УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Јована Рађеновић

РЕШАВАЊЕ ПРОБЛЕМА P-ЦЕНТРА СА ПОУЗДАНОМ МРЕЖОМ ПРИМЕНОМ МЕТОДЕ ПРОМЕНЉИВИХ ОКОЛИНА

мастер рад

Београд, 2022.

Ментор:

др Стефан Мишковић, доцент
Универзитет у Београду, Математички факултет

Чланови комисије:

др Мирослав Марић, редовни професор
Универзитет у Београду, Математички факултет

др Нина Радојичић Матић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 30.9.2022.

Породици и пријатељима

Наслов мастер рада: Решавање проблема p-центра са поузданом мрежом применом методе променљивих околина

Резиме: У општем случају, решавање локацијских проблема подразумева одабир локација за успостављање ресурса и алокацију корисника. У пракси, одабир локација представља дугорочну стратегијску одлуку. Може се десити да неки од ресурса који су успостављени, у случају природних катастрофа, протеста, терористичких напада и сличних ванредних ситуација, буду онеспособљени за употребу. Замена или оправка ових ресурса би изискивала велика новчана средства и пуно времена. Стога би у фази моделирања и решавања многих локацијских проблема било пожељно узети у обзир овакве ситуације и направити поуздану мрежу ресурса и корисника. Односно, успоставити ресурсе и повезати их са корисницима, тако да уз евентуалне мале измене, мрежа остане ефикасна и уколико неки од ресурса постане онеспособљен и дође до промена у потражњи корисника и другим улазним подацима. У оквиру овог рада разматран је проблем p-центра са поузданом мрежом (енг. *Reliable p-center facility location problem*). Решавање проблема подразумева иницијално успостављање p ресурса и алокацију корисника, као и накнадну реалокацију корисника у складу са новонасталим сценаријима. Сценарији садрже информације о онеспособљеним ресурсима и осталим измењеним улазним подацима. Како егзактне методе решавају проблем само за мале инстанце, за решавање проблема коришћен је метахеуристички приступ. Прецизније, развијен је алгоритам који представља итеративну варијанту основне методе променљивих околина (енг. *Iterated basic variable neighborhood search* – IBVNS). Такође, уочено је да се из постојећег модела проблема може изоставити једна променљива, чиме је модел додатно упрошћен. На основу нове формулације израђена је имплементација која користи CPLEX решавач. Резултати тестирања IBVNS-а упоређени су са резултатима CPLEX-а и још три егзактне методе из литературе. IBVNS се по сваком критеријуму показао као најефикаснија метода за решавање проблема p-центра са поузданом мрежом.

Кључне речи: локацијски проблеми, p-центар, робусна оптимизација, метахеуристичке методе, метода променљивих околина

Садржај

Глава 1

Увод

1.1 Оптимизациони проблеми

Најопштије речено, под појмом оптимизације подразумева се процес добијања најбољег решења проблема који се решава, при одговарајућим условима. Математички, проблем оптимизације се може дефинисати на следећи начин [18, 23, 43]: Нека је дат скуп S и функција $f : S \to \mathbb{R}$. Нека је $X \subseteq S$ сачињен од елемената из S који задовољавају одређени скуп ограничења. Наћи $x_0 \in X$ за које важи:

$$f(x_0) = \min_{x \in X} f(x) \tag{1.1}$$

Ограничења која дефинишу елементе скупа X су у склопу конкретног проблема исказана у виду једначина и неједначина. Уколико не постоје ограничења, односно уколико је $X = S$ говоримо о проблему *безусловне оптимизације*, у супротном ($X \neq S$) реч је о проблему *условне оптимизације*. Скуп S се најчешће назива *претраживачки простор*, а скуп X *допустив скуп*. Елементи скупа X називају се *допустивим решењима* проблема (1.1), док се елемент x_0 назива *оптималним решењем* задатог проблема. Може постојати више оптималних решења. У највећем броју случајева довољно је наћи једно од њих. Функција f назива се *функција циља*.

Елемент $x^* \in X$ за који важи $f(x^*) \leq f(x)$, за свако $x \in X$, назива се *глобалним минимумом* функције $f|_X$. За елемент $x' \in X$ кажемо да је *локални минимум* функције $f|_X$, ако постоји $\varepsilon > 0$ такво да за свако $x \in X$ за које је $\|x - x'\| \leq \varepsilon$, важи и $f(x') \leq f(x)$. Глобални минимум и локални минимум функције $f|_X$ се у оквиру оптимизационих проблема називају *глобалним оптимумом* и *локалним оптимумом* функције циља f, респективно. Може

се приметити да оптимално решење x_0 проблема (1.1) представља глобални оптимум функције циља f.

Према типу скупа S проблеми оптимизације се могу поделити на *дискретне (комбинаторне)*, *континуалне* и *мешовите*. Уколико је скуп S коначан или пребројиво бесконачан, реч је о проблему дискретне (комбинаторне) оптимизације. У случају континуалне оптимизације, скуп S је непребројив. Мешовити проблеми садрже елементе како дискретне, тако и континуалне оптимизације.

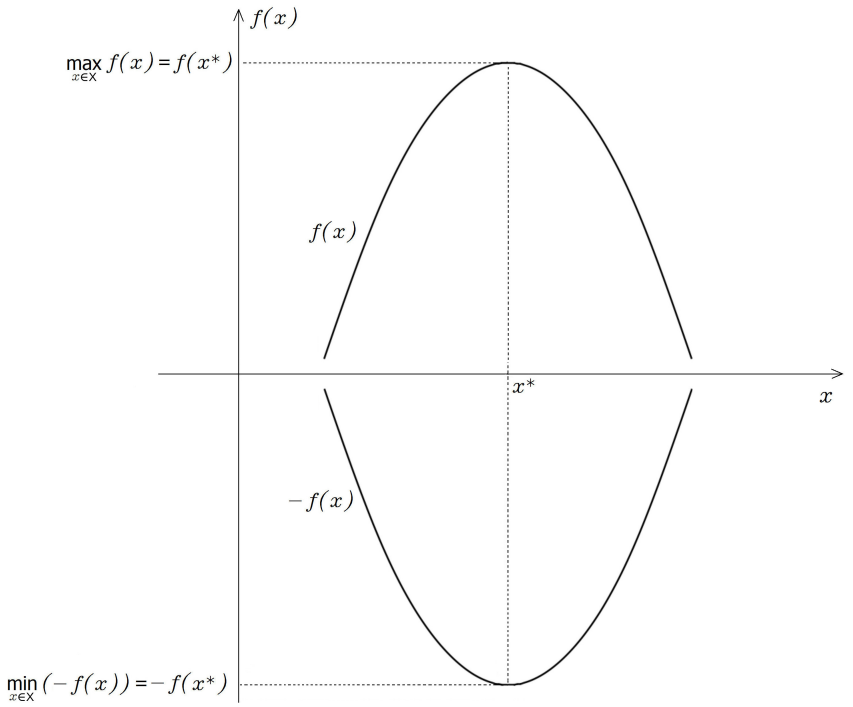
Проблем (1.1) се назива проблемом *минимизације*. Аналогно се може дефинисати проблем *максимизације*: Нека је дат скуп S и функција $f : S \to \mathbb{R}$. Нека је $X \subseteq S$ сачињен од елемената из S који задовољавају одређени скуп ограничења. Наћи $x_0 \in X$ за које важи:

$$f(x_0) = \max_{x \in X} f(x)$$

Међутим, како важи (видети слику 1.1):

$$\max_{x \in X} f(x) = -\min_{x \in X}(-f(x))$$

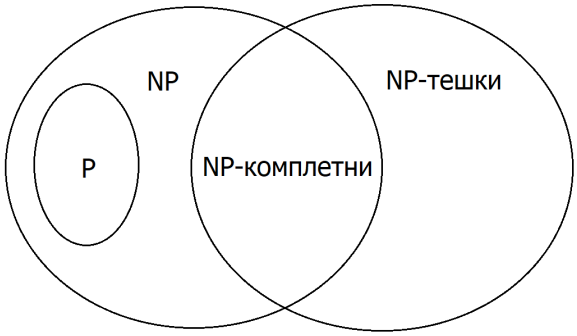
проблем максимизације се увек може свести на проблем минимизације, те је довољно говорити о једном од ова два проблема.



Слика 1.1: Илустративни пример тврђења: $\max_{x \in X} f(x) = -\min_{x \in X}(-f(x))$

Многи проблеми који се срећу у пракси, могу се формулисати као проблеми оптимизације. На пример, одређивање најбоље стратегије за остваривање максималног профита неке фирме, планирање процеса производње ради постизања максималне ефикасности, дизајнирање система цевовода са циљем минимизације трошкова [48], минимизација времена путовања од места A до места B, прављење распореда летова тако да се минимизује трошак, а испуне захтеви путника [36] и многи други проблеми из разних области науке и свакодневног живота (видети на пример [36, 56]).

Уколико за неки проблем постоји алгоритам полиномске сложености који га решава, каже се да тај проблем припада P класи. Проблеми који припадају овој класи, сматрају се релативно лаким за решавање. Ако се у полиномском времену може одредити да ли је нека вредност решење задатог проблема, за тај проблем се каже да припада NP класи. Уколико је проблем такав да се сваки проблем из NP класе може свести на њега, онда је он NP-тежак. Ако је проблем NP-тежак и припада NP класи, онда је тај проблем NP-комплетан. Наведене класе чине класе сложености. Међусобна зависност наведених класа приказана је на слици 1.2. Познато је да је P класа подскуп од NP класе, међутим није познато да ли су ове две класе заправо једнаке. Проблем једнакости P и NP класе је један од најбитнијих отворених проблема и налази се на листи такозваних миленијумских проблема [1]. Листа је доступна на [14]. Више о класама сложености се може наћи у [34, 54, 55]. Већина оптимизационих проблема спада у NP-тешке проблеме.



Слика 1.2: Венов дијаграм класа сложености

[1]Листа миленијумских проблема формирана је 2000. године од стране Математичког института Клеј. Исправно решење за било који проблем са листе доноси новчану награду од милион долара. Занимљиво је да је од седам проблема са листе, до данас решен само један. Проблем „Поенкареова хипотеза", решио је 2003. руски математичар Григориј Перелман. Григориј је одбио милион долара, као и све награде које су уследиле.

У овом мастер раду, решаван је проблем дискретне(комбинаторне) оптимизације, код кога се, при одређеним ограничењима, минимизује нека максимална вредност. Проблеми код којих је циљ минимизација неке максималне вредности називају се *min-max проблемима*. Више о овим проблемима дато је у секцији 1.1.1.

1.1.1 Min-max проблеми

У литератури постоје различите формулације min-max проблема. По узору на [43, 57], а у складу са ознакама коришћеним у секцији 1.1, дефинишемо min-max проблем облика:

$$\min_{x \in X} F(x) \tag{1.2}$$

где је

$$F(x) = \max_{s \in Sc} f^s(x) \tag{1.3}$$

и $f^s : S \to \mathbb{R}$, $s \in Sc$. Ако је скуп S коначан или пребројив, реч је о min-max проблему дискретне оптимизације. Скуп Sc је у општем случају дискретан скуп индекса. Као и у (1.1) $X \subseteq S$ сачињен је од елемената из S који задовољавају одређени скуп ограничења, а елементи скупа X називају се допустивим решењима проблема (1.1). Елемент $x_0 \in X$ за који важи

$$F(x_0) = \min_{x \in X} \max_{s \in Sc} f^s(x)$$

представља оптимално решење проблема (1.2). Оптималних решења може бити више, зависно од проблема и инстанци.

Min-max проблеми су изузетно присутни у литератури. Више о овим проблемима, као и о бројним примерима ових проблема, може се прочитати у [27].

1.1.2 Робусна оптимизација

У пракси се код многих оптимизационих проблема често наилази на *непоузданост* улазних података. Непоузданост може бити одраз непрецизних мерења или промена података током времена. Узроци поменутих промена су бројни. Уколико је, на пример, потребно изградити болнице на подручју државе у којој су честе тешке временске непогоде које могу онеспособити рад

успостављених установа, доступност изграђених болница након неке временске непогоде неће бити иста као при решавању почетног проблема. Уколико је пак, потребно одредити локације за изградњу продајних места неке фирме, у оквиру територије која је у процесу развоја, очекивано је да ће током времена доћи до пораста цена на тржишту. Самим тим доћи ће до промена захтева корисника, неопходних капацитета тих продајних објеката, цена транспорта и слично. Уколико се при моделирању проблема не узме у обзир непоузданост података, оптимално решење тако дефинисаног проблема у већини случајева неће бити оптимално, а можда чак ни допустиво, у тренутку када дође до промена вредности података.

Како би се укључила непоузданост података у математички модел проблема, често се користе стохастичко програмирање [19] или оптимизација са пробабилистичким условима [15]. Код ових метода потребно је знати могуће *сценарије* (комбинације варирања улазних података), као и њихове расподеле вероватноћа. Овако нешто је изузетно тешко постићи у пракси. Поред тога, узимање у обзир свих могућих сценарија ће у већини случајева довести до проблема великих димензија. Овакви проблеми, у зависности од методе решавања, или неће моћи да се реше због ограничене меморије у рачунару, или ће њихово решавање захтевати јако пуно времена. Због наведених мана стохастичког програмирања и оптимизације са пробабилистичким условима, све чешће се прибегава нешто новијој методи рада са непоузданим подацима, а то је *робусна оптимизација*.

Код робусне оптимизације, за непоуздане вредности у функцији циља и ограничењима, дефинишу се *скупови непоузданости* (енг. *uncertainty sets*). Претпоставка је да ће вредности варирати само у оквиру тих скупова. Ови скупови могу бити интервали, елипсоидни скупови, политопски скупови итд. Код дискретне робусне оптимизације скупови непоузданости се могу дефинисати као скупови сценарија [57]. Као што је претходно напоменуто сценарији су комбинације варирања улазних података. Прецизније, један сценарио представља једну могућу реализацију вредности непоузданих улазних података. Кардиналност скупа сценарија у дискретном случају дефинише ниво робусности који желимо да уведемо у модел. У општем случају ниво робусности се може дефинисати употребом такозваног буџета (видети нпр. [16]). Већи ниво робусности изискује дуже време извршавања. Циљ робусне оптимизације је да дође до решења које ће бити допустиво у сваком сценарију и оптимално за

најгори случај. Најинтуитивнији одабир облика функције циља при робусној оптимизацији, јесте одабир min-max облика. За проблем који се решава у оквиру овог мастер рада користи се дискретна робусна оптимизација. Општи облик проблема дискретне робусне оптимизације може се записати у облику (1.2), где Sc у (1.3) представља скуп сценарија.

Код класичне робусне оптимизације све одлуке о решењу доносе се истовремено на основу свих доступних података. Код *двостепене робусне оптимизације* ово није случај. Доношење одлуке о решењу проблема се раставља на два степана: *овде-и-сада* (енг. *here-and-now*) и *сачекај-па-види* (енг. *wait-and-see*). Први степен модела (*овде-и-сада*) односи се на доношење одлуке о делу решења које ће се применити пре него што се изрази непоузданост података. У другом степену модела (*сачекај-па-види*) доноси се одлука о промени дела решења када дође до промене у улазним подацима, са циљем да се минизације максимална вредност функције циља другог степена модела [26]. За конструкцију вишестепених робусних модела могу се применити методе као што су прилагодљива робусна оптимизација (енг. *Adaptive robust optimization* [11, 16] или *Adjustable robust optimization* [8]) и опорављива робусна оптимизација (енг. *Recoverable robustness* [40]).

Ниједна од наведених метода се не може прогласити за „најбољу". Одабир методе за рад са непоузданим подацима треба да зависи искључиво од природе конкретног проблема који се решава и доступних података. Уколико то има смисла за посматрани проблем, у робусни модел се могу уврстити и одређене претпоставке о расподелама. Односно, може се користити робусно стохастичка оптимизација или, на пример, прилагодљива дистрибутивна робусна оптимизација (видети [16]).

У овом мастер раду коришћена је двостепена робусна оптимизација. За више о робусној оптимизацији видети [7, 16, 33].

1.2 Локацијски проблеми

Локацијским проблемом може се назвати било који проблем одређивања локација неких објеката. Објекти за које је потребно одредити локације успостављања најчешће се називају *ресурси*, *сервиси*, *добављачи* или *услужни објекти*. Поред ових, јављају се и објекти које ће они услуживати. Они се најчешће називају *корисници*, *клијенти*, или *потрошачи*. У наставку рада биће

коришћени термини ресурс и корисник. Најчешћа питања на која се добијају одговори приликом решавања неког локацијског проблема су: а) Колико ресурса треба успоставити и на којим локацијама? (односно, кажемо да је потребно лоцирати ресурсе) и б) Који ресурс ће опслуживати које кориснике? (односно, кажемо да је потребно извршити алокацију корисника). Постојаност ових и додатних питања, као и то који су најадекватнији одговори на задата питања, зависе од конкретног проблема који се решава. Код локацијских проблема, најчешће је циљ да се, уз задовољавање одређених услова, минимизује укупно растојање, или највеће корисник-ресурс растојање. Појам *растојања* се у зависности од локацијског проблема може односити на физичко растојање, цену транспорта, време транспорта и слично. У наставку је изложено неколико примера локацијских проблема и класификација. Више о локацијским проблемима може се видети у [25, 28, 44].

1.2.1 Примери локацијских проблема

Размотримо ситуацију у којој одређена фирма жели да отвори нови продајни објекат у неком граду. Циљ наведеног локацијског проблема је да се одреди на којој од доступних локација у граду треба отворити радњу, тако да профит који би фирма остваривала буде максималан, узимајући у обзир конкуренцију у датим деловима града и поделе тржишта.

Одређивање локација за складишта неке компаније, како би се осигурала што бржа и јефтинија услуга корисника, представља још један локацијски проблем. При чему би у обзир требало узети и трошкове закупа складишта, трошкове транспорта и ограниченост капацитета складишта.

Градска управа за саобраћај и путеве суочава се са проблемом одређивања локација и броја станица јавног градског превоза. При формулацији овог проблема пожељно је узети у обзир локације постојећих станица, трошкове изградње, повезаност и дотупност локација, као и растојања између потенцијалних локација.

Код транспортних и телекомуникационих системима јавља се посебна врста локацијских проблема. То су такозвани *хаб локацијски проблеми*. Код ових проблема се уместо директног снабдевања сваког корисника од стране њему придруженог ресурса, транспорт ресурс-корисник одвија преко скупа хабова које је потребно лоцирати.

Листа локацијских проблема је изузетно дугачка. Због великог броја примена у пракси и значаја решавања датих проблема, постоји велики број радова везаних за теорију локације и проблеме ове природе.

1.2.2 Класификација локацијских проблема

У литератури постоје разне поделе локацијских проблема, међутим, строга класификација се показала као немогућа, јер се увек може наћи локацијски проблем који не припада ниједној класи. У наставку су дати најчешћи начини њихове поделе.

На основу места на којима је дозвољено успоставити ресурсе, локацијски проблеми се могу поделити на *дискретне*, *мрежне* и *континуалне*. Код дискретних поблема, локације за ресурсе се могу бирати само из унапред задатог коначног скупа локација. Код континуалних проблема ресурсе је дозвољено лоцирати у било којој тачки посматране регије. Док се код мрежних проблема ресурси успостављају на задатој мрежи.

У зависности од тога да ли је број ресурса које треба успоставити унапред познат, разликујемо *ендогене* и *егзогене* моделе. Код ендогених модела овај број је унапред задат. Ендогени модели се према задатом броју ресурса могу даље поделити на *једнообјектне* и *вишеобјектне*. Специјалан подтип ендогених проблема су и *локацијско-алокацијски* проблеми, код којих је поред одређивања локација ресурса, потребно одредити и који ресурс ће опслуживати ког корисника. Овакви проблеми се у пракси виђају далеко чешће од оних код којих су корисници унапред повезани са ресурсима. Код егзогених модела број ресурса које треба успоставити није унапред познат, већ се добија као резултат оптимизације.

Према капацитету разликујемо проблеме *ограничених капацитета* и проблеме *неограничених капацитета*. Код проблема ограничених капацитета јављају се ограничења капацитета ресурса у виду производње, транспорта, складиштења и слично, док код проблема неограничених капацитета оваквих ограничења нема.

Према броју критеријумских функција може се извршити подела на *једнокритеријумске* и *вишекритеријумске* проблеме.

На основу облика функције циља може се говорити о *min-sum* проблемима, код којих се врши минимизација суме растојања, и *min-max* локацијским проблемима, код којих се минимизује максимално корисник-ресурс растојање.

Односно, код min-max проблема циљ је оптимизација „најгоре варијанте", што је најчешће пожељно при лоцирању ресурса хитних служби (болница, полицијских станица, ватрогасних станица, итд). Min-sum проблеми код којих је потребно успоставити тачно p ресурса називају се проблеми *p-медијане*, док се min-max проблеми са задатком лоцирања p ресурса називају проблеми *p-центра*.

Још једна подела локацијских проблема јесте подела на *статичке* и *динамичке* проблеме. Највећи број локацијских модела су статичке природе, док су проблеми који се виђају у пракси често динамички. У статичким моделима улазни подаци се не мењају током времена, док је код динамичких проблема обрнут случај, те зато, модели за динамичке проблеме морају узети у обзир више временских периода. Примера ради, промене у улазним подацима код динамичких проблема (самим тим и потреба за разматрањем више временских периода) могу бити последица мењања цена на тржишту током година, разлике у захтевима корисника током радних дана у односу на викенд или у различитом добу дана и слично. Овакви улазни подаци се могу генерисати формирањем могућих сценарија на основу одређених претпоставки и досадашњег искуства, коришћењем одговарајуће расподеле и слично. Динамички проблеми се често моделирају статичким моделима, тако што се проблем посматра и решава само у једном временском периоду. У неким динамичким моделима ресурси се успостављају само једном тако да одабир што боље задовољава услове и ограничења у свим временским периодима, док се код других могу премештати или уклањати успостављени ресурси и лоцирати нови, у различитим временским периодима.

1.3 Методе решавања

За решавање оптимизационих проблема користе се *апроксимативне* и *егзактне методе* [54]. Егзактне методе налазе оптимална решења проблема. Међутим, услед ограничења времена и меморије у рачунару, имплементације егзактних метода при решавању NP-тешких проблема често не успевају да дођу до оптималног решења, а некад чак ни до допустивог. Од апроксимативних метода највећу примену имају *хеуристике*. За разлику од егзактних метода хеуристике увек налазе допустиво решење, али оно не мора бити оптимално. У пракси се хеуристике имплементирају тако да проблем решавају

брже од егзактних метода, а да при томе врате допустиво решење доброг квалитета. Оне налазе баланс између оптималности, потпуности и прецизности решења са једне стране и времена извршавања, са друге. Хеуристичке методе које нису ограничене на конкретан проблем, већ се уз одређена прилагођавања могу применити на велики број проблема, називају се *метахеуристике*. Метахеуристике почивају на два битна концепта: *диверсификација* и *интенсификација* претраге. Интенсификација подразумева детаљну претрагу у мањим просторима око решења која су квалитетна, док диверсификација настоји да претрагу прошири на што више делова претраживачког простора. У оквиру овог рада користи се метахеуристика која се назива *методом променљивих околина*. Више о овој методи дато је у секцији 3.1.

Глава 2

Проблем p-центра са поузданом мрежом

2.1 Опис проблема

Као што је наведено у уводном делу овог рада, основни циљ локацијских проблема је одабир локација за успостављање ресурса и алоцирање корисника успостављеним ресурсима. У случају локацијских проблема p-центра, потребно је успоставити тачно p ресурса са циљем да сваком кориснику буде пружена добра услуга. Акценат се ставља на корисника који ће имати најнижи квалитет услуге. Прецизније, циљ је минимизовати максимално корисник-ресурс растојање. У пракси, ови проблеми налазе примену у ситуацијама у којима се сваки корисник сматра подједнако битним, као што је случај при лоцирању ресурса хитних служби (болница, полицијских станица, ватрогасних станица итд). Одабир локација за успостављање ресурса представља дугорочну одлуку. Уколико се неки од података коришћених при решавању проблема промени током времена, одабир успостављених ресурса можда неће бити адекватан при новонасталим околностима. Код лоцирања ресурса хитних служби битно је да сво становништво има брз приступ овим ресурсима, чак и ако неки од њих постану неупотребљиви, што се може десити услед природних катастрофа, терористичких напада, протестних блокада и слично. Управо то је главни фокус проблема који се изучава у оквиру овог мастер рада.

Проблем који је решаван у оквиру овог рада је *Проблем p-центра са поузданом мрежом* (енг. *Reliable p-center facility location problem*), дат у [26].

Проблем је типа p-центра, при чему је као што је претходно напоменуто, циљ да решење проблема буде отпорно на могуће промене, односно да квалитет решења остане колико је могуће добар у случају да дође до онеспособљења неког ресурса и промена у растојањима и захтевима корисника. Решавање проблема подразумева иницијално успостављање ресурса и алокацију корисника, као и накнадну реалокацију корисника у складу са измењеним улазним подацима и доступношћу ресурса у новонасталој ситуацији.

2.2 Математичка формулација

У овој секцији, за разматрани проблем, изложен је двостепени робусни модел, предложен у [26]. Скуп непоузданости робусног модела сачињен је од сценарија поремећаја. У оквиру напоменутих сценарија дате су информације о томе који су ресурси онеспособљени, односно недоступни, као и ажуриране вредности растојања и захтева корисника. Први степен модела заснован је на детерминистичкој варијанти проблема p-центра, чија се математичка формулација, дата у виду мешовитог целобројног линеарног програмирања, може наћи у оквиру истог рада. При другом степену модела врши се реалокација корисника опсталим ресурсима у оквиру сценарија, према ажурираним улазним подацима (доступност ресурса, захтеви корисника и вредности растојања). У наставку ће бити коришћена следећа нотација [26]:

- I – скуп корисника;
- J – скуп потенцијалних локација ресурса;
- K – скуп сценарија;
- p – број ресурса које треба успоставити;
- c_{ij} – растојање од корисника $i \in I$ до ресурса $j \in J$, пре поремећаја;
- d_i – захтеви/потражња корисника $i \in I$, пре поремећаја;
- $c^2_{ij}(k)$ – растојање од корисника $i \in I$ до ресурса $j \in J$, у сценарију $k \in K$;
- $d^2_i(k)$ – захтеви/потражња корисника $i \in I$, у сценарију $k \in K$;

- $a_j(k)$ – доступност потенцијалне локације ресурса $j \in J$ у сценарију $k \in K$, прецизније

$$a_j(k) = \begin{cases} 1, & \text{ако је локација } j \in J \text{ недоступна у сценарију } k \in K \\ 0, & \text{иначе} \end{cases}$$

за свако $k \in K$ и $j \in J$.

Захтеви d_i, корисника $i \in I$, ће увек бити задовољени од стране успостављеног ресурса, који је најближи том кориснику. Односно, корисник $i \in I$ ће бити услужен од стране успостављеног ресурса $j \in J$, за који је c_{ij} најмање. Како сценарији поремећаја представљају ситуације у којима долази до онеспособљења одређених ресурса, у сваком сценарију може постати недоступан већи број локација ресурса. Како би постојало решење проблема, претпоставка је да важи $\sum_{j \in J} a_j(k) < p$. Бинарне променљиве одлуке, потребне за математичку формулацију су:

$$y_j = \begin{cases} 1, & \text{ако је ресурс } j \text{ успостављен} \\ 0, & \text{иначе} \end{cases}$$

за све $j \in J$,

$$x_{ij} = \begin{cases} 1, & \text{ако је корисник } i \text{ додељен ресурсу } j, \text{ пре поремећаја} \\ 0, & \text{иначе} \end{cases}$$

за све $i \in I$, $j \in J$, и

$$w_{ijk} = \begin{cases} 1, & \text{ако је корисник } i \text{ додељен ресурсу } j \text{ у сценарију } k \in K \\ 0, & \text{иначе} \end{cases}$$

за све $i \in I$, $j \in J$ и $k \in K$. Променљива x_{ij} односи се на алокацију корисника у периоду у ком још није дошло до онеспособљења одређених ресурса, док се w_{ijk} односи на реалокације у оквиру сценарија поремећаја. Користећи наведену нотацију, може се изложити двостепени робусни модел за задати проблем [26]:

$$\min_{x,y} \quad \alpha_1 L_1 + \alpha_2 Q(y) \tag{2.1}$$

при ограничењима:

$$L_1 \geqslant \sum_{j \in J} c_{ij} d_i x_{ij} \qquad \forall i \in I, \tag{2.2}$$

$$\sum_{j \in J} y_j = p, \tag{2.3}$$

$$x_{ij} \leqslant y_j \qquad \forall i \in I,\ j \in J, \tag{2.4}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I, \tag{2.5}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in I, j \in J, \tag{2.6}$$

где је $Q(y)$ дато кроз други степен модела:

$$Q(y) \ = \ \max_{k \in K} \min_{w_k} \quad L_2(k) \tag{2.7}$$

при ограничењима:

$$L_2(k) \geqslant \sum_{j \in J} c_{ij}^2(k) d_i^2(k) w_{ijk} \qquad \forall i \in I, \tag{2.8}$$

$$w_{ijk} \leqslant y_j \qquad \forall i \in I,\ j \in J, \tag{2.9}$$

$$w_{ijk} \leqslant 1 \ - \ a_j(k) \qquad \forall i \in I,\ j \in J, \tag{2.10}$$

$$\sum_{j \in J} w_{ijk} = 1 \qquad \forall i \in I, \tag{2.11}$$

$$w_{ijk} \in \{0, 1\} \qquad \forall i \in I, j \in J. \tag{2.12}$$

Израз (2.1) представља функцију циља и односи се на минимизацију тежинске суме максималног растојања између корисника и ресурса у првом степену модела (период пре поремећаја) означеног са L_1 и највећег од максималних растојања из свих сценарија другог степена модела (након што дође до онеспособљења неког ресурса) означеног са $Q(y)$. Тежине α_1 и α_2 у (2.1) одређују колико се вреднује минимизација максималног растојања пре, а колико после поремећаја, при доношењу одлуке о резултату. Постављање вредности за L_1 на максимално корисник-ресурс растојање у периоду пре поремећаја постиже се кроз ограничење (2.2). Одабир тачно p ресурса за

успостављање, осигуран је ограничењем (2.3). Имплицитна претпоставка је да важи $p \leq |J|$. Ограничењима (2.4) и (2.5) обезбеђује се да, у периоду пре поремећаја, корисници буду алоцирани само успостављеним ресурсима, као и то, да сваки корисник буде услужен од стране тачно једног ресурса, респективно. Ограничењем (2.6) успоставља се бинарна природа променљивих одлуке. Слична ограничења суделују и у другом степену модела. Кроз ограничење (2.8), вредност за $L_2(k)$ које конфигурише у (2.7), поставља се на максимално растојање између успостављених и доступних ресурса и њима алоцираних корисника у k-том сценарију. Односно, $L_2(k)$ представља максимално корисник-ресурс растојање у k-том сценарију. Према (2.7) максимална корисник-ресурс растојања се минимизују за сваки сценарио, а узимањем највећег од свих максималних растојања из свих сценарија, добија се $Q(y)$. Ограничењима (2.9) и (2.10) обезбеђује се да корисници буду алоцирани само успостављеним ресурсима који су достпупни у k-том сценарију. Ограничење (2.11) гарантује да ће сваки корисник, у k-том сценарију, бити услужен од стране тачно једног ресурса. Ограничење (2.12) успоставља бинарну природу променљиве одлуке.

По узору на поступак из [13], аутори у [26] су трансформисали претходно наведени двостепени модел у једностепени. Добијен је линеаран модел облика:

$$\min_{x,y,w_k} \quad \alpha_1 L_1 \; + \; \alpha_2 L_{max} \tag{2.13}$$

при ограничењима: (2.2) до (2.6) и (2.8) до (2.12) , за свако $k \in K$,

$$L_{max} \geqslant L_2(k) \qquad \forall k \in K. \tag{2.14}$$

У радовима [26], [52] и [3] претпоставка је да важи $I = J$. Према наведеној претпоставци претходни модел, какав је изложен у [26], може се преформулисати у:

$$\min_{x} \quad \alpha_1 L_1 \; + \; \alpha_2 Q(x) \tag{2.15}$$

при ограничењима:

$$L_1 \geqslant \sum_{j \in I} c_{ij} d_i x_{ij} \qquad \forall i \in I, \tag{2.16}$$

$$\sum_{j \in I} x_{jj} = p, \tag{2.17}$$

$$x_{ij} \leqslant x_{jj} \qquad \forall i, j \in I, \tag{2.18}$$

$$\sum_{j \in I} x_{ij} = 1 \qquad \forall i \in I, \tag{2.19}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in I, \tag{2.20}$$

где је $Q(y)$ дато кроз други степен модела:

$$Q(x) \; = \; \max_{k \in K} \min_{w_k} \; L_2(k) \tag{2.21}$$

при ограничењима:

$$L_2(k) \geqslant \sum_{j \in I} c_{ij}^2(k) d_i^2(k) w_{ijk} \qquad \forall i \in I, \tag{2.22}$$

$$w_{ijk} \leqslant x_{jj} \qquad \forall i, j \in I, \tag{2.23}$$

$$w_{ijk} \leqslant 1 \; - \; a_j(k) \qquad \forall i, j \in I, \tag{2.24}$$

$$\sum_{j \in I} w_{ijk} = 1 \qquad \forall i \in I, \tag{2.25}$$

$$w_{ijk} \in \{0, 1\} \qquad \forall i, j \in I. \tag{2.26}$$

Одговарајућа реформулација на модел са једним степеном, може се записати као:

$$\min_{x, w_k} \quad \alpha_1 L_1 \; + \; \alpha_2 L_{max} \tag{2.27}$$

при ограничењима: (2.16) до (2.20) и (2.22) до (2.26) , за свако $k \in K$,

$$L_{max} \geqslant L_2(k) \qquad \forall k \in K. \tag{2.28}$$

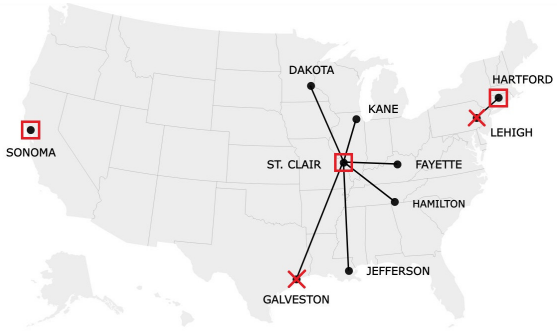
2.3 Илустративан пример проблема

Нека је дата инстанца која садржи 10 потенцијалних локација, односно 10 корисника ($|I| = |J| = 10$). Задате локације приказане су на слици 2.1а. Нека скуп K садржи 5 сценарија поремећаја ($|K| = 5$), при чему у сваком сценарију постају недоступна по 2 ресурса, и нека вредности тежинских коефицијената износе $\alpha_1 = 0.8$ и $\alpha_2 = 0.2$. Потребно је да се успостави тачно 3 ресурса ($p = 3$). Ради једноставности приказа, матрице $[c_{ij}]_{i\in I, j\in J}$, $[d_i]_{i\in I}$ и $[c^2_{ij}(k)]_{i\in I, j\in J}$, $[d^2_i(k)]_{i\in I}$, $\forall k \in K$, нису изложене, а генерисане су на начин описан у оквиру секције 4.1.

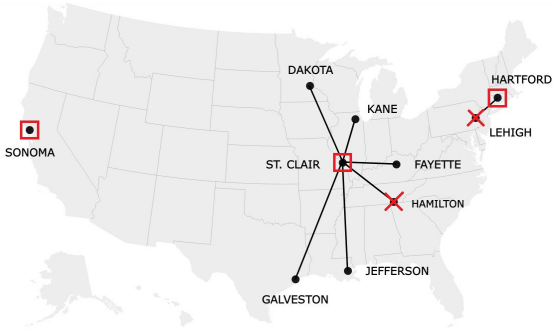
Графички приказ оптималног решења проблема p-центра за задату инстанцу дат је у оквиру слике 2.1. На слици 2.1б приказан је одабир ресурса које треба успоставити, као и алокације, за период пре поремећаја. На сликама 2.1в-2.1е приказани су сценарији поремећаја и адекватне реалокације корисника у складу са доступним ресурсима и ажурираним ценама и потражњом. Потребно је имати у виду да је скуп потенцијалних локација за ресурсе исти као скуп корисника, те да, локација која постаје недоступна (на сликама означене црвеним иксом) у одређеном сценарију и даље има улогу корисника. Са слика се види да је, за дату инстанцу, само у оквиру трећег сценарија било пожељно другачије алоцирање у односу на преостале сценарије. Вредност функције циља за задату инстанцу износи 565.780, док су $L_1 = 372.268$ и $Lmax = 1339.827$.

(а) Потенцијалне локације

(б) Период пре поремећаја

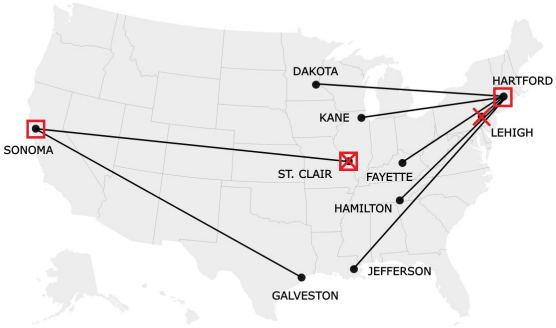


(в) Нулти сценарио

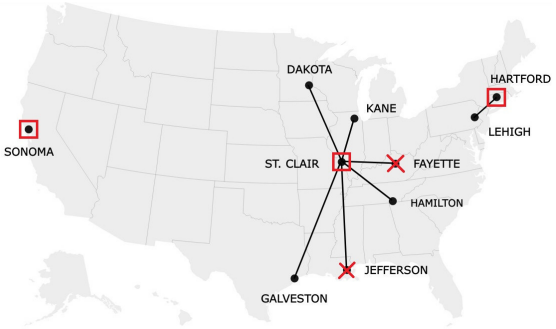


(г) Први сценарио

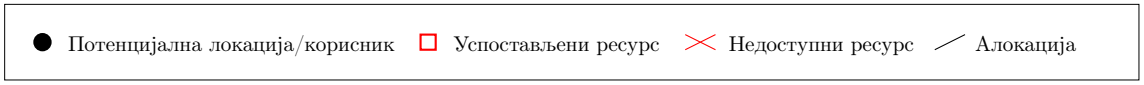
(д) Други сценарио



(ђ) Трећи сценарио



(е) Четврти сценарио



Слика 2.1: Графички приказ решења једне инстанце проблема p-центра са поузданом мрежом

2.4 Преглед релевантне литературе

Због свог практичног значаја локацијски проблеми су нашли значајно место у научној заједници. Постоји пуно радова који се баве разним локацијским проблемима, неки од њих наведени су у [12, 21, 24, 39, 43]. У оквиру ове секције направићемо кратак осврт на радове посвећене проблемима распоређивања хитних служби, раду са непоузданостима улазних података и примени робусне оптимизације при раду са непоузданостима.

Према [53] изучавање проблема распоређивања ресурса хитних служби, у области операционих истраживања, почело је још половином двадесетог века. За разматрање ових проблема најчешће се примењују модели типа медијане и центра, као и модели покривања [43]. У [38] се може наћи преглед многих радова на тему локацијских проблема за реаговање у хитним случајевима, као и модела коришћених за њихово решавање. У оквиру истог рада дат је уопштени модел погодан за хитне случајеве великих размера. Показано је да се предложени модел може преформулисати у модел за проблем покривања, проблем p-центра као и проблем p-медијане. Такође, дати су примери употребе модела ради одређивања оптималних локација медицинске помоћи за реаговање у хитним случајевима великих размера, на територији Лос Анђелеса. Према [46], проблем p-центра се показао као тежи за решавање, у односу на порблем p-медијане. Такође, вреди напоменути да постоји знатно мање радова који се баве проблемом p-центра.

У проблемима из праксе често се јавља непоузданост података. Аутори из [49] и [2] класификовали су непоузданости у локацијским проблемима као непоузданости на страни добављача, непоузданости на страни примаоца и међу-непоузданости. Непоузданости на страни добављача се односе на непоузданости везане за капацитет и доступност ресурса. Непоузданости на страни примаоца, односе се на, непоузданост потражње корисника, непоузданост локација корисника итд. Док се међу непоузданости односе на недостатак информација везаних за транспортну мрежу између ресурса и корисника у виду времена и трошкова транспорта. Непоузданости се могу класификовати на други начин, на непоузданости података (примере је могуће видети у нпр. [50]) и непоузданости у виду онеспособљења ресурса. Како је наведено у [52], непоузданости у виду онеспособљења ресурса се могу посматрати као непоузданости у самом решењу проблема. У оквиру истог рада предложени су први

модели за проблем p-медијане са поузданом мрежом (енг. *Reliability/Reliable p-median problem*) и локацијски проблем фиксних трошкова и неограничених капацитета са поузданом мрежом (енг. *Reliability/Reliable uncapacitated fixed-charge location problem* – RUFLP). Модели минимизују укупну суму трошкова транспорта, која обухвата трошкове у периоду пре поремећаја и предвиђене вредности трошкова након поремећаја. Сваком ресурсу додељена је иста вероватноћа онеспособљења. Сваки корисник се алоцира примарном ресурсу и низу резервних ресурса. Примарни ресурс услужује корисника све док не постане онеспособљен, након чега корисника почиње да услужује први из низа њему додељених резервних ресурса, који је и даље у функцији. За решавање проблема је коришћена Лагранжова релаксација (видети нпр. [1, 29]). За разлику од [52] где је вероватноћа онеспособљења сваког ресурса иста, у [10], [17] и [49] ресурсима се додељују различите вероватноће онеспособљења. У [10] је показано да се решење стохастичког проблема p-медијане поклапа са решењем детерминистичког проблема, када се вероватноће онеспособљења приближавају нули. Неколико егзактних приступа и хеуристика имплементирано је за решавање датог проблема. Аутори из [17] формулисали су RUFLP у виду проблема мешовитог целобројног линеарног програмирања и изложили модел непрекидних апроксимација. У [49] разматрана су два модела. Први модел је двостепени стохастички модел који користи принцип сценарија, а други формулише проблем као проблем целобројног нелинеарног програмирања. Аутори су нагласили да је лоша страна коришћења сценарија, то што време извршавања постаје велико са порастом броја сценарија. Предложено је и тестирано неколико хеуристика за решавање формулисаних проблема. У [42] је први пут примењено двостепено стохастичко програмирање ради преформулација дискретних локацијских проблема (простог локацијског проблема и проблема p-медијане). Још неки двостепени стохастички модели и методе решавања тако формулисаних проблема, могу се видети у [6, 41]. Више примера радова који укључују непоузданост података, може се наћи у нпр. [51] и [35].

На локацијске проблеме први пут је примењена робусна оптимизација у [5]. Разматран је вишепериодни локацијски проблем фиксних трошкова и ограничених капацитета. У оквиру рада коришћена су два вишедимензиона скупа непоузданости (енг. *Capacitated multi-period fixed-charge facility location problem*). У [4] и [30] примењена је двостепена робусна оптимизација са непо-

узданим захтевима корисника. У [3] предложен је двостепени робусни модел за проблем p-медијане са поузданом мрежом. За разлику од [4] и [30], у [3] се узимају у обзир могућа онеспособљења ресурса и дозвољавају реалокације у другом степену модела. Проблем је решаван коришћењем Бендерсове декомпозиције (енг. *Benders decomposition*) и методе генерисања колона и ограничења (енг. *Column and constraint generation* – C&CG). Наведени радови, а посебно [3] само су неки од радова који су инспирисали [26]. У [26] је изложен двостепени робусни модел за проблем p-центра са поузданом мрежом. Предложени модел може се видети у овом раду, у оквиру секције 2.2. Скуп непоузданости сачињен је од сценарија поремећаја. У првом степену модела врши се локација ресурса и алокација корисника успостављеним ресурсима, док се у другом степену модела врши реалокација корисника расположивим ресурсима. За решавање проблема аутори су користили три приступа: линеарну реформулацију модела (по угледу на [13]), Бендерсову дуалну методу одсецања равни (енг. *Benders dual cutting plane* – BD) и методу генерисања колона и ограничења. Линеарна реформулација модела се може решити коришћењем неког од комерцијалних решавача проблема мешовитог целобројног програмирања. BD су аутори дизајнирали као Бендерсову декомпозицију засновану на методи одложеног генерисања ограничења (енг. *Delayed constraint generation*) и методи одсецања равни (аутори упућују на [9, 31, 58]) уз конструкцију дуала другог степена двостепеног робусног модела (по угледу на [58]). Све наведене методе су егзактне. У оквиру овог рада решава се проблем из [26] користећи два приступа: имплементацију која користи егзактни CPLEX решавач (конструисана на основу (2.27)) и применом хеуристика, прецизније, урађена је имплементација верзије основне методе променљивих околина. Метода променљивих околина први пут је предложена у [45]. Више о овој методи дато је у секцији 3.1. Детаљи имплементације дати су у оквиру наредних секција овог рада.

Глава 3

VNS за проблем p-центра са поузданом мрежом

3.1 Метода променљивих околина

Метода променљивих околина (енг. *Variable neighbourhood search* – VNS) први пут је предложена у [45]. Основна идеја методе састоји се у сукцесивној примени унапред дефинисаних околина $U_l(x), 1 \leqslant l \leqslant l_{max}$, у циљу налажења бољег решења. Метода се заснива на следећим чињеницама:

- Локални минимум једне околине није нужно локални минимум друге.
- Глобални минимум је уједно и локални минимум сваке од околина.
- Код многих проблема локални минимуми су релативно близу једни других.

На основу прве и друге чињенице може се закључити да глобални минимум треба тражити у оквиру већег броја различитих типова околина. Трећа чињеница говори о томе да глобални минимум треба тражити у близини локалних минимума.

Постоји неколико варијанти алгоритма, међу којима су:

- Метода променљивог спуста (енг. *Variable neighbourhood descent* – VND) код које се у свакој итерацији разматра најбоље решење из околине тренутног. Кораци методе се могу видети у алгоритму 1. У случају $l_{max} = 1$, VND се своди на обичну локалну претрагу (енг. *Local search*).

Алгоритам 1 Метода променљивог спуста

Иницијализација: Генериши почетно решење x и одабери скуп околина $U_l(x), 1 \leqslant l \leqslant l_{max}$ које ће се користити за претрагу
repeat
  $l \leftarrow 1$
  while $l \leqslant l_{max}$ **do**
    Нађи најбоље решење x' у $U_l(x)$
    if x' је боље решење од x **then**
      $x \leftarrow x'$
      $l \leftarrow 1$
    else
      $l \leftarrow 1 + 1$
    end if
  end while
until Није задовољен критеријум заустављања

- Основна метода променљивих околина (енг. *Basic variable neighbourhood search* – BVNS) код које се на произвољно решење из тренутне околине врши примена неког типа локалне претраге. Детаљнији кораци су дати у алгоритму 2.

- Редукована метода променљивих околина (енг. *Reduced variable neighbourhood search* – RVNS) код које се у односу на BVNS изоставља локална претрага (видети алгоритам 3).

- Метода променљивих околина са декомпозицијом (енг. *Variable neighbourhood decomposition search* – VNDS) код које се за разлику од BVNS-а локална претрага не врши на нивоу целог претраживачког простора, већ на неком потпростору. На овај начин претрага се сужава, због чега се добија уштеда на времену.

- Адаптивна метода променљивих околина (енг. *Skewed variable neighbourhood search* – SVNS) код које се локална претрага примењује на произвољно решење из околине тренутног. На тај начин истражују се околине далеко од тренутног решења.

- Уопштена метода променљивих околина (енг. *General variable neighbourhood search* – GVNS) код које се уместо локалне претраге примењује VND.

Детаљније о методи променљивих околина може се прочитати у [32, 37].

Алгоритам 2 Основна метода променљивих околина

Иницијализација: Генериши почетно решење x и одабери скуп околина $U_l(x), 1 \leqslant l \leqslant l_{max}$ које ће се користити за претрагу
repeat
  $l \leftarrow 1$
  while $l \leqslant l_{max}$ **do**
    Изабери произвољно решење x' у $U_l(x)$
    Применом неког типа локалне претраге нађи локално најбоље решење x'
    if x'' је боље решење од x **then**
      $x \leftarrow x''$
      $l \leftarrow 1$
    else
      $l \leftarrow 1 + 1$
    end if
  end while
until Није задовољен критеријум заустављања

Алгоритам 3 Редукована метода променљивих околина

Иницијализација: Генериши почетно решење x и одабери скуп околина $U_l(x), 1 \leqslant l \leqslant l_{max}$ које ће се користити за претрагу
repeat
  $l \leftarrow 1$
  while $l \leqslant l_{max}$ **do**
    Изабери произвољно решење x' у $U_l(x)$
    if x' је боље решење од x **then**
      $x \leftarrow x'$
      $l \leftarrow 1$
    else
      $l \leftarrow 1 + 1$
    end if
  end while
until Није задовољен критеријум заустављања

3.2 Кодирање решења и рачунање вредности функције циља

У овој секцији најпре је описан начин кодирања решења за алгоритам VNS-а који је примењен на разматрани проблем (алгоритам је дат у секцији 3.3). Након тога описан је начин за рачунање вредности функције циља, а затим и његова модификација.

Решење проблема састоји се из бинарног низа x, дужине $|J|$, и целобројног низа *allocations*, дужине $|I| \cdot (|K| + 1)$. Низ x садржи информације о успостављеним ресурсима. Уколико бит на l-тој позицији низа има вредност 1, то означава да је l-ти ресурс успостављен. Уколико се, пак, на l-тој позицији налази 0, то значи да l-ти ресурс није успостављен. Како би решење било допустиво, неопходно је да x садржи тачно p јединица. На пример, $x = 011$ представља једно допустиво решење проблема у коме $|J| = 3$ и $p = 2$. На основу битова у низу x, види се да дато решење предлаже успостављање првог и другог ресурса (индексирање почиње од нуле). Низ *allocations* представља низ алокација. Првих $|I|$ битова односе се на алокације пре поремећаја, док се преосталих $|I| \cdot |K|$ битова односе на алокације при сценаријима поремећаја. Целобројна вредност на l-тој позицији овог низа, где је $l \leq |I|$, представља индекс ресурса којем је l-ти корисник алоциран, у периоду пре поремећаја. Вредност на $(l + |I| \cdot k)$-тој позицији овог низа, где је $k \in |K|$, представља индекс ресурса којем је алоциран l-ти корисник, у k-том сценарију. Нека је, на пример, $|I| = |J| = 3$, $|K| = 2$, $p = 2$, $x = 011$ и

$$allocations = \underbrace{1 \quad 2 \quad 2}_{\substack{\text{пре} \\ \text{поремећаја}}} \underbrace{1 \quad 2 \quad 2}_{\substack{\text{први} \\ \text{сценарио}}} \underbrace{1 \quad 1 \quad 1}_{\substack{\text{други} \\ \text{сценарио}}}.$$

У том случају, у периоду пре поремећаја и у првом сценарију, нулти корисник је алоциран првом ресурсу, док су први и други корисник алоцирани другом ресурсу. У другом сценарију сви корисници су алоцирани првом ресурсу. У даљем тексту ће се под решењем подразумевати низ x, док ће низ x заједно са низом *allocations* чинити *комплетно решење*.

Нека је решење за које се рачуна вредност функције циља означено са x. У складу са подацима који одговарају периоду пре поремећаја, за сваког корисника $i \in I$ потребно је да се одреди најмања од вредности $c_{ij} \cdot d_i$, које се рачунају за све успостављене ресурсе j. Нека је max_0 максимум од добијених

минималних вредности. Односно, нека је:

$$max_0 = \max_{i \in I} \min_{j \in J} \{c_{ij} \cdot d_i \mid x(j) = 1\}$$

На аналоган начин одреде се и $max_1, ..., max_{|K|}$ за сценарије поремећаја, с тим што се користе подаци за одговарајући сценарио. Односно, када се рачуна max_k, $k \in K$, уместо вредности $c_{ij} \cdot d_i$ рачунају се вредности $c_{ij}^2(k) \cdot d_i^2(k)$, за сваког корисника $i \in I$, по свим успостављеним и доступним ресурсима. Прецизније, за $k \in K$ важи:

$$max_k = \max_{i \in I} \min_{j \in J} \{c_{ij}^2(k) \cdot d_i^2(k) \mid a_j(k) = 0, x(j) = 1\}$$

Нека је максимум вредности $max_1, ..., max_{|K|}$ означен са max_{Sc}. Функција циља за решење x има вредност $\alpha_1 max_0 + \alpha_2 max_{Sc}$.

Прелазак у наредно решење се врши одабиром решења из околине $U_k(x)$, $1 \leqslant k \leqslant k_{max}$ тренутног решења x, односно, ослобађањем k ресурса у x и успостављањем k нових. Ово значи да $|J| - 2 \cdot k$ битова у новом решењу остају исти као у претходно разматраном решењу x. Уколико би се претходно изложен поступак рачунања вредности функције циља применио на сва решења која се узимају у разматрање током решавања проблема, дошло би до понављања великог броја израчунавања. Стога је урађена модификација кроз коју се чувају већ рачунате вредности, како би се избегла непотребна поновна израчунавања, и уштедело време.

Модификовани приступ користи структуру *saved*. На почетку решавања проблема p-центра са поузданом мрежом, цела структура *saved* је празна. У току решавања проблема структура се допуњује подацима добијеним за свако ново решење. Подаци у структури се не бришу, већ се допуњују и ажурирају. Структуру *saved* чине две подструктуре. Прва подструктура одговара периоду пре поремећаја и садржи матрицу *tot*, низ *res*, низ парова *minResPair* и вредност *max*. Зависност између наведених елемената се може визуелно приказати као на слици 3.1.

Низ *res* представља низ индекса свих ресурса који су били успостављени у оквиру неког од досад разматраних решења. Матрица *tot* садржи вредности $c_{ij} \cdot d_i$, рачунате за све кориснике $i \in I$ по свим елементима j из *res*. Минимум елемената једне врсте матрице *tot*, по ресурсима који су успостављени у оквиру тренутно разматраног решења, као и ресурс по коме је минимум остварен, чине пар који се бележи у низ *minResPair*. Вредност *max* одговара максимуму од свих забележених минимума у *minResPair*. Приликом

$$
\begin{array}{rc}
 & \\
 & \\
 & \\
tot = & \\
 & \\
 & \\
 & \\
res = & \left[\begin{array}{c} \uparrow \\ j_0, \end{array}\right.
\end{array}
\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
\end{array}
\begin{array}{c}
minResPair \\
\| \\
\left[\begin{array}{cccc}
c_{0j_0}\cdot d_0 & c_{0j_1}\cdot d_0 & \dots & c_{0j_m}\cdot d_0 \\
c_{1j_0}\cdot d_1 & c_{1j_1}\cdot d_1 & \dots & c_{1j_m}\cdot d_1 \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
\cdot & \cdot & & \cdot \\
c_{|I|j_0}\cdot d_{|I|} & c_{|I|j_1}\cdot d_{|I|} & \dots & c_{|I|j_m}\cdot d_{|I|}
\end{array}\right]
\begin{array}{c} \xrightarrow{min} \\ \xrightarrow{min} \\ . \\ . \\ . \\ \xrightarrow{min} \end{array}
\left[\begin{array}{c}
\{c_{0j_{k0}}\cdot d_0,\, j_{k0}\} \\
\{c_{1j_{k1}}\cdot d_1,\, j_{k1}\} \\
\cdot \\
\cdot \\
\cdot \\
\{c_{|I|j_{k|I|}}\cdot d_{|I|},\, j_{k|I|}\}
\end{array}\right] \\
\begin{array}{cccc} \uparrow & \uparrow & & \uparrow \\ j_1, & & \dots & j_m \end{array}\ \right]
\begin{array}{c} \\ \\ \\ \\ \\ \\ \downarrow \\ max \end{array}
\end{array}
$$

Слика 3.1: Елементи прве подструктуре у *saved*

рачунања вредности функције циља новог решења подаци у $minResPair$ и max се постављају на вредности које одговарају том решењу. Дакле, подаци у структури *saved* везани за минимуме и максимум одговарају тренутном решењу, док *res* и *tot* чувају све претходно срачунате вредности.

Као што је претходно поменуто, при почетку решавања проблема p-центра са поузданом мрежом, структура *saved* је празна. Иницијализација елемената структуре се врши при разматрању почетног решења x_0. Свако $j \in J$ за које је $x_0(j) = 1$, бележи се у *res*. За свако $i \in I$ и свако j из *res*, у матрицу *tot* бележе се вредности $c_{ij} \cdot d_i$. Једна врста матрице *tot* одговара подацима за једног корисника. Минимум елемената једне врсте, као и ресурс по коме је минимум остварен чине пар који се бележи у низ $minResPair$. У тренутном случају, када је једино разматрано решење x_0, минимум се посматра по свим елементима једне врсте. За сва наредна решења минимум врсте подразумеваће минимум по ресурсима који су успостављени у оквиру тренутно посматраног решења. Максимум од свих забележених минимума у $minResPair$, представља вредност max.

Нека је са x_1 означено наредно решење. У општем случају, прелазак у наредно решење врши се одабиром решења из околине $U_k(x)$ тренутног решења x, односно, ослобађањем k ресурса у x и успостављањем k нових. Нека је x_1 добијено из x_0 ослобађањем тачно једног ресурса и успостављањем другог (односно нека је $k = 1$). За $k > 1$ наредни поступак треба поновити k пута. Означимо ослобођени ресурс са r а успостављени са l. Потребно је ажурирати

структуру *saved*. Како би се узело у обзир успостављање ресурса l примењује се поступак дат кроз алгоритам 4. Уколико се новоуспостављени ресурс l не налази у res, то значи да тај ресурс није био успостављен ни у једном досад разматраном решењу (за сада је то само x_0). У том случају ресурс се убацује у res и у оквиру матрице tot додаје се нова колона са вредностима $c_{il} \cdot d_i$, $i \in I$. Уколико се l већ налазио у res овај корак се прескаче. У оба случаја је потребно извршити и ажурирање $minResPair$ и max, по поступку изложеном кроз алгоритам 5. Како је $minResPair$ низ парова, са $minResPair[i].first$

Алгоритам 4 Успостављање једног ресурса

Улазни подаци: tot, res, $minResPair$, max, l
if l није у res **then**
  Додај l у res
  $j_l \leftarrow |res| + 1$
  for $i = 0$ **to** $|I|$ **do**
    $tot[i][j_l] \leftarrow c_{il} \cdot d_i$
    Ажурирај minResPair и max (tot, minResPair, max, l, j_l, i)
  end for
else
  $j_l \leftarrow$ индекс ресурса l у низу res
  for $i = 0$ **to** $|I|$ **do**
    Ажурирај minResPair и max (tot, $minResPair$, max, l, j_l, i)
  end for
end if

Алгоритам 5 Ажурирај minResPair и max

Улазни подаци: tot, $minResPair$, max, l, j_l, i
if $tot[i][j_l] < minResPair[i].first$ **then**
  $minResPair[i] \leftarrow \{tot[i][j_l], l\}$
  if $minResPair[i].first > max$ **then**
    $max \leftarrow minResPair[i].first$
  end if
end if

означен је минимум i-те врсте матрице tot, по ресурсима који су успостављени у оквиру тренутно посматраног решења, а са $minResPair[i].second$ ресурс по коме се минимум остварује. Ажурирање се врши тако што се за свако $i \in I$ тренутни минимум $minResPair[i].first$ пореди са $tot[i][j_l]$, gde је j_l индекс у res, новоуспостављеног ресурса l. Уколико је вредност мања од

тренутног минимума, $minResPair[i].first$ се поставља на ту вредност. У том случају у $minResPair[i].second$ се уписује l. Истовремено се на основу вредности $minResPair[i].first$ испитује потреба за ажурирањем вредности max. У алгоритму 6 могу се видети промене које је потребно извршити услед ослобађања ресурса r. Подаци везани за ослобођени ресурс r не бришу се из res и tot. Уколико у $minResPair$ постоји вредност која је добијена баш за ресурс r потребно је поново срачунати минимум за тог корисника, по успостављеним ресурсима. Уколико је max био једнак баш минимуму који се мења, потребно је одредити нову вредност за max. Цео поступак се понавља за свако наредно решење.

Алгоритам 6 Ослобађање једног ресурса

Улазни подаци: tot, res, $minResPair$, max, r
for $i = 0$ **to** $|minResPair|$ **do**
  if $minResPair[i].second = r$ **then**
    Срачунај опет $minResPair[i]$
    По потреби ажурирај max
  end if
end for

Другу подструктуру у оквиру *saved* чине подаци везани за сценарије поремећаја. У овој подструктури налази се низ tot матрица, низ res низова, низ $minResPair$ низова, низ вредности max, као и вредност max_{Sc}. Сви низови су дужине $|K|$. Члан сваког низа са истим индексом одговара сценарију са тим индексом. Поступак ажурирања ових података аналоган је претходно изложеном поступку ажурирања података који одговарају периоду пре поремећаја.

Функција циља за решење x има вредност $\alpha_1 max_0 + \alpha_2 max_{Sc}$. Где се max_0 узима из прве, а max_{Sc} из друге подструктуре структуре *saved*. У оквиру алгоритма предложеног за решавање проблема p-центра са поузданом мрежом користи се модификовани приступ рачунања функције циља. Поређење ефикасности праволинијског и модификованог приступа дато је у секцији 4.2.1.

3.3 Структура алгоритма

У алгоритму 7 изложена је структура итеративне варијанте основне методе промењљивих околина (енг. *Iterated basic variable neighborhood search* – IBVNS), за решавање проблема p-центра са поузданом мрежом. Предложена хеуристика започиње претрагу од решења x_0 које се добија тако што се сваки бит поставља на 0 или 1, при чему вероватноћа да бит узме вредност 1 износи 0.25. Овај параметар је одређен експерименталним путем. При томе одабрано решење мора бити допустиво, односно, мора садржати тачно p јединица. За почетно решење се рачуна вредност функције циља $f(x_0)$ и истовремено се иницијализује структура *saved*. На добијено почетно решење x_0 примењује се BVNS у оквиру чега се добија ново решење x и врши ажурирање *saved*. Као што је поменуто у секцији 3.2, минимуми и максимуми у оквиру *saved* прате тренутно разматрано решење. Међутим, уколико се у току једне итерације алгоритма не добије решење са мањом вредношћу функције циља у односу на решење од ког је итерација започела, наредна итерација почиње од решења од ког је започела претходна. Уколико је то случај, вредности минимума и максимума у *saved* треба да прате то решење. Из тог разлога потребно је сачувати минимуме и максимуме из *saved* у $minsMaxes_{best}$, како би се на крају итерације поништиле начињене измене уколико је то потребно. Након тога, над решењем x се итеративно примењују размрдавање, $nbShake$ пута, и BVNS. Мотивација за примену размрдавања састоји се у повећању диверсификације, како би се избегло упадање у локалне минимуме. У оквиру овог, и осталих алгоритама у овој секцији, јавља се k_{max} околина. Околина $U_k(x)$, $1 \leqslant k \leqslant k_{max}$ представља скуп решења која се добијају од решења x, ослобађањем k ресурса и успостављањем k нових. Фаза размрдавања приказана је кроз алгоритам 8 и подразумева замену тренутног решења са произвољно одабраним решењем из неке од околина. У овом случају решење се бира из околине $U_{kI}(x)$ где k_I представља један од параметара алгоритма. О вредностима параметара дато је више у наставку. Добијене промене у решењу задржавају се при уласку у наредну итерацију само у случају да добијено решење има мању вредност функције циља од тренутно најбољег решења. У том случају ажурира се и $minsMaxes_{best}$. У супротном се минимуми и максимуми у *saved* постављају на вредности из $minsMaxes_{best}$. Као критеријум заустављања IBVNS-а узета је комбинација максималног броја итерација и

максималног броја понављања најбољег решења.

Алгоритам 7 IBVNS за решавање проблема p-центра са поузданом мрежом

Улазни подаци: $nbShake$, k_I, k_{max}, l
Иницијализуј почетно решење x_0
Израчунај $f(x_0)$ ▷ *Иницијализује се saved*
$x \leftarrow$ BVNS(x_0, k_{max}, l, *saved*) ▷ *Ажурира се saved*
$minsMaxes_{best} \leftarrow$ минимуми и максимуми из *saved*
repeat
  for $i = 0$ **to** $nbShake$ **do**
    $x' \leftarrow$ Размрдавање(x, k_I)
  end for
  Израчунај $f(x')$ ▷ *Ажурира се saved*
  $x'' \leftarrow$ BVNS(x', k_{max}, l, *saved*) ▷ *Ажурира се saved*
  if $f(x'') < f(x)$ **then**
    $x \leftarrow x''$
    $minsMaxes_{best} \leftarrow$ минимуми и максимуми из *saved*
  else
    Минимуми и максимуми у *saved* $\leftarrow minsMaxes_{best}$
  end if
until Није задовољен критеријум заустављања
Попуни *allocations*

Алгоритам 8 Фаза размрдавања

Улазни подаци: x, k
Изабери произвољно $x' \in U_k(x)$
$x \leftarrow x'$
Излазни подаци: x

Примењена BVNS метода дата је кроз алгоритам 9. Фаза размрдавања одговара претходно споменутом размрдавању, изложеном кроз алгоритам 8. Локална претрага, приказана кроз алгоритам 10, функционише по принципу l-тог побољшања. Прецизније, у једној итерацији локалне претраге врши се произвољан одабир l решења из околине $U_1(x)$, тренутно најбољег решења x. Уколико је пронађено боље решење од тренутно најбољег, x се ажурира. Критеријум заустављања за локалну претрагу представља одсуство бољег решења. Односно, уколико одабиром l решења из $U_1(x)$ није добијено квалитетније решење од x, локална претрага се прекида. У супротном се врши одабир нових l решења у околини новог најбољег решења. Критеријум заустављања

Алгоритам 9 BVNS

Улазни подаци: x, k_{max}, l, *saved*
repeat
  $k \leftarrow 1$
  $minsMaxes_{best} \leftarrow$ минимуми и максимуми из *saved*
  while $k \leqslant k_{max}$ **do**
    $x' \leftarrow$ Размрдавање(x, k)
    $x'' \leftarrow$ Локална претрага(x', l, *saved*) ▷ *Ажурира се saved*
    $k \leftarrow k + 1$
    if $f(x'') < f(x)$ **then**
      $x \leftarrow x''$
      $minsMaxes_{best} \leftarrow$ минимуми и максимуми из *saved*
      $k \leftarrow 1$
    else
      Минимуми и максимуми у *saved* $\leftarrow minsMaxes_{best}$
    end if
  end while
until Није задовољен критеријум заустављања
Излазни подаци: x

Алгоритам 10 Локална претрага

Улазни подаци: x, l, *saved*
repeat
  $x' \leftarrow x$
  $minsMaxes_{center} \leftarrow$ минимуми и максимуми из *saved*
  $i \leftarrow 0$
  while $i < l$ **do**
    Изабери произвољно $x'' \in U_1(x)$
    Израчунај $f(x'')$ ▷ *Ажурира се saved*
    if $f(x'') < f(x')$ **then**
      $x' \leftarrow x''$
      $minsMaxes_{best} \leftarrow$ минимуми и максимуми из *saved*
    end if
    Минимуми и максимуми у *saved* $\leftarrow minsMaxes_{center}$
    $i \leftarrow i + 1$
  end while
  if $f(x') < f(x)$ **then**
    $x \leftarrow x'$
    Минимуми и максимуми у *saved* $\leftarrow minsMaxes_{best}$
  end if
until Није задовољен критеријум заустављања
Излазни подаци: x

BVNS-а представља комбинацију максималног броја итерација и максималног броја понављања најбољег решења. У оквиру локалне претраге и BVNS-а се као код IBVNS-а води рачуна о томе да минимуми и максимуми у *saved* прате тренутно решење. Вредности свих параметара које се појављују у алгоритмима добијене су експерименталним путем и изложене су у оквиру одговарајућих секција које су везане за тестирање алгоритма.

Глава 4

Експериментални резултати

У овом поглављу представљени су резултати примене предложеног IBVNS алгоритма. Имплементација алгоритма извршена је у програмском језику C++. У оквиру истог програмског језика извршена је имплементација која користи CPLEX 20.1.0 решавач. Додатно, приказани су резултати тестирања два начина рачунања вредности функције циља, који су изложени у оквиру секције 3.2. Сва тестирања су извршена под Ubuntu 21.10 оперативним системом, у оквиру Dual Boot система, на рачунару са процесором Ryzen 5 3550H од 2.10 GHz и 8 GB RAM меморије.

4.1 Тест инстанце

Једини рад који разматра и решава проблем p-центра са поузданом мрежом, у облику у ком се он разматра у оквиру овог мастер рада је [26]. Стога би било пожељно да се при тестирању IBVNS алгоритма употребе инстанце које су користили аутори из [26]. Међутим, како није било могуће доћи до напоменутих инстанци, генерисане су нове на основу приступа коришћеног у [26]. Генерисане су две групе инстанци, при чему се поступак генерисања разликује само у одабиру корисника. Инстанце прве групе су величина $|I| = |J| \in \{10, 20, 30, 40, 50, 60, 100, 150\}$. Корисници и потенцијалне локације за ресурсе добијени су произвољном селекцијом округа из базе „2010 County Sorted 250", која садржи списак 250 најнасељенијих округа Сједињених Америчких Држава, из 2010. године. Инстанце друге групе су величина $|I| = |J| \in \{25, 49\}$. Инстанце величине $|I| = |J| = 49$ („49-node" база) преузете

су из [21][1], а локације укључују 48 главних градова континенталних Сједињених Америчких Држава и Вашингтон. Подаци су из 1990. године. Подаци за инстанце са 25 корисника преузети су из [3], а добијени су селекцијом градова из „49-node" базе. Наредни део поступка за генерисање инстанци поклапа се за обе групе. Поред назива, базе садрже и лонгитуду, латитуду и величину популације сваке од локација. Вредност c_{ij} је Еуклидско растојање између локација i и j, добијено на основу координата (лонгитуде и латитуде). За вредност d_i узета је величина популације локације i подељена са 10 000. За сваки сценарио k, вредности $c_{ij}^2(k)$ и $d_i^2(k)$ су произвољно (униформно) биране из интервала $[0.5c_{ij}, 1.5c_{ij}]$ и $[0.5d_i, 1.5d_i]$, респективно. Ресурси који постају недоступни у оквиру сценарија бирани су произвољно са једнаком вероватноћом, тако да важи $\sum_{j \in J} a_j(k) = p - 1$, $\forall k \in K$. За $|K|$ су узете вредности 5, 10, 20, 40, 50, 60, 80, 100, 150 и 200, док су за p узети цели бројеви најближи вредностима $|I|/5$, $|I|/4$ и $|I|/3$, осим у случају када је $|I| = 10$, када су за p узете вредности 2, 3 и 4. За тежинске коефицијенте узете су вредности $\alpha_1 \in \{0.2, 0.5, 0.8\}$ и $\alpha_2 = 1 - \alpha_1$.

4.2 Резултати и поређења

4.2.1 Поређење ефикасности два приступа рачунања вредности функције циља

У оквиру секције 3.2 описан је начин рачунања вредности функције циља и његова модификација. У табели 4.1 приказани су упоредни резултати тестирања праволинијског и модификованог приступа за рачунање функције циља. Тестирање је вршено над пет произвољно одабраних инстанци (више о коришћеним инстанцама дато је у секцији 4.1). Вредност сваке ћелије у табели представља просек резултата 5 извршавања IBVNS-а над одговарајућом инстанцом, уз коришћење једног од два испитивана приступа за рачунање вредности функције циља. Приликом тестирања, максималан број итерација IBVNS-а је био постављен на 5, док је максималан број итерација BVNS-а износио 1000. Како би било коректно поредити времена извршавања, у склопу критеријума заустављања није коришћен максималан број понављања најбољег решења. У табели 4.1, за сваку инстанцу су приказани број корисника,

[1]Базе „2010 County Sorted 250" и „49-node" се могу преузети на [20]

број сценарија, број ресурса које треба успоставити, тежински коефицијент α_1, као и:

- $t_{\text{tot}}^{\text{old}}$ – просечно укупно време извршавања IBVNS алгоритма, који користи праволинијски приступ за рачунање функције циља;
- $t_{\text{tot}}^{\text{new}}$ – просечно укупно време извршавања IBVNS алгоритма, који користи модификовани приступ за рачунање функције циља.

Сва времена изражена су у секундама.

$\|I\|$	$\|K\|$	p	α_1	$t_{\text{tot}}^{\text{old}}(s)$	$t_{\text{tot}}^{\text{new}}(s)$
30	20	7	0.2	224.631	28.952
40	50	8	0.5	1005.571	110.054
40	150	13	0.8	2858.832	206.225
50	200	12	0.2	5840.474	436.180
60	20	15	0.2	692.989	41.625

Табела 4.1: Поређење два приступа рачунања вредности функције циља

На основу изложених резултата може се видети да модификовани приступ доводи до знатног убрзања IBVNS алгоритма, како у просечном случају, тако и над сваком појединачном инстанцом. У просечном случају за изложене инстанце просечно укупно време извршавања IBVNS алгоритма који користи праволинијски приступ за рачунање функције циља износи 2124.299 секунди, док за IBVNS алгоритам који користи модификовани приступ за рачунање функције циља исто износи 164.607 секунди.

4.2.2 Поређење резултата IBVNS-а са резултатима из другог рада

Методе решавања проблема p-центра са поузданом мрежом, изложене у [26], тестиране су над инстанцама величина $|I| = |J| \in \{10, 20, 30, 40, 50, 60\}$, добијеним из базе „2010 County Sorted 250" на начин описан у оквиру секције 4.1. При томе је за сваку вредност $|I|$ из базе генерисано по 5 комбинација округа. За сваку од комбинација је даље разматрано $|K| \in \{5, 10, 20, 50, 100, 150, 200\}$ сценарија. Као што је напоменуто у оквиру секције 4.1, како нисмо имали приступ тим инстанцама генерисали смо нове по истом принципу. У табели 4.2 изложени су упоредни резултати три методе решавања (CC&G,

BD, LIP) изложене у раду [26] и IBVNS имплементације, за $|I| < 40$. Сваки ред табеле односи се на по 45 инстанци које имају исте вредности $|I|$ и $|K|$, а различите вредности за p и α_1. Колона $t(s)$ приказује просечно време, у секундама, за које су методе из [26] успешно достигле оптимално решење. Колона $t_{\text{best}}(s)$ приказује просечно време, у секундама, за које је IBVNS први пут достигао оптимално решење. Кроз колону $opt(\%)$ приказан је проценат инстанци (од 45) за које је проблем решен до оптималности. Оптималност резултата постигнутих IBVNS-ом проверена је коришћењем CPLEX-а. У колони $gap(\%)$ приказане су gap вредности, које у случају CC&G, BD, LIP метода представљају одступање између доње и горње границе, у случају када није достигнуто оптимално решење. Уколико се за свих 45 инстанци, једне групе, достигло оптимално решење, просечна gap вредност је нула. У том случају је одговарајућа ћелија табеле остављена празна. Код IBVNS-а gap вредност се рачуна у односу на најбоље познато решење (детаљније о томе у секцији 4.2.3). Иако тестирања нису вршена над истим инстанцама, резултати показују квалитете имплементираних алгоритама, у просечном случају.

У [26] је за тестирање коришћен рачунар са Intel Core 2 Quad Q9550 процесором са 2.83 GHz и 4 GB RAM меморије, под Windows 10 оперативним системом. Због разлике у платформама коришћеним за тестирање, извршена је нормализација времена извршавања изложених у [26]. Нормализација је извршена на основу оцена перформанси процесора датих на [47], користећи приступ из [22] и следећу формулу [43]:

$$\text{NAT(metoda)} = \text{AT(metoda)} \cdot \frac{\text{PCPUS(Intel Core 2 Quad Q9550 2.83 GHz)}}{\text{PCPUS(AMD Ryzen 5 3550H)}},$$

где је са AT(metoda) означено време за одговарајућу методу из [26], а PCPUS представља Passmark CPU скор. Као критеријум заустављања свих метода у [26] коришћено је максимално време извршвања, које је ограничено на 1000 секунди, односно 287.74 секунде након нормализације.

На основу резултата из табеле, може се видети да просечно време налажења оптималног решења за IBVNS износи 2.99 секунди, за CC&G 20.54 секунди, за BD 19.48 секунди и за LIP 47.88 секунди. Може се уочити да IBVNS најбрже долази до оптималних решења. Просечна вредност за opt износи 100.00% за IBVNS, 88.68% за CC&G, 95.03% за BD и 88.04% за LIP. На основу чега се види да IBVNS једини достиже оптимална решења за све

		IBVNS			CC&G			BD			LIP		
$\|I\|$	$\|K\|$	$t_{\text{best}}(s)$	$opt(\%)$	$gap(\%)$	$t(s)$	$opt(\%)$	$gap(\%)$	$t(s)$	$opt(\%)$	$gap(\%)$	$t(s)$	$opt(\%)$	$gap(\%)$
10	5	0.00	100.00		0.05	100.00		0.06	100.00		0.03	100.00	
	10	0.01	100.00		0.09	100.00		0.09	100.00		0.08	100.00	
	20	0.03	100.00		0.23	100.00		0.18	100.00		0.29	100.00	
	50	0.04	100.00		0.66	100.00		0.39	100.00		1.83	100.00	
	100	0.07	100.00		1.29	100.00		0.77	100.00		3.22	100.00	
	150	0.07	100.00		2.18	100.00		1.45	100.00		6.27	100.00	
	200	0.10	100.00		2.96	100.00		1.88	100.00		9.77	100.00	
20	5	0.03	100.00		0.36	100.00		0.57	100.00		0.21	100.00	
	10	0.09	100.00		0.84	100.00		0.94	100.00		0.63	100.00	
	20	0.21	100.00		2.14	100.00		1.41	100.00		2.24	100.00	
	50	0.42	100.00		9.41	100.00		4.21	100.00		11.33	100.00	
	100	0.76	100.00		23.04	100.00		7.62	100.00		45.11	100.00	
	150	2.69	100.00		30.39	95.56	7.50	16.31	100.00		83.97	95.56	24.55
	200	2.30	100.00		33.23	86.67	11.61	23.51	100.00		119.78	86.67	16.78
30	5	0.26	100.00		1.31	100.00		2.41	100.00		0.82	100.00	
	10	0.58	100.00		4.84	100.00		5.01	100.00		3.46	100.00	
	20	0.55	100.00		34.45	100.00		17.41	100.00		22.52	100.00	
	50	3.46	100.00		69.31	73.33	7.79	49.68	97.78	5.22	116.31	93.33	13.26
	100	8.53	100.00		83.99	35.56	21.07	99.73	88.89	26.04	187.71	44.44	50.13
	150	20.17	100.00		86.00	35.56	28.42	89.71	62.22	20.78	187.43	15.56	59.41
	200	22.46	100.00		45.49	35.56	31.04	85.78	46.67	26.52	202.41	13.33	55.23

Табела 4.2: Резултати тестирања IBVNS-а и упоредни резултати из литературе

разматране инстанце. Просечна вредност за *gap* износи 0.00% за IBVNS, 5.12% за CC&G, 3.74% за BD и 10.45% за LIP. На основу свих резултата, јасно се види да IBVNS показује најбоље перформансе у односу на све разматране алгоритме, у погледу успешности достизања оптималних решења и времена потребног за исто.

4.2.3 Поређење резултата IBVNS-а и CPLEX-а

У табели 4.3 приказани су резултати тестирања имплементиране IBNVS методе, као и CPLEX имплементације. По угледу на [3] за тестирање су коришћене инстанце димензија 25 и 49. Како би се потпуније испитала предложена IBNVS метода уведене су и веће инстанце, димензија 100 и 150, као и мање инстанце димензије 10. Све инстанце добијене су на начин описан у секцији 4.1. При томе су за потребе тестирања у овој секцији узете оне код којих је $|K| \in \{10, 50, 100, 150, 200\}$, $\alpha_1 \in \{0.2, 0.8\}$, и p цели бројеви најближи вредностима $|I|/5$ и $|I|/3$, односно 2 и 4 у случају када је $|I| = 10$. Време извршавања CPLEX-а било је ограничено на 3600 секунди. Параметри коришћени у IBVNS алгоритму добијени су експерименталним путем и могу се видети у оквиру табеле 4.3. Од свих разматраних комбинација вредности параметара, коришћењем вредности из табеле 4.3 алгоритам је достизао најквалитетније резултате.

Вероватноћа успостављања ресурса	0.25
k_{max}	3
$nbShake$	5
k_I	3
Максималан број итерација IBVNS-а	50
Максималан број понављања најбољег решења IBVNS-а	10
l у локалној претрази	5
Максималан број итерација BVNS-а	1000
Максималан број понављања најбољег решења BVNS-а	200

(а) За инстанце величина $|I| = 10$ и $|I| = 25$ за које $|K| < 100$

Вероватноћа успостављања ресурса	0.25
k_{max}	3
$nbShake$	5
k_I	3
Максималан број итерација IBVNS-а	150
Максималан број понављања најбољег решења IBVNS-а	45
l у локалној претрази	5
Максималан број итерација BVNS-а	3000
Максималан број понављања најбољег решења BVNS-а	700

(б) За инстанце величина $|I| = 25$ за које $|K| \geqslant 100$ и $|I| = 49$

Вероватноћа успостављања ресурса	0.25
k_{max}	3
$nbShake$	5
k_I	3
Максималан број итерација IBVNS-а	100
Максималан број понављања најбољег решења IBVNS-а	25
l у локалној претрази	5
Максималан број итерација BVNS-а	7000
Максималан број понављања најбољег решења BVNS-а	3000

(в) За инстанце величина $|I| = 100$ и $|I| = 150$

Табела 4.3: Вредности параметара за IBVNS

IBVNS је над сваком инстанцом покретан по n пута. За инстанце димензија 10, 25 и 49 узето је $n = 10$, док је за веће инстанце $n = 5$. Сваки ред табеле 4.3 односи се на резултате над једном инстанцом. У табели су за сваку инстанцу приказани број корисника, број сценарија, број ресурса које треба успоставити, тежински коефицијент α_1 и:

- $best_{sol}$ – најбоље познато решење;
- t – време извршавања CPLEX-а, у секундама;
- t_{best} – средње почетно време IBVNS-а, у секундама;
- t_{tot} – средње укупно време извршавања IBVNS-а, у секундама;
- $agap$ – средње одступање од оптималног/најбољег познатог решења, у процентима;

- σ – стандардна девијација, у процентима.

Како у литератури нису вршена тестирања над тест инстанцама коришћеним у оквиру овог рада, у колони $best_{\text{sol}}$ ће се наћи оптимално решење добијено CPLEX-ом, или најбоље решење добијено кроз n пуштања IBVNS-а над разматраном инстанцом. Средње почетно време IBVNS-а рачунато је по формули: $t_{best} = \frac{1}{n}\sum_{i=1}^{n} t_i$. Вредност t_i представља почетно време у i-том $(i = 1, ..., n)$ извршавању. Односно, време за које је IBVNS први пут достигао најбоље решење i-тог извршавања. У даљем тексту ће се за најбоље решење i-тог извршавања, користити ознака sol_i. Средње укупно време извршавања, t_{tot}, рачунато је као: $t_{tot} = \frac{1}{n}\sum_{i=1}^{n} t_{toti}$, где се t_{toti} односи на укупно време i-тог извршавања. За добијање вредности $agap$ коришћена је следећа формула: $agap = \frac{1}{n}\sum_{i=1}^{n} gap_i$, при чему $gap_i = 100\frac{|sol_i - best_{sol}|}{|best_{sol}|}$, $i = 1, ..., n$. Стандардна девијација рачуната је по формули $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(gap_i - agap)^2}$. У ћелије колоне $t(s)$ које одговарају инстанцама за које CPLEX није могао да нађе оптимално решење, у временском ограничењу од 3600 секунди, уписано је „-”.

					CPLEX	IBNVS			
$\|I\|$	$\|K\|$	p	α_1	$best_{\text{sol}}$	$t(s)$	$t_{\text{best}}(s)$	$t_{\text{tot}}(s)$	$agap(\%)$	$\sigma(\%)$
10	10	2	0.2	1730.828	0.326	0.003	0.937	0.000	0.000
	10	2	0.8	1268.397	0.211	0.003	0.945	0.000	0.000
	50	2	0.2	2379.220	1.147	0.013	4.200	0.000	0.000
	50	2	0.8	1477.835	0.975	0.007	4.017	0.000	0.000
	100	2	0.2	2835.180	3.676	0.048	7.989	0.000	0.000
	100	2	0.8	1575.006	2.417	0.029	8.259	0.000	0.000
	200	2	0.2	2946.214	13.039	0.070	16.515	0.000	0.000
	200	2	0.8	1602.765	10.310	0.039	16.104	0.000	0.000
	10	4	0.2	1074.730	0.217	0.016	0.691	0.000	0.000
	10	4	0.8	530.623	0.193	0.004	0.685	0.000	0.000
	50	4	0.2	1858.211	0.157	0.040	3.338	0.000	0.000
	50	4	0.8	805.747	0.533	0.038	2.916	0.000	0.000
	100	4	0.2	2056.363	1.251	0.067	5.243	0.000	0.000
	100	4	0.8	880.241	1.313	0.054	5.443	0.000	0.000
	200	4	0.2	2357.012	18.088	0.167	11.747	0.000	0.000
	200	4	0.8	937.873	7.716	0.087	10.519	0.000	0.000

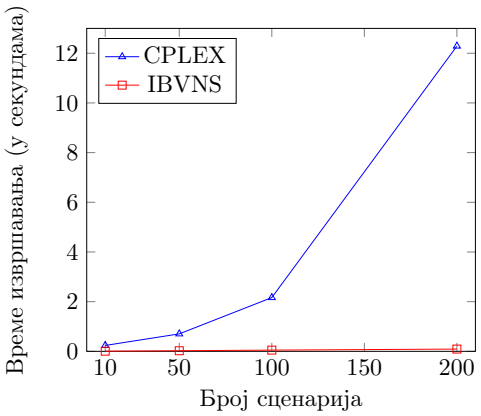
					CPLEX	IBNVS			
$\|I\|$	$\|K\|$	p	α_1	$best_{\text{sol}}$	$t(s)$	$t_{\text{best}}(s)$	$t_{\text{tot}}(s)$	$agap(\%)$	$\sigma(\%)$
25	10	5	0.2	461.596	2.596	0.222	3.613	0.000	0.000
	10	5	0.8	346.790	2.955	0.370	3.808	0.000	0.000
	50	5	0.2	612.896	40.217	0.341	13.242	0.000	0.000
	50	5	0.8	405.272	28.147	0.415	12.732	0.000	0.000
	100	5	0.2	695.324	225.788	6.864	368.801	0.000	0.000
	100	5	0.8	444.988	181.943	1.340	339.559	0.000	0.000
	200	5	0.2	882.252	1242.115	43.265	796.060	0.000	0.000
	200	5	0.8	474.509	468.210	8.465	835.048	0.000	0.000
	10	8	0.2	408.438	1.456	0.075	10.163	0.000	0.000
	10	8	0.8	186.065	0.889	0.264	12.322	0.000	0.000
	50	8	0.2	512.033	35.998	3.526	13.651	0.000	0.000
	50	8	0.8	226.123	14.883	2.885	13.749	0.000	0.000
	100	8	0.2	623.689	172.760	73.261	374.570	0.000	0.000
	100	8	0.8	259.526	124.917	3.273	265.823	0.000	0.000
	200	8	0.2	618.530	701.197	10.964	558.671	0.000	0.000
	200	8	0.8	267.653	380.724	20.924	554.022	0.000	0.000
49	10	9	0.2	655.321	13.390	1.218	82.658	0.000	0.000
	10	9	0.8	297.532	17.089	1.364	83.166	0.000	0.000
	50	9	0.2	495.750	466.365	28.584	505.216	0.000	0.000
	50	9	0.8	296.957	3318.425	95.574	499.323	0.000	0.000
	100	9	0.2	780.865	1495.110	8.505	766.012	0.000	0.000
	100	9	0.8	342.313	2030.050	187.096	912.119	0.000	0.000
	200	9	0.2	845.567	-	586.498	2412.058	0.000	0.000
	200	9	0.8	367.770	-	496.786	2249.180	0.000	0.000
	10	16	0.2	521.401	6.087	3.954	63.785	0.000	0.000
	10	16	0.8	190.013	4.517	4.509	60.849	0.000	0.000
	50	16	0.2	846.331	61.404	11.109	290.568	0.000	0.000
	50	16	0.8	271.245	53.650	16.603	276.257	0.000	0.000
	100	16	0.2	789.547	160.833	28.112	630.804	0.000	0.000
	100	16	0.8	265.768	137.721	77.727	704.925	0.000	0.000
	200	16	0.2	983.534	-	32.274	1194.122	0.000	0.000
	200	16	0.8	308.910	-	101.742	1349.449	0.000	0.000

					CPLEX	IBNVS			
$\|I\|$	$\|K\|$	p	α_1	$best_{\text{sol}}$	$t(s)$	$t_{\text{best}}(s)$	$t_{\text{tot}}(s)$	$agap(\%)$	$\sigma(\%)$
100	10	20	0.2	857.705	145.074	14.833	343.789	0.000	0.000
	10	20	0.8	431.634	-	25.724	345.610	0.000	0.000
	50	20	0.2	1062.753	-	745.753	3050.791	0.026	0.031
	50	20	0.8	447.254	-	1218.261	3743.142	0.000	0.000
	100	20	0.2	967.675	-	656.619	4608.705	0.000	0.000
	100	20	0.8	498.766	-	2775.276	6718.384	0.365	0.411
	200	20	0.2	1234.361	-	6202.353	14632.469	0.914	0.972
	200	20	0.8	551.642	-	6500.604	321.194	3.749	2.711
	10	33	0.2	1483.834	47.021	31.294	278.266	0.000	0.000
	10	33	0.8	471.887	135.125	69.761	304.202	0.000	0.000
	50	33	0.2	1192.543	-	499.824	1590.892	0.047	0.094
	50	33	0.8	389.868	-	395.637	1700.153	0.000	0.000
	100	33	0.2	1381.736	-	1574.086	3849.464	0.000	0.000
	100	33	0.8	434.073	-	2115.577	4313.489	0.373	0.278
	200	33	0.2	1417.510	-	2179.654	6732.323	0.022	0.034
	200	33	0.8	445.718	-	1893.904	6837.655	0.904	0.872
150	10	30	0.2	450.721	2743.217	401.359	1003.563	0.163	0.147
	10	30	0.8	268.365	-	465,992	1087.061	0.382	0.493
	50	30	0.2	2166.591	-	1849.255	4601.754	0.000	0.000
	50	30	0.8	666.464	-	1579.949	4212.761	0.063	0.077
	100	30	0.2	1083.215	-	1215.744	7171.615	0.000	0.000
	100	30	0.8	426.489	-	1702.338	7027.073	0.000	0.000
	200	30	0.2	2702.981	-	2532.393	13254.610	0.000	0.000
	200	30	0.8	800.561	-	5280.354	16920.274	0.000	0.000
	10	50	0.2	1682.167	228.967	177.205	677.370	0.004	0.003
	10	50	0.8	471.71	228.274	188.397	664.941	0.067	0.097
	50	50	0.2	1790.909	-	645.762	2787.381	0.008	0.016
	50	50	0.8	500.610	-	1565.394	3571.708	0.319	0.278
	100	50	0.2	2262.302	-	3494.427	8124.979	0.009	0.008
	100	50	0.8	616.743	-	3909.621	8656.295	0.126	0.110
	200	50	0.2	2898.129	-	2299.736	11297.618	0.015	0.011
	200	50	0.8	778.607	-	3267.213	11981.065	0.183	0.154

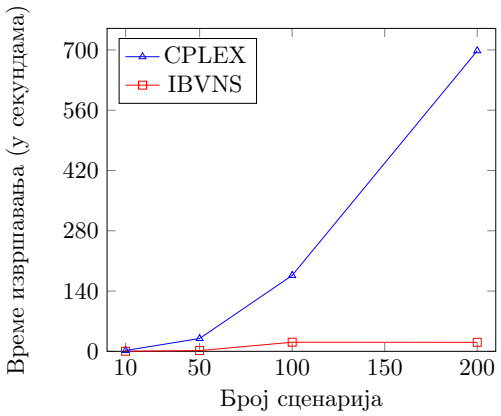
Табела 4.3: Резултати тестирања IBVNS-а и CPLEX-а

Тестирање је извршено над укупно 80 инстанци, од чега је CPLEX успешно решио проблем за 50 инстанци. На основу резултата из табеле 4.3, за сваки од разматраних бројева корисника ($|I| \in \{10, 25, 49, 100, 150\}$) израчуната су просечна времена, према броју сценарија, потребна за налажење најбољег познатог решења. Ова времена су за CPLEX рачуната на основу података из

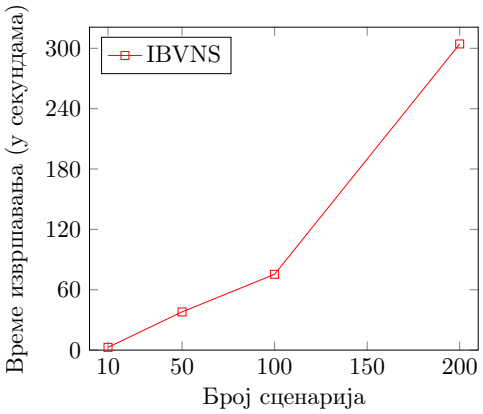
колоне $t(s)$, а за IBVNS из колоне $t_{\text{tot}}(s)$. Резултати су приказани у оквиру слике 4.1. Како CPLEX само за $|I| = 10$ и $|I| = 25$ налази решења за све величине сценарија, за те вредности $|I|$ су поред времена IBVNS-а приказана и времена CPLEX-а на сликама 4.1а и 4.1б. На основу резултата приказаних у оквиру слике 4.1, може се видети да сложеност проблема расте са порастом броја сценарија. Стога, није чудно да CPLEX успева да нађе оптимална решења за инстанце са већим бројем корисника и мањим бројем сценарија (видети резултате у табели 4.3, за нпр. $|I| = 100$ и $|K| = 10$), а не успева да реши проблем за мањи број корисника при већем броју сценарија (видети резултате у табели 4.3, за нпр. $|I| = 49$ и $|K| = 200$). За инстанце мањих димензија и CPLEX и IBVNS успевају да нађу оптимална решења, с тим што IBVNS за сваку од разматраних инстанци долази до оптималног решења брже од CPLEX-а. На сликама 4.1а и 4.1б види се приметна разлика у временима извршавања CPLEX-а и IBVNS-а. У табели 4.3 се може видети да IBVNS решава проблем 34.721 пута брже од CPLEX-а у најбољем случају (за $|I| = 49$, $|K| = 50$, $p = 9$, $\alpha_1 = 0.8$), а 1.002 пута брже у најгорем случају (за $|I| = 49$, $|K| = 10$, $p = 16$, $\alpha_1 = 0.8$). Узимајући у обзир само инстанце над којима обе методе долазе до оптималног решења, просечно време налажења оптималног решења за CPLEX износи 299.566 секунди, а за IBVNS 30.487 секунди. На основу чега се може закључити да IBVNS у просеку решава проблем 9.827 пута брже од CPLEX-а, над разматраним инстанцама. Просечна вредност *gap*-а за IBVNS, разматрана у односу на све тестиране инстанце, износи 0.108%. Просечна вредност стандардне девијације износи 0.085%. На основу ових резултата може се закључити да се IBVNS показао као стабилна и ефикасна метода за решавање проблема p-центра са поузданом мрежом.



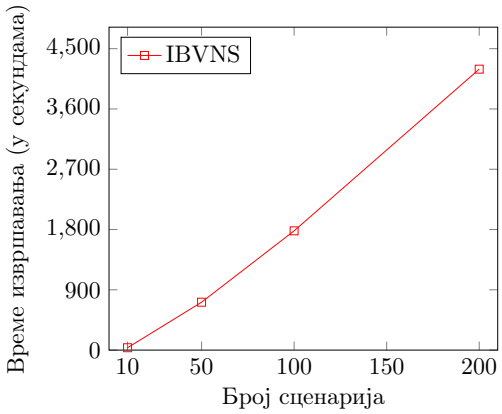
(а) $|I| = 10$



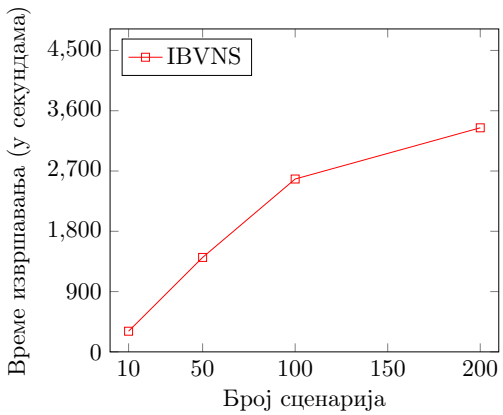
(б) $|I| = 25$



(в) $|I| = 49$



(г) $|I| = 100$



(д) $|I| = 150$

Слика 4.1: Утицај броја сценарија на времена извршавања. Случајеви су груписани према броју корисника $|I|$.

Глава 5

Закључак

У оквиру овог рада разматран је проблем p-центра са поузданом мрежом. Проблем је типа p-центра, при чему је циљ да решење проблема буде отпорно на могуће промене у мрежи. Односно, потребно је да квалитет решења остане колико је могуће добар у случају да дође до онеспособљавања неког ресурса и промена у захтевима корисника и растојањима. Решавање проблема подразумева иницијално успостављање p ресурса и алокацију корисника, као и накнадну реалокацију корисника у складу са новонасталим сценаријима. Сценарији садрже информацијаме о онеспособљеним ресурсима и осталим измењеним улазним подацима. Са порастом броја сценарија који се узимају у разматрање, као и са порастом броја корисника, расте сложеност проблема. Имајући то у виду, очекивано је да егзактне методе неће бити ефикасне у разматрању већег броја сценарија. Већи број сценарија даје већу робусност, односно већу сигурност да ће се нађено решење показати као добро у случају да дође до поремећаја ресурса. Како су у [26] разматране само егзактне методе, а у циљу постизања бољих резултата над већим инстанцама, за решавање проблема предложена је метахеуристичка метода. Конкретније, предложена је варијанта методе променљивих околина (IBVNS).

IBVNS је упоређен са три егзактне методе разматране у [26] и са резултатима постигнутим помоћу CPLEX решавача. IBVNS је показао најбоље перформансе у односу на све разматране методе. За разлику од метода из [26], IBVNS је за сваку инстанцу решио проблем до оптималности и то за мање времена. Највећи број корисника разматран у оквиру инстанци у [26] је 60, а највећи број сценарија 200. У оквиру овог рада разматране су и инстанце већих димензија, које садрже 100 и 150 корисника. Перформансе над

овим инстанцама упоређене су са CPLEX-ом. Над већим инстанцама IBVNS за разлику од CPLEX-а враћа решења која су у најмању руку допустива. Поред тога је примећено да се линеарна реформулација робусног двостепеног модела може додатно упростити за посматрани проблем и дата је нова формулација.

Будући рад се може фокусирати на унапређивању IBVNS алгоритма. На пример, може се урадити паралелизација у оквиру рачунања вредности функције циља (начин рачунања вредности функције циља дат је у оквиру секције 3.2). Прецизније, како се допуњавања врста матрица tot, као и ажурирања елемената низова $minResPair$ могу вршити независно једна од других, ове операције се могу вршити паралелно. Такође, како структура $saved$ садржи по једну матрицу tot и низ $minResPair$ за сваки сценарио, ажурирања се могу вршити паралелно за све сценарије, као и за период пре поремећаја. На овај начин постићи ће се додатно убрзање алгоритма. Поред паралелизације, може се разматрати имплементација која користи RVNS уместо BVNS-а. Уз овакав приступ очекујемо побољшања у погледу времена извршавања, али и решења лошијег квалитета над великим инстанцама. Стога, има смисла разматрати коришћење имплементације која користи RVNS за инстанце мањих димензија, а IBVNS-а над инстанцама већих димензија. Такође, може се разматрати примена IBVNS-а на друге проблеме који су по природи слични разматраном.

Библиографија

[1] Ahuja, R., Magnanti, T., Orlin, J. Network flows. 1993.

[2] Álvarez-Miranda, E., Fernández, E., Ljubić, I. The recoverable robust facility location problem. *Transportation Research Part B: Methodological*, 79:93–120, 2015.

[3] An, Y., Zeng, B., Zhang, Y., Zhao, L. Reliable p-median facility location problem: Two-stage robust models and algorithms. *Transportation Research Part B: Methodological*, 64:54–72, 2014.

[4] Atamturk, A., Zhang, M. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55:662–673, 2007.

[5] Baron, O., Milner, J., Naseraldin, H. Facility location: A robust optimization approach. *Production and Operations Management*, 20, 2011.

[6] Bayram, V., Yaman, H. Shelter location and evacuation route assignment under uncertainty: A benders decomposition approach. *Transportation Science*, 52, 2017.

[7] Ben-Tal, A., Ghaoui, L., Nemirovski, A. *Robust Optimization*. 2009.

[8] Ben-Tal, A., Goryashko, A., Guslitzer, E., Nemirovski, A. Adjustable robust solutions of uncertain linear programs1. *Mathematical Programming*, 99:351–376, 2004.

[9] Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[10] Berman, O., Krass, D., Menezes, M. Facility reliability issues in network p-median problems: Strategic centralization and co-location effects. *Operations Research*, 55:332–350, 2007.

[11] Bertsimas, D., Brown, D., Caramanis, C. Theory and applications of robust optimization. *SIAM Review*, 53, 2010.

[12] Brandeau, M. L., Chiu, S. S. An overview of representative problems in location research. *Management Science*, 35(6):645–674, 1989.

[13] Caunhye, A. M., Zhang, Y., Li, M., Nie, X. A location-routing model for prepositioning and distributing emergency supplies. *Transportation Research Part E: Logistics and Transportation Review*, 90:161–176, 2016.

[14] Clay Mathematics Institute. Millennium problems. `https://www.claymath.org/millennium-problems`, 2000. Poslednji put pristupljeno 18. jula 2022.

[15] Charnes, A., Cooper, W. Chance-constrained programming. *Management Science*, 6:73–79, 1959.

[16] Conejo, A. J. Adaptive robust optimization and its applications to power systems. `https://youtu.be/Zk6y8joQLNQ`, 2019. Poslednji put pristupljeno 20. jula 2022.

[17] Cui, T., Ouyang, Y., Shen, M. Reliable facility location design under the risk of disruptions. *University of California Transportation Center, University of California Transportation Center, Working Papers*, 58, 2010.

[18] Cvetković, D., Čangalović, M., Kovačević-Vujčić, V., Dugošija, D., Simić, S., Vuleta, J. *Kombinatorna optimizacija*. DOPIS, 1996.

[19] Dantzig, G. B. Linear programming under uncertainty. *Management Science*, 1(3/4):197–206, 1955.

[20] Daskin, M. S. Network and discrete location: Models, algorithms, and applications - chapter specific files. `https://daskin.engin.umich.edu/books/network-discrete-location/`. Poslednji put pristupljeno 10. marta 2022.

[21] Daskin, M. S. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, 2nd edition, 2013.

[22] Dongarra, J. J. Performance of various computers using standard linear equations software. *University of Manchester*, 2014.

[23] Dražić, Z. *Modifikacije metode promenljivih okolina i njihove primene za problem rasporedjivanja prenosa datoteka*. Doktorska disertacija, Matematički fakultet, Univerzitet u Beogradu, 2014.

[24] Drezner, Z. *Facility Location: A Survey of Applications and Methods*. Springer, 1995.

[25] Drezner, Z., Hamacher, H. W. *Facility location: Applications and theory*. 2001.

[26] Du, B., Zhou, H., Leus, R. A two-stage robust model for a reliable p-center facility location problem. *Applied Mathematical Modelling*, 77:99–114, 2020.

[27] Du, D. Z., Pardalos, P. M. *Minimax and Applications*. Kluwer Academic Publishers, 1995.

[28] Farahani, R. Z., Hekmatfar, M. Facility location: Concepts, models, algorithms and case studies. *Media*, 2009.

[29] Fisher M. L. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.

[30] Gabrel, V., Lacroix, M., Murat, C., Remli, N. Robust location transportation problems under uncertain demands. *Discrete Applied Mathematics*, 164:100–111, 2014.

[31] Geoffrion, A. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[32] Glover, F., Kochenberger, G. *Handbook of metaheuristics*. Kluwer Academic Publishers, 2003.

[33] Goerigk, M., Lendl, S., Wulf, L. Two-stage robust optimization problems with two-stage uncertainty. *European Journal of Operational Research*, 302(1):62–78, 2022.

[34] Goldreich, O. *P, NP, and NP-Completeness: The Basics of Complexity Theory*. 2010.

[35] Govindan, K., Fattahi, M., Keyvanshokooh, E. Supply chain network design under uncertainty: A comprehensive review and future research directions. *European Journal of Operational Research*, 2017.

[36] Guenin, B., Könemann, J., Tunçel, L. *A Gentle Introduction to Optimization*. 2014.

[37] Hansen, P., Mladenović, N., Moreno-P erez, J. A. Variable neighbourhood search: methods and applications. *4OR*, 6:319–360, 2010.

[38] Jia, H., Ordóñez, F., Dessouky M. A modeling framework for facility location of medical services for large-scale emergencies. *Iie Transactions*, 39:41–55, 2007.

[39] Laporte, G., Nickel, S., Saldanha da Gama, F. *Location Science*. Springer, 2015.

[40] Liebchen, C., Lübbecke, M., Möhring, R., Stiller, S. *The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications*. 2009.

[41] Ljubić, I., Mutzel, P., Zey, B. Stochastic survivable network design problems: Theory and practice. *European Journal of Operational Research*, 256(2):333–348, 2017.

[42] Louveaux, F. V. Discrete stochastic location models. *Annals of Operations Research*, 6(2):23–34, 1986.

[43] Mišković, S. *Rešavanje klase min-max problema robusne diskretne optimizacije sa primenama*. Doktorska disertacija, Matematički fakultet, Univerzitet u Beogradu, 2016.

[44] Mladenović, N. *Kontinualni lokacijski problemi*. Matematički institut SANU, 2004.

[45] Mladenović, N., Hansen, P. Variable neighborhood search. *Computers Operations Research*, 24(11):1097–1100, 1997.

[46] Mladenović, N., Labbé, M., Hansen, P. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:48–64, 2003.

[47] PassMark Software. Cpu benchmarks. `http://www.cpubenchmark.net/`. Poslednji put pristupljeno 20. avgusta 2022.

[48] Rao, S. *Engineering Optimization Theory and Practice*. Wiley, 5th edition, 2019.

[49] Shen, Z. J. M., Zhan, R. L., Zhang, J. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS J. on Computing*, 23(3):470–482, 2011.

[50] Snyder, L. Facility location under uncertainty: A review. *IIE Transactions*, 38, 2004.

[51] Snyder, L. V., Atan, Z., Peng, P., Rong, Y., Schmitt, A. J., Sinsoysal, B. OR/MS models for supply chain disruptions: a review. *IIE Transactions*, 48(2):89–109, 2016.

[52] Snyder, L. V., Daskin, M. S. Reliability models for facility location: The expected failure cost case. *Transportation Science*, 39(3):400–416, 2005.

[53] Swersey, A. J.. The deployment of police, fire, and emergency medical units. *Handbooks in Operations Research and Management Science*, 6:151–200, 1994.

[54] Talbi, E. G. *Metaheuristics: From Design to Implementation*. 2009.

[55] Wigderson, A. P, NP and mathematics - a computational complexity perspective. *Proceedings of the International Congress of Mathematicians*, 1:665–712, 2006.

[56] Yang, X. S. *Optimization Techniques and Applications with Examples*. 2018.

[57] Yu, G., Kouvelis, P. *On Min-Max Optimization of a Collection of Classical Discrete Optimization Problems*, 157–171, 1995.

[58] Zeng, B., Zhao, L. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.

Биографија аутора

Јована Рађеновић рођена је 7. јуна 1996. у Београду. Похађала је Основну школу „Браћа Барух" и паралелно са њом Музичку школу „Јосиф Маринковић". Након тога завршила је Прву београдску гимназију. Математички факултет у Београду, смер Примењена математика, уписала је 2015. године. Након чега је 2020. на Математичком факултету уписала мастер студије и положила све испите предвиђене програмом студија.

Током студија 2020. обавила је праксу на Математичком институту Српске академије наука и уметности. Од априла 2022. до данас, ради као сарадник у настави на Рударско-геолошком факултету у Београду, на Катедри за примењену математику и информатику.