоритам симулираног каљења за рјешавање ових проблема.

Проблем распореда за студенте подразумијева повезивање студената са секцијама курсева, који се нуде у различитим временским периодима унутар једне седмице. Циљ је испуњење студентских захтјева, и креирање распореда без конфликата уз поштовање ограничења у вези са просторијама, балансирањем величина секција. Као и у другим проблемима, могући су и додатни захтјеви и ограничења. Нека ранија истраживања овог проблема су допринијела развоју разних метода за рјешавање: "branch and bound" методе комбиноване са хеуристиком у [69], похлепног алгоритма који је унапријеђен интелигентним одређивањем распореда студената [93], симулирано каљење [30], циљно програмирање (енг. goal programming) у [86], те формулације засноване на току у [23].

Метода промјенљивих околина је у раду [41] успјешно примијењена на проблем додјељивања тема за семинарске радове студената. Аутори разматрају практичан проблем заснован на реалним подацима прикупљеним на неким њемачким универзитетима. Хеуристичким приступом који укључује случајну иницијалну додјелу и методу промјенљивих околина за побољшање рјешења, додјељују се теме за семинарске радове према студентовим жељама и потребама самог семинара.

Неки комбинаторни проблеми у организацији наставе спадају у проблеме рутирања возила (енг. vehicle routing problem). На примјер, у [82], приказан је хеуристички приступ рјешавања проблема организовања рута за наставнике (енг. the teaching assistants assignment routing problem - TAARP) који пружају теренску помоћ дјеци са посебним потребама у фламанској заједници (Белгија).

Уз поштовање специфичних ограничења, развијена је хеуристичка метода која комбинује тзв. аукцијски алгоритам и методу промјенљивих околина. Алгоритам је успјешно примијењен на конкретан реалан проблем и омогућио је значајно смањење трошкова.

Широку класу проблема, који користе методе комбинаторне оптимизације, чине и проблеми аутоматског организовања електронског образовног материјала на основу задатих полазних критеријума. Како је из дана у дан све масовнија употреба рачунара и других електронских уређаја у образовном процесу, све је већа и потреба за интелигентним алатима за подршку процесу електронског учења, заснованим на неком концепту вјештачке интелигенције. Прије свега, ова потреба потиче из чињенице да системима за учење преко интернета приступа на хиљаде студената, те је наставницима практично немогуће пружити сву неопходну подршку студентима. Поред тога, потребе студената за образовним садржајем су углавном хетерогене, те презентовање истог садржаја студентима различитих способноси и потреба, не представља само недостатак система, већ озбиљно нарушава и квалитет комплетног образовног процеса.

Неки од праваца који се у овом смислу интензивно истражују подразумијевају употребу хеуристичких метода у тзв. прилагодљивом електронском учењу (енг. adaptive eLearning) и у аутоматском креирању тестова.

Интензиван развој интернета и информационих технологија уопште, довели су до развоја и широке употребе великог броја алата за учење на даљину (енг. distance learning). Већина тих алата су, у основи, интернет засноване апликације и подразумијевају интеракцију студента (ученика) са системом, без или уз присуство наставника (тутора). Неки од новијих оваквих система описани су у [13, 19, 17, 18, 14, 52, 101]. Приступ образовном материјалу путем интернета омогућио је такозвани "било кад - било гдје" (енг. whenever - wherever) концепт који подразумијева да студенти, у произвољно вријеме, са произвољног мјеста преко разних система - алата, могу приступати образовном материјалу и тако учити. Са друге стране, већина оваквих система не укључује персонализацију садржаја према студентовим могућностима и способностима. Стога се за прилагођавање материјала специфичним потребама студентима морају ангажовати наставници (тутори), што у ситуацијама великог броја студената и различитих планова и програма представља изузетно захтјеван и скуп посао. Ове чињенице

оправдавају потребу за развојем интелигентних система који прилагођавају образовни материјал појединачним студентима, чиме се унапређује читав образовни процес.

Оптимизацијски проблеми који проистичу из овог разматрања се своде на одређивање оног сета образовног материјала који одговара студентовим потребама у најбољој могућој мјери. Модели којима се описују овакви системи најчешће одговарају тешким математичким проблемима (као на примјер проблемима мјешовитог цјелобројног линеарног програмирања или неким другим комбинаторним проблемима) и, с обзиром на огромну количину података који су у систему присутни, није могуће примијенити егзактне оптимизације. Стога је за рјешавање ових проблема оправдано користити приближне методе. Неке од хеуристичких метода кориштене за адаптивно електронско учење су:

- оптимизација помоћу роја честица (енг. Particle Swarm Optimization - PSO), [53] која се користи за креирање помоћног образовног материјала на основу креираних блогова;
- генетски алгоритам за креирање путева учења [53];
- приступ који комбинује методу k-тог најближег сусједа за класификацију објеката и генетски алгоритам [18] да би се идентификовали стилови студентовог учења унутар eLearning система;
- за креирање аутоматских одговора на студентова питања у [57] је развијен ефикасан генетски алгоритам,
- алат који користи генетски алгоритам за креирање персонализованих електронских курсева уз наглашено неукључивање неадекватног садржаја [15].

Проблеми аутоматског креирања тестова се такође могу математички моделовати. Тако је на примјер, у чланку [55] развијен модел мјешовитог цјелобројног линеарног програмирања за рјешавање проблема организовања тестова на основу великих банака питања, те двије хеуристичке методе које рјешавају дати проблем. За аутоматско креирање кориштени су разни еволуцијски приступи [56, 54, 71], те PSO алгоритми [103] или у [24]), гдје се PSO метод користи за креирање тестова заснованих на процјени релевантности, тежине посјећених тема, фреквенцији употребе и тежини одређених питања.

Глава 2

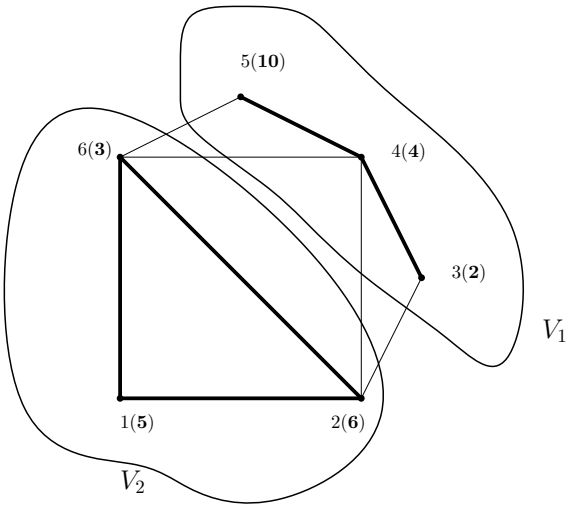
Проблем максимално балансиране повезане партиције у графу

2.1 Увод

Нека је $G = (V, E)$ повезан граф са скупом чворова $V = \{1, 2, ..., n\}$ и скупом грана E. Нека је $|E| = m$ и нека су w_i тежине на чворовима. За произвољан подскуп $V' \subset V$ уведимо ознаку $w(V')$ за суму тежина свих чворова из V', тј. $w(V') = \sum_{i \in V'} w_i$. Проблем максимално балансиране повезане партиције у графу (MBCP) је одређивање партиције скупа чворова V у два дисјунктна, непразна скупа V_1 и V_2, таква да су подграфови графа G индуковани са V_1 и V_2 повезани, а суме тежина чворова из V_1 и V_2 су по вриједности што ближе једна другој, тј. разлика између сума тежина је најмања могућа.

Формално, MBCP је проналажење партиције (V_1, V_2), такве да су пографови од G индуковани са V_1 and V_2 повезани, а вриједност $obj(V_1, V_2) = |w(V_1) - w(V_2)|$ је минимална.

Примјер 2.1. *Посматрајмо граф на слици 2.1. Нека су чворови означени бројевима* $1, 2, ..., 6$*, а тежине су наведене у заградама, поред ознака чворова. Укупна тежина свих чворова је 30. У најбољем случају, скупови чворова у партицији би требало да имају исту тежину (15), али у том случају она компонента која садржи чвор 5, (са тежином 10) би такође требала да садржи и чвор 1 (са тежином 5), или чворове 3 и 6, али у оба случаја компоненте* $\{1, 5\}$ *и* $\{3, 5, 6\}$ *нису повезане. Стога оптимално рјешење не може бити нула, већ најмање 2, јер је сума свих тежина паран број. Једно оптимално рјешење, које*



Слика 2.1: Примјер једноставног графа и рјешење за MBCP

је приказано и на слици 2.1, је $(V_1, V_2) = (\{3, 4, 5\}, \{1, 2, 6\})$*, и важи* $obj(V_1, V_2) = |16 - 14| = 2$*. Друго оптимално рјешење је* $(V_1', V_2') = (\{1, 2, 3, 6\}, \{4, 5\})$*, са једнаком вриједношћу функције циља* $obj(V_1', V_2') = |16 - 14| = 2$*.*

Мотивација за истраживање овог проблема се проналази у чињеници да многи алгоритми користе рекурзивни приступ: ако је потребно ријешити неки проблем на графу $G = (V, E)$, онда се прво граф подијели на два мања графа G_1 и G_2, који су приближно једнаких особина, те се онда дати алгоритам примијени на рјешавање потпроблема на добијеним мањим графовима. У случају да је полазни граф повезан, природно је захтијевати да су и новонастали, мањи графови повезани. Услов да подграфови G_1 и G_2 буду балансирани, што је више могуће, минимизује дубину рекурзије и различитост графова добијених у појединачним корацима рекурзивног процеса.

2.2 Примјена MBCP у образовању

У образовању, MBCP се може искористити за одређивање рјешења неких практичних организационих проблема, укључујући управљање курсевима, студентским групама или организовањем истраживача у повезане групе.

2.2.1 Неки практични примјери примјене

У овом одјељку укратко ће се навести неки практични примјери примјене МВСР у образовању: партиционисање (подјела) лекција курса на двије повезане, по тежини уједначене цјелине, организовање студената у групе и организацију истраживача на неком већем пројекту.

Материјал за неки курс се обично дијели у лекције и свакој лекцији се може додијелити тежина. Успостављање веза између лекција може се остварити по неколико критеријума, као што су сличност, аналогија, уопштење или условљеност. Подјела лекција курса у двије цјелине се врши уз логичан захтјев да свака цјелина буде повезана, а да тежине те двије цјелине буду у што већој мјери сличне.

Као други примјер можемо навести подјелу студената у двије мање групе. "Повезаност" студената се може на примјер дефинисати као "способност да раде заједно". Циљ би могао бити: подијелити студенте у двије мање групе, имајући у виду да групе треба да буду балансиране према способностима студената.

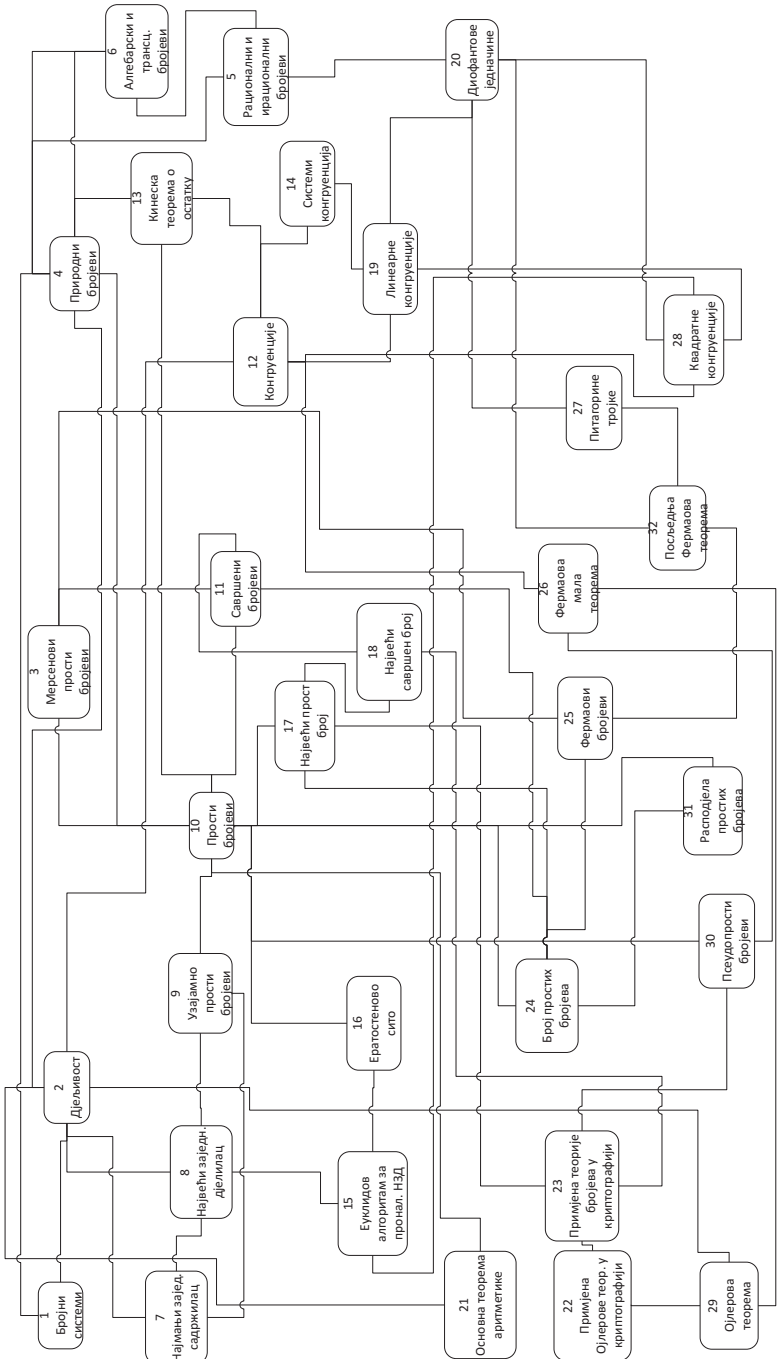
МВСР се може користити за организацију истраживача у неком већем пројекту. У овом случају, истраживачи се представљају чворовима у графу, а тежина чворова може бити дефинисана као "количина времена коју истраживач планира да проведе на пројекту". Релација "коауторство" је природна веза између истраживача и сличног је значења као поменута веза "способност да раде заједно" из претходног пасуса. Тако се проблем своди на одређивање партиције скупа свих истраживача, тако да је укупно вријеме потрошено на рад на пројекту у обје групе скоро подједнако.

2.2.2 Партиционисање курса из Увода у теорију бројева

У овом одјељку приказаћемо партицију курса Одабране теме из теорије бројева у двије балансиране партиције. Резултати презентовани у овом одјељку приказани су у раду [81].

Читав курс је подијељен у **32** лекције, а везе између лекција су приказане на слици 2.2.

Курс се представља као повезан граф код којег чворови одговарају лекцијама. Двије лекције су повезане само ако постоји директна семантичка или мето-



Слика 2.2: Лекције из курса Увод у теорију бројева

долошка веза. Прецизније, везе између лекција се успостављају у случају аналогије, условљености или се у обје лекције обрађују сличне математичке структуре.

На примјер, Фермаова мала теорема је директна посљедица Ојлерове теореме. Такође, она је у директној релацији са конгруенцијама и псеудо - простим бројевима, јер једна важна класа псеудо - простих бројева потиче из ње. Стога је Фермаова мала теорема повезана са ове три лекције: Ојлеровом теоремом, конгруенцијама и псеудо - простим бројевима. Са друге стране, у методолошком смислу, Фермаова мала теорема се тешко може довести у директну везу са Фермаовом посљедњом теоремом, иако обје потичу од истог математичара. Зато се између те двије теореме не успоставља веза. Семантичка веза је успостављена између лекција Прости бројеви и Савршени бројеви, док је између лекција Прости бројеви и Највећи прост број успостављена веза условљености. Између лекција Еуклидов алгоритам за проналажење НЗД-а и Ератостеново сито се успоставља веза сличности. Остале релације се идентификују на аналоган начин.

Треба напоменути да је одређивање веза између лекција флексибилно и у приличној мјери може да зависи од наставниковог мишљења и субјективне процјене. Детаљнија анализа успостављања веза између лекција у предложеном курсу се овдје изоставља, док се као основ за даље истраживање у овом смјеру предлажу књиге из Теорије бројева [83] и [50].

Одређивање тежине лекција такође може бити предмет субјективне процјене наставника и зависи од наставникове процјене о потребној количини градива, нивоу презентације лекције, циљева курса, студентских способности и слично. У сваком случају, могу се успоставити неки објективни критеријуми за одређивање тежине лекције. Да би се "измјерила" тежина лекције, потребно је увести факторе који утичу на тежину:

- Фактор I: обим лекције;
- Фактор II: ниво типичних студентских способности и знања прије почетка слушања лекције;
- Фактор III: важност лекције за наредне лекције;
- Фактор IV: важност лекције за математику уопште;

Јасно је да фактор I утиче на тежину лекције, јер су лекције које садрже више дефиниција и теорема, генерално теже него краће лекције. Други фактор се повезује са нивоом потребног студентовог знања и способности које су потребне да би лекција била лакше разумљива. Посљедња два фактора указују на ниво детаљности до ког се лекција обрађује, у смислу да лекције које су предуслови за наредне лекције или генерално математичко знање треба да буду обрађиване детаљније.

Да би се одредила тежина сваке лекције, уводимо сљедећу нотацију: Нека коефицијенти p_1, p_2, p_3 и p_4 означавају утицај фактора (I-IV) и нека је L лекција. Ако са $f_1(L), f_2(L), f_3(L)$ и $f_4(L)$ означимо вриједности додијељене сваком фактору (I-IV)за дату лекцију L, тада се тежина лекције L рачуна помоћу формуле

$$w(L) = 10 \sum_{i=1}^{4} p_i f_i(L). \tag{2.1}$$

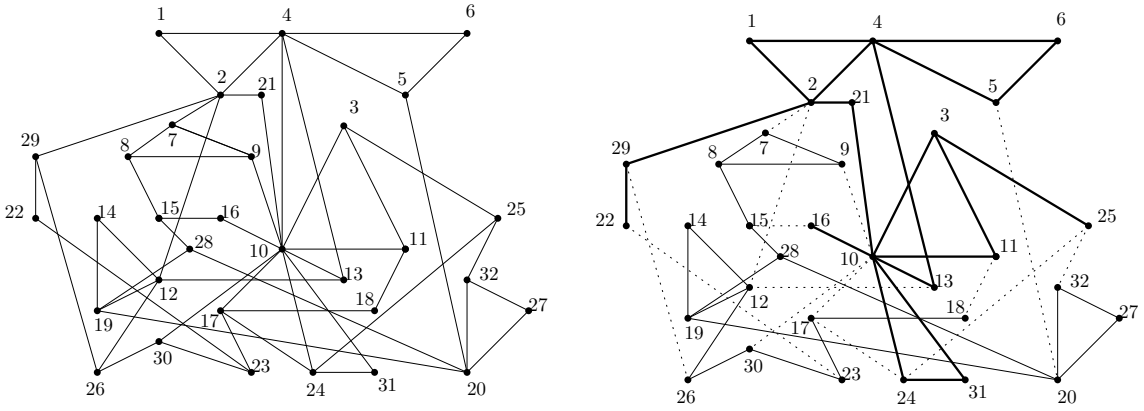
Све вриједности $p_i,\ i = 1..4$ се бирају из интервала $[0, 1]$ и важи да је укупна сума

$$\sum_{i=1}^{4} p_i = 1. \tag{2.2}$$

У нашем примјеру, коефицијенти $p_i,\ i = 1..4$ су постављени на вриједности: $p_1 = 0.45$, $p_2 = 0.25$, $p_3 = 0.15$ и $p_4 = 0.15$.

Табела 2.1 садржи информације о додијељеним вриједностима за тежине сваке појединачне лекције посматраног курса Одабране теме из теорије бројева. Прве двије колоне се односе на назив лекције (додијељени број и име). Наредне четири колоне садрже вриједности које се односе на факторе I-IV. Посљедња колона, названа $w(L)$ садржи вриједност тежине лекције, која се рачуна по формули (2.1).

Вриједности за сва четири фактора су скалиране у интервал $[0, 10]$, и с обзиром на (2.2), лако се види да за сваку лекцију L, вриједност $w(L)$ израчуната помоћу формуле (2.1) припада интервалу $[0, 100]$. На примјер, лекцији Прости бројеви су додијељене максималне вриједности за факторе f_1, f_3 и f_4. Тиме се одређује да је лекцији Прости бројеви, због обима лекције и њене примјене, како у даљим лекцијама тако и у математичком знању уопште, додијељена већа укупна тежина, која износи 85. Лекције које се односе на алгоритме за проналажење НЗД-а (Еуклидов алгоритам), односно простих бројева (Ератосте-



Слика 2.3: Графовска интерпретација (лијево) и рјешење (десно)

ново сито) не захтијевају високо предзнање, а по обиму су релативно мале. Овим лекцијама је дат мало већи значај за познавање математике уопште ($f_4 = 5$), јер се може сматрати да је познавање Еуклидовог алгоритма и принципа рада Ератостеновог сита ствар опште културе. Са друге стране, фактору f_4 је додијељен мањи утицај ($p_4 = 0.15$) на цјелокупну тежину, тако да свеукупно гледано, ове двије лекције немају велику укупну тежину и она за обје лекције износи по 22. Лекције које се односе на одређивање највећег простог броја (наравно, пошто је скуп простих бројева бесконачан, мисли се на одређивање највећег простог броја који се може тачно одредити) и на одређивање највећег познатог савршеног броја по обиму нису велике, али подразумијевају познавање техника теорије бројва, као и познавање техника које се односе на рад са великим бројевима на рачунару. Даље, ове двије лекције нису од великог значаја за наступајуће лекције, док им се придаје средњи значај за математичко знање уопште.

Опимтално рјешење добијено тоталном енумерацијом је 0.5, а одговарајућа оптимална партиција је:

$V_1 = \{1, 2, 3, 4, 5, 6, 10, 11, 13, 16, 21, 22, 24, 29, 31\}$

$V_2 = \{7, 8, 9, 12, 14, 15, 17, 18, 19, 20, 23, 25, 26, 27, 28, 30, 32\}$

Лако се може израчунати да је $w(V_1) = 713.5$ и $w(V_2) = 714$.

На слици 2.3 (лијево) је приказана графовска интерпретација датог курса. На десној страни, гране које припадају подграфу индукованом са V_1 су подебљане, док су гране из подграфа индукованог са V_2 у нормалној величини. Гране између ова два подграфа су приказане испрекиданим линијама.

Табела 2.1: Додјела тежина лекцијама

р.б.	Име лекције	f_1	f_2	f_3	f_4	$w(L)$
1	Бројни системи	3	4	3	5	35.5
2	Дјељивост	10	2	10	8	77
3	Мерсенови прости бројеви	3	4	2	3	31
4	Природни бројеви	6	2	4	10	53
5	Рационални и ирационални бројеви	6	2	4	8	50
6	Алгебарски и трансц. бројеви	6	4	4	8	55
7	Најмањи заједнички садржилац	5	1	6	10	49
8	Највећи заједнички дјелилац	5	1	6	10	49
9	Узајамно прости бројеви	3	4	2	2	29.5
10	Прости бројеви	10	4	10	10	85
11	Савршени бројеви	5	2	3	8	44
12	Конгруенције	10	6	10	8	87
13	Кинеска теорема о остатку	6	4	2	3	44.5
14	Системи конгруенција	6	6	4	4	54
15	Еуклидов алгоритам за рач. НЗД	2	1	2	5	22
16	Ератостеново сито	2	1	2	5	22
17	Највећи прост број	2	5	2	6	33.5
18	Највећи савршен број	2	5	2	6	33.5
19	Линеарне конгруенције	6	4	4	3	47.5
20	Диофантове једначине	6	4	6	6	55
21	Основна теорема аритемтике	6	4	6	8	58
22	Примјена Ојлерове теор. у криптограф.	6	4	4	3	47.5
23	Примјена теор. бројева у криптографији	6	4	6	3	50.5
24	Број простих бројева	6	4	2	3	44.5
25	Фермаови бројеви	3	4	2	3	31
26	Фермаова мала теорема	3	4	2	6	35.5
27	Питагорине тројке	3	1	2	5	26.5
28	Квадратне конгруенције	4	1	2	3	28
29	Ојлерова теорема	2	1	2	5	22
30	Псеудопрости бројеви	3	4	2	2	29.5
31	Распоред простих бројева	6	4	2	3	44.5
32	Посљедња Фермаова теорема	6	5	3	6	53

2.3 Ранији резултати

Први значајнији теоријски резултати који се односе на рјешавање MBCP се јављају у [25], гдје је доказано да је проблем NP тежак и сугерисан је једноставан приближни алгоритам полиномске временске зависности, са добрим фактором апроксимације 1.072.

Успјешна имплементација генетског алгоритма за рјешавање MBCP приказана је у [33]. Поменути GA користи бинарну репрезентацију хромозома, помоћу које се чворови распоређују у једну од двије компоненте. У случају неповезаних компоненти (самим тим некоректних рјешења), примјењује се казнена функција, помоћу које се GA усмјерава ка простору допустивих рјешења. Поузданост и ефективност овог GA је испитана на случајно генерисаним инстанцама које садрже до 300 чворова и 2000 грана.

MBCP припада широкој класи проблема који се односе на одређивање партиција у графу и као такав проналази примјену у различитим областима науке и технике.

На примјер, за управљање и рутирање у великим бежичним мрежама, у мрежи од N кластера сваки кластер одговара једној глави кластера (енг. claster head). Да би се поједноставило управљање мрежом, идеја је да се тако велика мрежа подијели у двије подмреже, којима би се управљало независно једном од друге. Мрежа се моделира као неусмјерен повезан граф, $G(H, A)$, гдје је H скуп глава кластера , $H = \{CH_i : i = 1..N\}$ а A је скуп неусмјерених веза (CH_i, CH_j), гдје су CH_i и CH_j двије главе кластера. Циљ је направити партицију графа G у повезане балансиране партиције и овај проблем одговара MBCP. У [98], аутори су прилагодили приступ презентован у [25] и искористили га да подијеле мрежу у двије мање, повезане подмреже.

2.4 Формулација мјешовитог цјелобројног линеарног програмирања

Ова секција садржи формулацију мјешовитог цјелобројног линеарног програмирања за MBCP. Резултати приказани у овој секцији су приказани у раду [79], који је у вријеме писања дисертације у фази рецензије.

Представљање проблема дискретне оптимизације помоћу формулација мјешовитог цјелобројног линеарног програмирања (енг. Mixed Integer Linear Programming - MILP) може бити корисно за проналажење оптималних рјешења датог проблема. Ово се обично постиже примјеном разних оптимизационих техника и употребом неких од добро познатих програма за рјешавање ових проблема [65, 1, 94].

Због саме природе функције циља, није тешко одредити њену линеарну формулацију у MILP моделу. Са друге стране, одређивање ограничења, које дефинишу повезаност индукованих подграфова, представља захтјеван задатак, посебно уколико је циљ направити модел са полиномским бројем ограничења. Једна идеја која води ка том рјешењу је кориштење приступа сличног као у мрежном току (енг. network flow). Овај приступ подразумијева да граф, на који се ток примјењује буде повезан и да свака грана има оријентацију. Пошто, приликом одређивања крајњег рјешења, није унапријед познато којој партицији (V_1 или V_2) припада који чвор, идеја је да се читавом графу придода један нови чвор (означен бројем 0) и да тај чвор буде полазна тачка за ток. Након тога се од тог чвора пушта ток величине $|V|$, док је циљ да се на сваком преосталом чвору графа остави ток величине 1. На тај начин се гарантује да ће сви чворови бити укључени у ток. Тако, додатни чвор рјешава проблем повезаности (чвор 0 у старту повежемо са свим чворовима из V).

Пошто наш граф није усмјерен, потребно је дефинисати оријентацију грана. Оријентација грана које су инцидентне са чвором 0 је јасна (све гране су усмјерене тако да је чвор 0 полазни). Да би се одредила оријентација преосталих грана, уводимо идеју покривајућег стабла. У стаблима, природно уређење чворова иде од "оца" ка "потомцима". Уколико на датим подграфовима буде успјела конструкција одговарајућих покривајућих стабала, то би значило да је испуњен услов повезаности (стабла су повезана). Ограничења која дефинишу стабло на подграфовима сама по себи нису компликована, јер је довољно испитати да ли је број грана у сваком индукованом подграфу G_1 и G_2 једнак $|V_1|-1$, односно $|V_2|-1$, уз искључење појаве контура на датим подграфовима. На примјер, да би се искључила могућност појаве контура, у подграфу G_1, потребно је и довољно испитати да ли сваки подграф индукован било којим скупом чворова S, $|S|>3, S \subset G_1$ има највише $|S|-1$ грану. У овом случају, увођење

мрежног тока се чак чини и непотребним. Међутим, овако се за дати граф у моделу линеарног програмирања јавља експоненцијалан број ограничења. Стога, прописани модел комбинује идеју мрежног тока и покривајућег стабла: покривајуће стабло дефинише оријентацију грана, док модел тока искључује појаву контура: у супротном, ако би у току постојала контура са током већим од 0, у том случају би се ток могао пропуштати кроз контуру неограничено много пута, што доводи до јасне контрадикције. Такође, треба примијетити да у свакој компоненти постоји по један чвор, који је полазни чвор за ток у компоненти.

Да би се конструисао модел линеарног програмирања заснован на горе описаном принципу, потребно је увести додатне елементе. Нека је $G = (V, E)$ граф. Прво уводимо чвор 0, који је повезан са свим чворовима из V. Означимо са $\overline{V}$ проширени скуп чворова, тј. $\overline{V} = V \cup \{0\}$. Проширеном скупу грана (означимо га са $\overline{E}$) је у односу на полазни скуп придодато нових $|V|$ грана које су инцидентне са чвором 0 и чворовима из V, $\overline{E} = E \cup \partial E$, гдје је $\partial E = \{(0, i) : i \in V\}$. Укупно суму свих тежина чворова из V означимо са $w_{sum} = \sum_{v_i \in V} w_i$. Циљ је одредити партицију скупа чворова у два непразна дисјунктна скупа V_1 и V_2, такву да су индуковани подргафови G_1 and G_2 повезани и вриједност $obj(V_1, V_2) = |w(V_1) - w(V_2)|$ је минимална. Нека су E_1 и E_2 скупови грана из G_1 и G_2, респективно.

Пошто су G_1 и G_2 повезани, садрже покривајућа стабла, $T_1(V_1, E_1')$ и $T_2(V_2, E_2')$, респективно. Нека су p и q, $p \in V_1$ и $q \in V_2$ два чвора. Дефинишимо скупове $\overline{E}_1' = E_1' \cup \{(0, p)\}$ и $\overline{E}_2' = E_2' \cup \{(0, q)\}$, као и графове $T = (V, E_1' \cup E_2')$ и $\overline{T} = (\overline{V}, \overline{E}_1' \cup \overline{E}_2')$.

Сада уводимо промјењиве за MILP модел

$$x_i = \begin{cases} 1, & i \in V_1 \\ 0, & i \in V_2 \end{cases}, \tag{2.3}$$

$$y_e = \begin{cases} 1 & e \in \overline{E}_1' \\ 0 & \text{иначе} \end{cases} \qquad z_e = \begin{cases} 1 & e \in \overline{E}_2' \\ 0 & \text{иначе} \end{cases}, \tag{2.4}$$

$$u_e \in [-n, n], e \in \overline{E}. \tag{2.5}$$

За сваку грану $e \in \overline{E}$, промјењива u_e означава вриједност тока на грани e.

MILP формулација за MBCP је:

$$\min \ -w_{sum} + 2\sum_{i=1}^{n} x_i w_i, \tag{2.6}$$

уз ограничења

$$2\sum_{i=1}^{n} x_i w_i \geq W, \tag{2.7}$$

$$y_e \leq \frac{1}{2}x_{i_e} + \frac{1}{2}x_{j_e}, \quad e \in E, \tag{2.8}$$

$$z_e \leq 1 - \frac{1}{2}x_{i_e} - \frac{1}{2}x_{j_e}, \quad e \in E, \tag{2.9}$$

$$y_e \leq x_{j_e}, \quad e \in \partial E, \tag{2.10}$$

$$z_e \leq 1 - x_{j_e}, \quad e \in \partial E, \tag{2.11}$$

$$u_e \leq n \cdot y_e + n \cdot z_e, \quad e \in \overline{E}, \tag{2.12}$$

$$u_e \geq -n \cdot y_e - n \cdot z_e, \quad e \in \overline{E}, \tag{2.13}$$

$$\sum_{e:j_e=i} u_e - \sum_{e:i_e=i} u_e = 1, \quad i \in V, \tag{2.14}$$

$$\sum_{e:i_e=0} u_e = n, \tag{2.15}$$

$$\sum_{e\in E} y_e + \sum_{e\in E} z_e = n - 2, \tag{2.16}$$

$$\sum_{e\in \partial E} y_e + \sum_{e\in \partial E} z_e = 2, \tag{2.17}$$

$$x_i, y_e, z_e \in \{0, 1\}, \ i \in V, \ e \in \overline{E}, \tag{2.18}$$

$$u_e \in [-n, n], \ e \in \overline{E}. \tag{2.19}$$

Као што се може видјети, сва три вектора y, z и u се односе на гране које постоје у графу, као и на гране између придодатог чвора 0 и преосталих

чворова. Лако се може израчунати да модел садржи укупно $2m + 3n$ бинарних варијабли, $m + n$ реалних варијабли $4m + 5n + 4$ ограничења.

Сада се obj_{MILP} дефинише као $obj_{MILP}(x, y, z, u) = -w_{sum} + 2\sum_{i=1}^{n} x_i w_i$ с обзиром на ограничења (2.7)-(2.19).

Покажимо да је рјешење добијено из MILP формулације уједно и рјешење за MBCP.

Сљедећа Лема даје гаранцију да у свакој компоненти постоји тачно по један чвор, повезан са додатним чвором 0.

Лема 1. *Из ограничења (2.8)-(2.17) слиједи да 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 $u_{(0,i)} = 0,\ i \neq p, q$.*

Доказ. Из (2.17), директно слиједи да постоје двије гране $(0, p)$ и $(0, q)$. Ове гране припадају скупу ∂E и за њих важи $y_e = 1$ или $z_e = 1$. Треба показати да p и q припадају различитим подскуповима V_1 и V_2. Супротно и без губљења општости, претпоставимо да оба чвора p и q припадају V_1. Из (2.15) слиједи $u_{(0,p)} + u_{(0,q)}{=}n$.

Даље, у случају када $i_e \in V_1$ и $j_e \in V_2$, десна страна ограничења (2.8) и (2.9) су једнаке $\frac{1}{2}$ и с обзиром да су промјењиве y_e и z_e бинарне, слиједи да је $y_e = 0$ и $z_e = 0$. За такве гране e, с обзиром на ограничења (2.12) и (2.13), слиједи да је $u_e = 0$. Слично, за гране за које важи да је $i_e \in V_2$ И $j_e \in V_1$, такође је $u_e = 0$.

Саберимо сва ограничења из (2.14).

$$|V_1| = \sum_{i \in V1} \Big(\sum_{e:j_e=i} u_e - \sum_{e:i_e=i} u_e \Big) =$$

$$= \sum_{e:j_e \in V_1} u_e - \sum_{e:i_e \in V_1} u_e =$$

$$= \sum_{e:i_e \in V_1 \wedge j_e \in V_1} u_e + \sum_{e:i_e=0 \wedge j_e \in V_1} u_e + \sum_{e:i_e \in V_2 \wedge j_e \in V_1} u_e - \left(\sum_{e:i_e \in V_1 \wedge j_e \in V_1} u_e + \sum_{e:i_e \in V_1 \wedge j_e \in V_2} u_e \right)$$

$$= \sum_{e:i_e=0 \wedge j_e \in V_1} u_e = u_{(0,p)} + u_{(0,q)} = n,$$

јер се прва и четврта сума анулирају, док трећа и пета сума имају све сабирке једнаке нула, како је горе и доказано. Тако се добија израз $|V_1| = n$, што је у контрадикцији са чињеницом да не могу сви x_i бити једнаки 1. Заиста, ако би сви x_i били једнаки 1, тада би једна партиција садржавала све чворове, а друга

ниједан чвор. Функција циља би у том случају била једнака збиру свих тежина, што је немогуће ако граф има бар два чвора. Тиме је доказ леме завршен. □

Теорема 1. *Партиција $V^* = (V_1^*, V_2^*)$ је оптимална ако и само ако постоји оптимално рјешење (x, y, z, u) проблема (2.6)-(2.19).*

Доказ. ($\Rightarrow$)

За партицију V^*, прво ће се конструисати промјенљиве (x, y, z, u) , такве да важи $obj_{MILP}(x, y, z, u) \leq obj(V_1^*, V_2^*)$.

Без губљења општости, претпоставимо да је $w(V_1^*) \geq w(V_2^*)$. Нека су промјенљиве x_i дефинисане према (2.3), а y_e и z_e према (2.4). Одмах се види да је ограничење (2.18) испуњено.

Промјенљиве $u_e, e \in \overline{E}$ се дефинишу на сљедећи начин:

За $e \in \partial E$, дефинишемо

$$u_e = \begin{cases} |V_1^*|, e_j = p \\ |V_2^*|, e_j = q \ . \\ 0, \text{ иначе} \end{cases} \tag{2.20}$$

Пошто је $|V_1^*| < n$ и $|V_2^*| < n$, $u_e \in [-n, n]$, за све $e \in \partial E$.

Да би се дефинисале вриједности u_e за преостале гране, $e \in \overline{E} \backslash \partial E$, користићемо чињеницу да у стаблу било који чвор може бити декларисан као коријен, док се сви остали чворови могу уредити према неком алгоритму за претраживање стабла. Не примјер, може се користити један од два стандардна алгоритма за претрагу: претрага у дубину или претрага у ширину. Нека су E_1' и E_2' покривајућа стабла од V_1^* и V_2^* добијена неким од алгоритама за претрагу. Користећи дефинисан редослијед чворова одређен тим алгоритмом, у случају када је чвор i_e "отац" чвора j_e, вриједност тока u_{i_e,j_e} се дефинише као број чворова у подстаблу, са коријеном j_e. У случају да је чвор j_e "отац" од i_e, вриједност тока u_{i_e,j_e} се дефинише као негативна вриједност броја чворова у подстаблу, са коријеном i_e. За све остале гране, које не припадају ни E_1' ни E_2', дефинишемо $u_e = 0$. Пошто је број чворова у сваком подстаблу мањи или једнак n, важи да u_e припада $[-n, n]$, за сваку грану $e \in E$. Тако је ограничење (2.19) задовољено за све $e \in \overline{E}$.

Неједнакост $obj_{MILP}(x, y, z, u) \leq obj(V_1^*, V_2^*)$ је задовољена због

$obj_{MILP}(x,y,z,u) = -w_{sum} + 2\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} w_i(2x_i - 1) = \sum_{i \in V_1^*} w_i(2x_i - 1) + \sum_{i \in V_2^*} w_i(2x_i - 1) = \sum_{i \in V_1^*} w_i - \sum_{i \in V_2^*} w_i = w(V_1^*) - w(V_2^*) = obj(V^*)$.

Треба примијетити да је доказано $obj_{MILP} = obj(V^*)$, што је јачи услов од потребног. Такође, с обзиром да је $obj(V^*) \geq 0$, испуњено је и (2.7).

Докажимо сада да су ограничења (2.8)-(2.17) задовољена.

Да би се доказала неједнакост (2.8), разматрају се два случаја:

i. $y_e = 0$. Неједнакости (2.8) су задовољене, јер је $x_{i_e},\ x_{j_e} \geq 0$ по дефиницији.

ii. $y_e = 1 \Rightarrow e \in E_1' \Rightarrow x_{i_e} = x_{j_e} = 1 \Rightarrow y_e \leq \frac{1}{2}x_{i_e} + \frac{1}{2}x_{j_e}$

Да би се доказале неједнакости (2.10), поново разматрамо два случаја:

i. $y_e = 0$. Неједнакости (2.10) су задовољене јер је $x_{j_e} \geq 0$ по дефиницији.

ii. $y_e = 1 \Rightarrow e \in \overline{E}_1'$. Пошто $e \in \partial E$, $e \in \overline{E}_1' \cap \partial E = \{(0,p)\}$, одакле слиједи $j_e = p \in V_1 \Rightarrow x_{j_e} = 1$.

Неједнакости (2.9) и (2.11) се доказују аналогно као у претходна два корака.

Да би се доказале неједнакости (2.12) и (2.13) поново анализирамо два случаја:

i. $e \in \overline{E}_1' \cup \overline{E}_2'$. Десне стране неједнакости (2.12) и (2.13) су једнаке n и $-n$, респективно. Пошто $u_e \in [-n, n]$, важи да су (2.12) и (2.13) задовољене.

ii. $e \notin \overline{E}_1' \cup \overline{E}_2'$. Тада је u_e једнако 0 по дефиницији. Неједнакости (2.12) и (2.13) су задовољене, јер су десне стране (2.12) ненегативне, а десне стране (2.12) непозитивне.

Да бисмо доказали неједнакост (2.14), без губљења општости, претпоставимо да $i \in V_1$. Посматрајмо гране из $\overline{E}_1'$, које имају или полазни или крајњи чвор i. Једна од ових грана полази од "оца" за дати чвор, док све остале гране воде ка "потомцима". За све друге гране инцидентне са i, а које не припадају $\overline{E}_1'$, вриједност u_e је једнака 0 и оне не утичу на (2.14).

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

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

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

У прве двије суме, постоји само једна грана која задовољава услове и та грана води од "оца" чвора i до чвора i. За ту грану, $|u_e|$ је једнако броју чворова у подстаблу са коријеном i. У посљедње двије суме, учествују све гране са полазним чвором i ка свим његовим "потомцима" у покривајућем стаблу T_1. За ове гране, вриједност $|u_e|$ као број чворова у одговарајућем подстаблу, са коријеном у неком "потомку" од i.

С обзиром да једино чвор i учествује у подстаблу код кога је коријен управо i важи $\sum_{e:j_e=i\wedge u_e>0} |u_e| + \sum_{e:i_e=i\wedge u_e<0} |u_e| - (\sum_{e:j_e=i\wedge u_e<0} |u_e| + \sum_{e:i_e=i\wedge u_e>0} |u_e|) = 1.$

Да бисмо доказали неједнакост (2.15), користимо дефиницију (2.20) од $u_e, e \in \partial E$:

$$\sum_{e:i_e=0} u_e = \sum_{e\in\partial E} u_e = u_{(0,p)} + u_{(0,q)} = |V_1^*| + |V_2^*| = n$$

Пошто су T_1 и T_2 покривајућа стабла од G_1 и G_2, важи $\sum_{e\in E} y_e = |E_1'| = |V_1| - 1$ и $\sum_{e\in E} z_e = |E_2'| = |V_2| - 1$. Одатле је $\sum_{e\in E} y_e + \sum_{e\in E} z_e = |V_1| + |V_2| - 2 = n - 2$. Тако је неједнакост (2.16) доказана.

За $e \in \partial E$, $y_e = 1$ за само једну грану e и то управо за ону грану $e = (0, p)$. За све преостале гране из $e \in \partial E$, $y_e = 0$, одакле слиједи да је $\sum_{e\in\partial E} y_e = 1$. Слично, $\sum_{e\in\partial E} z_e = 1$, одакле се закључује да је неједнакост (2.17) задовољена.

($\Leftarrow$). Претпоставимо да је (x^*, y^*, z^*, u^*) оптимално рјешење које задовољава услове (2.6)-(2.19). На основу тог рјешења, конструисаће се партиција (V_1, V_2), таква да је задовољено $obj(V_1, V_2) \leq\ obj_{MILP}(x^*, y^*, z^*, u^*)$.

Дефинишимо

$V_1 = \{i \in V : x_i = 1\}$, $\overline{E}_1' = \{e \in E : y_e = 1\}$ и

$V_2 = \{i \in V : x_i = 0\}$, $\overline{E}_2' = \{e \in E : z_e = 1\}$.

Ограничења (2.8)-(2.11) гарантују да су скупови $\overline{E}_1'$ и $\overline{E}_2'$ добро дефинисани, тј. све гране из $\overline{E}_1'$ имају крајње чворове у $V_1 \cup \{0\}$ док све гране из $\overline{E}_2'$ имају крајње чворове из $V_2 \cup \{0\}$:

i. $e \in E_1'$ слиједи $y_e = 1$. Из ограничења (2.8) и бинарне природе промјењивих x_{i_e} и x_{j_e}, слиједи да је $x_{i_e} = 1$ и $x_{j_e} = 1$, одакле $i_e,\ j_e \in V_1$.

ii. $e \in \partial E \cap \overline{E}_1'$ такође имплицира $y_e = 1$. Из ограничења (2.10) слиједи $x_{j_e} = 1$ одакле даље слиједи $j_e \in V_1$.

Слично, ограничења (2.9) и (2.11) обезбјеђују да све гране из $\overline{E}'_2$ имају крајње чворове из $V_2 \cup \{0\}$. Ограничење (2.16) обезбјеђује да је укупан број грана које су укључене у покривајуће стабло управо $n-2$. Већ је показано да је

$w(V_1) - w(V_2) = -w_{sum} + 2\sum_{i=1}^{n} x_i w_i$. Из ограничења (2.7) слиједи $w(V_1) - w(V_2) \geq 0$, тако је неједнакост $|w(V_1) - w(V_2)| \leq -w_{sum} + 2\sum_{i=1}^{n} x_i w_i$ испуњена.

Из неједнакости (2.12) и (2.13), слиједи да је $u_e = 0$, за све $e \notin \overline{E}'_1 \cup \overline{E}'_2$.

Сада се може показати да су графови (V_1, E'_1) и (V_2, E'_2) повезани. Доказаћемо повезаност графа (V_1, E'_1), док је за други граф доказ сличан.

Нека су S' S'' два произвољна подскупа од V_1, таква да је, $S' \cup S'' = V_1$ и $S' \cap S'' = \oslash$, $S', S'' \neq \oslash$. Доказаћемо да постоји грана $(\exists e \in E'_1)$, $i_e \in S' \wedge j_e \in S''$.

Саберимо сва ограничења дата са (2.14), за све $i \in S'$. Добијамо

$$\sum_{i \in S'} (\sum_{e:j_e \in S'} u_e - \sum_{e:i_e \in S'} u_e) = |S'|.$$

Ако развијемо суму на лијевој страни, добијамо сљедећи израз:

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

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

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

Означимо сабирке у посљедњој једнакости са A, B, C, D, E, F и G, и запишимо ту једнакост као

$$\sum_{i \in S'} (\sum_{e:j_e \in S'} u_e - \sum_{e:i_e \in S'} u_e) = A + B + C + D - (E + F + G)$$

Очигледно је $A = E$ и $D = G = 0$ (између V_1 и V_2 нема грана у E'). Даље, $C = 0$ или $C = |V_1|$ у зависности од тога да ли $p \in S'$ или $p \notin S'$, јер Лема 1 прописује да постоји тачно један чвор (p) из V_1 повезан са чвором 0. Тако добијамо

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

Из посљедње неједнакости слиједи $(\exists e)$ $(((i_e \in S') \wedge (j_e \in S'')) \vee ((i_e \in S'') \wedge (j_e \in S')))$ $u_e \neq 0$. Из $u_e \neq 0 \Rightarrow y_e = 1 \Rightarrow e \in E_1'$ и тако смо пронашли грану између S' и S'' .

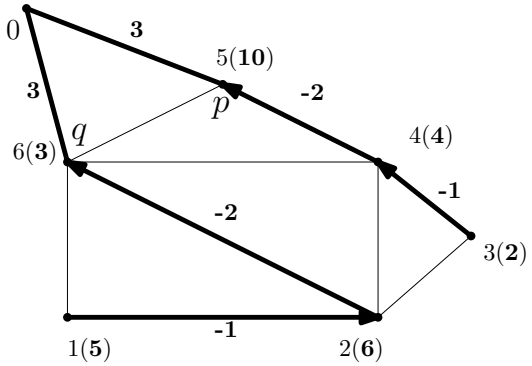
Тиме је доказана повезаност (V_1, E_1').

Дакле, конструисана партиција (V_1, V_2) је повезана и важи $obj(V_1, V_2) \leq obj_{MILP}(x^*, y^*, y^*, u^*)$. □

Примјер 2.2. *Нека је G граф из примјера 2.1. Оптимално рјешење, након рјешавања MILP-а (2.6)-(2.19) употребом CPLEX* [27] *рјешавача је:*

$x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 1, x_6 = 0$

Вриједности за векторе y, z и u су дате у табели 2.2, док је рјешење приказано на слици 2.4.



Слика 2.4: Рјешење за граф из Примјера 2.1

На основу Леме 1, додатни чвор је повезан са по тачно једним чвором по компоненти. Ако се држимо већ уведене нотације, за прву компоненту чвор p је чвор 5, док је за другу компоненту чвор q чвор 6. Из Теореме 1 слиједи да су токови на гранама $(0, p)$ и $(0, q)$ једнаки $|V_1|$ и $|V_2|$: тј. токови на гранама $(0, 5)$ и $(0, 6)$ су једнаки 3 у ова случаја, јер свака компонента садржи по 3 чвора. Свака грана која припада покривајућим стаблима има

Табела 2.2: Вриједности за y, z, u рачунате CPLEX-ом

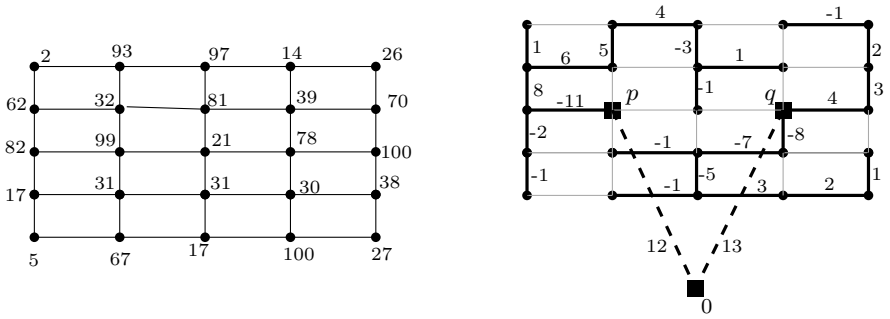
грана	1-2	1-6	2-3	2-4	2-6	3-4	4-5	4-6	5-6	0-1	0-2	0-3	0-4	0-5	0-6
y	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0
z	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
u	-1	0	0	0	-2	-1	-2	0	0	0	0	0	0	0	3

ток различит од нуле, док знак тока зависи од оријентације гране. Токови на гранама које не припадају покривајућим стаблима су једнаки нули. Иако је граф релативно мали, види се да је (апсолутна) вриједност тока за сваку грану (i, j) у покривајућем стаблу једнака броју чворова у одговарајућем подстаблу са коријеном у чвору j.

Примјер 2.3. *На слици 2.5 (лијево) приказана је правоугаона мрежа (енг. grid graph). Правоугаона мрежа димензије $n = r \times s$ је граф код којег чворови одговарају тачкама у равни са цјелобројним координатама. Два чвора су повезани граном кадгод су одговарајуће тачке на удаљености 1. Унутрашњи чворови правоугаоне мреже имају по четири сусједа, чворови дуж страна графа имају по три сусједа, док чворови у угловима (укупно четири чвора) имају по два сусједа. Правоугаоне мреже су ријетки графови и садрже укупно $(2\,rs - r - s)$ грана. Једна тежинска правоугаона мрежа, димензије 5×5 је приказан на слици 2.5 (лијево) и садржи укупно 25 чворова и 40 грана. Једно рјешење је приказано на слици 2.5 (десно). Да би се поједноставила слика 2.5 (десно), изостављене су тежине чворова, док знак тока одређује оријентацију. Као што се види са слике, додатни чвор 0 је повезан са чворовима 12 (чвор p) и 14 (чвор q). Токови на гранама $(0, p)$ и $(0, q)$ су једнаки 12 и 13, што указује да компоненте у партицији имају 12 односно 13 чворова. Као што је наведено у доказу теореме 1, апсолутна вриједност тока на свакој грани (i, j), која припада покривајућем стаблу, је једнака броју чворова у подстаблу са коријеном у чвору j. Ако се израчунају суме тежина чворова по компонентама, види се да је разлика тих сума једнака $|630 - 629| = 1$. Ова чињеница доказује да је рјешење приказано на слици 2.5 (десно) оптимално, јер је сума свих тежина непарна, а функција циља је ненегативна.*

2.5 Метода промјенљивих околина за рјешавање MBCP

У овој секцији описана је метода промјенљивих околина за рјешавање MBCP. Резултати приказани у овој секцији су приказани заједно са резултатима из претходне секције у раду [79]. Детаљнији опис VNS методе дат је у уводној глави, те се овдје због тога изоставља.



Слика 2.5: Правоугаона мрежа 05x05 (лијево) и партиција добијена MILP рјешавачем (десно)

2.5.1 Иницијализација и функција циља

Рјешење за дати граф $G = (V, E)$ је представљено као бинарни низ x дужине $|V|$. Елементи низа одговарају чворовима и указују на то којој компоненти дати чвор припада, тј. $i \in V_1$ ако $x_i = 1$ и $i \in V_2$ ако $x_i = 0$. Полазно рјешење се одређује на случајан начин.

Током процеса претраживања, јасно је да се оваквом репрезентацијом могу јавити недопустива рјешења, што значи да су један (или оба) подграфа индукована са V_1 и V_2 (G_1 и G_2 респективно) неповезани. Стандардан начин за превазилажење ситуације појаве некоректних рјешења је увођење казнене функције. Уопште, вриједност казнене функције би требало да буде мало већа него минимална цијена потребна да се одговарајуће некоректно рјешење "поправи" на коректно. Супротно, ако би вриједност казнене функције била мања од цијене минималне корекције, некоректна рјешења се могу јавити као коначно рјешење, што наравно није допуштено. Са друге стране, ако је вриједност казнене функције значајно већа од цијене минималне поправке, некоректна рјешења су тотално дискриминисана и у старту се одбацују. Ова појава није препоручљива, зато што се понекад добра, квалитетна рјешења лакше добијају поправком некоректних рјешења, него преко поправљања лоших и допустивих рјешења.

У случају MBCP, природно је да се у формулацију казнене функције укључи податак о броју компоненти повезаности. Овај број је потребно увећати одговарајући број пута, који зависи од конкретног графа. Експерименти су показали да у рачунање казнене функције треба укључити највећу тежину. Тако, да би се конструисала казнена функција, потребно је израчунати број компоненти

повезаности. За овај задатак, користи се алгоритам претраге у ширину (енг. breadth-first search algorithm - BFS). У случају када BFS пронађе неповезану компоненту (или обје) примјењује се казнена функција. Тако, ако су $nc(G_1)$ и $nc(G_2)$ бројеви компоненти повезаности од G_1, односно G_2, казнена функција f_{pen} се рачуна помоћу формуле

$$f_{pen} = (nc(G_1) + nc(G_2) - 2) * w_{max}, \tag{2.21}$$

гдје је w_{max} максимална тежина чвора у графу.

За дато рјешење, алгоритам рачуна вриједност функције циља помоћу формуле

$$obj(V_1, V_2) = |w(V_1) - w(V_2)| + f_{pen}. \tag{2.22}$$

Из (2.21), јасно је да ако су G_1 и G_2 повезани, казнена функција је једнака 0, јер је $nc(G_1) = nc(G_2) = 1$. У том случају, казнена функција нема утицај на рачунање вриједности функције циља (2.22). У случају појаве неповезаних компоненти, казнена функција значајно увећава вриједност функције циља, што значи да некоректна рјешења не могу дуго опстати. Чињеница да су завршна рјешења допустива указује на то да казнена функција има већу вриједност од цијене минималне поправке. Такође, према експерименталним резултатима, чини се да казнена функција није престрога.

На слици 2.6 је приказан псеудокод функције циља за дато рјешење S. За свако $i, i \in V$, вриједности $x[i]$ означавају компоненту за чвор i. Вриједност w_{max} означава максималну тежину, која се рачуна једном, приликом иницијализације. Вриједности $sum1$ и $sum2$ се рачунају као укупне суме тежина чворова из V_1 и V_2, док се вриједности $nc1$ и $nc2$ рачунају унутар функције ncomp(), која користи стандардни BFS алгоритам за одређивање компоненти повезаности.

Временска сложеност функције циља се одређује на сљедећи начин: временска зависност за рачунање суме тежина у одговарајућим подскуповима је $O(|V|)$. Временска сложеност за одређивање вриједности функције ncomp је једнака временској сложености BFS алгоритма, која износи $O(|V_1| + |E_1|)$ за прву и $O(|V_2|+|E_2|)$ за другу компоненту. Тако је укупна временска сложеност функције циља $O(|V| + |E|)$.

```
function Obj(S)
    sum1 := 0, sum2 := 0;
    nc1 := 0, nc2 := 0;
    for i ∈ V do
        if x[i] = 1
            then
                sum1 := sum1 + w[i];
            else
                sum2 := sum2 + w[i];
        endif
    endfor
    nc1:=ncomp(1);
    nc2:=ncomp(2);
    return abs(sum1 − sum2) + (nc1 + nc2 − 2) ∗ w_max
```

Слика 2.6: Приказ псеудо-кода за функцију циља

2.5.2 Околине и процедура размрдавања

У оквиру процедуре размрдавања, креира се ново рјешење x', ($x' \in N_k(x)$) засновано на тренутно најбољем рјешењу x. k-та околина се дефинише на сљедећи начин: Неких k чворова из V се бира на случајан начин. Сваком изабраном чвору мијења се компонента којој он припада: Ако је изабран чвор који припада V_1, тада се он пребацује у V_2 и обрнуто. Формално, k-та околина вектора x се записује као $N_k(x) = \{x' : \{i_1, i_2, ..., i_k\} \subset \{1, 2, ..., |V|\}\ x'_{i_j} = 1 - x_{i_j}\}$.

У алгоритму, k_{min} је постављено на вриједност 2, док се k_{max} дефинише као $k_{max} = \min\{30, \frac{|V|}{2}\}$. Разлог за овакву дефиницију вриједности k_{max} потиче из мотивације да буде задовољен теоријски услов који диктира да величина сваке наредне околине треба да буде већа од претходне. Пошто је величина k- те околине $\binom{|V|}{k}$, за $k < \frac{|V|}{2}$ слиједи $\binom{|V|}{k-1} < \binom{|V|}{k}$, тј. $|N_{k-1}(x)| < |N_k(x)|$ и услов је задовољен. За веће инстанце, експерименти су показали да је $k_{max} = 30$ довољна за достизање квалитетних рјешења.

Лако се види да се процедура размрдавања састоји од k корака временске сложености $O(|V|)$, тако да је укупна временска сложеност процедуре размрдавања $O(k_{max} \cdot |V|)$.

2.5.3 Локално претраживање

За рјешење x' добијено у процедури размрдавања позива се локално претраживање. У свакој итерацији локалне претраге, алгоритам размјењује компоненте за два чвора. На примјер, ако је у рјешењу x' $u \in V_1$ и $v \in V_2$, након

замјене, статус је $u \in V_2$ и $v \in V_1$. Нека је оваквом размјеном формирано ново рјешење x''. У случају да је x'' боље од x', тада x' постаје једнако x'', (у другом случају, x' се не мијења) и локално претраживање наставља замјеном сљедећег пара чворова.

Локално претраживање се завршава када се рјешење не може више побољшати. Ако је вриједност функције циља за рјешење x' строго веће од вриједности рјешења x, тада тренутно најбоље рјешење остаје x, а претрага се наставља са сљедећом околином.

Ако је вриједност функције циља за рјешење x' мање него за x, тада x' постаје тренутно најбоље рјешење ($x = x'$). Ако су вриједности функција циља за ова два рјешења иста, онда се са вјероватноћом p_{move} поставља $x = x'$ и алгоритам наставља претрагу са истом околином. У другом случају, претрага се са вјероватноћом $1-p_{move}$ наставља са истим рјешењем x и наредном околином.

Очигледно је да локална претрага формира парове чворова чији је укупан број $O(|V|^2)$. Замјена компоненти се врши у $O(1)$ времену, док се рачунање функције циља новог рјешења врши у времену $O(|V| + |E|)$. Стога је укупна временска зависност локалне претраге $O(|V|^3 + |E| \cdot |V|^2)$.

Пошто су разматране све околине, алгоритам наставља претраживање са првом околином, све док не буде испуњен критеријум за завршетак алгоритма. У нашем случају, критеријум за завршетак је достигнут максималан број итерација.

2.6 Експериментални резултати

У овом одјељку су приказани експериментални резултати који приказују ефикасност прописане MILP формулације и VNS метода.

Сви тестови су обављени на рачунару Intel Core 2 Quad Q9400 @2.66 GHz са 8 GB RAM. За тестирање MILP формулације кориштена су два MILP рјешавача: CPLEX 12.1 [27] и Gurobi 4.0 [45]. За оба рјешавача, вријеме извршења је ограничено на 7200 секунди. VNS имплементација је кодирана у програмском језику C. VNS је за сваку тестну инстанцу покретан 20 пута. За сваку инстанцу вриједност k_{min} је подешена на 2. Да би био испуњен теоријски услов $(\forall k)(k_{min} \leq k \leq k_{max} - 1)\ |N_k(x)| \leq |N_{k+1}(x)|$, за инстанце са мање од 60 чворова, k_{max} је подешено на $n/2$, а за веће инстанце на 30. Параметар p_{move} има вриједност

0.4. Алгоритам завршава са радном након 500 итерација. За све експерименте кориштена су иста два два скупа инстанци: правоугаоне мреже из [10], у овом раду означене као BR инстанце и случајно генерисане инстанце из [33], означене као DKTF инстанце.

Скуп BR инстанци се састоји од укупно 16 инстанци: најмања правоугаона мрежа има 25 чворова, (n = 05x05), а највећа 225 (n = 15x15). За инстанце чије име завршава словом (a), тежине чворова су цијели бројеви, изабрани униформно из интервала [1, 100], а у другом случају (инстанце чије име завршава словом (b)) вриједности тежина су униформно биране из интервала [1, 500]. Прва инстанца (са именом gg_05_05a) је управо инстанца приказана и анализирана у Примјеру 2.3.

Други скуп инстанци (DKTF инстанце) садржи укупно 21 повезан граф генерисан на случајан начин са реалним тежинама на чворовима, које су униформно биране из интервала (0, 100].

Табеле 2.3 и 2.4 приказују експерименталне резултате за два описана скупа инстанци. У прве три колоне приказана су имена инстанци, те број чворова и грана у графу. Наредна колона садржи оптималну вриједност, са знаком '-', у случају када оптимална вриједност није позната.

Наредне двије колоне се односе на два MILP рјешавача: CPLEX и Gurobi. Подаци у овим колонама су двојаки: у случају да је одговарајући MILP рјешавач успио да пронађе оптимално рјешење у мање од два сата (7200 секунди), приказано вријеме извршења. У другом случају, када рјешавач није пронашао оптимално рјешење у 7200 секунди, два подслучаја су анализирана: пронађено је рјешење, али није верификовано као оптимално, односно, рјешавач уопште није пронашао рјешење. У првом подслучају, приказано је то рјешење са ознаком (n.v.), која означава да рјешење није верификовано као оптимално. У другом подслучају, када MILP рјешавач није уопште успио да нађе рјешење, приказана је ознака N/A. Наредне двије колоне садрже добијене резултате и вријеме извршења GA из [33], са ознаком *opt* ако је GA за дату инстанцу достигао оптимално рјешење. Посљедње двије колоне се односе на прописани VNS: најбоља добијена вриједност, са ознаком *opt* ако је уједно и оптимална, те просјечно укупно вријеме извршења VNS алгоритма.

Лако се види да оптимално рјешење за BR инстанце не може бити мање

Табела 2.3: Експериментални резултати за BR инстанце

Инст.	$\|V\|$	$\|E\|$	opt	$t_{cplex}(s)$	$t_{gur}(s)$	GA	t_{GA}	VNS	$t_{VNS}(s)$
05x05a	25	40	1	11.26	1016.99	opt	0.36	opt	0.34
05x05b	25	40	1	56.41	536.27	opt	0.37	opt	0.33
05x06a	30	49	0	4.11	0.17	opt	0.41	opt	0.50
05x06b	30	49	1	101.58	7077.59	opt	0.43	opt	0.60
05x10a	50	85	1	866.49	2539.09	opt	0.72	opt	2.30
05x10b	50	85	0	N/A	N/A	opt	0.91	opt	2.47
05x20a	100	175	0	N/A	N/A	opt	1.73	opt	18.12
05x20b	100	175	1	N/A	N/A	opt	1.77	opt	20.81
07x07a	49	84	0	3139.73	2528.21	opt	0.804	opt	2.79
07x07b	49	84	1	1053.79	N/A	opt	0.743	opt	3.31
07x10a	70	123	1	N/A	1858.10	opt	1.249	opt	6.38
07x10b	70	123	0	6112.54	N/A	opt	1.186	opt	7.33
10x10a	100	180	1	N/A	N/A	opt	1.809	opt	22.40
10x10b	100	180	1	N/A	N/A	opt	1.633	opt	25.05
15x15a	225	420	0	N/A	N/A	opt	4.439	opt	195.49
15x15b	225	420	0	N/A	N/A	opt	4.669	opt	216.16

од 0, ако је сума свих тежина паран број, односно мања од 1 ако је сума тежина непарна, јер је вриједност функције циља ненегативна, а све тежине су цијели бројеви. Зато се закључује да су за све BR инстанце обје хеуристике достигле оптимално рјешење. Из Табеле 2.3 очигледно је да се MILP рјешавачи могу користити за проналажење оптималних рјешења за мање инстанце. У случају BR инстанци, оба рјешавача проналазе оптимална рјешења за већину инстанци до 70 чворова: CPLEX достиже 8 (од 10), док је Gurobi достигао 7 (од 10) оптималних рјешења. За веће инстанце, ниједан рјешавач није уопште пронашао рјешење, што је у табели означено ознаком N/A.

Из Табеле 2.4 се лако може уочити да је Gurobi рјешавач успјешнији у проналажењу оптималних рјешења за DKTF инстанце. За инстанце до 50 чворова, Gurobi проналази оптимална рјешења за све инстанце осим једне, док CPLEX проналази оптимално рјешење само за инстанце до 20 чворова и једну инстанцу која садржи 30 чворова. Такође, вријеме извршења Gurobi алгоритма је мање него код CPLEX-а.

Као што је очекивано, MILP рјешавачи нису могли достићи оптимална рјешења за веће инстанце, али за ове инстанце CPLEX проналази рјешење, које није доказано као оптимално у 7200 секунди. Гуроби је у овим случајевима

мање успјешан и проналази такво рјешење за само једну инстанцу.

VNS лако достиже оптимална рјешења за све BR инстанце, што се види из табеле 2.3. Такође, оптимално рјешење је достигнуто у свих 20 извршења, што показује високу поузданост VNS алгоритма. Времена извршења су прилично мала, до 216 секунди за највеће инстанце.

За прве двије DKTF инстанце, (rnd01 и rnd02 из Табеле 2.4), и GA и VNS достижу оптимално рјешење. За преостале DKTF инстанце, прописани VNS у свим случајевима достиже веома добре резултате, боље него GA. Као што се види из табеле 2.4, достигнути резултати су мањи од 10^{-3} (осим за три инстанце), док је за највеће инстанце, са 100 и више чворова и 300 и више грана, вриједност функције циља је мања од 10^{-4}. За двије инстанце, (rnd17 и rnd21), достигнути резултати су једнаки 0, што је засигурно оптимално рјешење, јер је вриједност функције циља ненегативна. Тако, приказани резултати у табелама 2.3 и 2.4 јасно указују да се VNS приступ може успјешно користити за инстанце великих димензија, у случајевима када егзактне методе не успијевају да пронађу оптимално рјешење. Упоређујући вријеме извршења двије хеуристике, очигледно је да GA брже проналази рјешења, док VNS достиже рјешења бољег квалитета.

2.7 Завршна разматрања за MBCP

Ова глава је посвећена проблему максимално балансиране повезане партиције у графу и примјени тог проблема у организовању курса. Ако се пође од претпоставке да курс треба подијелити на два, по тежини уједначена мања курса, уз очување релације повезаности, за тај задатак могуће је формирати одговарајући математички модел и њега ријешити неком методом комбинаторне оптимизације. Описан је метод одређивања повезаности, као и метод одређивања тежина лекција. На формирани повезан тежински граф примијењена је техника за рјешавање одговарајућег MBCP и предложено је оптимално рјешење подјеле курса из Теорије бројева.

У наставку Главе, приказан је теоријски резултат којим је формулисан модел мјешовитог цјелобројног програмирања, са пратећим доказом о коректности модела. Представљени модел користи полиномски број ограничења, што указује да се може користити и у теоријским и у практичним разматрањима.

Табела 2.4: Експериментални резултати за DKTF инстанце

Inst.	$\|V\|$	$\|E\|$	opt	t_{cplex}	t_{gur}	GA	t_{GA}	VNS	t_{VNS}
rnd01	20	30	1.14274	500.13	187.51	opt	0.52	opt	0.159
rnd02	20	50	0.01965	1899.18	480.89	opt	0.49	opt	0.198
rnd03	20	100	0	1307.85	36.10	0.00626	0.68	0.0008	0.281
rnd04	30	50	0	n.v:0.1817	5510.0	0.00915	0.58	0.00241	0.522
rnd05	30	70	0	5361.1	2081.6	0.01036	0.65	0.00148	0.505
rnd06	30	200	0	n.v:0.0108	403.56	0.00029	0.93	0.00007	1.036
rnd07	50	70	-	n.v:0.0328	n.v:0.4176	0.01608	0.92	0.00104	3.282
rnd08	50	100	0	n.v:0	5188.6	0.00531	1.10	0.00005	2.386
rnd09	50	400	0	n.v:0.0484	5112.0	0.00024	1.92	0.00008	5.864
rnd10	70	100	-	n.v:0.2083	N/A	0.01328	1.40	0.00022	7.678
rnd11	70	200	-	n.v:0	N/A	0.00023	1.737	0.00001	8.140
rnd12	70	600	-	n.v:0.0356	N/A	0.00232	2.910	0.00004	13.202
rnd13	100	150	-	n.v:0	N/A	0.00859	2.032	0.00037	23.473
rnd14	100	300	-	n.v:0	N/A	0.00137	2.355	0.00001	22.984
rnd15	100	800	-	n.v:0.7812	N/A	0.00155	3.812	0.00001	46.39
rnd16	200	300	-	n.v:0	N/A	0.00574	4.972	0.00006	203.31
rnd17	200	600	-	n.v:0	N/A	0.00058	5.255	0	206.34
rnd18	200	1500	-	n.v:0	N/A	0.00011	7.380	0.00001	411.47
rnd19	300	500	-	n.v:0	N/A	0.00039	7.747	0.00001	692.21
rnd20	300	1000	-	n.v:0	N/A	0.00025	7.560	0.00001	609.02
rnd21	300	2000	-	n.v:0	N/A	0.00008	11.61	0	1256.5

За рјешавање овог NP тешког проблема развијена је хеуристичка метода заснована на методи промјењивих околина. Погодно изабране околине су засноване на промјени компоненти за растући број чворова. Имплементирана је ефикасна и релативно брза локална претрага, која побољшава квалитет рјешења замјеном компоненти за парове чворова.

Да би се приказала употребна вриједност и поузданост развијених метода, спроведени су детаљни експерименти. MILP модел је покретан на два MILP рјешавача, CPLEX 12.1. и Gurobi 4.0. Добијени резултати указују да се на мањим инстанцама на овај начин могу добити оптимални резултати. Судећи по добијеним резултатима, VNS метод се показује да је веома успјешан. За мање инстанце проналази већину познатих оптималних рјешења, док за веће инстанце проналази веома добре резултате.

Истраживање презентовано у овој Глави може се проширити у неколико праваца. Развијене технике за одређивање тежине лекција засноване на објективним показатељима и субјективним процјенама професора могу се користити и у другим, сличним разматрањима. Предложена MILP формулација, као и прописани VNS алгоритам се у даљим истраживањима могу хибридизовати са другим егзактним и хеуристичким методама, како би се добили још бољи резултати. Прописани систем организације курса из Теорије бројева, по принципу подјеле курса на два по тежини уједначена дијела, може се примијенити на друге математичке или рачунарске курсеве.

Глава 3

Проблем проналажења максимално балансиране повезане партиције у графу са q партиција

3.1 Увод

У овом одјељку разматра се уопштење MBCP, такозваним проблемом проналажења максимално повезане партиције у графу са q партиција (енг. Balanced Connected Partition of graphs for q partitions - BCPq). За дати цио број q, гдје је $q \geq 2$, потребно је повезан граф са задатим тежинама на чворовима подијелити у q партиција $V_1, ..., V_q$, тако да је сваки индукован подграф повезан и да је распоред тежина по подграфовима што је могуће уједначенији. У [20] је дат приближни алгоритам са фактором 2 BCPq проблема за $q = 3$ и $q = 4$. У истом раду је доказано да је проблем NP тежак. Такође, разматран је и случај када q није фиксно, те је показано да у том случају проблем нема приближни алгоритам са фактором мањим од $6/5$, осим ако не важи да је $P = NP$.

Проблем сличан BCPq проблему је тзв. (l, u) партиционисање, гдје је циљ подијелити граф на компонентне, тако да укупна тежина у свакој компоненти буде најмање l, а највише u. Скорија истраживања овог проблема су приказана у [58, 59]. На примјер, у [58], аутори анализирају три проблема проналажења (l, u) партиције:

(a) проблем минималне (l, u) партиције, тј. проналажење (l, u) партиције са минималним бројем компоненти;

(б) проблем максималне (l, u) партиције, тј. проналажење (l, u) партиције са максималним бројем компоненти;

(ц) проблем проналажења (l, u) партиције са задатим бројем p компоненти.

Иако је проблем проналажења (l, u) партиција у општем случају NP тежак, за неке специјалне класе графова, проблем је рјешив у полиномском времену. На примјер, у [58], аутори су показали да и проблеми минималне и максималне партиције могу бити ријешени у $O(u^4 n)$ времену, а проблем проналажења p - партиције у $O(p^2 u^4 n)$ времену за било који серијско - паралелни граф са n чворова. Кратком анализом може се показати да се одређивање постојања (l, u) партиције може свести на рјешавање BCPq проблема: Ако је пронађено рјешење за BCPq проблем, тако да је вриједност функције циља за рјешење мање од δ, тада (l, u) партиција постоји за изабране $l = t\_sum/q - \delta$ и $u = t\_sum/q + \delta$, гдје је t_sum укупан збир свих тежина.

Примјер 3.1. *На лијевој страни слике 3.1 приказана је правоугаона мрежа са 25 чворова. Уз сваки чвор написана је тежина, док су имена чворова изостављена. Укупан збир свих тежина чворова у графу је 1259. Како су све тежине цијели бројеви, а укупна тежина није дјељива са три, оптимална вриједност функције циља за партицинисање графа на три повезана подграфа не може бити нула, већ у најбољем случају један. Ако чворове графа, са лијева на десно и одоздо на горе, обиљежимо бројевима од 1 до 25 (доњи лијеви чвор је чвор 1, десно до њега чвор 2 итд.), тада десној страни слике 3.1 одговара партиција*

$V_1 = \{1, 2, 6, 7, 8, 9, 11, 16, 21, 22\}$,

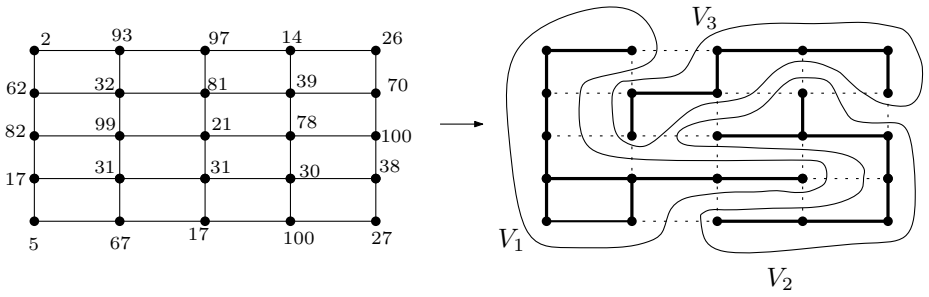
$V_2 = \{12, 17, 18, 20, 23, 24, 25\}$,

$V_3 = \{3, 4, 5, 10, 13, 14, 15, 19\}$.

Лако се може израчунати да је $w(V_1) = 420$, $w(V_2) = 419$ *и* $w(V_3) = 420$, *одакле закључујемо да је функција циља једнака 1, те је ова партиција уједно и оптимална.*

3.2 Ранији резултати

Као и MBCP, и BCPq припада широкој класи проблема који се односе на партиционисање графа и као такав има велику директну и индиректну примјену



Слика 3.1: Правоугаона мрежа димензије 05x05 (лијево) и партиција на три компоненте (десно)

у разним областима инжењерства, менаџмента, рјешавању неких социјалних проблема, као и проблема у образовању.

У обради слика, партиционисање се може искористити у ситуацијама када долази до погоршања квалитета слике усљед конвертовања из једног облика у други. У ситуацијама када нема информација о процесу деградације, једини начин да се побољша квалитет је да се повећа контраст и смање оштећења погодним измјенама на нивоима сиве боје. Да би се направио граф који одговара слици, одређује се скуп чворова који одговарају нивоима сиве боје, а тежине чворова се дефинишу као број појављивања одговарајуће нијансе у слици. Проналажење оптималне трансформације сивих нивоа се формулише као партиционисање одговарајућег графа, тако да збирови тежина чворова по свакој компоненти буду што уједначенији [77].

Приликом одређивања политичких подручја - изборних јединица, циљ је подијелити читаву област (мапу) на неколико региона са скоро подједнаким бројем гласача. Ако је G одговарајући дуални граф мапе M, сваки чвор v од G представља један регион, а вриједност $w(v)$ представља број гласача у датом региону v. Два чвора у графу су сусједна ако су сусједни одговарајући региони у M. Оптимизациони проблем се своди на проналажење партиције графа (дијељење читаве мапе на подобласти), тако да одговарајуће подобласти имају скоро подједнак број гласача, а региони унутар сваке подобласти су повезани. У [11] је за рјешавање оваквог проблема приказана хеуристика заснована на табу претраживању.

Рјешавање BCPq проблема се може успјешно примијенити и за унапређење означавања области полицијских патрола [6]. Најприје се за сваки дио неког града (или ширег региона) идентификује ниво криминала. Као циљ се задаје

подјела града на подобласти, тако да полицијске патроле дјелују у датим подобластима, а да ниво криминала по подобластима буде уједначен. Метод описан у [6] се састоји од двије фазе: иницијалног партиционисања и додатног побољшања. У првој фази се граф подијели на два подграфа, док се у другој фази, у циљу побољшања крајњег резултата, хеуристички врши размјена чворова између партиција.

3.3 Метода промјенљивих околина за рјешавање BCPq проблема

Сазнања добијена приликом конструисања методе промјенљивих околина за MBCP су у одређеној мјери искоришћена и за развој сличне методе за рјешавање општег случаја. Са друге стране, с обзиром на сложеније кодирање рјешења и сложенију функцију циља, није могуће успоставити потпуну аналогију са специјалним случајем. У овом одјељку презентован је VNS за рјешавање BCPq проблема.

3.3.1 Иницијализација и функција циља

Нека је дат полазни граф $G = (V, E)$ и нека је задат број q. Рјешење датог проблема се представља низом x дужине $|V|$, гдје елементи низа одговарају чворовима графа. Елементи низа x узимају вриједности из скупа $\{0, 1, 2, ..., q-1\}$ и указују на то којој компоненти дати чвор припада, тј. чвор i припада компоненти V_j ако $x_i = j - 1$. У полазном рјешењу се за сваки чвор на случајан начин бира припадајућа компонента, што значи да се за сваки чвор одговарајући елемент низа на случајан начин, са подједнаком вјероватноћом, бира из интервала $\{0, 1, 2, ..., q-1\}$.

Након креирања полазног рјешења, а и за вријеме процеса претраживања, очигледно је да оваква репрезентација може произвести недопустиво рјешење, јер подграфови индуковани чворовима унутар компоненти могу бити неповезани. Као и у случају за $q = 2$, и овдје се показује корисним увођење казнене функције. Експерименти показују да и у овом случају приликом рачунања казнене функције треба укључити у обзир број компоненти повезаности за сваки појединачни подграф, те тај број множити са максималном тежином чвора у графу. За

рачунање броја компоненти повезаности за сваки подграф користи се алгоритам претраге у ширину (BFS). Ако број компоненти повезаности у подграфу G_j означимо са $nc(G_j)$, тада се вриједност казнене функције f_{pen} рачуна помоћу формуле

$$f_{pen} = \sum_{j=1}^{q} (nc(G_j) - 1) * w_{max}, \tag{3.1}$$

гдје је w_{max} максимална тежина чвора у графу. Да би се израчунала вриједност фунције циља, прво је потребно одредити компоненте са најмањом, односно највећом тежином, односно њихове одговарајуће тежине:

$$mins = \min\{w(G_j) : j \in \{1, 2, ..., q\}\}, \tag{3.2}$$

и

$$maxs = \max\{w(G_j) : j \in \{1, 2, ..., q\}\}. \tag{3.3}$$

Вриједност функције циља se тада рачуна помоћу формуле

$$obj(V_1, V_2, ..., V_q) = maxs - mins + f_{pen}. \tag{3.4}$$

Из (3.1), јасно је да ако су сви подграфови повезани (број компоненти повезаности за сваки подграф је једнак 1), тада казнена функција има вриједност 0. У том случају, казнена функција нема утицај на рачунање вриједности функције циља (3.4). У случају појаве неповезаних компоненти, казнена функција значајно увећава вриједност функције циља, што процесу оптимизације омогућава да таква рјешења замијени допустивим.

3.3.2 Околине и процедура размрдавања

У процедури размрдавања се формира систем околина и у сваком кораку се из тренутне околине креира ново рјешење x', $(x' \in N_k(x))$ засновано на тренутно најбољем рјешењу x. Да би се формирала k-та околина, на случајан начин се бира неких k чворова V. За сваки изабрани чвор, мијења се припадајућа компонента, тако што се она, између свих осталих компоненти, бира на случајан начин: ако је изабрани чвор прије примјене размрдавања припадао компоненти V_j, тада се из скупа $\{\{1, 2, ..., q\} \backslash j\}$ на случајан начин бира број l, те се дати

чвор убацује у компоненту V_l.

3.3.3 Локално претраживање

Након одређивања рјешења x' унутар процедуре размрдавања, позива се процедура локалног претраживања.

У свакој итерацији локалне претраге, алгоритам размјењује компоненте за два чвора. Ако у рјешењу x' два чвора u и v припадају компонентама V_j и V_l респективно, након замјене, статус је $u \in V_l$ и $v \in V_j$. Нека је оваквом размјеном формирано ново рјешење x''. У случају да је x'' боље од x', тада x' постаје једнако x'', (у другом случају, x' се не мијења) и у локалном претраживању се наставља са замјеном сљедећег пара чворова.

Локално претраживање се завршава када се рјешење не може више побољшати. Након завршетка локалне претраге, разликујемо три могућа случаја:

(а) Ако је вриједност функције циља за рјешење x' строго веће од вриједности рјешења x , тада тренутно најбоље рјешење остаје x , а претрага се наставља са сљедећом околином.

(б) Ако је вриједност функције циља за рјешење x' мање него за x, тада x' постаје тренутно најбоље рјешење ($x = x'$) и претрага се наставља са истом околином.

(ц) Ако су вриједности функција циља за ова два рјешења иста, онда се са вјероватноћом p_{move} поставља $x = x'$ и алгоритам наставља претрагу са истом околином. У другом случају, претрага се са вјероватноћом $1 - p_{move}$ наставља са истим рјешењем x и наредном околином.

Пошто су разматране све околине, алгоритам наставља претраживање са првом околином, све док не буде испуњен критеријум за завршетак алгоритма. У нашем случају, критеријум је достигнут максималан број итерација.

3.4 Експериментални резултати

Експерименти приказани у овом одјељку садрже резултате добијене примјеном VNS методе на исте групе инстанци као у случају MBCP: правоугаоне мреже

Табела 3.1: Приказ резултата за BR инстанце

		$q=3$			$q=4$			$q=5$		
|V|	|E|	Min	Sr	$T_{tot}[s]$	Min	Sr	$T_{tot}[s]$	Min	Sr	$T_{tot}[s]$
25	40	1	2.35	0.58	4	4.6	0.81	8	11.8	0.94
25	40	4	7.9	0.60	23	23	0.77	71	82.9	0.99
30	49	0	0.6	1.11	4	6.9	1.49	8	11.3	1.80
30	49	5	8.2	1.33	9	13.45	1.52	18	39.1	1.93
50	85	0	0	6.49	1	2.35	9.60	3	6.55	11.43
50	85	1	1.6	6.49	2	9.5	10.30	8	25.15	12.37
100	175	0	55.1	57.33	1	81.1	105.72	1	58.4	139.15
100	175	0	199.3	67.60	4	452.25	119.09	3	312.25	153.24
49	84	1	1	6.64	0	2.1	10.18	2	5.85	12.15
49	84	0	1.55	7.21	3	10.6	10.64	11	28.45	13.19
70	123	1	1	19.29	1	6.25	33.33	1	3.3	44.78
70	123	1	1.25	20.87	3	5.7	35.79	5	19.5	47.27
100	180	1	1	61.67	1	20.8	109.19	2	17.05	158.38
100	180	0	49.6	72.08	1	27.6	146.38	3	58.65	182.40
225	420	1	81	611.94	1	131	1176.99	101	266.05	2005.51
225	420	0	612.5	729.46	0	931.6	1537.90	10	1153.4	2435.67

(BR инстанце), уведене у [10] и случајно генерисане графове (DKTF инстанце) из [33]. Сви тестови су обављени на рачунару Intel Core 2 Quad Q9400 @2.66 GHz са 8 GB RAM. VNS имплементација је кодирана у програмском језику C и за сваку тестну инстанцу алгоритам је покретан 20 пута. За сваку инстанцу вриједност k_{min} је подешена на 2. Да би био испуњен теоријски услов $(\forall k)(k_{min} \leq k \leq k_{max}-1)\ |N_k(x)| \leq |N_{k+1}(x)|$, за инстанце са мање од 60 чворова, k_{max} је подешено на $n/2$, а за веће инстанце на 30. Параметар p_{move} има вриједност 0.4, а алгоритам завршава са радом након 500 итерација.

У табелама 3.1 и 3.2 приказани су постигнути резултати описаног VNS метода на наведеним инстанцама, за случајеве $q=3$, $q=4$ и $q=5$. У прве двије колоне приказани су број чворова и грана у графу. Након ових колона, у обје табеле, респективно за вриједности $q=3$, $q=4$ и $q=5$ приказане су редом, најбоља и просјечна вриједност, те укупно просјечно вријеме рачунања алгоритма у секундама.

Као што се види из табела 3.1 и 3.2, предложена метода промјењивих околина постиже квалитетне резултате за већину инстанци. Са друге стране, за неке инстанце, као на примјер, за BR инстанце са 100 чворова и 175 грана, односно са 225 грана, примјећује се да средња вриједност знатно одступа од

Табела 3.2: Приказ резултата за случајне графове

		$q=3$			$q=4$			$q=5$		
|V|	|E|	Min	Sr	$T_{tot}[s]$	Min	Sr	$T_{tot}[s]$	Min	Sr	$T_{tot}[s]$
20	30	12.7831	13.468	0.26	62.873	72.066	0.32	110.862	115.219	0.34
20	50	0.0922	0.9774	0.44	6.491	8.976	0.52	27.791	29.263	0.54
20	100	0.0635	0.3619	0.64	0.900	2.104	0.91	1.821	3.413	1.02
30	50	1.0625	2.2651	0.96	4.128	10.715	1.31	25.089	43.319	1.51
30	70	0.0873	0.4659	1.18	1.065	4.124	1.69	8.273	15.453	2.08
30	200	0.0084	0.1124	1.98	0.156	0.572	2.71	0.558	1.484	3.36
50	70	0.2054	5.0229	5.89	3.813	28.353	7.56	30.481	76.349	8.46
50	100	0.0806	0.3347	6.23	1.297	3.211	9.43	6.029	11.346	10.96
50	400	0.0121	0.0518	11.03	0.088	0.269	15.76	0.195	0.518	20.47
70	100	0.1665	2.5334	17.98	5.479	19.442	23.26	19.686	51.471	27.49
70	200	0.0075	0.0469	18.92	0.257	0.505	30.67	0.735	1.391	41.93
70	600	0.0113	0.0322	29.0	0.085	0.116	44.89	0.106	0.240	59.14
100	150	0.1296	0.4784	63.7	1.507	4.726	96.0	3.701	14.919	113.3
100	300	0.0024	0.0295	47.3	0.062	0.211	71.2	0.508	0.774	109.7
100	800	0.0023	0.0084	89.0	0.013	0.046	120.8	0.064	0.142	151.8
200	300	0.0196	0.1170	490.7	0.389	1.131	826.2	1.647	5.112	1141.8
200	600	0.0022	0.0077	427.2	0.011	0.044	703.8	0.037	0.187	958.9
200	1500	0.0002	0.0015	794.3	0.001	0.008	1065.3	0.018	0.029	1259.1
300	500	0.0074	0.0313	1528.2	0.086	0.234	2752.8	0.331	0.853	3562.5
300	1000	0.0005	0.0026	1374.6	0.009	0.018	2246.1	0.012	0.056	2944.9
300	2000	0.00001	0.0007	2309.6	0.002	0.004	2888.6	0.007	0.015	3537.7

минималне, што указује на чињеницу да алгоритам у неким извршењима не може да избјегне "заглављивање" у локалним субоптималним рјешењима. Ова појава се у мањој мјери јавља и код DKTF инстанци са 50 чворова и 70 грана. Вријеме извршења алгоритма за највеће инстанце не прелази 3600 секунди. Како видимо из колона $T_{tot}[s]$ из обје табеле 3.1 и 3.2, вријеме извршења алгоритма за сваку инстанцу сразмјерно расте са бројем тражених компоненти.

3.5 Завршна разматрања за BCPq проблем

Метода промјенљивих околина, иницијално развијана за партиционисање повезаног графа на двије балансиране повезане партиције, успјешно је унапријеђена и за рјешавање општег случаја, партиционисања графа на произвољан број партиција. Показало се да се неки концепти специјалног случаја могу искористити и у општем (као што је употреба казнене функције или локално претраживање), док се други елементи алгоритма, као што су функција циља или околине, разликују у одређеној мјери. Метода промјенљивих околина, дакле, у оквиру процедуре размрдавања формира околине по принципу промјене припадајућих компоненти за растући број чворова, гдје се избор нове компоненте бира на случајан начин. У локалном претраживању рјешење се покушава унаприједити размјеном компоненти за парове чворова.

Експериментални резултати су вршени на истим инстанцама као и у случају

партиције на двије компоненте. Добијени резултати указују на релативно добар квалитет предложеног алгоритма. Експерименти показују да вријеме извршења алгоритма сразмјерно расте са повећањем задатог броја компоненти.

Релативно дуго вријеме извршења алгоритма за већи задати број партиција указује на правац даљег истраживања, који би подразумијевао убрзање локалног претраживања употребом неке друге стратегије, као и евентуално формирање другачијег система околина. С обзиром да овај проблем до сада није интензивно рјешаван другим метахеуристичким методама, било би интересантно испитати какве су могућности примјене других метахеуристика, као и њихове хибридизације.

Глава 4

Проблем p - арне транзитивне редукције у графу

4.1 Увод

У овој глави анализира се проблем p - арне транзитивне редукције у графу, гдје је p позитиван цио број, (енг. p - ary transitive reduction - TR_p problem) и предлажу се генетски алгоритам и метода промјенљивих алгоритама за рјешавање TR_p проблема. Резултати приказани у овој глави објављени су у [84]. TR_2 укључује проблем проналажења усмјерене Хамилтонове контуре у графу, те је стога, као уопштење NP - тешког проблема [40] и сам NP тежак. Специјалан случај TR_p проблема, када је $p = 1$, је познати проблем проналажења минималног еквивалентног диграфа (енг. minimum equivalent digraph - MED problem).

MED и TR_p су проблеми од практичног интереса. MED проблем се јавља у дизајнирању рачунарских мрежа које задовољавају услове повезаности [62]. Основни проблем дизајнирања мрежа у контексту усмјерених графова је сљедећи: за дати диграф, пронаћи најмањи скуп грана (које формирају минималан еквивалентан диграф), који садржи све релације "достижности" (енг. reachability), засноване на релацији "достижности" у полазном графу.

TR_1 се може примијенити у анализи и визуелизацији социјалних мрежа. Да би се испитале социјалне мреже и помогло процесу визуализације, у [34] је за рјешавање TR_1 проблема развијен праволинијски "похлепни" алгоритам, који је примијењен на добро познату широку социјалну мрежу познату као Енрон

(енг. Enron corpus).

Значајну примјену проблема TR_2 срећемо у биологији. TR_2 се користи за откривање начина преноса сигнала у биолошким мрежама. Већина биолошких карактеристика потиче од сложеног међудјеловања разних састојака ћелије, као што су ДНК, РНК, протеини и мали молекули. Ћелије користе путеве и механизме регулације да би координисале вишеструке функције, омогућавајући им да одговоре и прилагоде се на промјену окружења. У експерименталним методама за испитивање генома (енг. Genome-wide methods) се идентификују на хиљаде ћелијских компоненти. Експерименталне информације о међудјеловању у датој мрежи компоненти могу бити директне, тј. добијене посматрањем одређених интеракција између молекула у неком посебном експерименту, или индиректне, тј. оне које се добијају када утичемо на неку компоненту у мрежи и посматрамо њен утицај на друге компоненте. Да би се све ове информације синтетизовале у једну мрежу, мора се одредити како се различите интеракције добијене експериментима међусобно уклапају. TR_2, или бинарна транзитивна редукција (енг. binary transitive reduction - BTR), омогућава да се одреди најрјеђи граф (онај граф који има најмање грана) конзистентан на експериментална запажања. Овај приступ је кључни дио процедуре синтезе мреже, описан у [3].

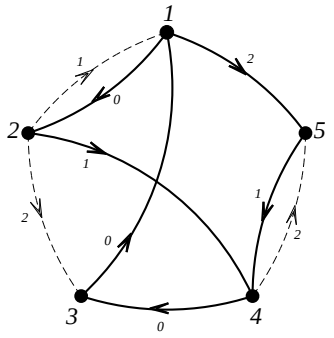
4.2 Математички модел

TR_p је прецизно дефинисан у [4]. За дати усмјерен граф $G = (V, E)$ са датом функцијом која дефинише лабеле на гранама $\omega : E \mapsto \{0, 1, 2, \ldots, p-1\}$ за неки фиксиран цио број $p > 0$, уводимо сљедеће дефиниције и нотацију:

- Сви путеви (могући су и самопресијецајући) су усмјерени путеви.
- "Паритет" (енг. parity) пута P из чвора u до чвора v је $\sum_{e \in P} \omega(e) \pmod p$.
- Нотација $u \overset{x}{\Rightarrow} v$ означава пут од u до v паритета $x \in \{0, 1, 2, \ldots, p-1\}$. Ако не наглашавамо паритет, тада пут једноставно означавамо као $u \Rightarrow v$. Грана се једноставно означава са $u \overset{x}{\rightarrow} v$ или $u \rightarrow v$.
- За подскуп скупа грана $E' \subseteq E$, скуп $\mathrm{dostupni}(E')$ је скуп свих уређених тројки (u, v, x) таквих да важи $u \overset{x}{\Rightarrow} v$ је пут у (под)графу (V, E').

Табела 4.1: Гране и одговарајуће лабеле

почетак(e)	крај(e)	w(e)
1	2	0
2	3	2
1	5	2
2	1	1
2	4	1
3	1	0
4	3	0
4	5	2
5	4	1



Слика 4.1: Оптимално рјешење за примјер 4.1.

Дефиниција проблема p-арне транзитивне редукције (TR_p) је сљедећа. Нека је $G = (V, E)$ усмјерен граф са датом функцијом $\omega : E \mapsto \{0, 1, 2, \ldots, p-1\}$ и датим скупом критичних грана $E_{kriticne} \subseteq E$. Задатак је пронаћи подграф $G' = (V, E')$ гдје је $E_{kriticne} \subseteq E' \subseteq E$, dostupni($E'$)=dostupni($E$) и $|E'|$ је што је могуће мањи скуп.

Примјер 4.1. *Нека је $p = 3$, $V = \{1, 2, 3, 4, 5\}$, $|E| = 9$ и $E_{kriticne} = \emptyset$. Гране и одговарајуће лабеле на гранама су дате у табели 4.1.*

Оптимално рјешење, добијено тоталном енумерацијом, је 6 и одговарајућа транзитивна редукција се састоји од грана (1,2), (1,5), (2,4), (3,1), (4,3) и (5,4). Полазни граф G и његова минимална транзитивна редукција је приказана на слици 4.1. Гране графа G, које нису укључене у оптимално рјешење, су приказане испрекиданим линијама. У графу G скуп dostupni(E) се састоји од

свих могућих тројки, тј. $dostupni(E)=\{(u,v,x)|\ (u,v)\in E, x\in\{0,1,\ldots,p-1\}\}$. *Провјеримо да све такве тројке такође припадају и скупу* (E').
$1 \overset{0}{\Rightarrow} 2$*:* $1 \overset{0}{\to} 2$.
$1 \overset{1}{\Rightarrow} 2$*:* $1 \overset{0}{\to} 2$, $2 \overset{1}{\to} 4$, $4 \overset{0}{\to} 3$, $3 \overset{0}{\to} 1$, $1 \overset{0}{\to} 2$.
Примијетимо да постоји циклус: $1 \overset{0}{\to} 2$, $2 \overset{1}{\to} 4$, $4 \overset{0}{\to} 3$, $3 \overset{0}{\to} 1$, *т.ј.* $1 \overset{1}{\Rightarrow} 1$ *Стога, ако за неки чвор* $v \neq 1$, *постоји* $1 \Rightarrow v$, *тада постоји* $1 \overset{x}{\Rightarrow} v$, *за сваки* $x \in \{0,1,\ldots,p-1\}$ *јер се по уоченом циклусу можемо кретати произвољан број пута, заправо можемо се кретати онолико пута колико је потребно.*

Тако, за сваки $x \in \{0,1,\ldots,p-1\}$, $1 \overset{x}{\Rightarrow} 3$ *постоји у* G' *јер постоји пут* $1 \Rightarrow 3$*:* $1 \to 5$, $5 \to 4$, $4 \to 3$.

Слично за $1 \Rightarrow 4$*:* $1 \to 5$, $5 \to 4$ *и* $1 \Rightarrow 5$*:* $1 \to 5$.

Сада, пошто смо потврдили да у графу G' *постоји пут од чвора* 1 *до било ког чвора, а из истога разлога и контура* $1 \overset{1}{\Rightarrow} 1$, *ако пронађемо пут (произвољног паритета)* $v \Rightarrow 1$, *за чвор* $v \neq 1$, *тада све тројке* (v,w,x), *гдје* $w \in V\backslash\{v\}$ *и* $x \in \{0,1,\ldots,p-1\}$ *припадају скупу* $dostupni(E')$. *Путеви* $2 \Rightarrow 1$, $3 \Rightarrow 1$, $4 \Rightarrow 1$, *и* $5 \Rightarrow 1$ *постоје у* G', *одакле слиједи да је* E' *транзитивна редукција полазног графа* G.

4.2.1 Ранији резултати

У [88], примијећено је да се било које рјешење MED проблема декомпонује у рјешења за строго повезане компоненте и рјешење за граф компоненти (граф добијен контракцијом сваке повезане компоненте). Тако се проблем редукује у линеарном времену за два случаја: граф је или ацикличан или строго повезан. Ако је граф ацикличан, MED проблем је еквивалентан проблему транзитивне редукције, за који је у [2] показано да је еквивалентан транзитивном затворењу, те је стога рјешив у полиномском времену. Проблем транзитивне редукције се састоји од проналажења подграфа датог графа G са истим транзитивним затворењем као и G, таквог да у себи не садржи прави пограф са истом особином.

У [97], је представљен алгоритам линеарне сложености за пронажење минималне транзитивне редукције за дати строго повезан дограф. У [61] се наводи да дати алгоритам није коректан.

У [63] је доказано да је у строго повезаним графовима, који немају усмјерених контура које садрже више од три гране, MED проблем еквивалентан

максималном бипартитном спаривању. Користећи ову чињеницу, аутори су унаприједили ранију апроксимацију за MED из [62] и приказали нову, са фактором 1.617.

У [95] је имплементиран и тестиран алгоритам симулираног каљења (SA) за рјешавање MED проблема. У сваком кораку, SA хеуристика разматра неку околину тренутног стања и пробабилистички одлучује да ли да систем пребаци у ту нову околину, или остане у тренутном стању. Вјероватноће се бирају тако да систем тежи ка преласку у стање са мањом енергијом. Могућношћу преласка и у стања са вишом енергијом се смањује нежељена појава заглављивања у локалним, субоптималним рјешењима.

У [8] је формално развијен генерички програм који одређује минималан подскуп који задовољава неке особине датог скупа, комбинујући неке инваријантне технике са теоријским и логичким рачунањима. Овај генерички приступ је даље примијењен на проналажење рјешења за неке познате графовске проблеме, а између осталих и на проналажење транзитивне редукције строго повезаног графа.

У литератури је описано и неколико приближних алгоритама за MED проблем. MED проблем је такође познат и као MAX-SNP-тежак проблем који је рјешив у полиномском времену са фактором $1.645 + \epsilon$ за сваку константу $\epsilon > 0$ [62]. Најновија апроксимација са фактором 1.5 дата је у [100]. За тежинску варијанту MED проблема, у којој свака грана има ненегативну реалну тежину, пронађен је приближни алгоритам са фактором 2 [38, 64]. С обзиром на повезаност MED и TR_1 проблема, слиједи да и за проблем TR_1 постоји приближни алгоритам са фактором 2.

Дефиниција TR_p проблема је уведена у [4]. Доказано је да је за усмјерене ацикличне графове проблем рјешив у линеарном времену за било који цио број $p > 0$, и приказан је приближни алгоритам за TR_1 са фактором 1.78, док је за графове уопште постигнута апроксимација са фактором $2 + o(1)$ за TR_p гдје је $p > 1$ прост број.

У [3] је уведен нови метод који комбинује синтезу и сазнања о мрежама биолошких сигнала. Главна идеја лежи у представљању посматраних индиректних условних веза као путева у мрежи и употреби техника комбинаторне оптимизације да се пронађе најрјеђи граф конзистентан са свим експерименталим

посматрањима. Рачунарско најзахтјевнији корак читавог приступа је BTR проблем. Аутори су доказали да се BTR рјешава у полиномском времену ако граф нема контура дужине веће од 3. Даље, описан је један "похлепни" приступ за рјешавање BTR проблема, који брише непотребне гране све док оне постоје и показано је да овај приступ води апроксимацији са фактором 3.

На основу ових резултата, у [60] је развијен софтвер који комбинује синтезу, међудјеловање и поједностављење мреже преносних сигнала (енг. signal transduction networks), [3]. Биолошка употребљивост софтвера илустрована је примјеном на претходно објављену мрежу преносних сигнала [73], те на његову употребу за истраживање смртности ћелија у неким обољењима.

У [9] је доказано постојање апроксимативног алгоритма са фактором 1.5 за MED, TR_1, и TR_p проблеме, гдје је p фиксан прост број.

4.2.2 Веза између TR_p и других проблема

У овом одјељку се описује веза између TR_p и других проблема, поменутих на почетку уводне секције. Специјалан случај BTR проблема, када све лабеле на гранама имају вриједност нула, укључује проблем проналажења усмјерене Хамилтонове контуре у графу. За дати усмјерен граф $G = (V, E)$, MED проблем је пронажење скупа $E' \subset E$ таквих да за свака два чвора u и v важи: ако постоји пут из u у v у E, тада постоји пут u у v у E'. Ако је $E_{kriticne} = \emptyset$, тада је TR_1 управо MED проблем.

У Додатку 2 чланка [3] приказана је формулација мјешовитог цјелобројног програмирања за проблем BTR (TR_2). Идеја је заснована на мрежним токовима између чворова.

Прво, за конструкцију новог графа $G_1 = (V_1, E_1)$ од полазног графа $G = (V, E)$, уз очување релације достижности и док се симултано елиминише потреба за лабелама, користи се сљедећа процедура. Свака грана $e = u \xrightarrow{1} v \in E$ се укључује у E_1 и ако $e \in E_{kriticne}$, онда се грана e означава и као критична у G_1. За сваку грану $e = u \xrightarrow{0} v \in E$, укључује се нови чвор w у G_1, у E_1 се укључују гране $e_1 = u \xrightarrow{1} w$ и $e_2 = w \xrightarrow{1} v$ и ако $e \in E_{kriticne}$, тада се обје гране e_1 и e_2 означавају као критичне у E_1.

Свака грана у G_1 има исту лабелу и стога можемо занемарити лабеле у G_1. Да би се одредила бинарна транзитивна редукција у G, мора се одредити

бинарна транзитивна редукција у G_1, те се ти резултати преносе назад у G. Благо нарушавајући нотацију, ознака $E_{kriticne}$ се користи за означавање и скупа критичних грана у G_1.

MILP модел заснован на мрежном току за BTR проблем примијењеном на G_1 уводи сљедеће промјенљиве:

- За сваку грану $e \in E$ промјенљива $x_e \in \{0, 1\}$ има вриједност 1, ако и само ако је грана e присутна у транзитивној редукцији од G_1,
- За све $u, v \in V, e \in E_1$, промјенљиве за ток $f^{even}_{u,v,e}$ и $f^{odd}_{u,v,e}$ узимају ненегативне реалне вриједности и називају се парни и непарни токови, респективно.

За дати проблем $G = (V, E)$, простор рјешења има $|V|^2 \cdot |E_1| + |E_1|$ промјенљивих, од којих је $|E_1|$ бинарних промјенљивих, које узимају вриједност из $\{0, 1\}$, док су преосталих $|V|^2 \cdot |E_1|$ реалне ненегативне промјенљиве. Уведимо још и додатну нотацију: ulazece(x) и izlazece(x) се односе на скупове $\{u | u \to x \in E_1\}$ и $\{u | x \to u \in E_1\}$, респективно. MILP модел за проблем бинарне транзитивне редукције је сљедећи:

$$min \sum_{e \in E_1} x_e, \tag{4.1}$$

уз ограничења:

$$x_e = 1, \qquad \text{за све } e \in E_1 \cap E_{kriticne} \tag{4.2}$$

$$\sum_{x \in \text{izlazece}(u)} f^{\text{even}}_{u,v,x} = 1, \qquad \text{за све } e = u \to v \in E_1 \tag{4.3}$$

$$\sum_{x \in \text{ulazece}(v)} f^{\text{even}}_{u,v,x} - f^{\text{odd}}_{u,v,x} = -1, \qquad \text{за све } e = u \to v \in E_1 \text{ и } w(e) = 0 \tag{4.4}$$

$$\sum_{x \in \text{ulazece}(v)} f^{\text{even}}_{u,v,x} - f^{\text{odd}}_{u,v,x} = 1,$$

$$\text{за све } e = u \to v \in E_1 \text{ и } w(e) = 1 \quad (4.5)$$

$$\sum_{y \in \text{ulazece}(x)} f^{\text{even}}_{u,v,y} \quad - \sum_{y \in \text{izlazece}(x)} f^{\text{odd}}_{u,v,y} = 0,$$

$$\text{за све } e = u \to v \in E_1 \text{ и све } x \in V_1 \backslash \{u, v\} \quad (4.6)$$

$$\sum_{y \in \text{ulazece}(x)} f^{\text{odd}}_{u,v,y} \quad - \sum_{y \in \text{izlazece}(x)} f^{\text{even}}_{u,v,y} = 0,$$

$$\text{за све } e = u \to v \in E_1 \text{ и све } x \in V_1 \backslash \{u, v\} \quad (4.7)$$

$$\sum f^{\text{even}}_{u,v,e} + f^{\text{odd}}_{u,v,e} \leq x_e, \qquad \text{за све } e \in E_1, u, v \in V. \quad (4.8)$$

Функција циља (4.1) обезбјеђује да ће рјешење бити минималан подграф који задовољава ограничења. Први скуп ограничења (4.2) обезбјеђује да ће рјешење садржавати све критичне гране.

Ограничења (4.3)-(4.8) се користе да би се осигурало да рјешење задовољава све релације достижности, које су присутне и у оригиналном графу.

У BTR проблему, постоје двије врсте паритета: парна и непарна. За сваку грану у графу, ограничења која се односе на ток у MILP моделу осигуравају постојање мрежног тока у рјешењу (парног или непарног тока, у зависности од тога да ли је грана у оригиналном графу означена са 0 или 1). Свака промјенљива за ток се везује за грану и односи се за ток и паритет у полазном проблему. Први скуп ограничења за ток (4.3) осигурава да ће постојати неки позитиван ток који полази из извора мрежног тока. Ограничења (4.4) и (4.5) осигуравају да ће ток улити у завршни чвор (ушће). Ограничења (4.6) и (4.7) осигуравају "услов конзервације". Посљедњи скуп ограничења (4.8) осигурава да ако се грана користи у току, тада је она и изабрана.

4.3 Генетски алгоритам за рјешавање TR_p проблема

У овој секцији се описује генетски алгоритам за рјешавање проблема TR_p. Резултати презентовани у овој секцији су објављени у раду [84].

4.3.1 Репрезентација јединки и креирање иницијалне популације

За репрезентацију јединки користи се бинарно кодирање. Свако рјешење се представља бинарним низом дужине $|E \backslash E_{kriticne}|$. Пошто критичне гране увијек припадају рјешењу, оне се не кодирају. Сваки бит у генетском коду одговара једној грани из скупа $E \backslash E_{kriticne}$. У случају да је грана укључена у рјешење, одговарајући бит има вриједност 1, док је у противном вриједност бита једнака нули.

Приликом иницијализације, најприје се користи *Floyd-Warshall*-ов алгоритам за рачунање свих уређених тројки (u, v, x) таквих да у датом графу G постоји пут од чвора u до чвора v, паритета x, тј. $u \overset{x}{\Rightarrow} v$. Иницијална популација од $N_{pop} = 150$ јединки се генерише на случајан начин, чиме се обезбјеђује максимална разноврсност генетског материјала. Међутим, случајно генерисане јединке могу бити некоректне. Додатно, пошто алгоритам користи стандардне генетске операторе (како ће то бити касније детаљније објашњено), примјеном тих оператора такође могу настати недопустива рјешења, што је заправо чест случај. Некоректне јединке се "поправљају" приликом рачунања функције циља, како је то објашњено у одјељку који слиједи.

4.3.2 Рачунање функције циља

Нека $E' \subseteq E$ означава скуп грана које су у датој јединки укључене у рјешење (те гране су у генетском коду кодиране јединицом). Одговарајући граф је тада $G' = (V, E')$. Користимо *Floyd-Warshall*-ов алгоритам за рачунање свих уређених тројки (u, v, x), таквих да у графу G' постоји $u \overset{x}{\Rightarrow} v$.

Сљедећи корак је одређивање тројки које постоје у G, али не постоје у G'. Ако нема таквих тројки, тада је скуп E' једна транзитивна редукција полазног графа G, тако да је дата јединка допустива. У противном, алгоритам проналази ону грану из E која већ није у E', чијим би се убацивањем укључио највећи број тројки. Тада у E' укључимо ту пронађену грану. Процес укључивања гране по гране у скуп E' се наставља по описаном принципу, све док E' не постане транзитивна редукција од G.

Вриједност функције циља је $|E'|$.

У најгорем случају, у рјешење је потребно је укључити свих $|E \backslash E_{kriticne}|$ грана. За сваку такву грану, на новоформирани граф G', потребно је покренути *Floyd-Warshall*-ову процедуру сложености $O(p^2 \cdot |V|^3)$. Такође, да би се пронашла она грана која највише одговара, у најгорем случају алгоритам мора проћи кроз

све кодиране гране, све тројке и све лабеле, што имплицира да је временска сложеност $O(p^2 \cdot |V|^2 \cdot |E|)$. Стога је укупна временска сложеност за рачунање функције циља $O(\max\{p^2 \cdot |V|^3, p^2 \cdot |V|^2 \cdot |E|\})$.

Нека max_{obj} означава максимум, а min_{obj} минимум функције циља, посматрано по свим јединкама у популацији. Прилагођеност јединки (fit_{jed}) у популацији се одређује као линеарна нормализација вриједности функција циља (obj_{jed}) у интервалу [0,1], рачуната преко формуле 4.9:

$$fit_{jed} = \frac{max_{obj} - obj_{jed}}{max_{obj} - min_{obj}}. \tag{4.9}$$

Стога је јединка са највећом прилагођеношћу она са минималном вриједношћу функције циља, и обрнуто, јединка са највећом вриједношћу функције циља има најмању прилагођеност.

4.3.3 Генетски оператори

Селекција

Приликом примјене оператора селекције, $N_{elite} = 50$ елитних јединки директно пролази у сљедећу генерацију и на њих се генетски оператори не примјењују. Посљедично, за те јединке се не мијења ни вриједност функције циља, па се самим тим она не мора ни рачунати. Тиме се у значајној мјери побољшавају укупне перформансе читавог алгоритма.

Оператор селекције бира јединке које ће учествовати у процесу рекомбинације и дати потомке. Овај процес фаворизује јединке са бољом прилагођеношћу. У овој имплементацији, користи се фино градирана турнирска селекција (енг. fine grained tournament selection operator - FGTS), представљена у [37]. Како је то већ описано у одјељку 1.2.3 уводне главе, FGTS оператор је унапређење класичне турнирске селекције који је заснован на увођењу рационалног параметра F_{tour}, који представља пожељну просјечну величину турнира. У овој имплементацији, вриједност параметра $F_{tour} = 5.4$.

Укрштање

Из скупа неелитних јединки, бира се $\lfloor (N_{pop} - N_{elite})/2 \rfloor$ парова јединки на случајан начин. На сваки пар се са вјероватноћом $p_{cross} = 0.85$ примјењује оператор укрштања. У прописаном алгоритму се користи стандардно једнопозиционо укрштање. Тачка укрштања се бира на случајан начин, а укрштање функционише по стандардном принципу: Бинарна репрезентација потомака

се формира копирањем генетског кода једног родитеља од почетка до тачке укрштања, а другог родитеља од тачке укрштања, до краја. Тако сваким укрштањем настаје два потомка.

Мутација

Како би се избјегла конвергенција у локална субоптимална рјешења и обновио изгубљени генетски материјал, на јединке се са неком малом вјероватноћом (која се зове ниво мутације) примјењује оператор мутације. Случајно одабраном гену на који се мутација примјењује обрће се вриједност у генетском коду са нуле на један и обрнуто. Алгоритам користи модификовани оператор просте мутације, који препознаје тзв. "смрзнуте" гене у генетском запису. Ген се назива "смрзнутим" ако све јединке имају исту вриједност бита на тој позицији. Концепт смрзнутих гена објашњен је у одјељку 1.2.3 уводне главе. У овој имплеменатацији вјероватноћа за мутацију несмрзнутих гена износи $\frac{0.4}{|E \setminus E_{kriticne}|}$, док је за смрзнуте гене вјероватноћа мутације 2.5 пута већа и износи $\frac{1.0}{|E \setminus E_{kriticne}|}$.

Елиминисање поновљених и сличних јединки

У популацији је могућа појава јединки са истим генетским кодом. Да би се спријечило да алгоритам западне у локална субоптимална рјешења, вриједност прилагођености за све ове јединке, осим једне, се поставља на нулу, чиме се омогућава да оператор селекције ове јединке изостави за сљедећу генерацију. Такође, ако би постојао значајан број јединки које немају исти генетски кôд, а имају исту вриједност функције циља, поново би дошло до смањења разноврности генетског материјала, што узрокује превремену конвергенцију. Стога је број оваквих јединки ограничен на вриједност $N_{rv} = 40$.

Кеширање

Рачунање вриједности функције циља је најзахтјевнији задатак у читавом алгоритму. Да би се унаприједиле перформансе алгоритма, користи се техника кеширања. Кеш је имплемантиран као *hash-queue* структура података, капацитета 5000. Када за дату јединку треба да се рачуна вриједност функције циља, прво се, за ту јединку, гледа да ли је та вриједност већ израчуната и похрањена у кешу. У случају да вриједност није већ израчуната, тек тада се она рачуна и похрањује у кеш. У случају да јесте израчуната, она се не рачуна поново, већ се само прочита из кеша. У случају да се кеш напуни, бришу се оне вриједности које најдуже нису кориштене. Оператор кеширања дјелимично је описан у

одјељку 1.2.3, док је детаљнији опис дат је у [66].

4.4 Метод промјенљивих околина за рјешавање TR_p

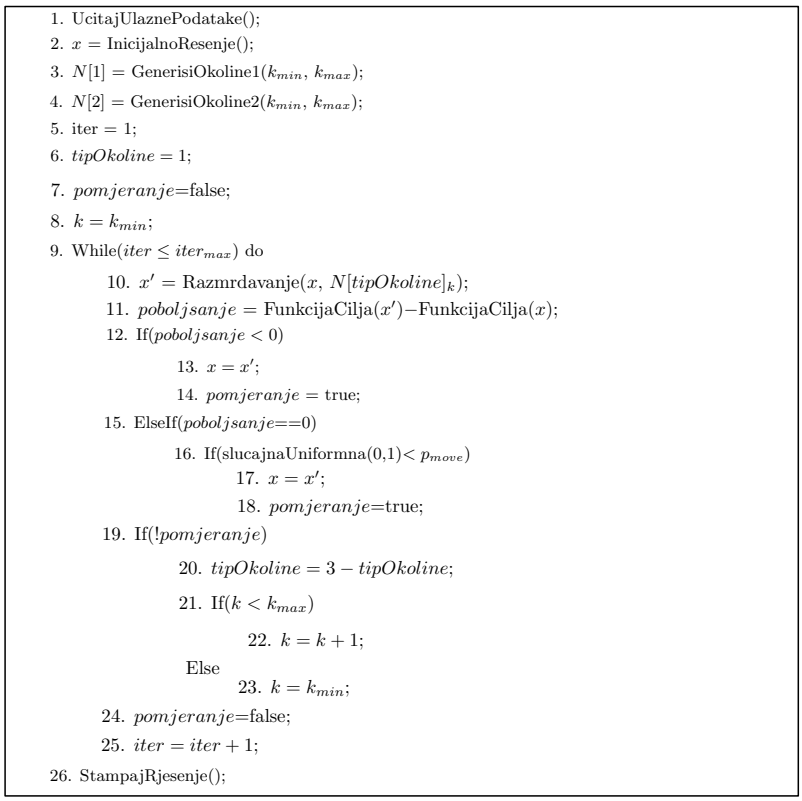
У овом одјељку ће бити описана метода промјенљивих околина развијена за рјешавање TR_p проблема. Резултати презентовани у овој секцији, заједно са резултатима из претходне секције, објављени су у раду [84].

Репрезентација рјешења је извршена на исти начин као код генетског алгоритма. Свако рјешење се представља бинарним низом дужине $|E \backslash E_{kriticne}|$. Функција циља се такође рачуна на исти начин као код генетског алгоритма.

За разлику од стандардног метода претраге помоћу промјенљивих околина, у овом алгоритму је примијењен другачији приступ: претрага се одвија само унутар процедуре размрдавања, док се локална претрага изоставља. Овакав приступ је чест у случајевима када се унутар процедуре размрдавања одвија "дубља" претрага у односу на само прелажење у нове околине, док је други разлог заснован на чињеници да је вријеме извршења функције циља већ прилично велико, те би претрага унутар процедуре за локално претраживање довела до још већег укупног времена извршетка алгоритма.

Сличне варијанте редукованог VNS-а (енг. Reduced VNS - RVNS) могу се срести на примјер у [28, 49].

Ток алгоритма је приказан на слици 4.2. Најприје се, на случајан начин, врши иницијализација полазног рјешења и генеришу се два система околина. Природа самих околина, као и начин прелажења из једне околине у другу, објашњени су у одјељку 4.4.2. У свакој итерацији алгоритма, ново рјешење x' се бира на случајан начин у некој околини тачке x. У зависности од вриједности функције циља за рјешење x', одлучује се да ли ће се прећи у ново рјешење x' или ће се остати у тренутном рјешењу. Ако је вриједност функције циља за x' боља (тј. стриктно мања) од вриједности за x, онда постављамо $x = x'$ и претрага наставља са истом околином. Ако је вриједност функције циља за x' стриктно већа у односу на рјешење x, тада се претрага наставља са старим рјешењем x и наредном околином. У случају да су ове двије вриједности једнаке, онда се са вјероватноћом p_{move} подешава $x = x'$ и претрага се наставља са истом околином, док се са вјероватноћом $1 - p_{move}$ оставља старо најбоље рјешење x и наредном околином. Након што се све околине размотре, претрага поново почиње са првом, све док критеријум за заустављање алгоритма не буде испуњен. У нашем случају, критеријум за заустављање је задат преко



Слика 4.2: Основна шема VNS алгоритма

достизања максималног броја итерација.

4.4.1 Иницијализација и функција циља

Као што је већ речено, репрезентација рјешења је извршена на исти начин као код генетског алгоритма. Свако рјешење се представља бинарним низом дужине $|E \backslash E_{kriticne}|$. У случају да је вриједност елемента низа једнака 1, одговарајућа грана је укључена у рјешење, а у случају да је елемент низа једнак 0, није.

Функција циља се такође рачуна на исти начин као код генетског алгоритма. У случају појаве некоректног рјешења, унутар функције циља, алгоритам "поправља" рјешење по истом принципу као код генетског алгоритма.

4.4.2 Процедура размрдавања

У овом алгоритму дефинишу се два типа околина.

$N_k^{obrni}(x)$: У првом типу околина, врши се инвертовање вриједности неких k елемената низа. Прецизније, у k - тој околини врши се избор неких k елемената низа који одговара тренутном рјешењу. Сваки елемент низа

одговара једној грани и за сваки елемент (сваку грану) се врши инвертовање, на сљедећи начин: у случају да је грана раније припадала рјешењу (одговарајући елемент низа је имао вриједност 1), тада се та грана уклања из рјешења. У случају да грана није припадала рјешењу (дати елемент низа је имао вриједност 0), тада се та грана укључује у рјешење, тј. елемент низа добија вриједност 1.

$N_k^{ukloni}(x)$: Други тип околина се формира тако што се у низу који одговара рјешењу x изабере неких k позиција и вриједности одговарајућих елемената се поставе на 0, док се преостале позиције не дирају. Практично, на овај начин се за неких k грана из скупа грана $E \backslash E_{kriticne}$ уклања из рјешења, без обзира на то да ли неке од тих грана већ ионако нису укључене у рјешење.

Примјер 4.2. *Анализирајмо граф G из примјера 1. На примјер, нека је тренутно рјешење x представљено низом* 110100110 *и нека је $k = 4$. Ако оператор размрдавања на случајан начин изабере позиције 2, 5, 7, и 9, рјешење x' у околини првог типа (obrni) би било* 100110011*, док је одговарајуће рјешење за други тип околина (ukloni)* 100100010*.*

Очигледно је да релација $k \leq |E \backslash E_{kriticne}|$ мора бити задовољена. Кардиналност сваке околине, с обзиром на то је $|N^k(x)| = \binom{|E \backslash E_{kriticne}|}{k}$. Пошто је релација $k_{max} < |E \backslash E_{kriticne}|/2$ задовољена за већину инстанци које су кориштене у одјељку 4.5, имамо да је $\binom{|E \backslash E_{kriticne}|}{k-1} < \binom{|E \backslash E_{kriticne}|}{k}$, тј. $|N^{k-1}(x)| < |N^k(x)|$, тако да за околине дефинисане на овај начин важи да је задовољен услов формирања околина са растућом кардиналошћу.

Избор одговарајуће околине

За дати број k, алгоритам креће са првим типом околина и користи га, све док се алгоритам помјера на ново рјешење. Након тога се "активира" други тип околина, који се примјењује све док важи исти принцип. Практично, алгоритам примјењује један тип околина све док та околина даје рјешења која поправљају тренутно најбоље рјешење, а након тога се узима други тип околине. Итеративно, вриједност k се увећава за 1, све док се не достигне вриједност k_{max}, након чега се k ресетује на k_{min}.

4.5 Експериментални резултати

Сви експерименти су обављени на AMD DualCore @ 2 GHz са 2 GB RAM рачунару. Критеријум за заустављање GA је достигнут максималан број од 500 генерација, или 200 генерација без побољшања вриједности функције циља.

RVNS се зауставља након 100 итерација. Други параметри за RVNS су: $k_{min} = 2$, $k_{max} \in \{8, 10, 15\}$, за инстанце које садрже $10, 15$ и 20 чворова респективно и $k_{max} = 20$ за инстанце које садрже више од 20 чворова.

Параметар p_{move} је подешен на 0.6. Тестирање GA и RVNS је извршено на истом, случајно генерисаном скупу инстанци. Генерисање инстанци је вршено са сљедећим улазним параметрима: број чворова ($|V|$), број грана ($|E|$), број критичних грана ($|E_{kriticne}|$) и укупан број лабела (p). Да би се одредило оптимално рјешење за мале инстанце које садрже до 30 грана, примијењена је техника тоталне енумерације. За сваку инстанцу оба метахеуристичка метода, GA и RVNS су покретана 20 пута.

Табела 4.2 садржи резултате од GA и RVNS за мање инстанце ($|E| < 30$ и без критичних грана), док табеле 4.3 и 4.4 садрже резултате добијене на већим инстанцама ($|E| \geq 30$). У све три табеле, прве колоне садрже број чворова, број грана и вриједност параметра p, респективно. У табели 4.2 наредна колона, названа *opt* садржи оптимално рјешење добијено тоталном енумерацијом. За све инстанце приказане у табели 4.2, и GA и RVNS проналазе оптимално рјешење. У табелама 4.3 и 4.4, четврта колона ($|E_c|$) садржи број критичних грана. За мање инстанце $|E_c| = 0$ те је ова колона у табели 4.2 изостављена. У петој колони (колона *best*) у табелама 4.3 и 4.4 за дату инстанцу је приказан најбољи добијени резултат, израчунат као $min\{GA_{sol}, RVNS_{sol}\}$.

Наредне четири колоне у све три табеле се односе на GA приступ. У колони *sol*, приказано је најбоље добијено рјешење, са ознаком *opt* (у табели 4.2) или ознаком *best* (табеле 4.3 и 4.4) у случајевима када GA достиже оптимално рјешење (табела 4.2) или најбоље познато рјешење (табеле 4.3 и 4.4). Наредна колона (t) у све три табеле садржи просјечно укупно вријеме (у секундама) потребно за завршетак GA. Квалитет рјешења у свих 20 извршења је процијењен у колонама *agap* и σ, гдје се просјечна релативна грешка (енг. average gap - agap) рачуна као $agap = \frac{1}{20}\sum_{i=1}^{20} gap_i$, гдје је $gap_i = 100 * \frac{sol_i - opt}{opt}$ у односу на оптимално рјешење *opt* ако је оно познато, односно у односу на најбоље познато рјешење *best*, тј. $gap_i = 100 * \frac{sol_i - best}{best}$ у случају када оптимално рјешење није познато, (sol_i представља рјешење добијено у i-том извршењу). Стандардна девијација је рачуната по формули $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$.

Посљедње четири колоне у све три табеле се односе на RVNS, а подаци су приказани на исти начин као и код GA.

Из табеле 4.2 јасно се види да и GA и RVNS у кратком времену достижу сва оптимална рјешења за инстанце које садрже до 30 грана. За све инстанце GA у свих 20 извршења достиже оптимално рјешење (у колони *agap*, која се односи на GA у свим овим случајевима је приказана вриједност 0), што није случај и са RVNS алгоритмом. Са друге стране, у већини случајева, RVNS је значајно бржи од GA, (од 10 до 30 пута). С обзиром на то да је GA метахеуристика заснована на популацији, рачунање функције циља временски захтјевна операција, а локална претрага у RVNS је изостављена, за очекивати је да ће GA у већини случајева бити спорији. Са друге стране, експериментали резултати показују да GA има боље механизме за излазак из субоптималних рјешења. Експерименти су такође показали да извршење RVNS у више од 100 итерација не унапређује крајње рјешење, тако да разлика у времену извршења ова два приступа може такође бити објашњена и релативно малим бројем итерација RVNS алгоритма.

Као што се види из табеле 4.2, вријеме извршења оба алгоритма очекивано расте са порастом броја грана у графу. Такође, вријеме извршења расте и порастом параметра p. Даље, можемо закључити да број p има већи утицај на вријеме извршења него број грана (на примјер, за инстанце са 15 чворова, вријеме извршења расте са порастом параметра p, чак и у случају када број грана у графу опада).

У случају већих инстанци, резултати јасно указују да и GA и RVNS проналазе рјешења у разумном времену. Поново, видимо да је GA спорији од RVNS и да вријеме извршења GA иде до неколико хиљада секунди. Са друге стране, GA добија боље резултате функције циља у односу на RVNS за већину инстанци.

И у случају великих инстанци, видимо да вријеме извршења природно зависи од димензије проблема. Утицај параметра p је такође значајан: порастом вриједности параметра p, оба приступа продужавају вријеме извршења. Експерименти показују да у случајевима $p = 1$ и $p = 2$, оба алгоритма брже проналазе рјешења (у поређењу са већим вриједностима параметра p), што указује на чињеницу да се оба приступа поуздано могу користити за рјешавање MED и BTR проблема. Број грана које припадају $E_{kriticne}$ не утиче значајно на пораст времена извршења (на примјер, у случају инстанци са 40 чворова у табели 4.4). Просјечна релативна грешка и стандардна девијација су релативно мали у већини случајева (посебно код GA), што доказује поузданост оба метода.

Иако поређење са другим радовима није могуће усљед недостатка заједнич-

Табела 4.2: Резултати добијени примјеном GA и RVNS на мале инстанце

				GA				RVNS			
$\|V\|$	$\|E\|$	p	opt	sol	t (sec)	$agap$ (%)	σ (%)	sol	t (sec)	$agap$ (%)	σ (%)
10	23	2	12	*opt*	1.411	0	0	*opt*	0.100	0.608	2.255
10	29	3	12	*opt*	5.388	0	0	*opt*	0.362	0.617	2.247
10	16	4	11	*opt*	2.040	0	0	*opt*	0.166	0.550	2.076
10	19	1	11	*opt*	0.201	0	0	*opt*	0.021	0.564	2.064
10	22	2	12	*opt*	1.324	0	0	*opt*	0.088	0.617	2.264
10	21	3	12	*opt*	2.682	0	0	*opt*	0.163	0.617	2.264
10	19	4	11	*opt*	3.758	0	0	*opt*	0.305	0.591	2.071
10	21	5	14	*opt*	3.311	0	0	*opt*	0.213	0.700	2.642
15	20	2	16	*opt*	2.390	0	0	*opt*	0.154	0.800	3.020
15	17	10	16	*opt*	32.798	0	0	*opt*	2.385	0.800	3.020
15	25	1	16	*opt*	1.080	0	0	*opt*	0.064	0.806	3.027
15	22	2	18	*opt*	5.660	0	0	*opt*	0.169	0.900	3.397
15	23	3	16	*opt*	7.502	0	0	*opt*	0.368	0.800	3.020
15	21	4	16	*opt*	11.074	0	0	*opt*	0.631	0.822	3.005
15	25	5	18	*opt*	19.816	0	0	*opt*	0.926	0.906	3.403
20	26	2	22	*opt*	5.655	0	0	*opt*	0.340	1.134	4.125
20	26	3	22	*opt*	19.617	0	0	*opt*	0.791	1.105	4.146
20	23	4	22	*opt*	35.309	0	0	*opt*	1.237	1.100	4.152

Табела 4.3: Резултати добијени примјеном GA и RVNS на велике инстанце

					GA				RVNS			
$\|V\|$	$\|E\|$	p	$\|E_c\|$	*best*	*sol*	t (sec)	*agap* (%)	σ (%)	*sol*	t (sec)	*agap* (%)	σ (%)
15	30	2	0	16	*best*	5.70	0.00	0.00	*best*	0.34	0.86	2.98
15	32	4	2	18	19	15.98	5.56	0.00	*best*	1.12	0.99	3.56
15	32	5	3	17	19	34.51	11.76	0.00	*best*	2.27	0.96	3.57
15	39	2	2	17	*best*	7.30	1.18	6.71	*best*	0.33	0.88	3.20
15	41	3	0	17	*best*	17.09	0.00	0.00	*best*	0.94	0.89	3.19
20	33	4	3	26	*best*	83.06	2.88	14.38	*best*	1.43	1.47	5.47
20	35	2	0	21	*best*	12.97	0.00	0.00	*best*	0.67	1.09	3.96
20	38	3	2	25	*best*	29.66	0.00	0.00	*best*	1.16	1.30	4.71
20	41	6	4	21	*best*	81.32	0.00	0.00	*best*	6.44	1.12	3.97
20	98	1	12	22	24	13.17	9.99	4.93	*best*	1.11	1.38	4.71
20	99	2	0	21	*best*	69.91	0.48	2.30	23	4.43	10.82	3.93
20	99	1	4	22	*best*	19.79	1.36	6.54	23	1.19	5.85	4.09
20	106	1	0	20	*best*	22.18	0.75	3.35	22	1.35	11.31	4.01
20	108	5	8	21	*best*	397.51	20.48	108.49	22	28.31	9.99	18.28
20	110	8	14	24	26	749.56	9.80	5.67	*best*	71.03	1.50	5.29
20	112	1	16	22	25	14.70	13.86	1.15	*best*	1.36	1.40	4.75
20	116	3	5	21	*best*	174.60	2.14	11.19	22	12.22	6.08	4.19
20	126	10	13	22	24	965.86	20.00	48.27	*best*	83.88	1.40	4.76
20	141	1	17	22	26	21.14	19.78	6.03	*best*	2.27	1.56	5.21
30	125	2	8	32	*best*	282.57	1.09	4.61	37	14.89	17.71	6.41
30	126	1	0	31	*best*	92.66	2.58	10.36	36	3.74	18.21	6.46
30	129	2	0	31	*best*	337.35	1.29	5.60	35	13.31	14.91	6.13
30	129	1	7	32	*best*	93.20	1.88	8.78	33	4.23	5.08	6.33

Табела 4.4: Резултати добијени примјеном GA и RVNS на велике инстанце (наставак)

					GA				RVNS			
$\|V\|$	$\|E\|$	p	$\|E_c\|$	*best*	*sol*	t (sec)	*agap* (%)	σ (%)	*sol*	t (sec)	*agap* (%)	σ (%)
40	183	1	0	42	*best*	344.49	2.14	9.83	50	11.48	22.00	8.58
40	187	2	0	42	*best*	1373.06	1.07	7.38	51	50.03	24.11	7.89
40	187	1	25	49	*best*	246.31	1.43	8.72	51	13.22	6.82	9.37
40	204	5	19	46	*best*	7619.83	1.96	8.28	51	311.80	13.75	9.22
50	104	5	11	56	*best*	5970.34	2.86	14.53	*best*	100.60	3.03	11.10
50	136	5	0	55	*best*	9285.69	0.82	3.46	61	188.87	14.14	10.60
50	213	1	19	60	*best*	618.03	1.67	6.86	68	24.70	17.15	12.12
50	226	2	22	61	*best*	2645.79	0.66	2.98	66	91.45	11.86	12.25
50	245	2	0	53	*best*	2873.49	1.32	6.04	68	98.78	32.01	10.38
50	277	1	0	53	*best*	1090.57	1.23	6.64	71	36.03	37.90	10.57
60	157	5	31	72	*best*	10977.84	0.56	3.14	77	383.03	10.89	13.54
60	179	2	0	67	*best*	5040.59	0.97	3.94	79	88.56	22.14	12.99
60	183	3	33	74	*best*	5191.59	1.08	6.09	80	141.24	11.49	14.25
60	188	1	0	67	*best*	1105.83	0.90	5.13	73	25.12	12.86	12.63
70	212	2	0	81	*best*	10213.43	3.40	17.22	102	128.92	31.39	16.00
70	230	1	0	76	*best*	2719.73	0.53	3.09	94	47.22	28.55	14.38
70	248	2	14	81	*best*	12698.17	6.05	32.37	95	177.91	22.28	15.47
80	183	2	0	89	*best*	12389.59	7.53	31.87	107	84.32	28.36	25.28
100	319	1	0	115	*best*	5662.65	0.87	3.88	150	68.71	38.23	22.27

ких инстанци, можемо закључити да оба алгоритма проналазе оптимално рјешење за мање инстанце, док за веће инстанце оба алгоритма проналазе рјешења у разумном времену.

4.6 Завршна разматрања за TR_p проблем

У овој глави описан је проблем p-арне транзитивне редукције и двије метахеуристичке методе које га рјешавају: генетски алгоритам и метод промјенљивих околина. Прописани GA користи бинарно кодирање и стандардне генетске операторе. Функција циља поправља некоректне јединке сукцесивним додавањем грана, на начин да се додавањем сваке гране даје највећа шанса да новонастало рјешење постане допустиво. Вријеме извршења алгоритма је побољшано употребом технике кеширања. Такође, употреба "смрзнутих гена" утиче на повећање разноврсности генетског материјала. У редукованом методу промјенљивих околина, претрага се одвија у процедуре размрдавања, у оквиру које се комбинују два типа околина: за k- ту околину, ново рјешење се бира инвертовањем вриједности неких k елемената низа везаног за полазно рјешење, или постављањем неких k елемената низа на нулу, чиме се обезбјеђује да тих k одговарајућих грана не улази у рјешење.

Експерименти извршени на случајно генерисаним инстанцама указују на поузданост употребе ових метода. За мање инстанце, достигнута су оптимална рјешења, која су верификована методом тоталне енумерације. За веће инстанце, оба метода достижу квалитетна рјешења у разумном времену. RVNS је значајно бржи од GA, док GA проналази рјешења бољег квалитета. Ови резултати представљају прве експерименталне резултате за случајеве $p > 2$.

Претходно описано истраживање се може проширити у неколико смјерова. Нумерички резултати добијени за вриједности $p = 1$ и $p = 2$ су охрабрујући за примјену алгоритма за рјешавање MED и BTR проблема, као и примјену на реалне инстанце из биологије. Други правац даљег истраживања може бити оријентисан ка имплементацији алгоритама на паралелним машинама што би значајно унаприједило перформансе самих алгоритама.

Глава 5

Проблем дијељења скупа

5.1 Увод

Нека је C колекција коначних подскупова скупа S. Проблем дијељења скупа (енг. Set splitting problem) је одређивање да ли постоји партиција скупа S на два непразна подскупа P_1 и P_2, таква да сваки елемент колекције C има непразан пресјек и са P_1 и P_2. Другим ријечима, ниједан елемент колекције C није подскуп ни P_1 ни P_2. Оптимизациона варијанта овог проблема се назива Проблем проналажења максималне подјеле скупа (енг. Maximum Set Splitting Problem - MSSP), гдје је циљ пронаћи ону партицију скупа S, такву да је број елемената колекције C са непразним пресјеком и са P_1 и са P_2 максималан. Подскупови колекције C са овом особином се називају *подијељени подскупови*. У тежинској варијанти овог проблема, подскупови колекције C имају тежине, а циљ је максимизовати укупну тежину подијељених подскупова.

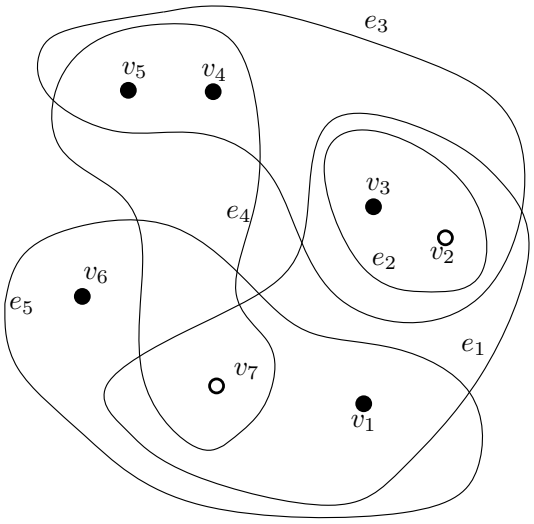
Доказ да је овај проблем NP тежак дао је L. Lovász у [76], гдје је аутор истраживао неке инваријанте хиперграфова. Хиперграф је уопштење графа, гдје за дати скуп чворова V, са једном граном, може бити инцидентан произвољан број чворова из V (оваква грана назива се хиперграна). Проблем проналажења минималног броја боја које су потребне да се обоје чворови графа на тај начин да ниједна грана нема све чворове обојене само једном бојом се показао подједнако тешким као и одређивање хроматског броја графа. Посебан случај је бојење чворова у тачно двије боје, а оптимизациона варијанта овог проблема се назива "максимално 2 - бојење хиперграфа" (енг. Max Hypergraph 2-Coloring - MH2C). Прецизније, 2 - бојење хиперграфа је одређивање да ли се сви чворови графа могу обојити са двије броје (на примјер у бијелу и црну), тако да свака хиперграна има најмање по један чвор сваке боје. Проблем максималног 2-

бојења хиперграфа је максимизовање броја хиперграна које садрже најмање по један бијели и један црни чвор.

Утврђено је да су MSSP и MH2C исти проблеми. Заиста, за дати скуп чворова V у хиперграфу, свака хиперграна дефинише један подскуп од V, који садржи елементе инцидентне са хиперграном. Тада је скуп свих хиперграна заправо једна колекција C подскупова од V. Тако, одређивање 2-бојења скупа чворова којим се максимизује број хиперграна које садрже чворове у обје боје је исти задатак као и проналажење партиције скупа V, такве да сваки подскуп из C садржи чворове из обје партиције.

Добро познати Max Cut проблем је специјална варијанта од MSSP. За дати граф $G = (V, E)$, Max Cut проблем је одређивање партиције скупа V у два подскупа V_1 и V_2, такав да је број грана са крајевима у различитим партицијама максимизован. Тако, може се рећи да је Max Cut проблем посебан случај MH2C проблема код кога је свака хиперграна инцидентна са тачно два чвора, или случај MSSP, код кога су сви скупови из колекције C кардиналности 2.

Примјер 5.1. *Нека је* $S = \{1, 2, 3, 4, 5, 6, 7\}$ *и* $C = \{S_1, S_2, S_3, S_4, S_5\}$, $S_1 = \{1, 2, 3, 7\}$, $S_2 = \{2, 3\}$, $S_3 = \{2, 3, 4, 5\}$, $S_4 = \{4, 5, 7\}$, $S_5 = \{1, 6, 7\}$. *На слици 5.1 је приказан хиперграф, са чворовима* $\{v_1, v_2, ..., v_7\}$ *који одговарају елементима скупа* S *и хипергранама које одговарају елементима из* C. *Чворови* v_2 *и* v_7 *су обојени бијелом бојом, а остали црном. Лако се види да свака хиперграна има барем по један чвор обојен у сваку боју. Тако је партиција* $(P_1, P_2) = (\{2, 7\}, \{1, 3, 4, 5, 6\})$ *оптимална, јер сваки подскуп садржи бар по један елемент у свакој партицији.*



Слика 5.1: Примјер максималног 2 - бојења хиперграфа

5.2 Примјена MSSP у образовању

Као и други слични проблеми који подразумијевају подјелу једне цјелине на мање дијелове по неком задатом критеријуму, и MSSP може бити примијењен за рјешавање неких организационих проблема. У овој секцији анализирају се неке опште примјене, као и једна детаљнија илустрација примјене подјеле лекција курса на два дијела.

5.2.1 Опште примјене MSSP у образовању

Претпоставимо да се у радној јединици располаже скупом запослених који требају да обаве неке задатке. Сваки запослени може да обавља више задатака и за сваки задатак може да постоји више запослених који могу да га обављају. Ако је циљ да се обезбиједи флексибилност у организацији тако што се скуп задатака разбија на два подскупа (на примјер обављање задатака на двије локације), на начин да број запослених који могу да раде барем један задатак на свакој локацији буде што је могуће већи, онда се овај проблем заправо своди на рјешавање MSSP, гдје се сваки задатак посматра као елемент скупа свих задатака, док се сваки запослени идентификује са подскупом тог скупа. Тај подскуп садржи све оне задатке које тај запослени може да обавља. Рјешење MSSP тада одговара постављеном циљу.

С обзиром да је однос задатака и запослених тзв. однос "више на према више", циљ се може поставити и обрнуто: Запослени се посматрају као елементи скупа, а задаци су подскупови тог скупа (подскуп садржи све оне запослене који могу да обављају дати задатак). У овој поставци скуп запослених треба разбити у два подскупа, који се на примјер везују за два дисјунктна временска периода, али тако да је број задатака који се могу обављати и у једном и у другом периоду максимизован.

У образовању, овај приступ се може примијенити, на примјер, у организовању љетњих и зимских школа. Претпоставимо да се располаже скупом $\{E_1, E_2, ..., E_m\}$ од m предавача и укупно n наставних области $\{t_1, t_2, ..., t_n\}$ за које треба организовати предавања. За сваку наставну област постоји листа предавача (подскуп скупа предавача), који могу да предају ту област. Због смањења трошкова, сваки предавач се ангажује само у једном периоду (или у љетњој или у зимској школи). Циљ је за сваког предавача одредити период када је ангажован, али тако да је број области које су покривене предавачима у оба периода максимизован. Овом циљу одговара рјешење одговарајућег MSSP.

Други модел примјене, који ће у наредном одјељку бити детаљније објашњен,

своди се на сљедеће разматрање. Материјал за неки курс се природно дијели по лекцијама. Претпоставимо да из неке научне области постоји више тематских цјелина које се у некој мјери покривају датим курсом. Једна лекција може да спада у различите тематске цјелине, а свака тематска цјелина садржи више лекција. Са методичког аспекта, било би корисно направити подјелу лекција у двије цјелине (на примјер на зимски и љетњи семестар), али тако да што више тематских цјелина буду присутне у оба семестра. Тиме би се омогућило да основе (и не само основе) сваке тематске цјелине буду поновљене и континуирано обрађиване током читаве академске године, што генерално доприноси побољшању квалитета учења [92]. Ако за овај приступ направимо одговарајући математички модел и лекције представимо елементима скупа, а тематске цјелине идентификујемо као подскупове који садрже одговарајуће лекције, тада рјешењу описаног задатка одговара рјешење одговарајућег MSSP. Ово разматрање је илустровано на конкретном курсу у наредном одјељку.

5.2.2 Подјела скупа лекција из курса Увод у рачунарство

За потребе илустрације подјеле курса на два дијела по принципу MSSP одабран је курс из Увода у рачунарство који се предаје на првој години студија на Студијском програму Математика и информатика Природно математичког факултета у Бањалуци. Циљ курса је да студентима омогући савладавање основних знања из архитектуре рачунара и основа програмирања, кроз програмски језик Python. Укупан фонд часова предавања је 90 (15 седмица по 3 часа седмично, у два семестра).

Да би се направио одговарајући математички модел за дати курс на који може да се примијени нека техника за рјешавање MSSP, потребно је да сеураде три задатка:

(а) идентификација лекција;

(б) идентификација тематских цјелина;

(ц) смјештање лекција у одговарајуће тематске цјелине.

Идентификација лекција

На основу плана и програма, доступног и на интернет страницама Факултета (http://www.matinf.pmfbl.org), види се да је садржај читавог курса распоређен у два семестра од по 15 седмица. С обзиром на фонд од 3 часа седмично, у зависности од обима лекција и тежине градива, у свакој седмици се најчешће

обрађују по двије или три лекције. На тај начин, идентификовано је укупно **76** лекција, које у математичком моделу представљају елементе скупа који се партиционише.

Списак свих лекција, приказан је у табели 5.1. Због уштеде простора и боље прегледности, називи неких лекција су скраћени. По званичном акредитованом програму, читав курс из Увода у рачунарство је подијељен и организован у два једносеместрална курса, Увод у рачунарство 1 и Увод у рачунарство 2. Тренутна подјела лекција по семестрима одговара редослиједу лекција у табели 5.1: првих 39 лекција су лекције из зимског, док су лекције од редног броја 40 до 76 из љетњег семестра. С обзиром на чињеницу да су лекције већ подијељене по семестрима, циљ истраживања приказан у овом одјељку је двојак:

(а) да се установи колико тренутна подјела курса одговара препоруци да што више тематских цјелина буду присутне у оба семестра;

(б) да се предложи прерасподјела лекција, у односу на постојећу, како би се постигла боља подјела, на основу формираног математичког модела датог MSSP.

Табела 5.1 – Списак лекција

Р.б.	Назив лекције
1	Хардвер и софтвер. CPU и главна меморија. Input/Output јединице. Категорије софтвера.
2	Аналогно и дигитално. Дигитално представљање података. Бинарни бројеви.
3	Спецификација рачунара. Меморија. Чување података у меморији. Врсте и капацитет меморије.
4	Структура CPU. Инструкциони циклус.
5	Мреже рачунара, начини повезивања. Интернет. TCP/IP. Домени. WWW. HTML
6	Функција рачунарског система. Структура рачунарског система. Организација и архитектура рачунарског система. Фон Најманова машина.
7	Азбука и кодови. Бројчани системи. Превођење цијелих бројева. Превођење разломљеног дијела. Хијерархија података у рачунару. Запис знаковних података у рачунару.
8	Сабирање и одузимање у бинарном и хексадецималном систему. Превођење из хексадецималног у декадни систем и обратно.
9	Запис неозначених и означених бројева, непотпуни и потпуни комплемент.
10	Конверзија између записа различитих дужина, промјена знака и сабирање и одузимање неозначених бројева, сабирање и одузимање бројева у непотпуном и потпуном комплементу.
11	Множење неозначених цијелих бројева, множење цијелих бројева у непотпуном и потпуном комплементу, дијељење цијелих бројева.
12	Децимална аритметика: промјена знака, сабирање и одузимање. Реални бројеви у непокретном и покретном зарезу. Запис са хексадецималном основом.
13	IEEE стандард 754. Сабирање и одузимање у непокретном и покетном зарезу. Множење и дијељење у непокретном и покретном зарезу.
14	Запис текста, слике и звука.
15	Дефиниција програма, програмирања и рачунарских наука.
16	Инсталација и опис радног окружења за програмски језик Python. Процес формирања програма у радном окружењу. Изрази у Python-у.
17	Аритметички и логички изрази. Типови у Python-у. Типови int и float. Приоритет оператора.
18	Варијабле и оператор додјеле вриједности. Сложени оператори и оператор додјеле вриједности.
19	Грешке и механизми откривања грешака. Појам функције и основне особине функција. Локалне и глобалне варијабле.
20	Уграђене функције (енг. Build-in) језика Python. Стилови при писању програма који садрже изразе и оператор додјеле. Коментари у програмима.
21	Стрингови и операције над стринговима. Escape секвенце. Стрингови у више линија. Команда print(). Форматирање излаза. Кориснички улаз и команда raw_input().
22	Модули и примјери модула. Импортовање модула. Дефинисање властитих модула. Шта се дешава приликом импортовања модула. Програмирање help-а у Python-у.

Табела 5.1 – Списак лекција - наставак

Р.б.	Назив лекције
23	Објекти и методе у Python-у.
24	Рад са сликама. Пиксели и боје.
25	Листе. Листе и индекси. Модификовање листи. Уграђене функције за рад са лстама. Обрада чланова листе. Издвајање дијелова листе (енг. slicing). Инверз листе и палиндроми. Псеудоними (aliasi). Методе листе. Угнијеждене листе. Разне врсте низова. Датотеке као листе (представљање датотеке листама). Аргументи командне линије.
26	Команде избора (гранања) у Python-у. Булова логика. Булови оператори. Релациони оператори. Примјена Булових оператора на int, float i string.
27	Команда if. Угњеждене команде гранања, оператор додјеле и услови гранања.
28	Команде понављања у Python-у. Команда for. Функција range(). Функција enumerate().
29	Угнијеждене for петље. Примјер обилажења дводимензионалне матрице. Команда while. Команде break и continue. Стилови при програмирању команди гранања и петљи.
30	Датотеке. Отварање датотека са рачунара за читање, додавање или писање. Отварање датотека са Interneta. Рад са датотеком која има један слог у линији. Слогови са више поља у линији. Слогови са позиционираним пољима у линији. Слогови са више линија. Look ahead при читању датотека.
31	Писање у датотеке.
32	Скупови у Python-у. Рјечници. Инвертовање рјечника. Алгоритми. Euklid-ов, Wirth-ов и школски алгоритам за НЗД.
33	Рекурзија. Рекурзивни и итеративни алгоритам за рачунање факторијала и Fibonacci-јевих бројева.
34	Алгоритми претраживања. Поређење програма по дужини времена извођења. Претраживање и сортирање.
35	Основно линеарно претраживање и претраживање са sentinel-ом. Мјерење времена линеарног претраживања. Бинарно претраживање.
36	Сорт мјехурићима. Сорт избором. Сорт уметањем. Quick сорт. Merge сорт. Поређење времена рада алгоритама сортирања.
37	Конструкције. Додатне особине функција.
38	Изузеци. Тестирање програма. Дебагирање програма.
39	Узорци (енг. Patterns)
40	Логичке основе обраде података: Булова алгебра, Пун систем функција, SDNF и SKNF, Методе минимизације логичких функција, Логички елементи за обраду података, Комбинационе мреже, Секвенцијалне мреже, Примјери.
41	Структура савременог рачунарског система: Фон Нојманова машина, Систем прекида, Брзина обраде података
42	Процесори: Организација централног процесора, Регистри, Технологије израде микропроцесора, Транзистори, Технологије израде брзих чипова, CISC и RISC архитектура микропроцесора, Особине RISC процесора.
43	Машинске инструкције: Карактеристике машинских инструкција, Формати и типови инструкција, Број адреса у инструкцији. Начини адресирања у машинским инструкцијама
44	Машински и асемблерски језици, Позиви подпрограма, Смјештање података у меморији.
45	Меморије и улазно/излазни подсистем: Унутрашња меморија, Карактеристике, Кеш меморија, Спољашње меморије, Магнетни и оптички медији,
46	У/И модули, Технике извршавања У/И операција, У/И уређаји и њихове карактеристике
47	Откривање и корекција грешака: Методе за откривање и методе за откривање и корекцију грешака
48	Претраживање: Налажење минимума у листи, Пролажење кроз листу, Мјерење времена претраживања,
49	Линеарно претраживање: while и for верзије линеарног претраживања, Sentinel, Тајминг линеарног претраживања.
50	Бинарно претраживање: Сложеност алгоритама бинарног претраживања, уграђена функција бинарног претраживања
51	Алгоритми, Програм
52	Сложеност сортова у најбољем и најгорем случају: Сорт мјехурићима (енг. Bubble sort), Сорт уметањем (енг. Insert sort), Сорт избором (енг. Choice sort).
53	Сорт спајањем (енг. Merge sort), Брзи сорт (енг. Quick sort)
54	Дефинниције функција: Подразумијевани (енг. default) параметри, Листе параметара, Параметри са називима.
55	Изузеци: tray и except, except објеката. Изузеци: Функције и изузеци, raise изузетак, Still изузетка.
56	Тестирање: Функционални тест, unit test, black-box test, glass-box test, Независност, Ограничења, Тестом вођен развој програма. Дебагирање.
57	Обрасци дизајнирања: Фиксне вриједности, Степери и каунтери, Најпожељнији носилац, Задњи носилац,
58	Контејнер, Акумулатор, Темпорална варијабла, Заставица. Класе: Атрибути, Методе, Конструктори.
59	Специјалне методе: init ; str ; repr ; add ; sub Методе dir и help.
60	Инкапсулација у објектно оријентисаном програмирању (енг. Encapsulation)
61	Полиморфизам и насљеђивање: ad hock полиморфизам, Параметарски полиморфизам, Хијерархијски полиморфизам.

Табела 5.1 – Списак лекција - наставак

Р.б.	Назив лекције
62	Насљеђивање: Примјери class Atom и class Molecule.
63	GUI: Основни појмови, Event-driven програмирање. Modul Tkinter: Widgets: Button, Canvas, Checkbutton, Entry, Frame, Label, Listbox, Menu, Message, Menubutton, Text, TopLevel.
64	Основне GUI конструкције: Frame, Entry, Лаки и тешки процеси. Mutable variables: intVar, StringVar, BooleanVar, DoubleVar.
65	Модели, погледи, контролери: Примјер. Кориштење Lambda: Примјер.
66	Стил писања GUI: Font, Color, Layout. Widget-i: Text, Checkbutton, Menu.
67	Објектно-оријентисани GUI: Примјер.
68	Базе података: Архитектура база података , sqlite3, MySQLdb, SQLAlchemy. sqlite3 функције: connect(), cursor(), execute(), commit().
69	SQL tipovi podataka: NULL, INTEGER, REAL, TEXT, BLOB. Табеле у SQL: CREATE TABLE TableName(ColumnName Type, ColumnName Type, ...), INSERT INTO VALUES.
70	Налажење података у Базама података: SELECT FROM, fetchone(), fetchall(), SELECT FROM ORDER BY. Упити са условима: SELECT FROM WHERE, =; ! =; >; <;>=;<=.
71	Апдејтовање и брисање у базама података: UPDATE SET WHERE, DELETE FROM WHERE, DROP TABLE.
72	Трансакције у Базама података. NULL као замјена за непостојеће податке. Кориштење JOIN за комбиновање табела(INNER JOIN).
73	Кључеви и ограничења у Базама података. Напредне технике База података: Агрегација, Груписање, SelfJoins, Угнијеждени упити.
74	Мрежни модули : Socket. Клијент/Сервер пар: Примјер. Мрежни модули urllib i urllib2: Отварање удаљених датотека, повраћај удаљених датотека.
75	Опис функција неких мрежних модула 1: asynchat, asyncore, cgi, Cookie, cookielib, email, ftplib,gopherlib, httplib, imaplib.
76	Опис функција неких мрежних модула 2: mailbox, mailcap, mhlib, nntplib, poplib, robotparser, impleXMLRPCServer, smtpd, smtplib, telnetlib, urlparse, xmlrpclib.

Идентификација тематских цјелина

С обзиром на огромну количину научног, образовног, комерцијалног, забавног и другог информатичког садржаја, те на прилично хетерогене изворе и класификације, одређивање тематских цјелина које покривају читаву област рачунарских наука представља захтјеван посао. Како би се осигурао објективан приступ, са једне стране, и комплетност и корекност са друге, као главна референца за одређивање тематских цјелина у овом раду кориштени су ресурси објављени од стране удружења *Association for Computing Machinery* (*ACM*), једног од најпознатијих и највећих свјетских образовних и научних удружења из области рачунарских наука. Један од задатака ове организације је континуирана испорука материјала који садржи препоруке за креирање планова и програма рачунарских курсева, као и читавих студијских програма, како би се успјешније пратили нагли напредак и брзе промјене у области рачунарства.

У свом посљедњем извјештају [26] из 2008. године, ACM је у сарадњи са одјелом за едукацију IEEE удружења (*IEEE Computer Society Education Activities Board*) представио нови Водич за формирање планова и програма из области компјутерских наука, заснован на претходном извјештају из 2001. године и новим сазнањима која су стечена у међувремену. Овај извјештај пружа валидну основу за креирање планова студијских програма из области рачунарских наука, као и распореда појединачних курсева и користи се у многим високошколским установама широм свијета. Као основни допринос приказан у

Табела 5.2: Списак области знања

Област	Област (енг.)	скр.	об.	из.
Дискретне структуре	Discrete Structures	DS	6	0
Основе програмирања	Programming Fundamentals	PF	8	0
Алгритми и комплексност	Algorithms and Complexity	AL	5	6
Архитектура и организација	Architecture and Organization	AR	6	4
Оперативни системи	Operating Systems	OS	6	8
Мрежно рачунарство	Net-Centric Computing	NC	3	6
Програмски језици	Programming Languages	PL	6	5
Интеракција човјека и рачунара	Human-Computer Interaction	HC	2	8
Графика и визуелно рачунање	Graphics and Visual Computing	GV	2	11
Интелигентни системи	Intelligent Systems	IS	3	8
Управљање информацијама	Information Management	IM	3	12
Друштвени и професион. аспекти	Social and Professional Issues	SP	7	4
Софтверски инжењеринг	Software Engineering	SE	8	6
Наука о израчунавању	Computational Science	CN	0	3

овом извјештају, идентификовано је тзв. "тијело знања" (енг. body of knowledge), које садржи све релевантне рачунарске области, подијељене у 14 "области знања" (енг. knowledge areas). У табели 5.2 приказане су те области, са називима на нашем и енглеском језику, скраћеним називом на енглеском језику, заједно са бројем кључних, односно изборних области.

Свака од ових области знања подијељена је на тематске цјелине, означене као кључне или изборне. Укупно је предложено 146 тематских цјелина, од којих су 65 кључне, а 81 изборне. У зависности од тога колики се значај посвећује посматраној цјелини, у извјештају [26] је предложен и број часова које треба посветити кључним тематским цјелинама. Природно, фундаменталним областима је дата већа важност, тј. предложен је већи број кључних тематских цјелина и већи предложен број часова. То су, прије свих, области: дискретне структуре, основе програмирања, алгоритми и комплексност, архитектура и организација, оперативни системи, програмски језици, софтверски инжењеринг, те друштвени и професионални аспекти. За остале области знања предложен је мањи број сати и мањи број кључних области. Тиме се креаторима планова и програма омогућава већа флексибилност и прилагођавање специфичним потребама.

Смјештање лекција у тематске цјелине

С обзиром на чињеницу да предложено "тијело знања" покрива читаву област рачунарских наука, за потребе истраживања и груписања лекција курса Увод у рачунарство у одговарајуће тематске цјелине, број тематских области је значајно

Табела 5.3: Списак тематских цјелина и припадајућих лекција

Р.б.	Назив тематске цјелине	Област (енг.)	Списак лекција
1	Функције, релације и скупови	DS	18,19,32,33,61,37
2	Основна логика	DS	17,26,40,70
3	Технике доказивања	DS	17,19,33,40
4	Основе пребројавања	DS	7,9,17,24,40,48
5	Графови и стабла	DS	33,35,50,74
6	Дискретна вјероватноћа	DS	36,52,53
7	Основе конструкција	PF	2,17,18,20,27,28,29,30,54
8	Алгоритамско рјешавање проблема	PF	15,32,33,34,35,38,47,51,56
9	Структуре података	PF	17,21,25
10	Рекурзија	PF	33,35,37,50
11	Програмирање засновано на догађајима	PF	38,55,63
12	Објектно оријентисана парадигма	PF	23,58,60,61,62,67
13	Сигурно програмирање	PF	19,47,55
14	Основна анализа алгоритама	AL	32,36,48,52,53
15	Алгоритамске стратегије	AL	20, 32,34,35,39,50
16	Основни алгоритми	AL	32,35,36,48,49,50
17	Дигитална логика и репрезентација података	AR	2,3,7,8,9,10,11,12,13,14
18	Рачунарска архитектура и организација	AR	1,3,4,6,40,41,42,43,44
19	Интерфејси и У/И стратегије	AR	1,30,46
20	Архитектура меморије	AR	1,3,44,45
21	Функционална организација	AR	1,3,4,42,43
22	Уређаји	AR	2,14,24
23	Основни преглед оперативних система	OS	1,45,46
24	Мрежне комуникације	NC	5,74,75,76
25	Преглед програмских језика	PL	15,16,58,59,60,61,62
26	Основе превођења језика	PL	15,16,19,20,68,69
27	Декларације и типови	PL	17,18,20,58
28	Механизми апстракције	PL	22,57,58,65
29	Објектно оријентисано програмирање	PL	16,58,60,61,62,63,64,65,67
30	Основе интеракције човјек-рачунар	HC	16,63,66
31	Изградња GUI интерфејса	HC	63,64,65,66,67
32	Евалуација кориснички оријентисаних система	HC	47,56,57
33	Дизајн GUI	HC	65,66
34	Основе технике графике и визуелног рачунања	GV	14,24
35	Информациони модели	IM	60,68
36	Језици упита	IM	68,69,70,71,72,73
37	Историја рачунарства	SP	2,15,41

редукован. Задржане су уводне фундаменталне тематске области, док је већина области које се односе на напредне курсеве из рачунарства изостављена.

Преглед уврштених тематских цјелина приказан је у табели 5.3. Прве три колоне садрже редни број, назив тематске цјелине и скраћени назив области знања којој тематска цјелина припада, док је у крајњој десној колони наведен списак редних бројева припадајућих лекција. Редни бројеви лекција одговарају редним бројевима из табеле 5.1. Као што се види из табеле 5.3, лекције су сврставане само у оне тематске цјелине за које се поуздано може сматрати да се проблематика тематске цјелине обрађује унутар лекције, а не само успутно помиње. Примјећујемо да су све фундаменталне тематске цјелине покривене већим бројем лекција, док се неке друге обрађују само у мањем броју лекција. У табели 5.4 систематизована је расподјела лекција по тематским цјелинама. Видимо да је за 21 тематску цјелину број лекција у којима се оне обрађују мањи или једнак од 4, за 11 тематских цјелина важи да се обрађују у 5, 6 или 7 лекција, док се 5 тематских цјелина обрађује у 9 и више лекција.

Табела 5.4: Распоред броја лекција по тематским цјелинама

Број лекција	Број тематских цјелина са датим бројем лекција
2	3
3	10
4	8
5	3
6	7
7	1
8	0
9	4
10	1

Табела 5.5: Списак цјелина за које не важи тражени услов

Р.б.	Тематска цјелина	Област (енг.)	Присутна само у
22	Уређаји	AR	зимском
31	Изградња GUI интерфејса	HC	љетњем
32	Евалуација кориснички оријентисаних система	HC	љетњем
33	Дизајн GUI	HC	љетњем
34	Основе технике графике и визуелног рачунања	GV	зимском
35	Информациони модели	IM	љетњем
36	Језици упита	IM	љетњем

Анализа тренутне подјеле лекција и приједлог побољшања

На основу тренутне подјеле курса на два дијела, направљена је анализа колико та подјела одражава полазну препоруку да што више тематских цјелина буде обрађено у оба семестра. Поновимо да се по тренутној расподјели, (табела 5.1), лекције са редним бројевима 1-39 обрађују у зимском семестру, а лекције 40-76 у љетњем. Према распореду лекција по тематским цјелинама приказаном у табели 5.3, за 30 тематских цјелина важи да се оне обрађују у бар по једној лекцији у оба семестра, док за 7 тематских цјелина тај услов не важи. Примијетимо да алгоритам којим се одређује број подијељених тематских цјелина одговара алгоритму верификације рјешења одговарајућег MSSP. У прве двије колоне табеле 5.5 су приказани редни бројеви и имена тематских цјелина за које услов подијељености не важи. У наредној колони је приказан скраћени енглески назив одговарајуће области знања, а у крајњој десној колони информација у ком семестру је цјелина присутна.

Као што видимо из табеле 5.5, од седам "неподијељених цјелина", по једна

Табела 5.6: Систематизација предложених рјешења

Поступак	Ефекат на тематске цјелине	Посљедица на укупно рјешење
Пребацивање лекције 24 у љетњи семестар	Тематске цјелине 22 и 34 ће бити подијељене	Смањење броја неподијељених лекција за 2
Пребацивање лекције 56 у љетњи семестар	Тематска цјелина 32 ће бити подијељена	Смањење броја неподијељених лекција за 1
Пребацивање лекције 65 у зимски семестар	Тематске цјелине 31 и 33 ће бити подијељене	Смањење броја неподијељених лекција за 2
Пребацивање лекције 68 у зимски семестар	Тематске цјелине 35 и 36 ће бити подијељене	Смањење броја неподијељених лекција за 2

се односи на архитектуру рачунара, те графику и визуелно рачунање, док се три односе на интеракцију човјека и рачунара и двије на управљање информацијама. Примјећујемо такође да се областима које се односе на интеракцију човјека и рачунара и управљању информацијама пажња посвећује у љетњем семестру, док у зимском семестру овој проблематици није посвећена значајнија пажња. Стога би правац унапређења Плана и програма курса могао да се заснива на укључивању по једне лекције из графичког корисничког окружења, те база података у зимском семестру. Пажљивом анализом тренутног распореда лекција, може се примијетити да би се пребацивањем полазне лекције у вези са базама података и једне лекције у вези са интеракцијом човјека и рачунара у зимски семестар, могао ријешити проблем "неподијељености" цјелина 35 и 36, односно 31 и 33, а да се не наруше други методички аспекти, као што су контитуитет или зависност лекција.

Такође, може се закључити да би се размјеном лекција 24 и 56 ("Рад са сликама. Пиксели и боје." односно "Тестирање: Функционални тест, *unit test*, *black-box test*, *glass-box test*, Независност, Ограничења, Тестом вођен развој програма. Дебагирање.") постигла ситуација да су цјелине 22, 32 и 34 подијељене, а подијељеност преосталих тематских цјелина се не би нарушила.

Систематизација предложених унапређења приказана је у табели 5.6.

Овом прерасподјелом лекција добија се стање да су све тематске цјелине подијељене.

На крају овог одјељка, значајно је нагласити да се приказаном анализом дошло до сазнања да тренутна подјела курса у значајној мјери "прати" пропи-

сани приступ подјеле лекција. Као што се види из табеле 5.6, минорним размјенама лекција између семестара тај распоред се може учинити и оптималним. Ова чињеница указује на претпоставку да су аутори Плана и програма, мотивишући се овим или неким сличним приступом, у значајној мјери водили рачуна да се фундаменталне цјелине разматрају у оба семестра, како би се очувао континуитет у изучавању.

5.3 Ранији резултати

MSSP проблем је посљедњих година предмет истраживања и са комбинаторне и са алгоритамске тачке гледишта. Поновимо да је први доказ да је проблем NP тежак дат у [76] и проблем остаје NP тежак чак и ако су сви подскупови кардиналности мање или једнако 3 [40]. Чак и ако су сви подскупови фиксне кардиналности $r, r \geq 2$, проблем опет остаје NP тежак. Даље, MSPP је и APX комплетан, тј. не може се априксимовати у полиномском времену фактором већим од 11/12 [46]. Поједностављена варијанта проблема је случај када сваки елемент скупа C има највише два елемента. Варијанта одлучивања тада постаје "проблем 2 бојења графа" који се може ријешити у полиномском времену, на примјер, примјеном добро познатог алгоритма претраживања у дубину.

Пошто је проблем NP тежак и не може се рјешавати егзактним методама у разумном времену, оправдана је конструкција алгоритама који проналазе приближно рјешење неког гарантованог, доказивог квалитета. Тако је у [5] квадратна цјелобројна формулација за MSSP искоришћена за конструкцију приближног алгоритма са фактором 0.724. У [106], аутори су додавањем неких нових неједнакости и побољшањем начина заокруживања, презентовали приближан алгоритам са фактором 0.7499.

Неколико резултата који се односе на методе кернеларизације су унаприједили горњу границу за вријеме извршења алгоритма. За дати број k подијељених подскупова и величину проблема N, првобитно су презентоване границе времена $O(72^k N^{O(1)})$ и $O(8^k N^{O(1)})$ из [31] и [32], а у [74] је постигнуто ново унапређење које износи $O(2.65^k N^{O(1)})$. У [21, 22] је презентован алгоритам са временом $O(2^k + N)$ за тежинску варијанту проблема, док је у [75], употребом класичне теореме дуалности за повезаност у хиперграфовима, постигнут резултат $O(1.96^k + N)$.

Алгоритам за рјешавање MSSP заснован на "ДНК" приступу је презентован у [16]. Простор рјешења ДНК хеликса је конструисан уз помоћ "стикера" (енг. sticker based model), а након тога су на Адлеман - Липтон модел примијењени

биолошки оператори.

Први модел цјелобројног линеарног програмирања (енг. Integer linear programming - ILP) и један генетски алгоритам за рјешавање MSSP су презентовани у [70]. Употребом напредних софтверских алата (CPLEX) за рјешавање проблема линераног програмирања, тестиран је прописани ILP модел и показано је да CPLEX успјешно проналази оптимална рјешења за све мање и већину средњих тестних инстанци. GA имплементација користи бинарно кодирање и стандардне генетске операторе прилагођене датом проблему, који су унапријеђени употребом кеширања. Експериментални резултати су извршени на два скупа инстанци, чиме је доказана ефикасност и поузадност прописане методе.

Хеуристика заснована на електромагнетизму (EM) је презентована у [67]. Ова EM хеуристика је заснована на хибридном приступу у којем се комбинује механизам "привлачење - одбијање" за помјерање честица са техником скалирања. EM користи брзу локалну претрагу која додатно унапређује ефикасност читавог система. Алгоритам је тестиран на истим инстанцама као у [70] и добијени резултати јасно показују да је прописани EM приступ користан алат за рјешавање MSSP.

Као и други проблеми који се односе на партиције скупова или графова, MSSP се примјењује у разним областима науке. Једна корисна примјена овог приступа приказана је у [105, 104]. Аутори користе тзв. тернарну меморију са адресабилним садржајем (енг. ternary content addressable memory - TCAM) да би се ријешио класификацијски проблем са вишеструким спаривањем (енг. multi-match classification problem), који се користи за неке мрежне примјене: системи за детекцију напада или праћење испоручених пакета података у мрежи. Да би се унаприједиле перформансе употребе TCAM меморије, аутори користе алгоритам за дијељење скупа (set splitting algorithm - SSA) да би подијелили филтере у вишеструке групе. SSA се даље паралелно покреће на одвојеним групама. Овим приступом најмање половина пресјека се избацује када се скуп филтера разбије у два скупа. То резултира уштеду у меморији и боље перформансе читавог алгоритма.

5.4 Математичка формулација

У овом одјељку се представљају двије познате формулације цјелобројног програмирања: квадратна цјелобројна формулација (енг. quadratic integer programming QIP) из [5] и линеарна цјелобројна формулација (ILP) из [70]. Прије него што се наведу формулације, уведимо одговарајућу нотацију. Нека је S коначан

скуп кардиналности $m = |S|$ и нека је $C = \{S_1, S_2, ..., S_n\}$ колекција подскупова од S. Без губљења општости, можемо претпоставити да је $S = \{1, 2, ..., m\}$. Нека је (P_1, P_2) партиција од S. Поновимо да за скуп $S_j \in C$ кажемо да је "подијељен", ако S_j има непразан пресјек и са P_1 и са P_2.

Квадратна цјелобројна формулација (5.1)-(5.4) уводи два скупа промјенљивих:

$$y_i = \begin{cases} 1, & i \in P_1 \\ -1, & i \in P_2 \end{cases}, \quad \text{за } i = 1, ..., m,$$

и

$$z_j = \begin{cases} 1, & S_j \text{ је подијељен} \\ 0, & S_j \text{ није подијељен} \end{cases}, \quad \text{за } j = 1, ..., n.$$

QIP се дефинише као

$$\max \sum_{j=1}^{n} z_j \tag{5.1}$$

уз ограничења

$$\frac{1}{|S_j| - 1} \sum_{\substack{i_1, i_2 \in S_j \\ i_1 \neq i_2}} \frac{1 - y_{i_1} \cdot y_{i_2}}{2} \geq z_j, \quad \text{за све } S_j \in C \tag{5.2}$$

$$z_j \in \{0, 1\}, \quad \text{за } j = 1, ..., n \tag{5.3}$$

$$y_i \in \{-1, 1\}, \quad \text{за } i = 1, ..., m \tag{5.4}$$

За ILP уводимо параметре

$$s_{ij} = \begin{cases} 1 & i \in S_j \\ 0 & i \notin S_j \end{cases} \quad \text{за } i = 1, ..., m, \ j = 1, ..., n,$$

који означавају да ли елемент i из S припада подскупу S_j. Промјенљиве се дефинишу као

$$x_i = \begin{cases} 1 & i \in P_1 \\ 0 & i \in P_2 \end{cases}$$

$$y_j = \begin{cases} 1 & S_j \text{ је подијељен} \\ 0 & S_j \text{ није подијељен} \end{cases}$$

ILP модел за MSSP се формулише као

$$\max \sum_{j=1}^{n} y_j \tag{5.5}$$

уз ограничења

$$y_j \leq \sum_{i=1}^{m} s_{ij} \cdot x_i, \qquad \text{за све } j = 1, ..., n \tag{5.6}$$

$$y_j + \sum_{i=1}^{m} s_{ij} \cdot x_i \leq |S_j| \quad , \qquad \text{за свеј} = 1, ..., n \tag{5.7}$$

$$y_j \in \{0, 1\} \quad , \qquad \text{за све } j = 1, ..., n \tag{5.8}$$

$$x_i \in \{0, 1\} \quad , \qquad \text{за све } i = 1, ..., m \tag{5.9}$$

Лако се види да ILP формулација располаже са $m + n$ бинарних промјенљивих и укупно $2 \cdot n$ ограничења.

5.5 Метод промјенљивих околина за MSSP

У овом одјељку описује се метод промјенљивих околина примијењен и развијен за рјешавање MSSP. Резултати презентовани у овом одјељку објављени су у [78].

5.5.1 Иницијализација и функција циља

Нека је S коначан скуп и $|S| = m$. Партиција (P_1, P_2) скупа S, која је једно рјешење, представља се као бинарни низ x дужине m. Елементи низа одговарају елементима скупа S и означавају ком од два подскупа дати елемент припада. Формално, $i \in P_1$ ако $x_i = 1$ и $i \in P_2$ ако је $x_i = 0$. Полазно рјешење се бира на случајан начин.

Сваки скуп колекције C је представљен једним низом који садржи одговарајуће елементе. За дату партицију (P_1, P_2), и за сваки елемент $S_j \in C$, вриједност y_j се рачуна помоћу формуле

$$y_j = \begin{cases} 1, & S_j \text{ је подијељен} \\ 0, & S_j \text{ није подијељен} \end{cases}$$

За дато рјешење, алгоритам рачуна вриједност функције циља помоћу фор-

муле

$$obj(P_1, P_2) = \sum_{j=1}^{n} y_j. \tag{5.10}$$

За свако $j \in \{1, .., n\}$ временска сложеност за одређивање вриједности y_i је $O(m)$, тако да је укупна временска сложеност за рачунање функције циља $O(mn)$.

5.5.2 Околине и процедура размрдавања

У процедури размрдавања се креира ново рјешење x', ($x' \in N_k(x)$) које је засновано на тренутно најбољем рјешењу x. Да би дефинисала k-та околина, у алгоритму се на случајан начин бира неких k елемената из S. Сваком изабраном елементу мијења се припадајућа компонента, тј. сви изабрани елементи који припадају P_1 се пребацују у P_2 и обрнуто. Формално, k-та околина вектора x се може записати као $N_k(x) = \{x' : \{i_1, i_2, ..., i_k\} \subset \{1, 2, ..., |S|\}\ x'_{i_j} = 1 - x_{i_j}\}$.

У прописаном алгоритму, k_{min} је постављено на вриједност 2. Да би се задовољио теоријски услов који прописује да величина сваке наредне околине треба да буде већа од претходне, k_{max} се дефинише као $k_{max} = \min\{20, |S|/2\}$. Пошто је величина k - те околине $\binom{|S|}{k}$, за $k < |S|/2$ слиједи да је $\binom{|S|}{k-1} < \binom{|S|}{k}$, тј. $|N_{k-1}(x)| < |N_k(x)|$ и услов је задовољен. За веће инстанце, експерименти су показали да је довољно узети вриједност $k_{max} = 20$. Лако се види да се процедура размрдавања састоји од k корака временске сложености $O(|S|)$, тако да је укупна временска сложеност процедуре размрдавања $O(k_{max} \cdot |S|)$.

5.5.3 Локално претраживање

За рјешење x', које је добијено у процедури размрдавања, позива се локално претраживање. У свакој итерацији локалне претраге алгоритам покушава да поправи рјешење тако што мијења припадајуће компоненте за парове елемената из S. На примјер, ако $a \in P_1$ и $b \in P_2$, тада након размјене имамо $a \in P_2$ и $b \in P_1$. Означимо ново рјешење које је добијено на овај начин са x''. У случају да је x'' боље од рјешења x', тада x' постаје једнако x''. Локална претрага престаје са радом након првог таквог побољшања. У другом случају, ако није дошло до побољшања (x'' није боље од x'), локална претрага наставља са сљедећим паром елемената.

Када се локална претрага заврши, анализирају се сљедећа три случаја:

a. Ако функција циља за рјешење x' има мању вриједност у односу на рјешење x, тада се претрага наставља са истим рјешењем x и наредном околином.

б. У случају да функција циља за рјешење x' има већу вриједност у односу на рјешење x, тада тренутно најбоље рјешење x добија вријеност x'.

в. У случају да функције циља два рјешења x и x' имају исту вриједност, тада се поставља $x = x'$ са вјероватноћом p_{move} и алгоритам наставља претрагу са истом околином. У другом случају, претрага се понавља са истим рјешењем x и наредном околином, са вјероватноћом $1 - p_{move}$. У презентованом алгоритму, p_{move} је постављена на вриједност 0.4.

Експерименти су показали да прописани алгоритам ради поуздано чак и за $p_{move} = 0$, али се у том случају алгоритму не даје никаква шанса да се пребаци са једног на друго рјешење са истом вриједношћу функције циља. Са друге стране, иако се теоријски p_{move} бира из интервала $[0, 1]$, вриједности за p_{move} блиске броју 1 могу проузроковати циклирање по рјешењима са истом вриједношћу функције циља. Стога је у већини случајева најбоље p_{move} подесити на вриједност која је између двије крајње вриједности.

Лако се види да локална претрага формира парове елемената и да је укупан број тих парова ограничен са $O(m^2)$. Замјена компоненти се врши у времену $O(1)$, а рачунање функције циља у $O(mn)$. Стога је укупна временска сложеност локалне претраге $O(m^3 n)$.

По завршетку разматрања свих околина, алгоритам поново почиње са првом околином све док се не испутни кретеријум за завршетак алгоритма. У нашем случају, то је задати максималан број итерација.

5.6 Експериментални резултати

У овом одјељку су приказани експериментални резултати добијени примјеном прописаног VNS метода. Имплементација алгоритма је извршена у C програмском језику. Сва тестирања су урађенана на Intel Core 2 Quad Q9400 @2.66 GHz са 8 GB RAM рачунару. Тестови су извршени на два скупа инстанци: *minimum hitting set* инстанцама (MHS) из [29] и *Steiner triple systems* инстанцама (STS), које су описане у [39]. Прва класа инстанци (MHS) садржи укупно десет инстанци, са различитим бројем елемената ($m = 50,\ 100,\ 250,\ 500$) и различитим бројем подскупова ($n = 100,\ 10000,\ 50000$). Класа STS инстанци садржи седам инстанци: најмања има 9 елемената и 12 подскупова, а највећа 243 елемента и 9801 подскуп. STS инстанце су означене као теже, јер ILP рјешавач није могао одредити оптимално рјешење за средње и велике инстанце, које садрже 27 и више елемената [70]. За сваку инстанце VNS је покретан 20 пута, а свако покретање се завршава након 100 итерација.

Табела 5.7: VNS резултати за MSSP на MHS инстанцама

m	n	o/b	VNS	t_{VNS}	agap	σ
50	1000	1000	opt	0.17835	0.0	0.00
50	10000	10000	opt	2.71175	0.0	0.00
100	1000	1000	opt	0.2969	0.0	0.00
100	10000	10000	opt	3.7231	0.0	0.00
100	50000	50000	opt	124.589	0.0	0.00
250	1000	1000	opt	0.80325	0.0	0.00
250	10000	10000	opt	12.3331	0.0	0.00
500	1000	1000	opt	1.64985	0.0	0.00
500	10000	10000	opt	20.11155	0.0	0.00
500	50000	50000	opt	225.2309	0.0	0.00

У табелама 5.7 и 5.8 су приказани резултати тестирања прописаног VNS алгоритма на MHS и STS инстанцама. Обје табеле су организоване на исти начин: прве двије колоне садрже информације о броју елемената (m) и броју подскупова (n). Наредна колона (o/b) садржи оптимално, односно најбоље рјешење, означено знаком '*', ако није познато да ли је оно и оптимално. Најбоља вриједност добијена VNS алгоритмом је приказано у колони VNS, са ознаком *opt* у случајевима када VNS достиже унапријед познато оптимално рјешење, односно са знаком *best* ако VNS достиже до сада најбоље познато рјешење.

Очигледно је да било које рјешење не може бити веће од укупног броја подскупова. Стога, за прву класу инстанци (MHS инстанце), јасно је да су достигнуте вриједности оптималне. За другу класу инстанце (STS инстанце), оптимални резултати за прве двије инстанце су доказани употребом егзактне методе у [70]. За преостале STS инстанце не може се гарантовати да су достигнута најбоља рјешења оптимална. Просјечно вријеме потребно да VNS достигне своје најбоље рјешење приказано је у колони са ознаком t. Квалитет рјешења у 20 извршења ($i = 1, 2, ..., 20$) је приказана преко колона *agap* и σ, гдје се просјечна релативна грешка (*agap*) рачуна као $agap = \frac{1}{20}\sum_{i=1}^{20} gap_i$, гдје је $gap_i = 100 * \frac{Opt.sol - sol_i}{Opt.sol}$ у односу на оптимално рјешење $Opt.sol$ ако је оно познато, односно у односу на најбоље познато рјешење $Best.sol$, i.e. $gap_i = 100 * \frac{Best.sol - sol_i}{Best.sol}$ у случају када оптимално рјешење није познато, (sol_i представља рјешење добијено у i-том извршењу). Стандардна девијација је рачуната по формули $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$.

Из табела 5.7 и 5.8 се јасно види да VNS достиже све познате оптималне и најбоље резултате. Такође, ови резултати се достижу у свих 20 покретања, што

Табела 5.8: VNS резултати за MSSP на STS инстанцама

m	n	o/b	VNS	t_{VNS}	agap	σ
9	12	10	opt	0.0009	0.0	0.00
15	35	28	opt	0.002	0.0	0.00
27	117	91*	best	0.00915	0.0	0.00
45	330	253*	best	0.04215	0.0	0.00
81	1080	820*	best	0.27625	0.0	0.00
135	3015	2278*	best	1.2607	0.0	0.00
243	9801	7381*	best	11.84055	0.0	0.00

указује на добру поузданост алгоритма. Вријеме извршења алгоритма је мало чак и за највеће MHS инстанце са 100 и 500 елемената и 50 000 подскупова. За ове инстанце, вријеме извршавања иде до 124 и 225 секунди, респективно. За све STS инстанце, VNS лако достиже сва оптимална/најбоља рјешења у кратком времену. Табеле 5.9 и 5.10 садрже упоредне резултате добијене на MHS и STS инстанцама примјеном различитих метода из [70, 67], те прописаног VNS метода. Колоне табела су организоване на сљедећи начин:

- прве двије колоне садрже m и n;
- трећа колона садржи оптималну вриједност ако је позната, односно најбољу познату вриједност са ознаком '*';
- наредне двије колоне садрже вријеме потребно CPLEX рјешавачу да пронађе оптимално рјешење, уз ознаку N/A ако рјешење није пронађено усљед временских и меморијских ограничења [70];
- наредне двије колоне садрже вриједност и вријеме извршења GA [70];
- наредне двије колоне садрже вриједност и вријеме извршења EM [67];
- посљедње двије колоне садрже вриједност и вријеме извршења VNS;

У свим случајевима, ако је алгоритам достигао оптимално (респективно најбоље) рјешење, кориштена је ознака opt(best) умјесто оптималне(најбоље) вриједности.

Подаци приказани у табелама 5.9 и 5.10 указују на високе перформансе прописаног VNS алгоритма. Као и GA, VNS достиже сва позната оптимална/најбоља рјешења, док EM не успијева да пронађе најбоље рјешење за MHS инстанцу са 10000 елемената и 50000 подскупова. Поређењем времена извршавања датих алгоритама, види се да је за већину инстанци GA 5-10 пута спорији од преостале двије хеуристике. За већину MHS инстанци VNS је до два пута бржи од EM,

Табела 5.9: Компаративни резултати за MHS испанце

m	n	o/b	CPL	t_{CPL}	GA	t_{GA}	EM	t_{EM}	VNS	t_{VNS}
50	1000	1000	opt	0.078	opt	2.582	opt	0.158	opt	0.17835
50	10000	10000	opt	3.265	opt	60.039	opt	3.212	opt	2.71175
100	1000	1000	opt	0.188	opt	4.67	opt	0.334	opt	0.2969
100	10000	10000	opt	8.297	opt	168.603	opt	10.593	opt	3.7231
100	50000	50000	opt	155.2	opt	683.147	49998	216.316	opt	124.589
250	1000	1000	opt	0.219	opt	8.626	opt	1.062	opt	0.80325
250	10000	10000	opt	30.063	opt	336.894	opt	45.393	opt	12.3331
500	1000	1000	opt	0.500	opt	13.325	opt	2.336	opt	1.64985
500	10000	10000	opt	106.09	opt	437.909	opt	94.473	opt	20.11155
500	50000	50000	N/A	-	opt	2086.517	opt	486.124	opt	225.2309

Табела 5.10: Компаративни резултати за STS испанце

m	n	o/b	CPL	t_{CPL}	GA	t_{GA}	EM	t_{EM}	VNS	t_{VNS}
9	12	10	opt	0.031	opt	0.193	opt	0.001	opt	0.0009
15	35	28	opt	0.343	opt	0.233	opt	0.003	opt	0.002
27	117	91*	N/A	-	best	0.382	best	0.005	best	0.00915
45	330	253*	N/A	-	best	0.914	best	0.030	best	0.04215
81	1080	820*	N/A	-	best	2.893	best	0.173	best	0.27625
135	3015	2278*	N/A	-	best	7.858	best	0.905	best	1.2607
243	9801	7381*	N/A	-	best	65.409	best	14.953	best	11.84055

(за инстанце са 10000 подскупова чак и 3-5 пута бржи). На STS инстанцама времена извршења VNS и EM алгоритама су слична. Иако су скупови инстанци релативно мали, једино праведно поређење између различитих метода може бити урађено само употребом истних тестних података. Експериментални резултати указују на то да VNS постиже боље перформансе у односу на GA из [70] и EM из [67]. Постигнути резултати дају поуздану претпоставку да се прописани VNS може ефикасно користити у рјешавању MSSP.

5.7 Завршна разматрања за MSSP

У овом одјељку је разматран проблем маскималне подјеле скупа. Разматране су неке практичне примјене овог проблема и приказана је метода промјенљивих околина за његово рјешавање.

MSSP је од практичног значаја и у образовању. Ако се разматра методолошки

приступ подјеле курса на два дијела (на примјер на два семестра), на начин да што више тематских цјелина буде присутно у оба дијела, онда математичка поставка овог приступа одговара поставци MSSP. У овој глави, приказана је подјела курса Увод у рачунарство и анализирана тренутна подјела. Анализом се дошло до сазнања да тренутна подјела курса у значајној мјери "прати" прописани приступ, док се минорним размјенама лекција између семестара, тај распоред може учинити и оптималним. Ова чињеница указује на претпоставку да су аутори Плана и програма, мотивишући се овим или неким сличним приступом, у значајној мјери водили рачуна да се фундаменталне цјелине разматрају у оба семестра, како би се очувао континуитет у изучавању. Истраживање приказано у овој глави пружа могућност додатног унапређења посматраног плана и програма.

VNS метод користи стандардне процедуре: размрдавање и локално претраживање. У оквиру процедуре размрдавања, алгоритам формира систем околина, које су засноване на промјени компоненти за растући број елемената. Овај приступ усмјерава претрагу у добром правцу и омогућава ефикасну примјену локалног претраживања на тренутно рјешење. Да би се у оквиру локалне претраге унаприједило тренутно рјешење, алгоритам размјењује компоненте за парове елемената, покушавајући да се пребаци у боље рјешење из локалне околине.

Према експерименталним резултатима, примијењени VNS приступ се показује успјешним. VNS достиже сва позната оптимална, односно најбоља рјешења у кратком времену. Експерименти указују да VNS алгоритам показује боље перформансе од других метода кориштених за рјешавање овог проблема.

Могући правци даљих истраживања укључују могућност примјене поменутог приступа на подјелу других курсева. Такође, било би интересантно примијенити овај приступ на читав сет курсева (на примјер на цјелокупне студије из области рачунарских наука). Тиме би се број лекција и тематских цјелина значајно повећао, што би оправдало примјену метахеуристичког приступа за рјешавање датог проблема. Са теоријског аспекта, истраживања би могла да укључују комбинацију предложеног VNS приступа са другим хеуристичким или егзактним методама на овај или сличне проблеме.

Глава 6

Закључак

У овом раду су истражени актуелни проблеми комбинаторне оптимизације, као и могућност примјене неких од њих у организовању наставе. Анализиране су и представљене различите методе рјешавања сљедећих проблема: проблем проналажења максималне повезане партиције, проблем проналажења максимално балансиране повезане партиције у графу са q партиција ($q \geq 2$), проблем проналажења подјеле скупа на двије партиције и проблем проналажења p - арне транзитивне редукције у диграфу.

За сваки од ових проблема приказане су метахеуристике за рјешавање: метод промјењивих околина је развијен за сва четири проблема, док је за рјешавање TR_p развијен и генетски алгоритам.

За MBCP развијен је и модел мјешовитог цјелобројног линеарног програмирања, који омогућава проналажење тачног рјешења за инстанце мањих димензија. Приказан је детаљан доказ корекности предложеног модела, те његова примјена за рјешавање поменутог модела употребом савремених софтверских рјешавача.

Метода промјењивих околина за проблеме MBCP, MSSP и TR_p користе бинарно кодирање, а за BCPq цјелобројно кодирање. Генетски алгоритам примијењен на проблем TR_p такође користи бинарно кодирање. У генетском коду алгоритма за TR_p проблем сваки бит одговара једној грани која јесте или није укључена у рјешење.

За VNS алгоритме који се односе на проблеме MBCP и BCPq конструисана је специфична функција циља, која унутар себе користи казнену функцију, омогућавајући проширење претраге и на некоректне јединке. За TR_p проблем функција циља је иста за GA и за VNS и поправља некоректне јединке сукцесивним додавањем грана, на начин да се додавањем сваке гране даје највећа

шанса да новонастало рјешење постане допустиво. За сваки од VNS алгоритама развијени су специфични системи околина, којима се омогућава усмјеравање претраге са једног на (могуће бољи) други минимум, смјештен у некој околини тренутног рјешења.

Развојем ефикасних метода локалног претраживања за проблеме MBCP, MSSP и BCPq значајно се унапређује читав VNS за рјешавање поменутих проблема чиме свеукупна претрага добија на квалитету, уз разумно повећање времена извршења. За TR_p проблем, значајна претрага се одвија у оквиру процедуре размрдавања, те је због побољшања комплетних перформанси самог алгоритма, локална претрага за овај проблем изостављена.

Генетски алгоритам за TR_p проблем користи стандардни генетски оператор укрштања, док је стандардна мутација унапријеђена концептом смрзнутих гена, чиме се за поједине гене даје већи ниво мутације. У овом алгоритму се користи унапријеђени оператор фино градиране турнирске селекције. У оквиру овог генетског алгоритма се користи и кеширање, чиме се значајно убрзава рачунање функције циља.

У раду је разматрана примјена неких од наведених проблема у организацији наставе. Показало се да се се MBCP и MSSP успјешно могу примијенити у подјели курсева, како је то приказано и на конкретним примјерима. Развијене су технике за начине повезивања лекција, као и за одређивање њихових тежина, заснованим на објективним показатељима и субјективним процјенама професора. Тиме је постигнуто да се читав курс представи као повезан тежински граф, што је отворило могућност за примјену алгоритама за рјешавање MBCP на формирани граф. Детаљна анализа овог приступа илустрована је на примјеру курса Увод у теорију бројева.

Придруживањем лекција одговарајућим категоријама (тематским цјелинама) унутар једног курса, креира се фамилија подскупова (тематских цјелина) читавог скупа лекција. Ако претпоставимо да лекције курса треба разбити у два дисјунктна подскупа, тако да што више тематских цјелина буде покривено у оба та подскупа, тада се наведени проблем своди на рјешавање MSSP. У дисертацији је детаљно анализиран овај приступ на примјеру курса Увод у рачунарство.

Из изложених модела употребе MBCP и MSSP у организовању наставе, може се закључити да ова два приступа предлажу прилично различите, чак и супротстављене концепте, те се стога и не могу користити у истим ситуацијама. Подјела курса заснована на MBCP приступу подразумијева балансираност дијелова курса по тежини, уз очување повезаности лекција. Насупрот том концепту, у MSSP приступу се не разматрају тежине лекција и њихова повезаност, већ

се позитиван методички ефекат понављања основа тематских цјелина (свих или бар већине) постиже кроз квалитетну расподјелу лекција по семестрима, имајући у виду припадност лекција одговарајућим тематским цјелинама.

Синтеза методичког и математичког приступа, изложена у овом раду, указују на правце даљег истраживања у циљу побољшања квалитета наставног процеса, употребом и разних других проблема комбинаторне оптимизације, као што су локацијски проблеми, проблеми рутирања, складиштења итд.

6.1 Научни допринос рада

Најважнији резултати који представљају научни допринос овог рада су:

- Формулација мјешовитог цјелобројног програмирања за MBCP.
- Развој различитих система околина од којих је сваки посебно пројектован за дати проблем у оквиру VNS методе, за проблеме MBCP, BCPq, MSSP и TR_p.
- Развој ефикасних метода локалног претраживања у уквиру VNS методе, за проблеме MSSP, MBCP и BCPq.
- Развој специфичног начина кодирања јединки у GA и специфичне репрезентације рјешења за VNS методу за TR_p проблем.
- Развој функције циља за GA и VNS за TR_p проблем у оквиру које се врши корекција јединки у GA и корекција рјешења у VNS методи.
- Примјена проблема MBCP и MSSP у настави, односно приједлог новог приступа организацији курсева употребом метода комбинаторне оптимизације, који омогућава флексибилност и ефикасност у организовању наставе.

Рјешења приказана у овом раду успјешно се могу примијенити на рјешавање описаних, али и сличних проблема. Научно истраживање описано у овом раду даје допринос областима комбинаторне оптимизације и метахеуристика, а са друге стране, особама укљученим у организовање наставног процеса предлаже ефикасан алат за анализу квалитета постојећих и прављење нових планова и програма.

Литература

[1] K. Afshar, M. Ehsan, M. Fotuhi-Firuzabad, and N. Amjady. "Cost-benefit analysis and MILP for optimal reserve capacity determination in power system". *Applied Mathematics and Computation* 196(2) (2008), pp. 752 –761.

[2] A. V. Aho, M. R. Garey, and J. D. Ullman. "The transitive reduction of a directed graph". *SIAM Journal of Computing* 1(2) (1972), pp. 131–137.

[3] R. Albert, B. DasGupta, R. Dondi, S. Kachalo, E. Sontag, A. Zelikovsky, and K. Westbrooks. "A novel method for signal transduction network inference from indirect experimental evidence". *Journal of Computational Biology* 14 (2007), pp. 927–949.

[4] R. Albert, B. DasGupta, R. Dondi, and E. D. Sontag. "Inferring (biological) signal transduction networks via transitive reductions of directed graphs". *Algorithmica* 51 (2008), pp. 129–159.

[5] G. Andersson and L. Engebretsen. "Better approximation algorithms for Set Splitting and Not-All-Equal Sat". *Information Processing Letters* 65(6) (1998), pp. 305 –311.

[6] V. Assunção T.and Furtado. "A Heuristic Method for Balanced Graph Partitioning: An Application for the Demarcation of Preventive Police Patrol Areas" in *Advances in Artificial Intelligence – IBERAMIA 2008*. Ed. by H. Geffner, R. Prada, I. Machado Alexandre, and N. David. Vol. 5290. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 62–72.

[7] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis, 2000.

[8] R. Berghammer. "A generic program for minimal subsets with applications" in *Proceedings of the 12th international conference on Logic based program synthesis and transformation*. Lecture Notes in Computer Science 2664, Springer Verlag, 2003, pp. 144–157.

[9] P. Berman, B. DasGupta, and M. Karpinski. "Approximating transitive reductions for directed networks" in *Proceedings of the 11th International Symposium on Algorithms and Data Structures Lecture Notes in Computer Science 5664*. 2009, pp. 74–85.

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

[11] B. Bozkaya, E. Erkut, and G. Laporte. "A tabu search heuristic and adaptive memory procedure for political districting". *European Journal of Operational Research* 144(1) (2003), pp. 12 –26.

[12] M. Carter and G. Laporte. "Recent developments in practical course timetabling" in *Practice and Theory of Automated Timetabling II*. Ed. by E. Burke and M. Carter. Vol. 1408. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 3–19.

[13] T. Chang and Y. Chen. "Cooperative learning in E-learning: A peer assessment of student-centered using consistent fuzzy preference". *Expert Systems with Applications* 36(4) (2009), pp. 8342 –8349.

[14] T. Chang, P. Cheng, and Y. Chen. "A teacher evaluation system based on association rule and certainty factor inference: how to judge a course is successful or not?" in *International Conferenceone-Commerce, e-Administration, e-Society, e-Education, and e-Technology*. 2011, pp. 18–20.

[15] T. Chang and Y. Ke. "A personalized e-course composition based on a genetic algorithm with forcing legality in an adaptive learning system". *Journal of Network and Computer Applications* 36(1) (2013), pp. 533 –542.

[16] W. Chang, M. Guo, and M. Ho. "Towards solution of the set-splitting problem on gel-based DNA computing". *Future Generation Computer Systems* 20(5) (2004), pp. 875 –885.

[17] Y. Chang, Y. Huang, and C. Chu. "B2 model: A browsing behavior model based on High-Level Petri Nets to generate behavioral patterns for e-learning". *Expert Systems with Applications* 36(10) (2009), pp. 12423 –12440.

[18] Y. Chang, W. Kao, C. Chu, and C. Chiu. "A learning style classification mechanism for e-learning". *Computers & Education* 53(2) (2009), pp. 273 – 285.

[19] D. Charsky and W. Ressler. “Games are made for fun: Lessons on the effects of concept maps in the classroom use of computer games”. *Computers & Education* 56(3) (2011), pp. 604 –615.

[20] F. Chataigner, L. Salgado, and Y.Wakabayashi. “Approximation and inaproximability results on balanced connected partitions of graphs”. *Discrete Mathematics & Theoretical Computer Science* 9 (2007), pp. 177 –192.

[21] J. Chen and S. Lu. “Improved Algorithms for Weighted and Unweighted Set Splitting Problems” in *Computing and Combinatorics*. Ed. by G. Lin. Vol. 4598. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 537–547.

[22] J. Chen and S. Lu. “Improved Parameterized Set Splitting Algorithms: A Probabilistic Approach”. *Algorithmica* 54 (4 2009), pp. 472–489.

[23] E. Cheng, S. Kruk, and M. Lipman. “Flow Formulations for the Student Scheduling Problem” in *Practice and Theory of Automated Timetabling IV*. Ed. by E. Burke and P. Causmaecker. Vol. 2740. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 299–309.

[24] S. Cheng, Y. Lin, and Y. Huang. “Dynamic question generation system for web-based testing using particle swarm optimization”. *Expert Systems with Applications* 36(1) (2009), pp. 616 –624.

[25] J. Chlebikova. “Approximating the maximally balanced connected partition problem in graphs”. *Information Processing Letters* 60(5) (1996), pp. 225 – 230.

[26] *Computer Science Curriculum 2008: An Interim Revision of CS 2001 Report from the Interim Review Task Force*. Tech. rep. ACM/IEEE-CS Joint Interim Review Task Force., 2008.

[27] *CPLEX solver, IBM company.*
www.ibm.com/software/integration/optimization/cplex-optimizer.

[28] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenović. “Cooperative Parallel Variable Neighborhood Search for the p-Median”. *Journal of Heuristics* 10 (2004), pp. 293–314.

[29] V. Cutello and G. Nicosia. “A Clonal Selection Algorithm for Coloring, Hitting Set and Satisfiability Problems” in *Neural Nets*. Ed. by B. Apolloni, M. Marinaro, G. Nicosia, and R. Tagliaferri. Vol. 3931. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 324–337.

[30] L. Davis and L. Ritter. “Schedule optimization with probabilistic search” in *Proceedings of the 3rd IEEE Conference on Artifficial Intelligence Applications*. IEEE, 1987, pp. 231–236.

[31] F. Dehne, M. Fellows, and F. Rosamond. “An FPT Algorithm for Set Splitting” in *Graph-Theoretic Concepts in Computer Science*. Ed. by H. Bodlaender. Vol. 2880. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 180–191.

[32] F. Dehne, M. Fellows, F. Rosamond, and P. Shaw. “Greedy Localization, Iterative Compression, and Modeled Crown Reductions: New FPT Techniques, an Improved Algorithm for Set Splitting, and a Novel 2k Kernelization for Vertex Cover” in *Parameterized and Exact Computation*. Ed. by R. Downey, M. Fellows, and F. Dehne. Vol. 3162. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 271–280.

[33] B. Djurić, J. Kratica, D. Tošić, and V. Filipovič. “Solving the Maximally Balanced Connected Partition Problem in Graphs by Using Genetic Algorithm”. *Computing and Informatics* 27(3) (2008), pp. 341 –354.

[34] V. Dubois and C. Bothorel. “Transitive reduction for social network analysis and visualization” in *IEE/WIC/ACM International Conference on Web Intelligence*. 2005, pp. 128–131.

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

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

[37] V. Filipović, J. Kratica, D. Tošić, and I. Ljubić. “Fine grained tournament selection for the simple plant location problem” in *Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5*. 2000, pp. 152–158.

[38] G. N. Frederickson and J. Jàjà. “Approximation algorithms for several graph augmentation problems”. *SIAM Journal of Computing* 10(2) (1981), pp. 270–283.

[39] D. Fulkerson, G. Nemhauser, and L. Trotter. “Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems” in *Approaches to Integer Programming*. Ed. by M. Balinski. Vol. 2. Mathematical Programming Studies. Springer Berlin Heidelberg, 1974, pp. 72–81.

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

[41] M. J. Geiger and W. Wenger. "On the assignment of students to topics: A Variable Neighborhood Search approach". *Socio-Economic Planning Sciences* 44(1) (2010), pp. 25 –34.

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

[43] A. Gunawan and K. M. Ng. "Solving the Teacher Assignment Problem by Two Metaheuristics". *International Journal of Information and Management Sciences* 22 (2011), pp. 73–86.

[44] A. Gunawan, H. L. Ong, and K. M. Ng. "A genetic algorithm for the teacher assignment problem". *International Journal of Information and Management Sciences* 19(1) (2008), pp. 1–16.

[45] I. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2012.

[46] V. Guruswami. "Inapproximability Results for Set Splitting and Satisfiability Problems with No Mixed Clauses" in *Approximation Algorithms for Combinatorial Optimization*. Ed. by K. Jansen and S. Khuller. Vol. 1913. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 155–166.

[47] P. Hansen, N. Mladenović, and J. A. Moreno-Pérez. "Variable neighbourhood search: methods and applications (invited survey)". *4OR: A Quarterly Journal of Operations Research* 6 (2008), pp. 319–360.

[48] P. Hansen, N. Mladenović, and J. A. Moreno-Pérez. "Variable neighbourhood search: algorithms and applications". *Annals of Operations Research* 175 (2010), pp. 367–407.

[49] P. Hansen, N. Mladenović, and D. Pérez-Brito. "Variable Neighborhood Decomposition Search". *Journal of Heuristics* 7(4) (2001), pp. 335–350.

[50] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. USA: Oxford University Press, 1979.

[51] J. Holland. *Adaption in Natural and Artificial Systems*. Ed. by A. Arbor. University of Michigan Press, 1975.

[52] S. Hsieh, Y. Jang, G. Hwang, and N. Chen. "Effects of teaching and learning styles on students' reflection levels for ubiquitous learning". *Computers & Education* 57(1) (2011), pp. 1194 –1201.

[53] T. Huang, Y. Huang, and S. Cheng. "Automatic and interactive e-Learning auxiliary material generation utilizing particle swarm optimization". *Expert Systems with Applications* 35(4) (2008), pp. 2113 –2122.

[54] Y. Huang, Y. Lin, and S. Cheng. "An adaptive testing system for supporting versatile educational assessment". *Computers & Education* 52(1) (2009), pp. 53 –67.

[55] G. Hwang, B. Lin, and T. Lin. "An effective approach for test-sheet composition with large-scale item banks". *Computers & Education* 46(2) (2006), pp. 122 –139.

[56] G. Hwang, P. Yin, and S. Yeh. "A tabu search approach to generating test sheets for multiple assessment criteria". *IEEE Transactions on Education* 49(1) (2006), pp. 88–97.

[57] G. Hwang, P. Yin, T. Wang, J. Tseng, and G. Hwang. "An enhanced genetic approach to optimizing auto-reply accuracy of an e-learning system". *Computers & Education* 51(1) (2008), pp. 337 –353.

[58] T. Ito, X. Zhou, and T. Nishizeki. "Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size". *Journal of Discrete Algorithms* 4(1) (2006), pp. 142 –154.

[59] T. Ito, T. Uno, X. Zhou, and T. Nishizeki. "Partitioning a Weighted Tree to Subtrees of Almost Uniform Size" in *Algorithms and Computation*. Ed. by S.-H. Hong, H. Nagamochi, and T. Fukunaga. Vol. 5369. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 196–207.

[60] S. Kachalo, R. Zhang, E. Sontag, R. Albert, and B. DasGupta. "NET-SYNTHESIS: a software for synthesis, inference and simplification of signal transduction networks". *Bioinformatics* 24(2) (2008), pp. 293–295.

[61] P. Kelsen and V. Ramachandran. *The complexity of finding minimal spanning subgraphs*. Tech. rep. CS-TR-91-17, 1991.

[62] S. Khuller, B. Raghavachari, and N. Young. "Approximating the minimum equivalent digraph". *SIAM Journal on Computing* 24(4) (1995), pp. 859–872.

[63] S. Khuller, B. Raghavachari, and N. Young. "On strongly connected digraphs with bounded cycle length". *Discrete Applied Mathematics* 69 (1996), pp. 281–289.

[64] S. Khuller, B. Raghavachari, and A. Zhu. "A uniform framework for approximating weighted connectivity problems" in *19th Annual ACM-SIAM Symposium on Discrete Algorithms*. 1999, pp. 937–938.

[65] J. Kojić. "Integer linear programming model for multidimensional two-way number partitioning problem". *Computers and Mathematics with Applications* 60(8) (2010), pp. 2302 –2308.

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

[67] J. Kratica. "An Electromagnetism-like method for the maximum set splitting problem". *Yugoslav Journal of Operations Research* 23(1) (2013).

[68] J. Kratica, Z. Stanimirović, D. Tošić, and V. Filipović. "Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem". *European Journal of Operational Research* 182(1) (2007), pp. 15–28.

[69] G. Laporte and S. Desroches. "The problem of assigning students to course sections in a large engineering school". *Computers & Operations Research* 13(4) (1986), pp. 387 –394.

[70] B. Lazović, M. Marić, V. Filipović, and A. Savić. "An integer linear programming formulation and genetic algorithm for the maximum set splitting problem". *Publications de l'Institut Mathematique* 92(106) (2012), pp. 25–34.

[71] C. Lee, C. Huang, and C. Lin. "Test-Sheet Composition Using Immune Algorithm for E-Learning Application" in *New Trends in Applied Artificial Intelligence*. Ed. by H. Okuno and M. Ali. Vol. 4570. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 823–833.

[72] R. Lewis. "A survey of metaheuristic-based techniques for University Timetabling problems". *OR Spectrum* 30 (1 2008), pp. 167–190.

[73] S. Li, S. M. Assmann, and R. Albert. "Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling". *PLoS Biology* 4(10) (2006), pp. 1732–1748.

[74] D. Lokshtanov and C. Sloper. "Fixed parameter set splitting, linear kernel and improved running time" in *ACiD*. 2005, pp. 105–113.

[75] D. Lokshtanov and S. Saurabh. "Even Faster Algorithm for Set Splitting!" in *Parameterized and Exact Computation*. Ed. by J. Chen and F. Fomin. Vol. 5917. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 288–299.

[76] L. Lovász. "Coverings and colorings of hypergraphs" in *Proc. 4th Southeastern Conf. on Comb.* Springer Berlin Heidelberg, 1973, pp. 3–12.

[77] M. Lucertini, Y. Perl, and B. Simeone. "Most uniform path partitioning and its use in image processing". *Discrete Applied Mathematics* 42(2-3) (1993), pp. 227 –256.

[78] D. Matić. "A Variable Neighborhood Search Approach for Solving the Maximum Set Splitting Problem". *Serdica Journal of Computing* 6(4) (2012), pp. 369–384.

[79] D. Matić. "A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem" (submitted).

[80] D. Matić and M. Božić. "Rješavanje nekih organizacionih problema u nastavi primjenom pronalaženja maksimalno balansirane povezane particije u grafu" in *Proc. of Simposium Mathematics and Application*. Faculty of Mathematics, University of Belgrade, 2012.

[81] D. Matić and M. Božić. "Maximally Balanced Connected Partition Problem in Graphs: Application in Education". *The Teaching of Mathematics* 15(2) (2012), pp. 121–132.

[82] P. Maya, K. Sorensen, and P. Goos. "A metaheuristic for a teaching assistant assignment-routing problem". *Computers & Operations Research* 39(2) (2012), pp. 249 –258.

[83] V. Mićić and Z. Kadelburg. *Uvod u teoriju brojeva*. Beograd: Društvo matematičara Srbije, 1989.

[84] M. Milanović, D. Matić, A. Savić, and J. Kratica. "Two Metaheuristics Approches to Solving the p-ary Transitive Reduction Problem". *Applied and Computational Mathematics* 10(2) (2011), pp. 294–308.

[85] S. MirHassani and F. Habibi. "Solution approaches to the course timetabling problem". *Artificial Intelligence Review* (2011), pp. 1–17.

[86] I. Miyaji, K. Ohno, and H. Mine. "Solution method for partitioning students into groups". *European Journal of Operational Research* 33(1) (1988), pp. 82 –90.

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

[88] D. M. Moyles and G. L. Thompson. "An algorithm for finding the minimum equivalent digraph". *SIAM Journal on Computing* 24 (1995), pp. 859–872.

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

[90] C. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, 1982.

[91] S. Petrovic and E. Burke. "University timetabling" in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter 45*. Chapman Hall/CRC Press, 2004.

[92] F. Rutherford and A. Ahlgren. *Science for All Americans*. Oxford University Press, USA, 1991.

[93] G. Sabin and G. Winter. "The impact of automated timetabling on universities - a case study". *Journal of the Operational Research Society* 37(7) (1986), pp. 689 –693.

[94] A. Savić, J. Kratica, M. Milanović, and D. Dugošija. "A mixed integer linear programming formulation of the maximum betweenness problem". *European Journal of Operational Research* 206(3) (2010), pp. 522 –527.

[95] J. Saxtorph and J. Clausen. "Finding minimum equivalent digraphs" in *Sixth Workshop on Applied/Advanced Research in Combinatorial Optimization*. Copenhagen, Denmark, 2000.

[96] A. Schaerf. "A Survey of Automated Timetabling". *Artificial Intelligence Review* 13(2) (1999), pp. 87–127.

[97] K. Simon. "Finding a minimal transitive reduction in a strongly connected digraph within linear time" in *Proc. 15th International Workshop WG'89*. Lecture Notes in Computer Science 411, Springer Verlag, 1989, pp. 245–259.

[98] I. Slama, B. Jouaber, and D. Zeghlache. "Topology Control and Routing in Large Scale Wireless Sensor Networks". *Computing and Informatics* 2 (2010), pp. 584 –598.

[99] Z. Stanimirović. *Genetski algoritmi za rešavanje nekih NP-teških hab lokacijskih problema, doktorska disertacija*. 2007.

[100] A. Vetta. "Approximating the minimum strongly connected subgraph via a matching lower bound" in *12th ACM-SIAM Symposium on Discrete Algorithms*. 2001, pp. 417–426.

[101] T. Wang. "Developing Web-based assessment strategies for facilitating junior high school students to perform self-regulated learning in an e-Learning environment". *Computers & Education* 57(2) (2011), pp. 1801 –1812.

[102] Y. Wang. "An application of genetic algorithm methods for teacher assignment problems". *Expert Systems with Applications* 22(4) (2002), pp. 295 – 302.

[103] P. Yin, K. Chang, G. Hwang, G. Hwang, and Y. Chan. "A Particle Swarm Optimization Approach to Composing Serial Test Sheets for Multiple Assessment Criteria". *Educational Technology & Society* 9(3) (2006), pp. 3–15.

[104] F. Yu, R. Katz, and T. Lakshman. "Efficient Multi-Match Packet Classification for Network Security Applications". *IEEE Journal on Selected Areas in Communications* 24(2) (2006).

[105] F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz. *SSA: A Power and Memory Efficient Scheme to Multi-Match Packet Classification.* Tech. rep. UCB/CSD-05-1388. EECS Department, University of California, Berkeley, 2005.

[106] J. Zhang, Y. Ye, and Q. Han. "Improved approximations for max set splitting and max NAE SAT". *Discrete Applied Mathematics* 142(1-3) (2004), pp. 133 –149.

Биографија

Драган Матић је рођен 23. августа 1977. године у Сремској Митровици. Основну школу и Гимназију завршио је у Бијељини (Босна и Херцеговина).

Дипломирао је 2001. године на Природно-математичком факултету у Новом Саду са просјечном оцјеном 9,46, а на истом Факултету 2009. године завршава и постдипломске-мастер студије, одбранивши мастер рад под називом Генетички алгоритми и музика.

Докторске студије на Катедри за методику Математичког факултета у Београду је уписао 2008. године.

Од октобра 2001. године ради на Природно математичком факултету у Бањалуци, прво у звању асистента (2001-2009), те у звању вишег асистента (од краја 2009. године до данас), на Катедри за информатику (ужа научна област Програмски језици). Током асистентског стажа држао је вјежбе из већег броја информатичких и неких математичких предмета: Основе програмирања, Основе рачунарских система, Објектно-оријентисано програмирање, Анализа и дизајн алгоритама, Математичко програмирање, Методика наставе рачунарства, Теорија аутомата и формалних језика, као и основне курсеве из основа информатике за студенте хемије, географије и биологије.

У периоду од 2006. до 2012. ангажован је и као консултант на развоју и имплементацији система за електронско учење, као и на другим домаћим и међународним пројектима у Босни и Херцеговини из области информационих технологија.

Прилог 1.

Изјава о ауторству

Потписани Драган Матић

број индекса 2024/2008

Изјављујем

да је докторска дисертација под насловом

Рјешавање неких проблема у настави примјеном метода комбинаторне оптимизације

- 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 докторанда

У Београду, 20.5.2013.

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора: Драган Матић

Број индекса __2024/2008_

Студијски програм Математика

Наслов рада: Рјешавање неких проблема у настави примјеном метода комбинаторне оптимизације

Ментор др Владимир Филиповић, доцент

Потписани Драган Матић

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

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 Београду.

Потпис докторанда

У Београду, 20.5.2013.

Матић Драган

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Рјешавање неких проблема у настави примјеном метода комбинаторне оптимизације

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 Београду 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

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 20.5.2013.

_____

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.