Универзитет у Београду

Математички факултет

Оптимизација трошкова у мрежи снабдевања применом методе променљивих околина

Мастер рад

Студент:

Перица Трајков
1023/2012

Ментор:

проф. др Зорица Станимировић
Математички факултет
Универзитет у Београду

Београд, 2016.

Студент:

Перица Трајков
1023/2012

Ментор:

проф. др Зорица Станимировић
Математички факултет
Универзитет у Београду

Чланови комисије:

проф. др Миодраг Живковић
Математички факултет
Универзитет у Београду

проф. др Мирослав Марић
Математички факултет
Универзитет у Београду

Оптимизација трошкова у мрежи снабдевања применом методе променљивих околина

Апстракт: Локацијски проблеми представљају значајну класу проблема оптимизације са широком применом, посебно при оптимизацији трошкова транспортних и телекомуникацијских мрежа. Предмет проучавања у овом раду је локацијски проблем неограничених капацитета са једним извором снабдевања и више типова производа (енгл. Uncapacitated single-source multi-product facility location problem, USSMP-FLP). С обзиром на то да је овај проблам НП-тежак, развијен је алгоритам методе променљивих околина (енгл. Variable neighborhood search, VNS) за решавање овог проблема. У алгоритму су дефинисана три типа околина које се користе у фази „размрдавања“, а које обезбеђују добру диверсификацију претраге. У алгоритму је коришћена и варијанта локалног претраживања (енгл. Local search, LS), која ефикасно води алгоритам методе променљивих околина до решења високог квалитета. Предложени VNS алгоритам је тестиран на постојећим инстанцама за разматрани проблем. Најбоља решења добијена предложеним алгоритмом упоређена су са оптималним или тренутно најбољим познатим решењима из литературе. Експериментални резултати показују да предложена метода променљивих околина у највећем броју случаја достиже оптимална или најбоља позната решења у краћем процесорском времену израчунавања у односу на постојеће методе, док су у случају тест инстанци великих димензија, најбоља позната решења из литературе побољшана.

Кључне речи: Локацијски проблем, метода променљивих околина, мреже снабдевања, транспортне и телекомуникацијске мреже.

Cost optimization in supply networks by using Variable Neighborhood Search

Abstract: Facility location problems are an important class of optimization problems widely used in practice, especially when optimizing transportation and telecommunication networks. This study deals with the Uncapacitated single-source multi-product facility location problem (USSMP-FLP). Since this problem is NP-hard, exact methods can provide optimal solution for small-size instances only. Therefore, a Variable Neighborhood Search (VNS) metaheuristic is developed as a solution method to USSMP-FLP. The proposed VNS uses three types of neighborhoods that are adapted to the considered problem. All three neighborhoods are explored within the shaking phase, ensuring good searching diversification. The proposed VNS also involves a variant of Local Search (LS) algorithm that efficiently leads the VNS to high-quality solutions. The proposed VNS is benchmarked on three existing test data sets for the considered problem. The best VNS solutions are compared with the optimal or best-known ones from the literature. Computational experiments show that the proposed VNS in most of the cases reaches the optimal or best-known solutions in short CPU running times, while in the case of large problem dimensions, best-known solutions from the literature are improved.

Keywords: Facility location problem, variable neighborhood search, supply networks, transportation and telecommunication networks.

Садржај

1 Увод

Локацијски проблеми (енгл. Facility location problems, FLP) чине значајну класу проблема оптимизације, који налазе широку примену у пракси, посебно при оптимизацији трошкова у транспортним и телекомуникацијским мрежама. У општем случају, локацијски проблеми се састоје у одређивању оптималних локација за изградњу постројења која пружају неку услугу снабдевања, као и придруживање сваког корисника једном или више успостављених постројења, тако да се минимизују трошкови изградње мреже снабдевача и укупни транспортни трошкови, при условима који зависе од карактеристика конкретне мреже. Већина локацијских проблема спада у класу НП-тешких проблема, за које не постоји ефикасан алгоритам који би у полиномијалном времену нашао оптимално решење. Због тога се за њихово решавање често користе разне хеуристичке методе, које достижу позната оптимална решења или дају решења високог квалитета у кратком времену извршавања. Значај локацијских проблема лежи у великом броју примена, као што су управљање дистрибуцијом производа, унапређивање транспортне мреже, унапређивање телекомуникацијских мрежа, али и других система од јавног значаја, као што су здравствени системи, системи за пружање хитне помоћи, мреже продајних објеката, банака, школа, итд.

1.1 Историјат локацијских проблема

Историја локацијских проблема сеже назад све до Пјера де Ферма[1], за кога многи аутори сматрају да је поставио први локацијски проблем. Проблем који је Ферма формулисао састоји се у налажењу тачке унутар троугла такве да је сума растојања тражене тачке до сваког од темена троугла минимална. Овај проблем је познатији као проблем одређивања геометријске медијане.

На практични значај локацијских проблема први је указао Алфред Вебер[2]. Са њим је и формално започело изучавање теорије локацијских проблема. Он је 1929. године формулисао Веберов проблем, један од најпознатијих проблема у теорији локацијских проблема [38]. Веберов проблем је практичне природе – пронаћи оптималну позицију постројења, тако да укупни траснпортни трошкови од постројења до посматраног скупа

[1] Пјер де Ферма (Pierre de Fermat, 1601-1665), француски математичар и правник. Био је један од најзначајнијих математичара 17. века, између осталог бавио се теоријом бројева, алгебром, вероватноћом, геометријом и био зачетник диференцијалног рачуна.

[2] Алфред Вебер (Alfred Weber, 1868-1958), немачки економиста, географ и социолог, чији рад је био утицајан у развоју модерне економске географије

корисника буду минимални, при чему су задате различите цене транспорта по јединици количине робе од постројења до сваког од посматраних корисника. Овај проблем представља уопштење проблема геометријске медијане. До сада су у литератури предложене бројне методе за решавање Веберовог проблема и модификација, што је још један од показатеља његовог практичног значаја [6].

1.2 Класификација локацијских проблема

Постоји више начина за поделу локацијских проблема према различитим критеријумима. До данас је у литератури предложен велики број класификација локацијских проблема, али се ни за једну од њих не може тврдити да обухвата све локацијске проблеме. Локацијски проблеми се могу класификовати према неким од следећих карактеристика:

- Простор одлуке,
- Тип функције циља,
- Број објеката који се успоставља,
- Ограничење капацитета,
- Променљивост параметара проблема.

Према простору одлуке, локацијски проблеми се могу поделити на континуалне, дискретне и мрежне локацијске проблеме. Континуални локацијски проблеми могу за скуп потенцијалних локација објеката имати неограничен скуп тачака у равни или простору. За разлику од њих, код дискретних локацијских проблема се за локације услужних објеката може бирати само унапред одређен коначан скуп локација. Код мрежних локацијских проблема, проблем се може представити помоћу мреже, односно графа, где чворови обично представљају потражњу корисника, гране представљају успостављену везу између корисника и услужног објеката, а тежине грана су цене одговарајућих транспортних трошкова од услужног објекта до корисника.

Према типу функције циља, разликују се три типа локацијских проблема: min-sum, min-max и проблеми покривања. Сва три типа се односе на лоцирање тражених објеката, али се разликују по критеријумима на основу којих се врши одабир локација. Код локацијских проблема типа min-sum, функција циља минимизује укупну суму транспортних трошкова од корисника до додељених услужних објеката којима су придружени, док min-max групи припадају проблеми код којих се минимизује максимално растојање, односно транспортни трошкови између корисника и услужног објекта који му је придружен. Са друге стране, проблеми покривања не узимају експлицитно у разматрање растојање корисника до изграђеног објекта, већ подразумевају да сви корисници који су довољно близу додељеном услужном објекту (на растојању које није веће од тзв. „радијуса покривања“), могу бити опслужени са истим трошковима (слика 1.2).

Слика 1.2: Локацијски проблеми покривања

Према броју објеката који се успостављају, постоје егзогени и ендогени локацијски проблеми. Код егзогених локацијских проблема, број услужних објеката који се успоставља није унапред познат, већ зависи од резултата оптимизације. Са друге стране, код ендогених локацијских проблема, унапред је познат број услужних објеката који треба успоставити.

Према ограничењу капацитета, локацијски проблеми се деле на локацијске проблеме неограничених капацитета и локацијске проблеме ограничених капацитета. Код локацијских проблема ограничених капацитета, ограничења могу бити наметнута на капацитет услужних објеката, гране које повезују кориснике и услужне објекте и друго. Са друге стране, код локацијских проблема неограничених капацитета, оваква ограничења не постоје.

Према променљивости параметара проблема, локацијски проблеми се деле на статичке и динамичке локацијске проблеме. Статички локацијски проблеми посматрају само одређени репрезентативни период у оквиру којег су сви параметри проблема непроменљиви. Са друге стране, динамички локацијски проблеми разматрају више периода који су међусобно повезани и укључују параметре који се мењају током разматраних периода.

Детаљан преглед типова локацијских проблема, заједно са областима примене, приказан је у [34].

1.3 Методе за решавање локацијских проблема

Методе за решавање локацијских проблема се могу поделити на егзактне и хеуристичке методе. Егзактне методе проналазе оптимално решење и верификују његову оптималност, али се оне не могу увек применити, нарочито при решавању инстанци проблема већих димензија. Егзактне методе су добре за решавање инстанци проблема мањих димензија, јер обично дају оптимално решење у прихватљивом времену. Међутим, ако се димензија проблема повећа, простор могућих решења постаје толико велики да егзактне методе не могу у прихватљивом времену дати оптимално решење, а понекад чак ни допустиво решење. У случају неких НП-тешких проблема, дешава се да егзактне методе не могу решити чак ни инстанце проблема мањих димензија, или је време извршавања неприхватљиво дуго са практичног аспекта. Због тога се често користе хеуристичке методе, које не могу да гарантују оптималност решења, али, уколико се добро конструишу, дају решења која су субоптимална. Хеуристичке методе често достижу оптимална решења, али се оптималност добијеног решења не може увек гарантовати. У литератури постоји велики број различитих хеуристичких метода које су предложене за решавање локацијских проблема. Међутим, не постоји универзални хеуристички метод који би се могао применити на сваки локацијски проблем, већ је имплементацију хеуристичке методе неопходно прилагодити карактеристикама разматраног проблема. Такође, може се приметити да један или више хеуристичких метода даје боље резултате за један тип локацијских проблема у односу на остале методе. Због тога су честе и хибридизације хеуристичких метода које комбинују добре аспекте појединачних хеуристичких метода у циљу добијања што квалитетнијих решења.

1.3.1 Егзактне методе

Егзактне методе које се најчешће користе за решавање локацијских проблема су:

- Метода одсецања (енгл. Cutting plane),
- Метода гранања и ограничавања (енгл. Branch-and-bound, BnB),
- Метода гранања и одсецања (енгл. Branch-and-cut, BnC).

Методу одсецања за решавање проблема целобројног програмирања предложио је Гомори у раду [12]. Уколико се у проблем целобројног програмирања уведе релаксација услова целобројности променљивих одлуке, тада се добија проблем линераног програмирања за који се може користити симплекс метод [30] за добијање решења. Уколико је решење таквог релаксираног проблема целобројно, онда је то решење и полазног проблема. Прецизније, важи теорема 1.3.1 из [12]:

Теорема 1.3.1: *Оптимално решење релаксације проблема целобројног програмирања представља горњу границу оптималног решења полазног проблема.*

Користећи ову теорему, може се конструисати линеарна неједначина која задовољава сва допустива полазна решења, а не задовољава нецелобројна решења проблема добијеног релаксацијом. Ова једначина се назива одсецајућом равни, јер одсеца део који садржи нецелобројна решења. Она се може додати скупу полазних ограничења и на тај начин би се добио проблем који је еквивалентан полазном. У суштини, поступак решавања целобројног проблема се састоји у увођењу нових одсецајућих равни и додавању истих полазном проблему све док решење релаксираног проблема (добијеног укидањем услова целобројости) не буде целобројно и тиме представља решење полазног проблема. Главна мана ове методе је спора конвергенција ка решењу.

Метода гранања и ограничавања се користи за решавање целобројног проблема оптимизације код којег је скуп претраживања коначан. Предложили су је Ланд и Доиг [24]. Главна идеја ове методе се састоји у томе да се полазни проблем разложи на више мањих и једноставнијих потпроблема од којих се неки неће ни разматрати ако се процени да не могу имати боље решење од тренутно најбољег решења. Дакле, ова метода се базира на принципу „подели па владај“. Поменута подела проблема на мање се врши ограничавањем вредности променљивих.

Алгоритам се може описати на следећи начин:

1. Поделити постојећи проблем на мање проблеме (методом гранања и ограничавања).
2. За сваки од потпроблема извршити релаксацију уклањањем захтева целобројности и решити тај проблем.
   2.1. Уколико решење релаксације потпроблема нема допустивог решења, одбацује се потпроблем.
   2.2. Уколико решење релаксације има допустиво целобројно решење, упоређује се то решење са тренутно најбољим и ажурира уколико је потребно, док се потпроблем одбацује.
   2.3. Уколико потпроблем има оптимално решење које је лошије од оптималног решења полазног проблема, тада се потпроблем одбацује.
   2.4. Уколико потпроблем има нецелобројно решење које је боље од тренутно најбољег целобројног решења полазног проблема, онда се рекурзивно дели овај потпроблем на нове потпроблеме и понавља исти поступак.

Лоша особина ове методе је њена неполиномијалност.

Метода гранања и одсецања, предложена од стране Падберга и Риналдија у [33], представља комбинацију метода одсецања равни и гранања са ограничавањем. Ова метода је до сада успешно примењена на бројне локацијске проблеме и друге проблеме

комбинаторне оптимизације као што су хаб-локацијски проблем, проблеми рутирања возила, проблеми трговачког путника и др.

1.3.2 Хеуристичке методе

Постоји велики број проблема код којих егзактне методе не могу дати добре резултате у прихватљивом временском периоду са становишта практичних потреба. То је најчешће случај код НП-тешких проблема, за које не постоје алгоритми полиномијалне сложености [11]. Због тога се за ефикасно решавање ових проблема користе разне хеуристичке методе. Предност хеуристичких метода је што у кратком времену извршавања могу дати висококвалитетна решења, што је од великог практичног значаја. Неретко су ова решења и оптимална али гаранција оптималности код њих не постоји. Међутим, за успешну примену хеуристика, неопходно је да се хеуристика прилагоди разматраном проблему. У супротном, решења могу бити незадовољавајућег квалитета, или чак недопустива. Најпознатије хеуристичке методе су:

Локална претрага (енгл. Local search, LS) је вероватно и најпознатија хеуристичка метода. Састоји се у генерисању почетног решења, а онда се почев од њега креће по околини тог решења док се не пронађе локални оптимум. Околина решења се унапред дефинише на адекватан начин, имајући у виду природу разматраног проблема. Предност ове хеуристике је једноставност, док је главна мана немогућност изласка из локалног оптимума, који често није и глобални оптимум.

Похлепни алгоритам (енгл. Greedy algorithm) се заснива на разматрању оног решења које краткорочно гледано може дати најбоље резултате. Дакле, претрага се усмерава ка тренутно најбољим решењима у нади да ће та решења бити и глобално најбоља.

Итеративна локална претрага (енгл. Iterated local search, LS), представља модификацију локалне претраге која се заснива на понављању обичне локалне претраге. Локална претрага може довести до решења која су локално оптимална али не и глобално. Таква решења постају и коначна решења методе осим ако се не уведе стратегија да се локална претрага настави изван околине достигнутог локалног оптимума. Једна од стратегија је да се локална претрага понови више пута, тако што се сваки пут започне од случајно одабраног решења, или од решења које је блиско решењу добијеном у претходним итерацијама локалне претраге.

2 Локацијски проблем неограничених капацитета са једним извором снабдевања и више типова производа

2.1 Опис проблема и математичка формулација

Локацијски проблем неограничених капацитета са једним извором снабдевања и више типова производа (енгл. Uncapacitated single-source multi-product facility location problem, USSMP-FLP) је предложен од стране Незада и сарадника у раду [31]. Да би проблем био дефинисан неопходно је увести следећу нотацију:

Нека су дати следећи скупови:

$I = \{1, \dots, m\}$ – скуп потенцијалних локација за изградњу постројења (ресурса, фабрика),

$J = \{1, \dots, n\}$ – скуп корисника,

$K = \{1, \dots, p\}$ – скуп типова производа.

Сваки од корисника из скупа J има одређену потребу за сваким од типова производа скупа K, а које обезбеђује снабдевањем из одређеног постројења које је претходно изграђено на некој од локација из скупа *I*. Свако постројење може производити само један тип производа, док више постројења могу производити исти тип производа. Целокупна потреба једног корисника за једним типом производа може бити задовољена снабдевањем из само једног изграђеног постројења. Не постоје ограничења за количину типа производа која може бити произведена у сваком од постројења. Свако од постројења има фиксну цену изградње, као и цену производње за сваки од типова производа.

Укупни трошкови обухватају иницијалне трошкове изградње постројења, трошкове производње сваког од типова производа, као и трошкове транспорта сваког типа производа од постројења у којем се производи до крајњег корисника.

Циљ проблема је одредити локације за изградњу постројења, и тип производа који ће производити свако од њих, као и одредити план снабдевања сваког корисника за сваки тип производа, тако да потребе корисника буду задовољене а укупни трошкови система снабдевања буду минимални.

Алгоритам за решавање овог проблема мора пружити одговоре на следећа питања:

- Који подскуп скупа *I* треба изабрати за изградњу постројења?
- Који тип производа ће се производити у сваком од отворених постројења?
- Како доделити кориснике изабраним постројењима?

За математичку формулацију проблема ће бити коришћене следеће ознаке:

q_{jk} – потражња производа типа $k \in K$ од стране корисника $j \in J$

c_{ijk} – транспортни трошкови производа типа $k \in K$ од постројења $i \in I$ до корисника $j \in J$

f_{ik} – иницијални трошкови производње производа типа $k \in K$ у постројењу $i \in I$

p_{ik} – трошкови производње једног производа типа $k \in K$ у постројењу $i \in I$

Користе се два скупа бинарних променљивих одлучивања:

$$x_{ijk} = \begin{cases} 1, & \text{ако се корисник } j \text{ снабдева производом типа } k \text{ из постројења } i, \\ 0, & \text{иначе,} \end{cases}$$

и

$$y_{ik} = \begin{cases} 1, & \text{ако постројење } i \text{ производи производ типа } k, \\ 0, & \text{иначе.} \end{cases}$$

Користећи наведену нотацију, локацијски проблем неограничених капацитета са једним извором снабдевања и више типова производа се може формулисати на следећи начин [31]:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} \gamma_{ijk}\, x_{ijk} + \sum_{i=1}^{m} \sum_{k=1}^{p} f_{ik}\, y_{ik} \qquad (1)$$

при условима:

$$\sum_{i=1}^{m} x_{ijk} = 1, \qquad \forall j \in J, k \in K, \qquad (2)$$

$$x_{ijk} \leq y_{ik}, \qquad \forall i \in I, j \in J, k \in K, \qquad (3)$$

$$\sum_{k=1}^{p} y_{ik} \leq 1, \qquad \forall i \in I, \qquad (4)$$

$$x_{ijk}, y_{ik} \in \{0,1\}\, \forall i \in I, j \in J, k \in K, \qquad (5)$$

и притом важи:

$\gamma_{ijk} = (c_{ijk} + p_{ik}) q_{jk}$.

Функција (1) минимизује укупне трошкове система снабдевања. Ограничење (2) обезбеђује да не постоје два постројења која опслужују истог корисника истим типом производа.

Ограничење (3) обезбеђује да корисник може бити снабдевен једним типом производа само из постројења која су успостављена и која га производе. Ограничење (4) обезбеђује да свако успостављено постојење може производити само један тип производа. Ограничење (5) дефинише бинарну природу променљивих одлучивања, тј. да оне могу узимати само вредности из скупа $\{0,1\}$.

Разматрани проблем је НП-тежак као уопштење простог локацијског проблема. Наиме, ако се посматра само један тип производа, разматрани проблем се може свести на прост локацијски проблем без ограничења капацитета за који је показано да је НП-тежак [23].

Локацисјки проблем неограничених капацитета са једним извором снабдевања и више типова производа има велики значај. Честе су ситуације да велика корпорација која производи више типова производа, услед обимности посла као и специфичности сваког од типова производа, мора да подели производњу истог типа производа на више постројења. Тада се природно поставља питање где изградити тражена постројења, и како направити план снабдевања корисника, тако да трошкови изградње, производње и транспорта буду што мањи, а потребе корисника задовољене.

2.2 Пример проблема

Нека су скупови I, J, K из поставке задатака следећи:

$I = \{F1, F2, F3\}$ – скуп од 3 постројења (фабрике) *F1*, *F2* и *F3*,

$J = \{K1, K2, K3, K4, K5\}$ – скуп од 5 корисника *K1*, *K2*, *K3*, *K4* и *K5*,

$K = \{P1, P2\}$ – скуп од 2 типа производа *P1* и *P2*.

Нека су вредности улазних параметара q_{jk}, c_{ijk}, f_{ik}, p_{ik} дате у табелама 2.2.а - 2.2.г:

Табела 2.2.а: Потражње производа типа $k \in K$ од стране корисника $j \in J$

q_{jk}	*P1*	*P2*
K1	5	0
K2	0	3
K3	2	0
K4	0	8
K5	8	0

Табела 2.2.б: Транспортни трошкови производа типа $k \in K$ од постројења $i \in I$ до корисника $j \in J$

c_{1jk}	P1	P2	c_{2jk}	P1	P2	c_{3jk}	P1	P2
K1	7	23	K1	17	8	K1	23	12
K2	5	34	K2	15	3	K2	15	43
K3	2	18	K3	33	5	K3	11	22
K4	9	22	K4	23	2	K4	15	21
K5	3	19	K5	26	4	K5	13	11

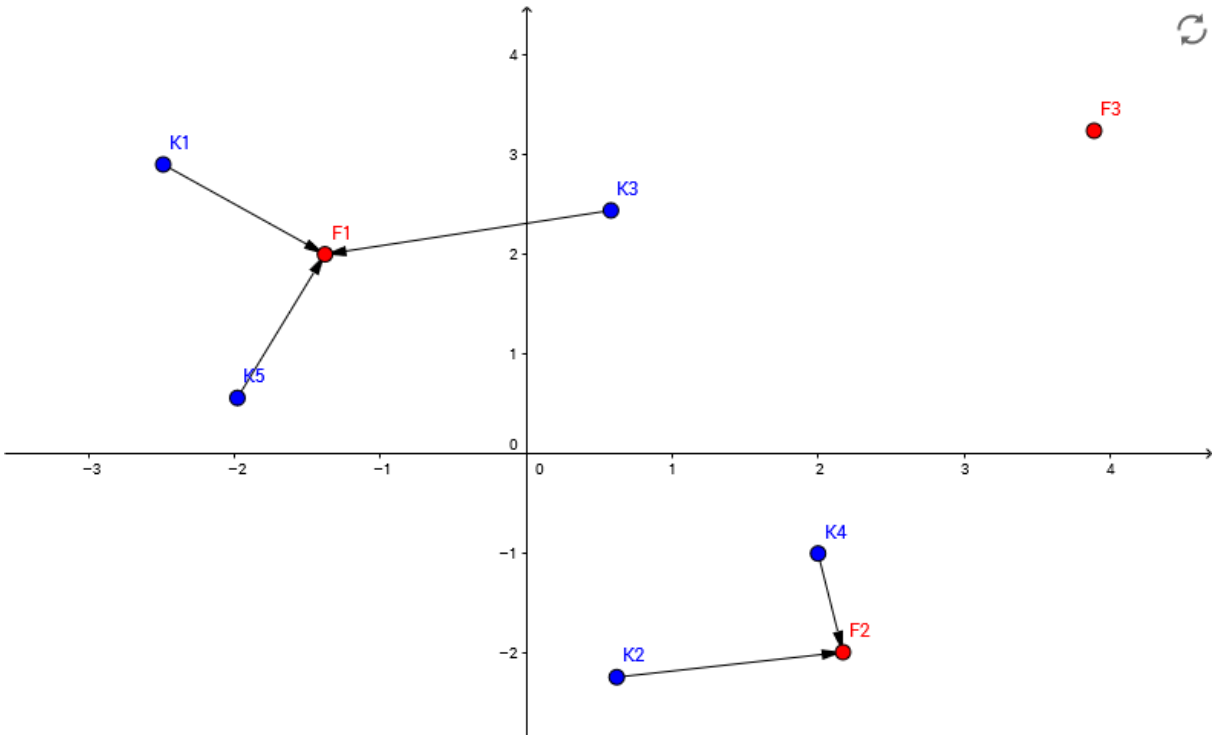
Табела 2.2.в: Иницијални трошкови производње производа типа $k \in K$ у постројењу (фабрици) $i \in I$

f_{ik}	*P1*	*P2*
F1	3	1509
F2	1667	18
F3	1233	4533

Табела 2.2.г: Трошкови производње производа типа $k \in K$ у постројењу (фабрици) $i \in I$

p_{ik}	*P1*	*P2*
F1	2	34
F2	54	1
F3	54	12

На основу табеле 2.2.а се види да корисници *K1, K3, K5* имају потребу само за производом типа *P1*, док корисници *K2* и *K4* имају потребу само за производом типа *P2*. Са друге стране, на основу табела 2.2.б, 2.2.б и 2.2.г, може се закључити да постројење *F1* има знатно повољније услове за производњу производа типа *P1*, јер су иницијални трошкови и транспортни трошкови тако знатно мањи него кад би то постројење производило производ типа *P2*. Такође, очигледно је да постројење *F2* има повољније услове за производњу производа типа *P2*, док постројење *F3* нема повољне услове за производњу било ког типа производа. Из наведених разлога, очекује се да ће у оптималном решењу овог примера проблема бити изграђене само постројења *F1* и *F2*, да ће корисницима *K1*, *K3* и *K5* бити додељено постројење *F1*, а корисницима *K2* и *K4* постројење *F2*. Оптимално решење које је добијено коришћењем комерцијалног CPLEX решавача, потврђује ова очекивања. Графички приказ додељивања корисника постројењима дат је на слици 2.2 (већа растојања на слици одговарају већим транспортним трошковима)



Слика 2.2: Графички приказ оптималног решења из примера 2.2

2.3 Преглед релевантне литературе

Локацијски проблеми су до сада доста проучавани у литератури, али су још увек актуелни, имајући у виду њихову широку примену у бројним областима. Преглед локацијских проблема са применама може се наћи у бројним прегледним радовима као што су [2], [10], [32] и други. Међутим, локацијски проблем који је тема овог рада је проучаван једино у раду Незада и сарадника [31], публикованог 2013. године. У истом раду, дата је математичка формулација проблема и предложена хеуристика заснована на Лагранжовој релаксацији.

Лагранжова реласкација је метода која се користи за решавање широког спектра проблема оптимизације, укључујући и локацијске проблеме. Први пут је описана од стране Хелда и Карпа у [18] и [19]. Ова хеуристичка метода комбинује елементе егзактних и других хеуристичких метода. Своди се на релаксацију скупа ограничења за која се зна да чине проблем тешким за решавање, и њихово укључивање у функцију циља уз додавање казне (енгл. penalty) за добијену релаксацију. Овај трансформисан проблем је лакше решити, и важи да је његово оптимално решење доња граница полазног проблема. Допустиво решење полазног проблема се добија коришћењем неке друге ефикасне хеуристичке методе. Затим

се казна додата на релаксацију прилагођава, све док је растојање између најбоље доње и горње границе довољно мало.

Лагранжова метода је у литератури примењена на разне варијанте локацијских проблема, попут локацијског проблема са капацитетима (енгл. Capacitated FLP, CFLP) где постројења имају извесно ограничење капацитета [3], затим варијанте локацијског проблема са капацитетима са једним извором снабдевања (енгл. Single source CFLP, SSCFLP), где сваки корисник може бити опслужен само из једног постројења [35], и многи други. Том идејом су се водили и аутори рада [31], који су применили методу Лагранжове релаксације на проблем локацијског проблема неограничених капацитета са једним извором снабдевања и више типова производа. Они су развили две Лагранжове хеуристике и у њих укључили три различите хеуристике локалног претраживања, како би побољшали горњу границу и убрзали процес претраге. Суседна решења у локалној претрази добијена су разменом типова производа између два постројења, разменом корисника два постројења, као и успостављањем новог постројења уз доделу одговарајућег типа производа. За добијање експерименталних резултата коришћена су три скупа тест инстанци: инстанце класе 1 (мале), инстанце класе 2 (средње) и инстанце класе 3 (велике). Због недостатка јавно доступних инстанци, ове инстанце су добијене модификацијом и надоградњом већ постојећих јавно доступних инстанци Барета [4], Билда и Крарупа [5] и Кочетова и Иваненка [22]. Класа 1 садржи 17 инстанци код којих је број корисника између 21 и 117, број локација за изградњу постројења се креће између 5 и 15, док је број типова производа између 2 и 12. Код инстанци класе 2, број корисника је између 80 и 100, број локација за изградњу постројења имеђу 30 и 100, а број типова производа између 15 и 40. Инстанце класе 3 су добијене модификацијом инстанци класе 2, како би се добиле теже инстанце, па је и распон параметара исти као и код инстанци класе 2. Анализом експерименталних резултата аутори су показали да се применом Лагранжових хеуристика добијају добре горње границе оптималних решења проблема. Поређењем резултата Лагранжових хеуристика са резултатима егзактног CPLEX решавача, може се закључити да у случају инстанци из класе 1, које су мањих димензија, хеуристике достижу оптимална решења. За инстанце из класа 2 и 3, решења Лагранжових хеуристика у неким случајевима достижу или побољшавају горње границе добијене CPLEX-ом, док су у неким случајевима решења добијена хеуристиком лошијег квалитета.

У овом раду предложена је метахеуристичка метода променљивих околина која на ефикасан начин решава разматрани проблем, односно достиже сва оптимална решења и поправља до сада позната најбоља решења презентована у раду [31]. Детаљи предложене имплементације описани су у поглављу 4.

3 Метода променљивих околина

Методу променљивих околина (енгл. Variable neighborhood search, VNS), предложили су Младеновић и Хансен у раду [28], иако се идеја о оваквом приступу јавила још 1995. године [27]. Идеја ове метахеуристике је веома једноставна: увођење више околина и систематична промена околина између локалних претрага.

Нека је дат проблем минимизације функције $f(x)$ на скупу X. Нека је $\mathcal{N}_k, k = 1, \dots, k_{max}$, коначан скуп унапред дефинисаних околина, а $\mathcal{N}_k(x)$ скуп решења у k -тој околини од x. Нека је x_{opt} – допустиво решење где је достигнут минимум функције $f(x)$ на скупу X. Са $x' \in X$ се означава локални минимум у односу на $\mathcal{N}_k$, уколико не постоји решење $x \in \mathcal{N}_k(x') \subseteq X$, такво да је $f(x) < f(x')$. Метода променљивих околина је базирана на три основне чињенице [28]:

1. Локални минимум у односу на једну околину не мора бити и локални минимум у односу на другу околину,
2. Глобални минимум представља локални минимум у односу на све околине,
3. За већину проблема оптимизације локални минимуми у односу на једну или више околина су релативно блиски један другом.

Прва чињеница даје потребан разлог за увођење различитих структура околина. Друга говори да глобални оптимум треба тражити искључиво међу локалним оптимумима, тј. уколико неко решење није локални оптимум, тада сигурно није ни глобални оптимум. Трећа чињеница имплицира да локални оптимум пружа неке информације и о глобалном оптимуму, због чега је оправдана претрага у најближој околини тог локалног оптимума, јер се ту очекује поправљање текућег глобалног оптимума.

Постоје различите варијанте методе променљивих околина: детерминистичка метода променљивих околина (енгл. Variable neighborhood descent, VND), редукована метода променљивих околина (енгл. Reduced variable neighborhood search, RVNS), основна метода промељивих околина (енгл. Basic variable neighborhood search, BNS), метода променљивих околина са декомпозицијом (енгл. Variable neighborhood decomposition search, VNDS), уопштена метода променљивих околина (енгл. General variable neighborhood search, GVNS), искошена метода променљивих околина (енгл. Skewed variable neighborhood search, SVNS), и друге. У наредним потпоглављима биће дат кратак опис сваке од наведених варијанти методе променљивих околина. У сваком од потпоглавља са $N_k, k = 1, \dots, k_{max}$ се означава коначан скуп унапред дефинисаних структура околина, док се са $N_k(x)$ означава скуп решења у k-тој околини од x. За опис детерминистичке методе променљивих околина, користи се и ознака за други коначан скуп изабраних структура околина $N'_k,\ k = 1, \dots, k'_{max}$. Избор критеријума заустављања и вредност одговарајућих параметара критеријума заустављања је значајан аспект сваке од наведених варијанти VNS методе. Критеријум

заустављања може бити максимално процесорско време, максималан број итерација, максималан број итерација при којима није дошло до побољшања решења, или нека њихова комбинација.

Метода променљивих околина се користи за решавање бројних проблема комбинаторне и глобалне оптимизације. Преглед примена VNS методе може се наћи у радовима [14] и [16]. У досадашњој литератури налазимо бројне примере успешне примене VNS методе на разне дискретне локацијске проблеме, на пример у [8], [9], [13], [21], [25], [26], [29], [36] и другим радовима.

3.1 Детерминистичка метода променљивих околина

Детерминистичка метода променљивих околина се заснива на чињеници да локални оптимимум једне околине $N_k(x)$ не мора бити и локални оптимум друге околине $N'_k(x)$. Улаз у алгоритам представљају k_{max} – број различитих околина, и x – полазно решење проблема. Основни кораци детерминистичке методе променљивих околина приказани су у алгоритму 3.1.

Алгоритам 3.1: Детерминистичка метода променљивих околина

Улаз: x, k'_{max}

```
1.  repeat
2.      k ← 1
3.      repeat
4.          x' ← argmin_{y ∈ N'_k(x)} f(x) /*Naći najboljeg suseda u N_k(x)*/
5.          if f(x') < f(x)
6.              x ← x'
7.              k ← 1
8.          else
9.              k ← k + 1
10.     until k = k'_max
11. until nije nadjeno poboljšanje
```

3.2 Редукована метода променљивих околина

Редукована метода променљивих околина се заснива на стохастичком приступу. Састоји се у систематској промени околина и избору случајног решења у свакој од околина $N_k(x)$. Кораци одлучивања базирани су на једном, случајно одабраном решењу из текуће околине, и једноставнији су јер нема процеса локалног прераживања. Овакав приступ је од велике користи код проблема великих димензија, јер се избегава сложена и дуготрајна процедура локалног претраживања. Кораци редуковане методе променљивих околина приказани су у алгоритму 3.2. Функцијом $razmrdaj\ (x,k)$ у кораку 4, генерише се тачка x' на случајан начин у k-тој околини решења x, тј. $x' \in N_k(x)$.

Алгоритам 3.2: Редукована метода променљивих околина

Улаз: x, k_{max}

1. ***repeat***
2. $k \leftarrow 1$
3. ***repeat***
4. $x' \leftarrow razmrdaj\ (x,k)$
5. ***if*** $f(x') < f(x)$
6. $x \leftarrow x'$
7. $k \leftarrow 1$
8. ***else***
9. $k \leftarrow k+1$
10. ***until*** $k = k_{max}$
11. ***until*** *kriterijum zaustavljanja je zadovoljen*

3.3 Основна метода променљивих околина

Основна метода променљивих околина представља комбинацију детерминистичке и редуковане методе променљивих околина. Састоји се у детерминистичкој промени околина, случајном избору почетног решења у тренутној околини и примени локалне претраге из тако добијеног решења. Основна метода променљивих околина је најчешће коришћена у литератури у односу на остале варијанте VNS методе. Кораци основне методе променљивих околина дати су у алгоритму 3.3.

Алгоритам 3.3: Основна метода променљивих околина

Улаз: x, k_{max}

```
1.  repeat
2.      k ← 1
3.      repeat
4.          x' ← razmrdaj(x, k) /*u odnosu na okolinu N_k*/
5.          x'' ← lokalna_pretraga(x', k) /*u odnosu na okolinu N_k*/
6.          if f(x'') < f(x')
7.              x ← x''
8.              k ← 1
9.          else
10.             k ← k + 1
11.     until k = k_max
12. until kriterijum zaustavljanja je zadovoljen
```

Корак 4. у алгоритму 3.3 на случајан начин бира решење у околини $N_k(x)$, док се у кораку 5 врши локална претрага почев од случајног решења добијеног у претходном кораку. Постоје три варијанте локалне претраге, једна, где се претрага завршава чим се пронађе боље суседно решење (енгл. First-improvement LS), друга, где се процес претраге наставља све док се околина потпуно не претражи и не пронађе локални оптимум (енгл. Best-improvement LS) и трећа, где се процес претраге наставља све док се не постигне тачно k побољшања решења (енгл. k-th improvement LS).

3.4 Метода променљивих околина заснована на декомпозицији

Методу променљивих околина засновану на декомпозицији предложили су Хансен и сарадници у [15]. Ова метода проширује основну методу променљивих околина у двонивовску варијанту методе, тако што раставља полазни проблем на мање потпроблеме. Ово проширење је засновано на декомпозицији проблема на мање потпроблеме. Показало се да тако добијена метода генерално брже долази до решења у поређењу са основном методом променљивих околина, посебно у случају решавања инстанци великих димензија. Кораци методе променљивих околина засноване на декомпозицији, приказани су у алгоритму 3.4, у коме је са t_d означен параметар који представља време решавања мањих проблема добијених декомпозицијом. Не губећи на општости, може се претпоставити да решење x представља скуп само неких елемената. Са y је означен скуп од k својстава решења x' којих нема у x $(y \leftarrow x' \setminus x)$. Параметар y' представља локални оптимум са

својствима из скупа y. Са x'' се обележава оптимално решење на целом простору X $(x'' \leftarrow (x' \setminus y) \cup y')$, док x''' означава локални оптимум у односу на цео простор X користећи x'' као почетно решење.

Алгоритам 3.4: Метода променљивих околина заснована на декомпозицији

Улаз: x, k_{max}, t_d

1. **repeat**
2. $k \leftarrow 2$
3. **repeat**
4. $x' \leftarrow razmrdaj(x,k)$; /*u odnosu na okolinu N_k*/
5. $y \leftarrow x' \setminus x$
6. $y' \leftarrow BVNS(y,k,t_d)$; $x'' \leftarrow (x'\ y) \cup y'$
7. $x''' \leftarrow lokalna\_pretraga\ (x'')$
8. **if** $f(x''') < f(x)$
9. $x \leftarrow x'''$
10. $k \leftarrow 1$
11. **else**
12. $k \leftarrow k + 1$
13. **until** $k = k_{max}$
14. **until** *kriterijum zaustavljanja je zadovoljen*

Поред максималног времена извршавања, као критеријум заустављања у методи променљивих околина заснованој на декомпозицији, може се користити и други критеријум за целу методу као и за потпроблеме, попут максималног броја итерација за целу методу и максималног броја итерација за сваки од мањих потпроблема.

3.5 Уопштена метода променљивих околина

Уопштена метода променљивих околина представља модификацију основне методе променљивих околина. Све што важи за основну методу променљивих околина, важи и овде, осим што се уместо корака локалне претраге користи детерминистичка метода променљивих околина. Примена ове методе дала је одличне резултате код неких проблема (видети [1], [6], [7]). Кораци уопштене методе променљивих околина приказани су у алгоритму 3.5.

Алгоритам 3.5: Уопштена метода променљивих околина

Улаз: x, k_{max}, k'_{max}

```
1.  repeat
2.      k ← 1
3.      repeat
4.          x' ← razmrdaj (x, k) /*u odnosu na okolinu N_k*/
5.          x'' ← VND(x', k'_max)
6.          if f(x'') < f(x')
7.              x ← x''
8.              k ← 1
9.          else
10.             k ← k + 1
11.     until k = k_max
12. until kriterijum zaustavljanja je zadovoljen
```

3.6 Искошена метода променљивих околина

Искошену методу променљивих околина предложили су Хансен и сарадници у [17]. Заснива се на истраживању делова простора решења који се налазе даље од тренутно најбољег пронађеног решења. Када се пронађе најбоље решење у некој области претраге, неопходно је прећи у неку удаљенију област претраге у нади да ће тај скок довести до проналажења бољих решења. Уколико би се прелазак вршио на случајан начин у удаљеним областима, добила би се решења која се доста разликују од тренутно најбољег решења, али би се ова метода у некој мери трансформисала у вишестартну хеуристику (енгл. Multistart heuristic), која није веома ефикасна. Из тог разлога, мора се направити одређена врста компромиса о удаљености решења у које се прелази од тренутно најбољег решења. Ова идеја представља основу искошене методе променљивих околина. Кораци искошене методе променљивих околина приказани су у алгоритму 3.6. У алгоритму се користи функција $\rho(x, x'')$ за мерење растојања од тренутно најбољег решења x до нађеног локалног оптимума x''. Уколико за локални оптимум важи да је $f(x'') - \alpha\rho(x, x'') < f(x)$, тј. уколико је услов компромиса о удаљености испуњен, врши се померање простора претраге. Параметар α се бира тако да се омогући истраживање делова простора решења која су далеко од x када је $f(x'')$ мало веће од $f(x)$.

Алгоритам 3.6: Искошена метода променљивих околина

Улаз: x, k_{max}, α

1. **repeat**
2. $k \leftarrow 1$; $x_{best} \leftarrow x$
3. **repeat**
4. $x' \leftarrow razmrdaj\,(x,k)$ /*u odnosu na okolinu N_k*/
5. $x'' \leftarrow lokalna\_pretraga(x')$
6. **if** $f(x_{best}) < f(x)$
7. $x \leftarrow x_{best}$
8. **if** $f(x'') - \alpha\rho(x,x'') < f(x)$
9. $x \leftarrow x''$
10. $k \leftarrow 1$
11. **else**
12. $k \leftarrow k + 1$
13. **until** $k = k_{max}$
14. $x \leftarrow x_{best}$
15. **until** *kriterijum zaustavljanja je zadovoljen*

4 Примена основне методе променљивих околина на проблем неограничених капацитета са једним извором снабдевања и више типова производа

У овом поглављу биће објашњена реализација основне методе променљивих околина, прилагођена разматраном проблему. У наредним потпоглављима дати су описи битних елемената предложене методе.

4.1 Основна структура алгоритма

Основни део програма се налази унутар процедуре која учитава улазне податке, налази почетно решење, и имплементира основну методу променљивих околина. Псеудокод ове процедуре је дат у алгоритму 4.1.

Алгоритам 4.1: Структура предложене основне методе променљивих околина

Улаз: из датотеке

```
1.  ucitaj_ulazne_podatke
2.  generisi_pocetno_resenje
3.  najbolja_vrednost ← funkcija_cilja(x_najbolje);
4.  k ← 0; nema_poboljsanja ← 0
5.  while nema_poboljsanja < max_nema_poboljsanja
6.      k ← 0
7.      while k < kmax
8.          x′ ← razmrdaj(k,x_najbolje)
9.          x′′ ← lokalna_pretraga(k, x′)
10.         trenutna_vrednost ← funkcija_cilja(x′′)
11.         if trenutna_vrednost < najbolja_vrednost
12.             kopiraj(x′′,x_najbolje)
13.             nema_poboljsanja ← 0
14.             najbolja_vrednost=trenutna_vrednost
15.             k ← 0
16.         else
17.             k ← k + 1
18.     nema_poboljsanja++
```

Након учитавања улазних података, алгоритам конструише почетно решење. Затим се извршава спољашња петља све док не буде испуњен услов заустављања. Услов заустављања у имплементацији алгоритма 4.1 представља максималан број итерација спољашње петље који је означен параметром `max_nema_poboljsanja`. У оквиру спољашње петље редом се извршавају процедуре *razmrdaj*, *lokalna_pretraga* као и ажурирање најбољег решења. Алгоритам је имплементиран тако да се не генеришу кодови који одговарају неком недопустивом решењу. О детаљима имплементације свих процедура, као и осталим битним аспектима алгоритма, биће речи у наредним потпоглављима.

4.2 Репрезентација решења

Свако решење је представљено структуром коју чини комбинација три помоћне подструктуре: postrojenje_proizvod, proizvod_postrojenje и postrojenje_korisnik. Подструктура postrojenje_proizvod представља једнодимензиони низ дужине *m* чији елементи одговарају постројењима, а вредности индексу типа производа које одговарајуће потројење производи. У случају да постројење не производи нити један тип производа, одговарајућа вредност елемента низа је -1. Подструктура *proizvod_postrojenje* представља једнодимензиони низ дужине *p*, чији елементи одговарају типовима производа, а вредности одговарају листи постројења у којима се проиводи одговарајући тип производа. У случају да се тип производа не производи ни у једном од постројења, одговарајућа вредност елемента низа је null показивач. Подструктура *postrojenje_korisnik* представља дводимензиону бинарну матрицу формата *m* x *n*, чије врсте одговарају постројењима а колоне корисницима, а вредности елемената матрице су 1, уколико постројење снабдева одговарајућег корисника, или 0, у супротном. Иако су структуре *postrojenje_proizvod* и *proizvod_postrojenje* међусобно еквивалентне по садржини информација, обе се користе у опису решења због ефикасности алгоритма. Пример кодирања оптималног решења које одговара проблему из поглавља 2.2 дат је у наставку у табелама 4.2.а, 4.2.б и 4.2.в:

Табела 4.2.а: Једнодимензиони бинарни низ *postrojenje_proizvod*

postrojenje_proizvod[0]	*postrojenje_proizvod[1]*	*postrojenje_proizvod[3]*
0	1	-1

Табела 4.2.б: Једнодимензиони бинарни низ листи постројења *proizvod_postrojenje*

proizvod_postrojenje[0]	*proizvod_postrojenje[1]*
0 =>	1 =>
null	null

Табела 4.2.в: Бинарна дводимензиона матрица придруживања корисника постројењима *postrojenje_korisnik*

postrojenje_korisnik	*K1*	*K2*	*K3*	*K4*	*K5*
F1	1	0	1	0	1
F2	0	1	0	1	0
F3	0	0	0	0	0

4.3 Генерисање почетног решења

Генерисање квалитетног почетног решења од великог је значаја за квалитет крајњег решења и брзину извршавања алгоритма. Што је почетно решење квалитетније, већа је вероватноћа да ће VNS алгоритам добити крајње решење које је високог квалитета у краћем времену извршавања. Због тога је значајна пажња пружена конструисању алгоритма за налажење доброг почетног решења. Предложени алгоритам представља неку врсту похлепног алгоритма (енгл. Greedy algorithm), и прилагођен је коришћеној репрезентацији решења и карактеристикама проблема. Његов псеудокод је дат у алгоритму 4.3.

Алгоритам 4.3: Генерисање почетног решења

Улаз: -

```
1. for k=1:broj_proizvoda
2.     izabrano_postrojenje ← 1
3.     for i=2:broj_postrojenja
4.         if slobodna(postojenje_i) and
           cena(postrojenje_i, proizvod_k) < cena(izabrano_postrojenje,proizvod_k)
5.             izabrano_postrojenje ← i
6. dodeli_korisnike_postrojenjima
```

Главна идеја процедуре приказане алгоритмом 4.3 је да се за сваки тип производа одабере оно постројење које би било најповољније за производњу тог типа производа, под претпоставком да се сви корисници снабдевају тим типом производа управо из тог постројења. На тај начин се добија квалитетно полазно решење, јер се фаворизују добре локације за производњу сваког од типова производа. Након додељивања производа постројењима, процедура додељује кориснике успостављеним постројењима за снабдевање одређеним типом производа користећи похлепни алгоритам *dodeli_korisnike_postrojenjima*. Овај алгоритам додељује сваког корисника оном постројењу које му је најповољније за снабдевање датим типом производа. Када су локације успостављених постројења познате,

као и типови производа које она производе, тада се корисници на јединствен начин могу оптимално доделити тим постројењима, јер ће се сваки корисник снабдевати из постројења које има најмању цену транспорта одговарајућег производа до корисника.

4.4 Рачунање функције циља

Процедура за рачунање функције циља за улаз има структуру која представља неко допустиво решење. Алгоритам користи дефиницију функције циља (1) разматраног проблема дату у поглављу 2.1. Кораци процедуре за рачунање функције циља дати су у алгоритму 4.4. На почетку процедуре врши се иницијализација променљиве *funkcija_cilja*, која чува вредност функције циља. Свака од наредне две петље повећава вредност променљиве *funkcija_cilja* користећи дефиницију (1) из поглавља 2.1.

Алгоритам 4.4: Рачунање функције циља

Улаз: текуће решење *x*

```
1.  funkcija_cilja ← 0
2.  for i = 1, ..., m
3.      for j = 1, ..., n
4.          for k = 1, ..., p
5.              if x.postrojenje_korisnik[i,j] and x.postrojenje_proizvod[i] == k
6.                  funkcija_cilja ← funkcija_cilja + (c[i,j,k]+p[i,k])*q[j,k]
7.  for i = 1, ..., m
8.      if x.postrojenje_proizvod[i] != -1
9.          funkcija_cilja ← funkcija_cilja + f[i][x.postrojenje_proizvod[i]]
```

4.5 Структура околина

Околина неког решења представља скуп решења која се могу добити применом процедуре за њено генерисање. Правилан избор структуре околина од пресудног је значаја за добијање квалитетног крајњег решења. Да би се могло изаћи из тренутног локалног оптимума, и да би се проширила област претраге на неке неистражене области, потребно је да избор околина буде пажљиво изабран. Процедуре за генерисање околине решења морају бити такве да се њиховом применом не могу добити недопустива решења. У наставку су детаљније описане структуре околина које су коришћене у имплементацији основне методе променљивих околина, као и процедуре које их генеришу.

4.5.1 Процедура *Dodaj_ili_oduzmi_postrojenje*

Ова процедура је уведена како би било могуће мењати број постројења која су изграђена. Често се дешава да у решењу постоји већи број постројења која производе исти тип производа, док се анализом оптималног решења испоставља да једно или више њих не мора бити успостављено, већ је довољно користити остала постројења. Смањивање броја успостављених постројења има смисла, јер успостављање сваког постројења носи извесне трошкове, те се на тај начин смањује вредност укупних трошкова. Уколико би цена успостављања сваког од постројења била једнака нули, тада би имало смисла одмах успоставити сва постројења, и не смањивати њихов број, јер њихово успостављање не повећава вредност функције циља, чак и уколико се не користе. Међутим, како то није случај, има смисла покушати са смањивањем броја успостављених постројења. Са друге стране, додавање постројења, односно њихово успостављање, такође има смисла, јер може водити ка смањењу транспортних трошкова, а самим тим и смањењу вредности функције циља. Примена ове процедуре је таква да се никад не уклања постројење које једино производи неки тип производа, јер би се на тај начин дошло до недопустивог решења. Приказ ове процедуре дат је у алгоритму 4.5.1.

Алгоритам 4.5.1: Процедура *Dodaj_ili_oduzmi_postrojenje*

Улаз: текуће решење *x*

```
1.  random ← slučajni broj iz skupa {0,1}
2.  if random = 0
3.      S ← skup_neuspostavljenih_postrojenja
4.      izaberi_slučajno_postrojenje_iz_skupa S - postrojenje(j)
5.      izaberi_slučajan_tip_proizvoda – proizod(k)
6.      uspostavi_postrojenje postrojenje(j) i dodeli joj proizvod(k)
7.  if random = 1
8.      P ← skup_proizvoda_sa_vise_postrojenja
9.      izaberi_slučajni_proizvod_iz_skupa P – proizvod(k)
10.     F ← skup_postrojenja_koje_proizvode_proizvod proizvod(k)
11.     izaberi_slučajno_postrojenje_iz_skupa F - postrojenje(j)
12.     ukloni_postrojenje postrojenje(j)
```

На самом почетку алгоритма врши се случајан избор, и на основу њега се успоставља случајно изабрано постројење из скупа неуспостављених (кораци 3-6), или се случајно изабрано постројење из скупа успостављених брише из тог скупа и додаје скупу неуспостављених постројења (кораци 8-12).

4.5.2 Процедура *Promeni_proizvod_postrojenju*

За разлику од процедуре *Dodaj_ili_oduzmi_postrojenje* која врши промену броја постројења, у процедури *Promeni_proizvod_postrojenju* се не мења њихов број, већ тип производа који производи неко од њих. Промена типа производа који производи једно постројење, може довести до промене мреже дистибуције типа производа од постројења до корисника. Постројење које мења тип производа који производи бира се на случајан начин. Промена типа производа који се производи у случајно изабраном постројењу врши се такође на случајан начин, за разлику од процедуре генерисања почетног решења где се сваки тип производа додељује постројењу похлепним алгоритмом. Овде је неопходно увести елемент случајности како би се могло доћи до нових и неистражених простора претраге. Када се не би случајно бирао тип производа који се додељује постројењу, тада би сваком постројењу увек био додељен исти тип производа, што није добро. Приликом конструисања процедуре *Promeni_proizvod_postrojenju* водило се рачуна о томе да постројење које мења тип производа који производи не буде једино постројење које производи тај тип производа. Псеудокод процеудре *Promeni_proizvod_postrojenju* приказан је у алгоритму 4.5.2.

Алгоритам 4.5.2: Процедура *Promeni_proizvod_postrojenju*

Улаз: текуће решење *x*

```
1. P ← skup_proizvoda_sa_više_postrojenja
2. izaberi_slučajni_proizvod_iz_skupa P - proizvod(k)
3. F ← skup_postrojenja_koje_proizvode_proizvod proizvod(k)
4. izaberi_slučajno_postrojenje_iz_skupa F - postrojenje(f)
5. izaberi_slučajan_proizvod proizvod(k_novi)
6. dodeli_postrojenju postrojenje(f) proizvod proizvod(k_novi)
```

4.5.3 Процедура *Razmeni_proizvode_postrojenjima*

Трећа, и последња, структура околина која је коришћена у предложеној VNS имплементацији, уведена је како би надоместила недостатке претходне две околине. Наиме, понекад се може десити да није довољно само додати или одузети неко постројење, или променити тип производа који постројење производи. Наведене промене понекад нису довољне да се изађе из локалног оптимума у који алгоритам може упасти. Из тог разлога неопходна је јача диверсификација претраге, односно већа измена текућег решења. Трећа структура околина генерише се процедуром *Razmeni_proizvode_postrojenjima*. Идеја ове процедуре је да се веће промене текућег решења врше у каснијим итерацијама алгоритма. Улаз у процедуру представљају текуће решење и индекс текуће итерације алгоритма. У почетним итерацијама само један пар постројења мења типове производа које она

производе, док је касније тај број већи, до максималне промене која је једнака целом делу природног логаритма текуће итерације. Процедура *Razmeni_proizvode_postrojenjima* се може извести из прве две процедуре: размена типова производа два постројења је заправо промена типа производа првог постројења на тип производа другог, и затим, у новом циклусу, промена типа производа другог постројења на тип производа првог. Међутим, овакав след операција се у пракси ретко дешава. Са друге стране, интензивнија промена решења није могућа користећи само прве две структуре околина, и због тога је увођење треће процедуре неопходно, што показују и прелиминарни експериментални резултати. Псеудокод процедуре *Razmeni_proizvode_postrojenjima* дат је у алгоритму 4.5.3.

Алгоритам 4.5.3: Процедура *Razmeni_proizvode_postrojenjima*

Улаз: текуће решење *x*, $tekuća\_iteracija$

```
1. broj_parova_za_razmenu ← [log(tekuća_iteracija)]
2. for i = 1 : broj_parova_za_razmenu
3.     postrojenje(1) ← rand(m); postrojenje(2) ← rand(m);
       postrojenje(1) != postrojenje(2)
4.     razmeni_proizvode_postrojenjima(postrojenje(1),postrojenje(2))
```

4.6 Размрдавање решења

Размрдавање решења (енгл. Shaking) представља значајан део основне методе променљивих околина. Идеја процедуре је да на случајан начин изврши генерисање решења x' у k-тој околини решења x. Избор процедуре за генерисање решења x' врши се у зависности од вредности улазног параметра *k*. Псеудокод процедуре за размрдавање решења приказан је уа алгоритму 4.6.

Алгоритам 4.6: Размрдавање решења

Улаз: текуће решење *x*, *k*

```
1. if k == 1
2.     return Dodaj_ili_oduzmi_postrojenje(x)
3. if k == 2
4.     return Promeni_proizvod_postrojenju(x)
5. if k == 3
6.     return Razmeni_proizvode_postrojenjima(x)
```

4.7 Локална претрага

Један од главних елемената основне методе променљивих околина је и локална претрага. Након промене околине, алгоритам врши локалну претрагу почевши од решења које је добијено из фазе размрдавања. Квалитетна локална претрага ће обићи околину решења и доспети у локални оптимум који може бити бољи од тренутно најбољег решења. У имплементацији предложеног алгоритма, суседно решење у локалној претрази се добија у околини коју генерише процедура *Razmeni_proizvode_postrojenjima*, с тим што се типови производа размењују само једном пару постројења. Локална претрага се зауставља након првог пронађеног бољег решења (first-improvement local search), јер су резултати прелиминарних тестирања показали да је та варијанта локалне претраге доста бржа, а у случају разматраног проблема, даје резултате једнаке или приближно једнаке best-improvement варијанти. Псеудокод алгоритма локалне претраге приказан је у алгоритму 4.7.

Алгоритам 4.7: Локална претрага

Улаз: текуће решење x

```
f0 ← izračunaj_funkciju_cilja(x)
for i = 1:broj_postrojenja-1
    for j = i+1:broj_postrojenja
        if proizvod_postrojenja(i) != proizvod_postrojenja(j)
            x′ ← razmeni_proizvode_postrojenjima
            f1 ← izračunaj_funkciju_cilja(x′)
            if f1 < f0
                return x′
return x
```

5 Експериментални резултати

Експериментални резултати добијени су покретањем тестова на рачунару базираном на процесору Intel Core I7-4600U на 2.4 GHz и 8 GB RAM меморије. За програмску имплементацију методе променљивих околина коришћен је програмски језик C. Вредност параметра *max_nema_poboljsanja*, који се односи на критеријум заустављања, у имплементацији је постављен на 100, док је параметар *k_max*, који представља број различитих структура околина, постављен на три. На свакој тест инстанци метода променљивих околина је покретана 10 пута. За добијање оптималних резултата и њихово поређење, коришћен је комерцијални CPLEX решавач, верзија 12.1 [20].

За поређење добијених решења са оним која су тренутно позната у литератури коришћена су три скупа тест инстанци: инстанце класе 1 (мале), инстанце класе 2 (средње) и инстанце класе 3 (велике). Начин генерисања инстанци описан је у раду [31]. Класа 1 садржи инстанце које имају од 21 до 117 корисника, 5-15 потенцијалних локација за изградњу постројења и 2-12 типова производа, и код њих се лако добијају оптимална решења. Инстанце класе 2 су добијене модификовањем иницијалних трошкова и трошкова производње у инстанцама класе 1, и на тај начин су добијене инстанце које је знатно теже решити (видети [31]). Ове инстанце имају од 80 до 100 корисника, 30-100 потенцијалних локација за изградњу постројења и 15-40 типова производа. Инстанце класе 3 садрже до 100 корисника, 10-30 потенцијалних локација за изградњу постројења и 20-30 типова производа, и управо на овим инстанцама се може уочити највећа разлика у квалитету добијених решења коришћењем приступа који је овде описан, у односу на досадашње резултате.

Резултати добијени основном методом променљивих околина на инстанцама из класа 1-3, упоређени су са оптималним резултатима или горњим границама добијеним CPLEX решавачем у временском ограничењу од једног сата, као и са најбољим резултатима који су добијени бољом од две хеуристике засноване на Лагранжовој релаксацији из [31].

У табелама 5.1-5.3, резултати су приказани на следећи начин. У првој колони налази се ознака инстанце. У наредне три колоне налазе се параметри који се односе на димензије тест инстанце: *m*, *n* и *p*. Следеће две колоне приказују решење добијено коришћењем CPLEX решавача, као и одговарајуће време извршавања. Уколико је CPLEX нашао неко допустиво решење проблема, али у задатом року од 1h није успео да нађе оптимално решење, тада је у колони Sol_{CPLEX} приказана вредност функције циља за најбоље допустиво решење које је CPLEX пронашао. Та вредност представља горњу границу (енгл. Upper bound) за вредност функције циља оптималног решења и означено је са UB, а у одговарајућем пољу за време извршавања у колони $t_{CPLEX}(s)$ унета је ознака t.l. (енгл. Time limit). Уколико CPLEX у задатом времену извршавања од 1h није успео да нађе допустиво решење, ознака "–" је уписана у одговарајуће поље. Наредне две колоне садрже

информацију о најбољој доњој (LB_{LR}) и горњој граници (UB_{LR}) решења добијеног коришћењем неке од две Лагранжове релаксације предложених у [31]. Вредност функције циља најбољег VNS решења које је добијено кроз 10 покретања алгоритма дато је у колони $Best_{VNS}$, док колона t_{VNS} (s) приказује просечно процесорско време у секундама потребно VNS алгоритму да први пут дође до најбољег решења. Колона agap приказује средње одступање (у процентима) од најбољег познатог резултата (енгл. Best-known solution). У последњем реду сваке табеле приказане су просечне вредности колона. Најбољи познати резултати су истакнути у све три табеле.

Табела 5.1: Поређење резултата добијених на инстанцама класе 1

Инстанца	m	n	p	Sol_{CPLEX}	t_{CPLEX}(s)	LB_{LR}	UB_{LR}	$Best_{VNS}$	t_{VNS} (s)	agap (%)
K1-1	5	21	2	**48760.68**	1.13	**48760.68**	**48760.68**	**48760.68**	1.16	0.00
K1-2	5	22	3	**22440.72**	2.12	**22440.72**	**22440.72**	**22440.72**	0.02	0.00
K1-3	5	29	4	**55761.30**	1.23	**55761.30**	**55761.30**	**55761.30**	0.02	0.00
K1-4	5	32	5	**114349.69**	0.82	**114349.69**	**114349.69**	**114349.69**	0.02	0.00
K1-5	5	36	5	**3582.29**	0.75	**3582.29**	**3582.29**	**3582.29**	0.02	0.00
K1-6	5	50	5	**3161.75**	2.12	**3161.75**	**3161.75**	**3161.75**	0.03	0.00
K1-7	10	75	8	**12075.28**	2.35	**12075.28**	**12075.28**	**12075.28**	1.30	0.00
K1-8	10	100	9	**13135.73**	0.55	**13135.73**	**13135.73**	**13135.73**	0.98	0.00
K1-9	2	12	2	**524.43**	0.66	**524.43**	**524.43**	**524.43**	0.02	0.00
K1-10	15	55	12	**13143.04**	3.13	**13143.04**	**13143.04**	**13143.04**	4.80	0.00
K1-11	7	85	6	**10128.93**	3.57	**10128.93**	**10128.93**	**10128.93**	0.16	0.00
K1-12	4	318	4	**31146511.76**	1.23	**31146511.76**	**31146511.76**	**31146511.76**	0.08	0.00
K1-13	5	27	2	**26699.36**	0.79	**26699.36**	**26699.36**	**26699.36**	0.03	0.00
K1-14	8	34	6	**89316.89**	2.12	**89316.89**	**89316.89**	**89316.89**	0.41	0.00
K1-15	8	88	8	**380294636.11**	1.35	**380294636.11**	**380294636.11**	**380294636.11**	0.12	0.00
K1-16	10	150	10	**5312709655.55**	5.43	**5312709655.55**	**5312709655.55**	**5312709655.55**	0.53	0.00
K1-17	14	117	12	**24546.82**	12.23	**24546.82**	**24546.82**	**24546.82**	12.37	0.00
Просек				336740495.90	2.45	336740495.90	336740495.90	336740495.90	1.30	0.00

Табела 5.2: Поређење резултата добијених на инстанцама класе 2

Инстанца	m	n	p	Sol_{CPLEX}	t_{CPLEX}(s)	LB_{LR}	UB_{LR}	$Best_{VNS}$	t_{VNS} (s)	agap (%)
K2-1	50	100	20	4534673.70^{UB}	t.l.	4204368.18	4646350.02	**4527617.27**	1066.79	0.81
K2-2	50	100	25	7263879.67^{UB}	t.l.	6659568.29	7252438.01	**6958671.61**	470.34	0.77
K2-3	50	100	30	10981107.40^{UB}	t.l.	9853587.03	**10338547.73**	10370130.84	1074.10	0.81
K2-4	50	100	25	6715340.97^{UB}	t.l.	5645216.66	6751291.49	**6350585.64**	252.05	1.14
K2-5	50	100	30	10002621.55^{UB}	t.l.	8413468.82	9865435.37	**9210249.10**	353.04	1.07
K2-6	50	100	25	13514196.26^{UB}	t.l.	12129724.10	13434197.00	**13135188.27**	517.42	0.83
K2-7	30	80	27	8733519.46^{UB}	t.l.	8466238.63	8740157.66	**8730483.63**	31.26	0.54
K2-8	30	80	28	**17434782.02**	16.44	17434782.03	**17434782.03**	17908665.16	42.44	1.12
K2-9	30	80	29	**48574841.15**	7.52	48574841.16	**48574841.16**	49979114.48	25.94	1.60
K2-10	50	100	30	9778568.54^{UB}	t.l.	7815656.61	9779846.02	**8913386.06**	365.90	0.85
K2-11	50	100	35	14242716.16^{UB}	2495.41	13014296.45	14071557.74	**13743764.98**	1013.42	0.68
K2-12	50	100	40	21862726.57^{UB}	469.65	21523697.46	**21850868.66**	22047440.91	875.30	0.71
K2-13	100	100	25	28348123.45^{UB}	t.l.	27544949.18	28650738.26	**28258551.43**	1695.75	0.82
K2-14	100	100	20	**18034783.57**	1583.13	17734127.61	18276778.59	18049812.87	2267.30	1.21
K2-15	100	100	15	**10754899.72**	68.88	10754464.04	**10754899.72**	10947321.92	1536.04	0.89
Просек				15385119.68	-	14651265	15361514	15275398.94	772.47	0.92

Табела 5.3: Поређење резултата добијених на инстанцама класе 3

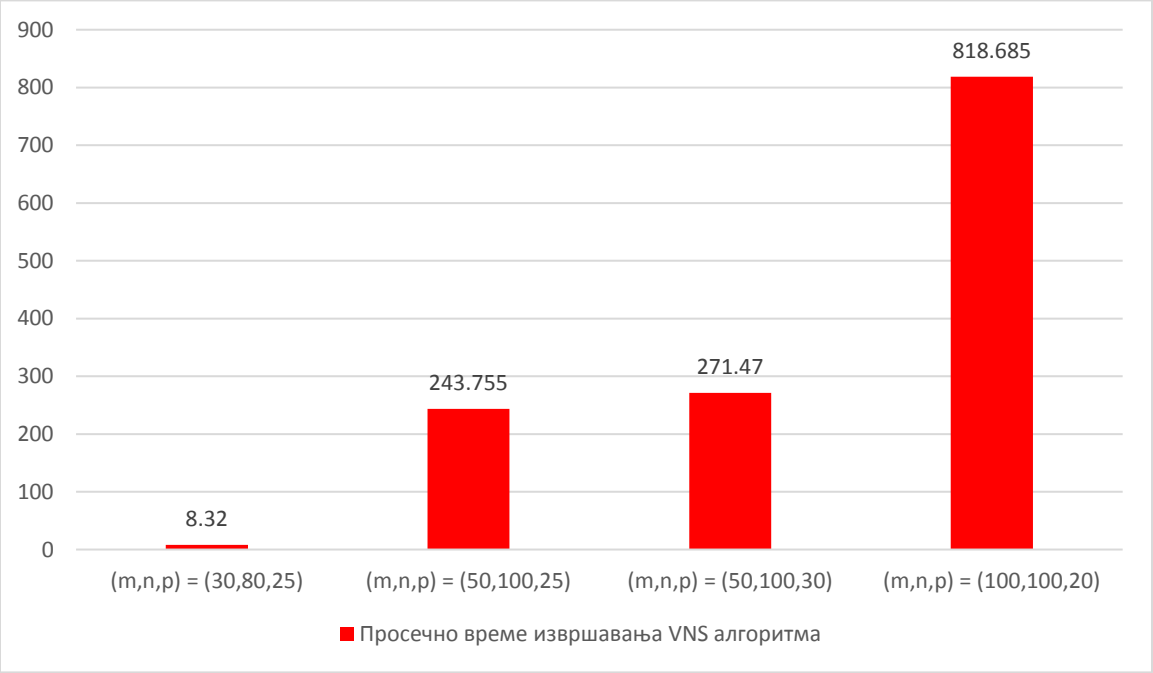
Инстанца	m	n	p	Sol_{CPLEX}	t_{CPLEX}(s)	LB_{LR}	UB_{LR}	$Best_{VNS}$	t_{VNS} (s)	agap (%)
K3-1	50	100	30	81652.67^{UB}	t.l.	73469.17	80752.62	**77899.78**	115.19	0.30
K3-2	50	100	30	73615.15^{UB}	t.l.	65431.66	73074.29	**69874.06**	187.52	0.24
K3-3	50	100	30	57540.11^{UB}	t.l.	49356.62	56723.46	**53792.12**	126.53	0.39
K3-4	50	100	30	-	t.l.	446545.10	451939.79	**450733.05**	75.10	0.04
K3-5	50	100	30	-	t.l.	87457.87	92852.55	**91782.38**	56.35	0.07
K3-6	50	100	30	59043.50^{UB}	t.l.	51549.14	56943.82	**55777.41**	89.48	0.20
K3-7	30	80	25	517285.26^{UB}	t.l.	515108.76	517245.50	**516996.63**	7.06	0.00
K3-8	30	80	25	47198.14^{UB}	t.l.	44984.60	47142.96	**46762.74**	7.26	0.11
K3-9	30	80	25	23769.32^{UB}	t.l.	21517.07	23441.75	**23313.95**	10.63	0.11
K3-10	50	100	25	211679.29^{UB}	t.l.	210174.66	**210948.44**	**210948.40**	21.87	0.00
K3-11	50	100	25	184338.51^{UB}	t.l.	177656.94	183596.21	**181357.66**	83.92	0.05
K3-12	50	100	25	180103.40^{UB}	t.l.	173421.51	179858.91	**177210.37**	116.93	0.05
K3-13	100	100	20	**396809.92**	15.93	**396809.87**	**396809.87**	**396809.90**	33.04	0.00
K3-14	100	100	20	124915.94^{UB}	1492.39	124200.30	135738.68	**124845.67**	293.24	0.33
K3-15	100	100	20	74971.12^{UB}	3102.00	73320.75	79008.11	**74338.23**	681.16	0.36
Просек				-	-	167399.79	172404.62	170162.823	127.02	0.15

За све инстанце класе 1, метода променљивих околина је успела да достигне оптимална решења у времену које је приближно једнако времену које је било потребно CPLEX решавачу. У сваком од покретања, VNS метода је пронашла оптимална решења, па је agap једнак нули за сваку од инстанци из класе 1. Резултати две Лагранжове релаксације предложених у [31] су такође оптимални са сличним временима извршавања.

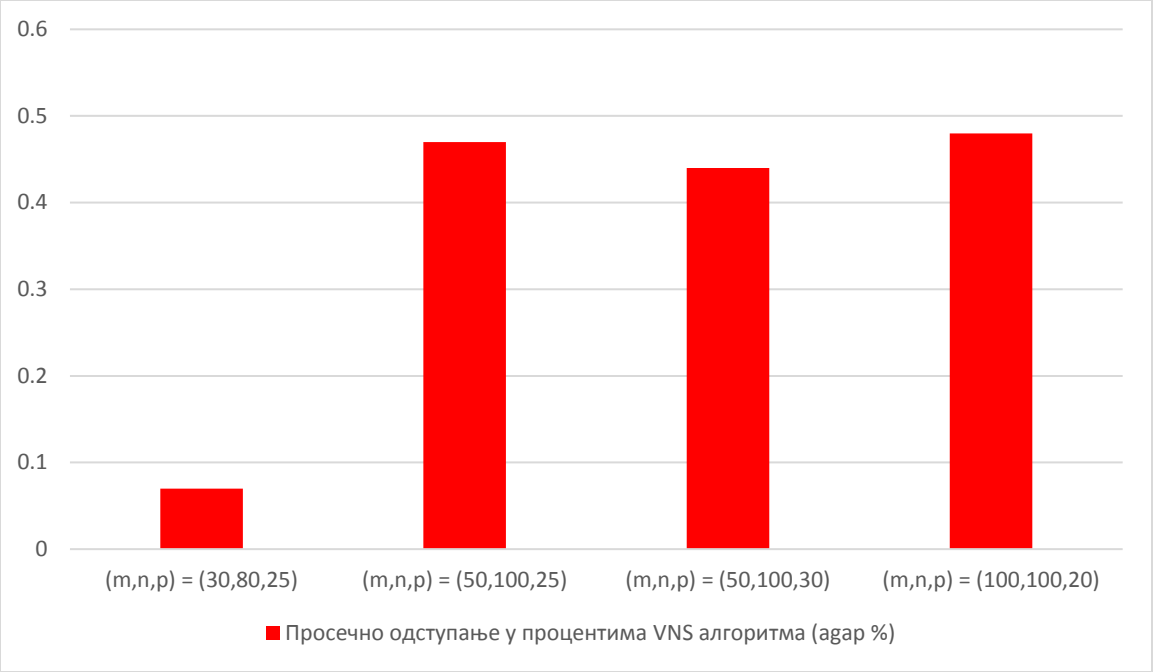
За инстанце класе 2, предложени алгоритам у већини случајева успева да достигне или побољша горњу границу добијену Лагранжовом релаксацијом или CPLEX-ом, у релативно кратком временском интервалу. У 10 од 15 случајева, предложена метода променљивих околина је успела да спусти горњу границу решења добијену коришћењем две Лагранжове релаксације у [31]. У 11 од 15 случајева добијено оптимално решење је било боље од оптималног решења добијеног коришћењем CPLEX решавача. Средње одступање у процентима је за 10 од 15 тест инстанци испод 1%, док је у пет случајева тај проценат био мало већи (до 1.6%)

Предности предложене методе променљивих околина најуочљивије су на тест инстанцама већих димензија (класа 3). На овим инстанцама метода променљивих околина је на свих 15 тест инстанци успела да побољша досадашња најбоља решења позната у литератури, односно да спусти горњу границу решења добијену применом боље од две Лагранжове релаксације из [31]. Такође, на 12 тест инстанци, предложена VNS метода побољшала је горње границе добијене CPLEX решавачем, и достигла оптимално решење које је CPLEX дао за само једну инстанцу класе 3. У случају две инстанце за које CPLEX није успео да пронађе чак ни допустиво решење, VNS је у кратком времену извршавања дао решења која су боља од познатих горњих граница добијених Лагранжовим хеуристикама из [31]. Средње одступање VNS решења од најбољег познатог је у свих 15 случајева било мање од 0.5%.

На слици 5.1 приказан је график зависности просечног времена извршавања предложеног VNS алгоритма од димензија проблема. Слика 5.2 приказује график зависности agap-а од димензија проблема. За сваку од четири разматране комбинације улазних параметара (m, n, p), приказаних на Ox оси, на Oy оси представљене су одговарајуће просечне вредности времена извршавања (слика 5.1), односно agap-а (слика 5.2), над скупом инстанци са истом комбинацијом улазних параметара.



Слика 5.1: Зависност просечног времена извршавања VNS алгоритма од димензија проблема



Слика 5.2: Зависност просечног одступања решења од димензија проблема

6 Закључак

У овом раду разматран је проблем неограничених капацитета са једним извором снабдевања и више типова производа (USSMP-FLP), који има велики практични значај приликом оптимизације мреже снабдевања, али и других транспортних и телекомуникацијских система. Изложена је математичка формулација проблема као и претходни начини решавања, засновани на егзактном приступу и хеуристикама Лагранжове релаксације. У овом раду је предложена метода променљивих околина као хеуристички приступ решавању проблема, посебно у случају инстанци проблема великих димензија за која оптимална решења нису позната. Имплементирана је основна варијанта методе промељивих околина са пажљиво изабраним структурама околина које су прилагођене карактеристикама разматраног проблема и адекватним фазама размрдавања и локалне претраге. Дат је детаљан опис имплементације алгоритма заједно са свим помоћним структурама, процедурама и одговарајућим псеудокодовима.

Предложена VNS метода је тестирана на три класе инстанци из литературе. Најбоља решења VNS методе су упоређена са резултатима добијеним CPLEX решавачем, као и горњим границама боље од две хеуристике Лагранжове релаксације из рада [31]. На тест инстанцама класе 1, предложени алгоритам је достигао сва оптималана решења. На инстанцама класе 2 и 3, VNS је побољшао најбоље познате резултате из литературе на укупно 25 од 30 инстанци, значајним смањењем до сада познатих горњих граница функције циља проблема. Приказани резултати потврђују потенцијал примене методе променљивих околина за решавање наведеног проблема. Део резултата истраживања представљених у овом раду објављен је у [37].

Овај рад пружа добру основу за даља истраживања, на пример, хибридизацију методе променљивих околина са неком другом метахеуристиком или егзактном методом, у циљу додатних побољшања добијених резултата.

Литература

[1] Andreatta, A., Ribeiro, C. (2002). Heuristics for the phylogeny problem. *Journal of Heuristics*, 8(4), 429–447.

[2] Arabani, A. B., Farahani, R. Z. (2012). Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1), 408-420.

[3] Barcelo, J., Casanovas, J. (1984). A heuristic Lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15, 212–226.

[4] Barreto, S., Ferreira, S., Paixao, J., Santos, B.S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179, 968–977.

[5] Bilde, O., Krarup, J. (1977). Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics*, 1, 79–97.

[6] Brimberg, J., Hansen, P., Mladenović, N., Taillard, É. (2000). Improvements and comparison of heuristics for solving the multisource Weber problem. *Operations Research*, 48(3), 444–460.

[7] Canuto, S., Resende, M., Ribeiro, C. (2001). Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 31(3), 201–206.

[8] Đenić A., Marić M., Stanimirović Z., Stanojević P. (2016). A variable neighbourhood search method for solving the long-term care facility location problem. *IMA Journal of Management Mathematics*

[9] Đenić, A., Radojičić, N., Marić, M., Mladenović, M. (2016). Parallel VNS for Bus Terminal Location Problem. *Applied Soft Computing*, 42, 448-458.

[10] Drezner, Z., Hamacher, H. (2002). *Facility Location: Applications and Theory*. Springer Verlag, Heidelberg, Germany.

[11] Garey, M. R., Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completness*, W. H. Freeman and Company, New York, USA

[12] Gomory, R.E. (1963). An algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming*, 269-302.

[13] Hansen, P., Mladenović, N. (1997). Variable neighborhood search for the p-median. *Location Science*, 5(4), 207-226.

[14] Hansen, P., Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3), 449-467.

[15] Hansen, P., Mladenović, N., Pérez-Brito, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics*, 7 (4), 335–350.

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

[17] Hansen, P., Jaumard, B., Mladenović, N., Parreira, A. (2000). Variable neighborhood search for weighted maximum satisfiability problem. *Les Cahiers du GERAD*, 62, Montréal, Canada.

[18] Held, M., Karp, R.M. (1970). The traveling salesman problem and minimum spanning trees. *Operations Research*, 18, 1138–1162.

[19] Held, M., Karp, R.M. (1971). The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming 1*, 6–25.

[20] IBM ILOG CPLEX Optimization Studio. http://www03.ibm.com/software/products/en/ibmilogcpleoptistud. Pristupljeno: 08.09.2016.

[21] Ilić, A., Urošević, D., Brimberg, J., Mladenović, N. (2010). A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. *European Journal of Operational Research*, 206(2), 289-300.

[22] Kochetov, Y., Ivanenko, D. (2005). Computationally difficult instances for the uncapacitated facility location problem. *Metaheuristics: Progress as Real Problem Solvers*, 351–367.

[23] Krarup, J., Pruzan, P.M. (1983). The simple plant location problem: survey and synthesis. *European Journal of Operational Research*, 12.1, 36-81.

[24] Land, A. H., Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28 (3), 497–520.

[25] Marić, M., Stanimirović, Z., Milenković, N., Đenić A. (2015). Metaheuristic Approaches to Solving Large-Scale Bilevel Uncapacitated Facility Location Problem With Clients' Preferences. *The Yugoslav Journal of Operations Research* - YUJOR, 25 (3), 361-378.

[26] Mišković, S., Stanimirović Z., Grujičić I. (2014). An efficient variable neighborhood search for solving a robust dynamic facility location problem in emergency service network. *Electronic Notes in Discrete Mathematics*, 47, 261-268.

[27] Mladenović, N. (1995). A Variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. *Abstracts of papers presented at Optimization Days*, 112.

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

[29] Mladenović, N., Labbé, M., Hansen, P. (2003). Solving the p- Center problem with Tabu Search and Variable Neighborhood Search. *Networks*, 42(1), 48-64.

[30] Nelder J. A., Mead R. (1965). A Simplex Method for Function Minimization. *The computer journal*, 7(4), 308-313.

[31] Nezhad, A.M., Manzour, H., Salhi, S. (2013). Lagrangian relaxation heuristics for the uncapacitated single-source multi-product facility location problem. *International Journal of Production Economics*, 145, 713–723.

[32] Nickel, S., Puerto, J. (2005). *Location Theory: A Unified Approach*. Springer, Heidelberg, Germany.

[33] Padberg, M., Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *Siam Review*, 60–100.

[34] ReVelle C.S., Eiselt H.A. (2005). Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165, 1-19.

[35] Sridharan, R. (1993). A Lagrangian heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research*, 66, 305–312.

[36] Stanimirović, Z., Marić, M., Radojičić, N., Božović, S. (2014). Two Efficient Hybrid Metaheuristic Methods for Solving the Load Balance Problem. *Applied and Computational Mathematics*, 13(3), 332-349.

[37] Trajkov, P., Stanimirovic, Z. (2016). Variable Neighborhood Search for the Uncapacitated Single-Source Multi-Product Facility Location Problem. *Proceedings of XLIII International Symposium on Operational Research*, Tara, 20-23 September, (u štampi)

[38] Weber, A. (1929). *Uber den Standort der Industrien (Alfred Weber's Theory of the Location of Industries)*, University of Chicago, USA.