УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Александар Стојановић

ПРИМЕНА МЕТОДЕ ПРОМЕНЉИВИХ ОКОЛИНА ЗА РЕШАВАЊЕ ВАРИЈАНТЕ ДВОНИВОСКОГ ДИНАМИЧКОГ ЛОКАЦИЈСКОГ ПРОБЛЕМА

мастер рад

Београд, 2022.

Ментор:

др Зорица Станимировић, редовни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Зорица Дражић, доцент
Универзитет у Београду, Математички факултет

др Стефан Мишковић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: _____

Мами и тати

Наслов мастер рада: Примена методе променљивих околина за решавање варијанте двонивоског динамичког локацијског проблема

Резиме: У овом раду разматран је проблем динамичке локације објеката са избором добављача уз количински попуст са неограниченим капацитетима (енгл. Uncapacitated Dynamic Facility Location With Supplier Selection Under Quantity Discount − UDFLP-QD) који представља једну варијанту двонивоског проблема локације објеката. Посматрани проблем је NP-тежак као уопштење проблема локације објеката. Коришћењем математичког модела проблема у оквиру егзактног решавача CPLEX може се наћи оптимално решење само за инстанце малих димензија, док је код инстанци великих димензија егзактни решавач немоћан услед недостатка меморије и ограничења времена. Како се у реалним применама овог проблема углавном јављају инстанце великих димензија, неопходно је осмислити и применити адекватне апроксимативне методе у циљу налажења решења довољно доброг квалитета имајући у виду меморијска и временска ограничења. Зато је за решавање проблема у раду предложено неколико метахеуристичких приступа заснованих на познатој методи променљивих околина (енгл. Variable Neighborhood Search − VNS). Прецизније, дизајнирано је и имплементирано неколико метахеуристика прилагођених разматраном проблему: основна метода променљивих околина (енгл. Basic Variable Neighborhood Search − BVNS), метода променљивог спуста (енгл. Variable Neighborhood Descent − VND) и општа метода променљивих околина (енгл. General Variable Neighborhood Search − GVNS). Перформансе имплементираних метахеуристика су међусобно упоређене кроз тестирања на инстанцама различитих димензија, док су на инстанцама малих димензија резултати метахеуристика упоређени са оптималним или најбољим решењима добијених егзактним решавачем CPLEX. Такође је извршена статистичка анализа резултата добијених тестирањем метахеуристика.

Кључне речи: комбинаторна оптимизација, мешовито целобројно линеарно програмирање, метода променљивих околина, локацијски проблеми, метахеуристике

Title: Application of variable neighborhood search method for solving variant of two-level dynamic facility location problem

Abstract: In this paper, the problem of uncapacitated dynamic facility location with supplier selection under quantity discount − UDFLP-QD is presented, which is a variant of the two-level problem of facility location. The considered problem is NP-hard as a generalization of the facility location problem. Using a mathematical model of the problem within the CPLEX exact solver, the optimal solution can be found only for small instances, while for large instances the exact solver is powerless due to lack of memory and time constraints. As large-scale instances generally occur in real applications of this problem, it is necessary to devise and apply adequate approximate methods in order to find a solution of sufficiently good quality, taking into account memory and time constraints. Therefore, several metaheuristics based on the well known Variable neighborhood search − VNS method are proposed as solution approaches to the considered problem. More precisely, the following metaheuristics have been designed and implemented: the Basic Variable Neighborhood Search − BVNS method, the Variable Neighborhood Descent − VND method and the General Variable Neighborhood Search − GVNS. The performance of implemented metaheuristics was compared with each other through testing on instances of different size, while on instances of small size the results of metaheuristics were compared with the optimal or best solutions obtained with the exact CPLEX solver. Statistical analysis of the results obtained by testing metaheuristics was also performed.

Keywords: combinatorial optimization, mixed integer linear programming, variable neighborhood search, two-level facility location problem, metaheuristics

Садржај

Глава 1

Увод

1.1 Проблем комбинаторне оптимизације

Проблем комбинаторне оптимизације се може дефинисати на следећи начин[10]:
Дат је коначан или пребројиво бесконачан, дискретан скуп S и функција $f : S \to \mathbb{R}$. Наћи минимум функције f на скупу S, тј. решити задатак

$$\min_{x \in S} f(x). \tag{1.1}$$

Скуп S се назива *допустив скуп*, а функција f *функција циља*. Тачка $x \in S$ је *допустиво решење* проблема 1.1.
Како важи:

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

проблем максимизације се једноставно може свести на проблем минимизације.

Дефиниција 1.1.1. Затворена околина решења x, у ознаци $N(x)$, у простору S се може дефинисати у односу на метрику δ простора S као [23]:

$$N(x) = \{y \in S \mid \delta(x, y) \leq \varepsilon\}, \tag{1.2}$$

где је ε задат позитиван број.

Дефиниција 1.1.2. Решење x^* је локални минимум проблема 1.1, у околини $N(x^*)$ ако важи:

$$\forall x \in N(x^*) \quad f(x^*) \leq f(x).$$

Дефиниција 1.1.3. Решење x^* је глобални минимум проблема 1.1, ако важи:

$$\forall x \in S \quad f(x^*) \leq f(x).$$

Локалних минимума ако постоје може бити више, а у случају да постоји само један локални минимум он је уједно и глобални. Глобалних минимума такође може бити више (са истом вредношћу функције циља), па је најчешће довољно наћи само један од њих. У пракси се углавном јављају проблеми код којих постоји више локалних минимума, а циљ је пронаћи глобални минимум.

Проблеми оптимизације се јављају у разним областима: привреди, инжењерству, транспорту робе, планирању изградње јавних установа (полицијских станица, домова здравља, пошта), телекомуникационих мрежа итд. Главни проблем је што већина ових проблема има велики број (неки чак и бесконачно) локалних минимума нарочито код проблема који имају велики број променљивих, а за успешно решавање проблема је потребно наћи онај где је вредност функције циља најмања.

1.2 Локацијски проблем

Још у седамнаестом веку Пјер де Ферма[1] је поставио следећу загонетку: „Дате су три тачке у равни, наћи четврту тачку у равни тако да збир растојања те тачке од остале три тачке буде што је могуће мањи.". Прво (геометријско) решење овог проблема је дао Торичели[2]. Због поновног откривања, сличних формулација и проналажења нових начина за његово решавање овај проблем је у литератури познат као Фермаов проблем, Торичелијев проблем, Штајнеров[3] проблем или Веберов[4] проблем.

[1]Pierre de Fermat (1601 – 1665) – француски математичар
[2]Evangelista Torricelli (1608 – 1647) – италијански физичар и математичар
[3]Jakob Steiner (1793 – 1863) – швајцарски математичар
[4]Alfred Weber (1868 – 1958) – немачки економиста, географ и социолог

Овај оригинални проблем, као и неке његове варијанте су били скоро заборављени све док га Вебер 1909. године није преформулисао и применио га на практичне проблеме који се јављају у индустрији. Прецизније, Вебер је оригинални проблем преформулисао тако што је две фиксиране тачке означио као складишта сировина, а једну као корисника. Свим тачкама је доделио одређену тежину. Циљ је одредити локацију четврте тачке на којој би се изградило индустријско постројење које користи сировине из складишта за производњу неког производа, а затим се готови производ дистрибуира кориснику, тако да укупни трошкови транспорта били минимални. Уопштење Веберовог проблема са више од једног корисника се може формулисати на следећи начин [17]:
Наћи координате (x, y) индустријског постројења X тако да се минимизује збир растојања до n познатих тачака са координатама (a_i, b_i) и тежинама w_i, $i = 1, 2, \ldots n$ (цене транспорта по јединици растојања). Математички може да буде записано као:

$$\min z(X) \tag{1.3}$$

где је $z(X) = \sum_{i=1}^{n} w_i d(X, P_i)$, а $d(X, P_i)$ Еуклидско[5] растојање, које се изражава формулом $d(X, P_i) = \sqrt{(x - a_i)^2 + (y - b_i)^2}$, од индустријског постројења X до складишта P_i, $i = 1, 2, \ldots n$.

Веберов проблем се може лако решити применом диференцијалног рачуна. Испоставља се да су координате (x, y) складишта X функције растојања $d(X, P_i)$, које је непознато. Проблем је решио Ендру Вазоњи[6] користећи итеративни алгоритам.

Природно уопштење Веберовог проблема је верзија у којој се траже оптималне локације за више тачака. Како се морају пронаћи локације за више објеката, проблем алокације се решава паралелно са проблемом локације, што проблем чини тешким за решавање. У најједноставнијем случају се претпоставља да је свако индустријско постројење придружено најближем складишту. Овај случај је матаматички дефинисао и предложио хеуристику за његово решавање Леон Купер[7] 1963. године.

[5] Еуклид 4. век п.н.е. – старогрчки математичар
[6] Andrew Wazsonyi (1916 – 2003) – мађарски математичар
[7] Leon Cooper (1930–) – амерички физичар

Због све већих захтева за решавањем проблема у јавном и приватном сектору научници и инжењери се баве дефинисањем специфичних локацијских проблема, формулисањем математичких модела и дизајнирањем одговарајућих метода за њихово решавање. Генерално, задатак локацијских проблема је одабир локација за одређен скуп објеката да би се повезали са постојећим објекатима који имају одређене захтеве. Објекти за које се врши одабир локација су нека врста услужних објеката па се они обично називају *услужни објекти*, *добављачи* или *ресурси*. Објекти који већ постоје и који имају одређене захтеве који треба да буду испуњени се називају *клијенти*, *крајњи корисници* или *потрошачи*. Код класичних локацијских проблема циљ је задовољити потребе корисника уз минималне трошкове или максималан профит. Локацијски проблеми могу укључивати и додатне услове који одговарају конкретној ситуацији у пракси, на пример фиксиран број успостављених снабдевача, ограничени капацитети снабдевача и/или крајњих корисника итд. Више о локацијским проблемима, одговарајућим математичким моделима и методама решавања може се наћи у [14], [17] и [19].

1.3 Сложеност алгоритама и проблема

1.3.1 Сложеност алгоритама

Да би неки алгоритам решио неки проблем мора да има два важна ресурса на располагању: време и простор (меморију). Временска сложеност алгоритма је број корака који су му потребни за решавање проблема величине n. Циљ одређивања рачунске сложености алгоритма није добијање тачног броја корака, већ одређивање асимптотске границе броја корака.
У наставку подпоглавља наведене су дефиниције које су преузете из [48].

Дефиниција 1.3.1. Велико-O нотација. Алгоритам има сложеност $f(n) = O(g(n))$ ако постоје позитивне константе n_0 и c такве да важи $\forall n > n_0,\ f(n) \leq c \cdot g(n)$. Сложеност алгоритма $f(n)$ је ограничена одозго функцијом $g(n)$.

Дефиниција 1.3.2. Полиномска функција степена k се дефинише као:

$$p(n) = a_k n^k + \ldots + a_j n^j + \ldots + a_1 n + a_0,$$

где је $a_k \in \mathbb{R} \setminus \{0\}$, а $a_j \in \mathbb{R}$ за $0 \leq j \leq k-1$.

Дефиниција 1.3.3. Полиномијални алгоритам. Алгоритам је полиномијалан ако је његова сложеност $O(p(n))$, где је $p(n)$ полиномска функција од n.

Дефиниција 1.3.4. Експоненцијални алгоритам. Алгоритам је експоненцијалан ако је његова сложеност $O(c^n)$, где је c реална константа строго већа од 1.

Дефиниција 1.3.5. Велико-Ω нотација. Алгоритам има сложеност $f(n) = \Omega(g(n))$ ако постоје позитивне константе n_0 и c такве да важи $\forall n > n_0,\ f(n) \geq c \cdot g(n)$. Сложеност алгоритма $f(n)$ је ограничена одоздо функцијом $g(n)$.

Дефиниција 1.3.6. Велико-Θ нотација. Алгоритам има сложеност $f(n) = \Theta(g(n))$ ако постоје позитивне константе n_0, c_1 и c_2 такве да важи $\forall n > n_0,\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$. Сложеност алгоритма $f(n)$ је ограничена одоздо функцијом $g(n)$.

1.3.2 Сложеност проблема

Сложеност проблема је еквивалентна сложености најбољег алгоритма за решавање тог проблема. Проблем је лак за решавање ако постоји полиномијални алгоритам за његово решавање. Проблем је тежак за решавање ако не постоји полиномијални алгоритам за његово решавање.

Теорија сложености се бави проблемима одлчивања. Проблем одлучивања увек има *да* или *не* као одговор. Проблем оптимизације се може увек редуковати на проблем одлучивања. Најважнији задатак теорије сложености је да категоризује проблеме у класе сложености. Класа сложености представља скуп свих проблема који могу бити решени користећи унапред дате ресурсе. Постоје две важне класе сложености проблема: P и NP. Њихов однос је представљен на слици 1.1.

Дефиниција 1.3.7. Детерминистички алгоритам. Детерминистички алгоритам је полиномијалан за проблем A ако је његова сложеност ограниче-

на полиномском функцијом $p(n)$ где n представља величину улазне инстанце I.

Дефиниција 1.3.8. Класа сложености P. Класа сложености P представља скуп свих проблема који се могу решити помоћу детерминистичких алгоритама у полиномијалном времену.

Стога класа P представља фамилију проблема за које постоје полиномијални алгоритми за њихово решавање. Проблеми који припадају класи P су релативно лаки за решавање.

Недетерминистички алогритам може да се схвати као алгоритам који садржи кораке у којима се грана на две фазе. Те фазе су међусобно независне, а извршавају се истовремено. Оваквих корака гранања може бити више, па се рад алгоритма одвија у више паралелних процеса. Те две фазе могу да се опишу на следећи начин [40]:

1) **Фаза погађања (недетерминистичка):** У овој фази улаз је само инстанца проблема, а излаз је неки низ карактера S. Низ карактера S може бити схваћен као решење за улазну инстанцу проблема.

2) **Фаза верификације (детерминистичка):** У овој фази су улаз и инстанца проблема и низ карактера S. У овој фази могу да се догоде три случаја. Може да се заврши са излазом „тачно", што може да се протумачи да је одговор на улазну инстанцу проблема „да", тј. низ карактера S је решење улазне инстанце проблема. Може да се заврши са излазом „нетачно", тј. одговор на улазну инстанцу проблема је „не". У последњем случају може да се догоди да се фаза уопште не завршава тј. да је ушла у неку бесконачну петљу.

Дефиниција 1.3.9. *Полинимијални недетерминистички алгоритам* је недетрминистички алгоритам чија је фаза верификације полиномијалан алгоритам.

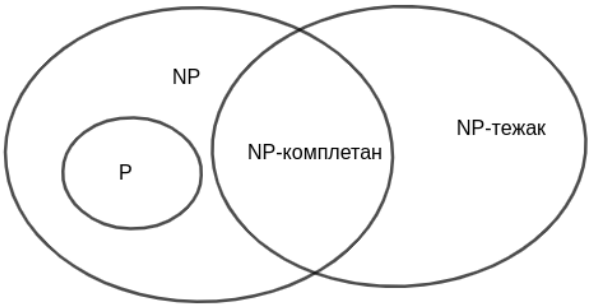
Дефиниција 1.3.10. Класа сложености NP. Класа сложености NP представља скуп свих проблема који се могу решити помоћу полиномијалних недетерминистичких алгоритама.

Дефиниција 1.3.11. Полиномијална редукција. Проблем одлучивања A се може *полиномијално редуковати* на проблем одлучивања B ако постоји полиномијална функција L тако да важи: I_A је улазна инстнаца проблема A за коју је одговор позитиван ако и само ако је $I_B = L(I_A)$ улазна инстанца проблема B за коју је одговор позитиван.

Дефиниција 1.3.12. Проблем одлучивања $A \in NP$ је NP-*комплетан* ако сви проблеми класе NP могу да се полиномијално редукују на проблем A.

Дефиниција 1.3.13. Проблеми чији су одговарајући проблеми одлучивања NP-комплетни називају се NP-*тешки*.

Већина проблема оптимизације који се срећу у пракси су NP-тешки проблеми за које не постоје доказиво ефикасни алгоритми за њихово решавање. Ти проблеми захтевају експоненцијално много времена у зависности од величине улазне инстанце проблема да би било пронађено њихово глобално оптимално решење.



Слика 1.1: Однос између класа сложености проблема

1.4 Методе оптимизације

Методе оптимизације се генерално могу поделити у две класе: егзактне и апроксимативне. Теоријски, егзактне методе су у стању да нађу оптимална решења и могу да гарантују њихову оптималност, под условом да нема

ограничења времена или меморије за њихову примену. У пракси се испоставља да примена егзактних метода на велики број NP-тешких проблема оптимизације, не доводи до оптималних решења, а понекад чак ни до допустивих решења. Са друге стране апроксимативне методе могу да нађу веома квалитетна решења у разумном времену, али не могу да гарантују њихову оптималност. Ипак, у неким случајевима могуће је оценити квалитет решења добијених неком апроксимативном методом.

1.4.1 Егзактне методе

Неки од најпознатијих алгоритама који спадају у класу егзактних метода су: динамичко програмирање (енгл. Dynamic Programming), метода гранања и ограничавања (енгл. Branch and Bound), метода одсецања равни (енгл. Cutting Plane Method), метода гранања и одсецања (енгл. Branch and Cut), метода гранања и оцењивања (енгл. Branch and Price). Оно што је заједничко за ове методе је да претрага креће од целог претраживачког простора који се дели на мање делове и претражује по одређеном систему, а решавање проблема обухвата низ потпроблема који су дефинисани на одређени начин.

Динамичко програмирање је засновано на рекурзивној подели проблема који се разматра на мање потпроблеме. Оптимизација се одвија у етапама као резултат низа делимичних одлука. Поступак избегава претрагу целог претраживачког простора одсецањем делимичних секвенци одлука које не воде до оптималног решења.

Метода гранања и ограничавања се базира на имплицитном пребројавању свих решења посматраног проблема. Претраживачки простор се претражује динамички генерисаним дрветом чији корен представља проблем који се решава и њему придружени претраживачки простор. Листови су потенцијална решења проблема, а унутрашњи чворови су потпроблеми на целом претраживачком простору.

Метода одсецања равни се састоји у одсецању делова допустивог скупа линеарне релаксације проблема који не садрже целобројне тачке. Одсецање се врши помоћу нових ограничења која се генеришу из информација о оптималном решењу проблема линеарне релаксације.

Метода гранања и одсецања је хибридизација методе гранања и ограничавања и методе одсецања равни. Хибридизација се остварује тако што се пре корака гранања у методи гранања и ограничавања примењује метод одсецања равни.

Метода гранања и оцењивања се базира на методи гранања и ограничавања и примењује се на проблеме целобројног програмирања са великим бројем променљивих. Основна идеја методе је у томе да се неке колоне у линеарној релаксацији проблема изоставе јер је проблем са великим бројем колона непрактичан за решавање, такође се у великом броју случајева испоставља да њима одговарајућа променљива има вредност нула у оптималном решењу. Затим да би се проверила оптималност решења линеарне релаксације, решава се потпроблем, који се зове *проблем оцењивања*, да би се евентуално пронашле колоне са исплативим смањеним трошковима. Ако се пронађу такве колоне, проблем линеарне релаксације се поново оптимизује. Гранање се дешава када се не пронађу исплативе колоне, а решење проблема линеарне релаксације проблема не задовољава услове целобројности својих променљивих.

Егзактне методе могу бити примењене на решавање инстанци малих димензија тешких проблема. Величина инстанци није једини индикатор који описује тежину проблема, већ је ту и његова структура. Тако се може десити да за неки одређени проблем инстанце малих димензија не могу да буду решене егзактним методама, а неке инстанце великих димензија истог проблема могу бити решене истим егзактним методом.

1.4.2 Апроксимативне методе

Апроксимативне методе могу се поделити на две групе: апроксимативни алгоритми и хеуристички алгоритми. За разлику од хеуристика, које обично могу да пронађу довољно „добра" решења у разумном времену, помоћу апроксимативних алгоритама се добијају решења чији се квалитет може доказати.

Дефиниција 1.4.1. $\varepsilon-$**Апроксимативни алгоритам.** Алгоритам има апроксимациони фактор ε ако му је временска сложеност полиномијална

и ако за сваку улазну инстанцу проблема даје решење a за које важи:

$$a \leq \varepsilon \cdot s \text{ ако је } \varepsilon > 1,$$

$$\varepsilon \cdot s \leq a \text{ ако је } \varepsilon < 1,$$

где је s глобално оптимално решење, а ε дефинише границу грешке. Апроксимациони фактор ε може бити константа или може зависити од димензије инстанце проблема или од неког другог параметра инстанце проблема.

Дефиниција 1.4.2. ε-апроксимативни алгоритам има границу грешке ε ако важи:

$$(s - \varepsilon) \leq a \leq (s + \varepsilon),$$

где је s глобално оптимално решење, а a решење добијено помоћу апроксимативног алгоритма.

Дефиниција 1.4.3. Полиномијална апроксимациона шема (енгл. **Polynomial-Time Approximation Scheme – PTAS**). Проблем припада класи PTAS ако за његово решавање постоји полиномијални $(1+\varepsilon)$-апроксимативни алгоритам за свако фиксирано $\varepsilon > 0$.

Дефиниција 1.4.4. Потпуно полиномијална апроксимациона шема (енгл. **Fully Polynomial-Time Approximation Scheme – FPTAS**). Проблем припада класи FPTAS ако за његово решавање постоји $(1 + \varepsilon)$-апроксимативни алгоритам који је полиномијалан у зависности од величине улазне инстанце проблема и полиномијалан у зависности од фактора $\frac{1}{\varepsilon}$ за свако фиксирано $\varepsilon > 0$.

Разлика између класа проблема PTAS и FPTAS је у томе што за проблем из класе FPTAS постоји $(1+\varepsilon)$-апроксимативни алгоритам који је полиномијалан у зависности од величине улазне инстанце проблема и полиномијалан у зависности од фактора $\frac{1}{\varepsilon}$, док за проблем који припада класи PTAS постоји $(1+\varepsilon)$-апроксимативни алгоритам који је полиномијалан у зависности од величине улазне инстанце проблема, али може бити експоненцијалан у зависности од фактора $\frac{1}{\varepsilon}$, за свако фиксирано $\varepsilon > 0$. Из овога се закључује да је класа проблема FPTAS подскуп класе проблема PTAS.

Проучавање апроксимативних алгоритама даје сазнања о тежини проблема и може помоћи у дизајнирању ефикасних хеуристика за његово решавање. Апроксимативни алгоритми су специфични за проблем који се решава (зависе од проблема), па им ова њихова карактеристика ограничава употребу. Штавише у пракси су решења добијена помоћу њих далеко од глобалних оптималних решења, па је њихова употреба у пракси веома ограничена.

Хеуристике налазе „добра" решења на инстанцама великих димензија. Могу се поделити у две групе: *специјалне хеуристике* и *метахеуристике*.

1.5 Метахеуристичке методе

За разлику од егзактних метода, метахеуристике су у стању да реше инстанце великих димензија дајући задовољавајућа решења у разумном времену. Са друге стране нема никакве гаранције да ће наћи оптимална решења, чак ни ограничена решења. Метахеуристике су веома популарне зато што је њихова употреба показала њихову ефикасност за решавање различитих великих и комплексних проблема. Специјалне хеуристике су дизајниране за решавање специфичних проблема и/или инстанци. Метахеуристике се могу схватити као алгоритми за општу употребу и могу бити примењени на решавање скоро сваког проблема оптимизације.

При дизајнирању метахеуристика у обзир морају бити узете две чињенице: истраживање претраживачког простора (диверзификација претраге) и искоришћавање најбољег пронађеног решења (интензификација претраге). Региони претраживачког простора се означавају као „обећавајући" уколико се у њима налазе решења доброг квалитета (а могуће је да је међу њима и глобални оптимум) која даље низом трансформација или потеза могу довести до глобалног оптимума. Циљ интензификације је да се истражи мање окружење неког решења, а циљ диверзификације је скок у неки други део претраживачког простора да би се избегло евентуално упадање у локални оптимум, а и да би били претражени сви обећавајући региони.

Метахеуристике се могу класификовати на различите начине[48]:

- **Метахеуристике инспирисане природом и оне који нису ин-**

спирисане природом: Многе метахеуристике су инспирисане природним процесима. *Еволуционни алгоритми* (енгл. *Evolutionary Algorithms – EA*) су инспирисани процесом еволуције као биолошким процесом. У еволутивним алгоритмима генерисана допустива решења представљају јединке које током генерација еволуирају слично као и код природног процеса еволуције кроз укрштање, мутацију и селекцију. *Вештачки имуни системи* (енгл. *Artificial Imune Systems – AIS*) су инспирисани биолошким имуним системима. Имуни систем је скуп одбрамбених механизама који бране тело од болести тако што откривају, идентификују и убијају патогене. Код вештачких имуних система патоген представља проблем оптимизације који треба да буде решен, а антитела су кандидати за решења. Тражење допустивих решења је аналогно процесу препознавања патогена од стране антитела. *Оптимизација колонијом пчела* (енгл. *Bee Colony Optimization – BCO*) је заснована на понашању пчела у заједници, а кораци процеса претраживања су инспирани начином на који пчеле траже нектар у природи. Алгоритам опонаша начин на који пчеле трагачи у природи обавештавају остале пчеле о изворима нектара изводећи плес на основу којег остале пчеле одлучују коју пчелу трагача ће пратити. *Оптимизација колонијом мрава* (енгл. *Ant Colony Optimization – ACO*) је инспирисана понашањем мрава у колонији при тражењу хране, опонашајући начин на који мрави остављају трагове феромона док се крећу од мравињака до извора хране. Док се мрави крећу од мравињака до извора хране они остављају траг феромона за собом да би остали мрави могли да их прате. Што више мрава прође истим путем од мравињака до извора хране на том путу ће траг феромона бити јачи па ће остали мрави моћи да прате тај траг. *Оптимизацијаројем честица* (енгл. *Particle Swarm Optimization – PSO*) је базирана на понашању риба и птица у јатима у природи. Базира се на комуникацији између јединки у јатима. Свака јединка у јату представља једно допустиво решење оптимизационог проблема. Јединка своје кретање усмерава ка најбољој јединки, аналогно претрага се креће ка најбољем решењу. *Симулирано каљење* (енгл. *Simulated Annealing – SA*) је инспирисано физичким

процесом каљења челика. Челик се загрева до тачке топљења, а затим постепено хлади док његови атоми стварају кристалну решетку и заузимају стабилно стање које када се охлади ствара чврсту структуру која има минималну унутрашњу енергију. Аналогија методе са физичким процесом се огледа у томе што решења проблема оптимизације одговарају стањима физичког система метала, а вредност функције циља одговара унутрашњој енергији система.

- **Методе које користе историју претраге и оне које не користе историју претраге:** Неке метахеуристичке методе не користе информације до којих дођу током свог рада. *Локална претрага* (енгл. *Local Search – LS*) спада у оне методе које не користе информације до којих су дошле током свог рада у даљем раду. Принцип локалне претраге подразумева да је дефинисана нека структура околина у простору допустивих решења. Сваком решењу се придружује неки подскуп скупа допустивих решења који представља околину тог решења. Од почетног решења се у њему одговрајућој околини тражи према унапред дефинисаним правилима сусед који ће представљати ново решење. Ефикасност конкретне методе локалне претраге базиране на претходно описаном принципу умногоме зависи од начина на који се дефинише појам „околине". *Табу претрага* (енгл. *Tabu Search – TS*) је типичан представник метода које користе информације о претходним фазама претраге. Табу претрага се се базира на принципу локалне претраге, али користи и такозвану адаптивну меморију, тј. памти неке податке о претходним фазама претраживања, који даље утичу на то који ће делови претраживачког простора бити претраживани.

- **Детерминистичке и стохастичке:** Детерминистичке методе решавају оптимизациони проблем доносећи детерминистичке одлуке, нпр. *локална претрага, табу претрага*, *метода променљивог спуста* (енгл. *Variable Neighborhood Descent – VND*). Метода промељивог спуста је базирана на методи локалне претраге, с тим што се код ове методе дефинише неколико структура околина у простору допустивих решења. Детерминизам се у овој методи огледа у томе што се свака од унапред

дефинисаних околина неког решења систематски претражује да би се пронашло најбоље решење у тој околини. У стохастичким методама, се током претраге неке одлуке доносе на случајан начин (нпр. *симулирано каљење, еволутивни алгоритми*). У детерминистичким методама, полазећи од истог почетног решења увек се долази до истог крајњег решења, док код стохастичких метода то не мора да буде случај, тј. могу се добити различита крајња решења полазећи од истог почетног решења.

- **Популационе и методе са једним решењем:** Методе са једним решењем (нпр. *локална претрага, симулирано каљење, метода промењљивих околина*) на почетку генеришу једно решење и њега трансформишу током претраге. Популационе методе (нпр. *оптимизацијаројем честица, еволутивни алгоритам, оптимизација колонијом пчела*) на почетку свог рада генеришу целу популацију решења и свако од тих решења се трансформише током претраге. Ове две класе метода имају комплементарне карактеристике. Методе са једним решењем су орјентисане ка интензификацији претраге тј. претрага се врши локално у некој области претраживачког простора. Популационе методе су орјентисане ка диверзификацији претраге, тј. претрага се врши у различитим областима претраживачког простора. Због својих комплементарних карактеристика популационе методе и методе са једним решењем се често комбинују и стварају се хибридне метахеуристике.

- **Итеративне и похлепне:** Итеративне методе на почетку генеришу комплетно решење (или популацију решења) и транформишу то решење кроз итерације. Типични представници су: *локална претрага, табу претрага, симулирано каљење, еволутивни алгоритми, оптимизацијаројем честица, оптимизација колонијом пчела, оптимизација колонијом мрава*, итд. Похлепне методе почињу своју претрагу од празног решења и у сваком кораку се додаје једна компонента решења док се не конструише комплетно допустиво решење. Пример похлепних метода је *похлепна случајно адаптивна процедура претраге* (енгл. *Greedy Randomized Adaptive Search Procedure – GRASP*). Похлепна случајно

адаптивна процедура претраге се у свакој итерацији састоји од две фазе: фазе конструкције и фазе локалне претраге. У фази конструкције се од понуђених кандидата на похлепан начин бира прва компонента решења, овај поступак се наставља док се не конструише цело допустиво решење. У фази локалне претраге се то решење побољшава. Ове фазе се понављају у одређеном броју итерација, а за оптимално решење се узима најбоље пронађено у свим итерацијама. Већина метахеуристичких метода су итеративне.

Више о метахеуристикама, класификацији, карактеристикама и применама на различите проблеме оптимизације може се наћи у [13] и [48].

Глава 2

Проблем динамичке локације објеката са избором добављача уз количински попуст

2.1 Дефиниција и опис проблема

Локацијски проблем објеката се бави одабиром локација за успостављање снабдевача и њиховим повезивањем са крајњим корисницима уз поштовање одређених ограничења у циљу задовољења потражње корисника, тако да укупни трошкови буду минимални. Ограничења која могу бири укључена су: фиксиран број локација снабдевача, ограничења капацитета снабдевача, снабдевачи су распоређени у два или више нивоа, трошкови успостављања снабдевача, афинитети корисника према одређеним снабдевачима итд. Прост локацијски проблем (енгл. Simple Plant Location Problem, Uncapacitated Facility Location Problem) је проблем који има један ниво услужних објеката (снабдевача) и нема додатних ограничења. У овом проблему се за услужне објекте бирају локације такве да укупни трошкови транспорта од услужних објеката до крајњих корисника буду минимални. Сложенија варијанта овог проблема је прост локацијски проблем са ограниченим капацитетима (енгл. Capacitated Facility Location Problem) тј. услужни објекти имају ограничене капацитете у смислу количине робе коју могу да ускла-

диште и њоме опслуже крајње кориснике. Даљим додавањем услова да снабдевачи буду распоређени у два нивоа, на најједноставнији локацијски проблем долази се до двонивоског локацијског проблема први пут изложеног у раду [24] где се поред услужних објеката првог нивоа посматрају и услужни објекти другог нивоа. Код овог проблема мрежа снабдевања се састоји од скупа услужних објеката првог нивоа, скупа услужних објеката другог нивоа и крањих корисника чије захтеве треба испунити. Циљ проблема је одредити локације за услужне објекте оба нивоа тако да буду испуњени сви захтеви корисника уз минималне трошкове снабдевања. И овај проблем може имати варијанту са неограниченим капацитетима услужних објеката, као и сложенију, варијанту са ограниченим капацитетима услужних објеката.

Код наведених проблема се претпоставља да су трошкови куповине робе од добављача (услужних објеката другог нивоа) линеарне функције количине наручене робе тј. да се јединична цена робе не мења у зависности од количине наручене робе [11], [19], као и да су захтеви крајњих корисника константни тј. не мењају се са временом. Овакве претпоставке су прилично строге и не одговрају у потпуности ситуацији која се среће у пракси. Наиме у пракси се често дешава да се продавци (услужни објекти првог нивоа) договарају са добављачима (услужни објекти другог нивоа) око дисконтне цене тј. цене која зависи од количине наручене робе, па се самим тим смањује и цена робе за крајње кориснике [39], [16].

Одабир локација за успостављање продавница (услужних објеката првог нивоа) је дугорочна стратегијска одлука, и доношење такве одлуке је веома сложен задатак, нарочито када се услови мењају са временом. У ту сврху се разматрају такозвани *динамички локацијски проблеми* где се посматра промена услова са временом, како би се донела прецизнија и исплативија одлука. По [19] динамички локацијски модели могу бити подељени на дискретне и континуалне у зависности од тога да ли се промене посматрају у континуалном или дискретном времену тј. временским периодима одређеног трајања, као и на коначне и бесконачне у зависности за колико времена унапред се доносе одлуке. Увођење више периода у локацијски проблем и посматрање динамичке варијанте може довести до значајног смањења укупних трошкова (чак и до 50%) [2]. Иако су овакви динамички модели прецизнији када

се ради са временски зависним улазним подацима, комплексност оваквих проблема се драстично увећава са порастом броја посматраних периода [29].

У литератури су проблеми са избором добављача уз количински попуст и динамички локацијски проблем изучавани одвојено док се у раду [18] први пут ова два проблема интегришу и посматрају као један проблем који је назван проблем динамичке локације објеката са избором добављача уз количински попуст (енгл. Dynamic Facility Location With Supplier Selection Under Quantity Discount-DFLP-QD).

У овом мастер раду, посматрана је варијанта проблема из рада [18] са неограниченим капацитетима која је добијена тако што је уклоњен захтев да продавнице (услужни објекти првог нивоа) имају ограничене капацитете у погледу количине робе коју могу да потражују од добављача и да продају крајњим корисницима, тј. њихови капацитети су неограничени.

У овом мастер раду први пут у литератури је разматран проблем са неограниченим капацитетима који представља потпроблем из рада [18] и може се назвати *Проблем динамичке локације објеката са избором добављача уз количински попуст са неограниченим капацитетима* (енгл. Uncapacitated Dynamic Facility Location With Supplier Selection Under Quantity Discount UDFLP-QD). Циљ проблема UDFLP-QD је одредити локације за успостављање продавница (услужних објеката првог нивоа) за сваки период, одабрати добављаче од којих се поручује роба (услужне објекте другог нивоа) за сваки период, тако да се задовоље потребе крајњих корисника за једним производом минимизујући укупне трошкове мреже снабдевања који представљају трошкове отварања и одржавања продавница, трошкове набављања робе, трошкове транспорта робе од добављача до продавница, као и трошкове транспорта робе од продавница до крајњих корисника. У поставци проблема, сваки добављач нуди робу свим продавницама са унапред дефинисаним количинским попустима. Ти унапред дефинисани попусти су различити за различите добављаче. Такође сви добављачи имају једнак број попуста, али је количина поручене робе која је потребна да се оствари неки од тих попуста различита за различите добављаче. Са друге стране продавци посматрају промене на тржишту где се периодично јавља већа потражња неке робе. Претпоставка је да крива потражње има сличан облик као крива

потражње у Басовом дифузионом моделу [5] када се уводи нови производ на тржиште. Одлука о отварању продавница се обично доноси на дужи временски период (најмање неколико година), а затварање је неисплативо осим уколико се не ради о веома дугачком временском периоду [26]. Циљ продавница је да задовоље потребе крајњих корисника уз минималне трошкове одређујући (i) оптималне локације за отварање, (ii) период отварања сваке продавнице и (iii) количину робе која се поручује од сваког добављача у сваком периоду. У поставци проблема, продавци потражују робу од добављача и продају је крајњим корисницима. Са друге стране, добављачи нуде попуст у зависности од количине наручене робе, а крајњи корисници мењају своје захтеве у зависности од периода. У оваквом окружењу продавци траже најбоље локације за своје објекте са циљем да задовоље потребе крајњих корисника уз минималне укупне трошкове. Без умањења општости претпоставља се да сви добављачи нуде исти број дисконтних нивоа. Уколико неки од добављача нуди мањи број дисконтних нивоа онда се цена на преосталим дисконтним нивоима сматра константном. Цене и попусти на истим дисконтним нивоима варирају од добављача до добављача, као што варирају и границе дисконтних нивоа на којима добављачи нуде смањену цену робе.

2.2 Математичка формулација

У овој подсекцији, разматрани проблем биће формулисан као проблем мешовитог целобројног програмирања. У математичком моделу су коришћене следеће ознаке за скупове и параметре:

- $\mathscr{M}-$ скуп крајњих корисника $\{1,\ldots,M\}$ индексиран са m,
- $\mathscr{I}-$ скуп добављача $\{1,\ldots,I\}$ индексиран са i,
- $\mathscr{J}-$ скуп потенцијалних локација продаваца $\{1,\ldots,J\}$ индексиран са j,
- $\mathscr{L}-$ скуп периода $\{1,\ldots,L\}$ индексиран са l,

- $\mathscr{D}-$ скуп дисконтних нивоа $\{1, \ldots, D\}$ индексиран са d,
- $\mathscr{D}_1-$ скуп дисконтних нивоа без првог $\{2, \ldots, D\}$ индексиран са d,
- $a_{id}-$ горња граница количине наручене робе за дисконтни ниво d за добављача i,
- $p_{id}-$ цена робе (по јединици количине) купљене од добављача i на дисконтном нивоу d,
- $\lambda_{ml}-$ захтев крајњег корисника m у периоду l,
- $c^1_{ij}-$ цена транспорта (по јединици количине робе) од добављача i до продавца на локацији j,
- $c^2_{jm}-$ цена транспорта (по јединици количине робе) од продавца на локацији j до крајњег корисника m,
- F_j- фиксна цена успостављања продавнице на локацији j,
- o_j- цена одржавања (по периоду) продавнице на локацији j,
- $C-$ велика позитивна константа,
- $K_{id}-$ помоћни параметар за додатну цену робе купљене од добављача i на дисконтном нивоу d.

За све добављаче се претпоставља да робу нуде у D дисконтних нивоа. Добављач i продаје робу по цени p_{id} по комаду када је укупна количина робе које се продаје између $a_{id-1} + 1$ и a_{id} (претпоставља се да је $a_{i0} = 0$ за сваког добављача i). На сваком дисконтном нивоу се уводи „додатна" вредност на цену робе због веће цене робе на претходном дисконтном нивоу. Ова додатна цена је означена са K_{id} за добављача i на дисконтном нивоу d. На пример, нека добављач продаје комад робе по цени од 100 динара за првих 100 комада, по цени од 90 динара по комаду за следећих 100 комада и по цени од 80 динара по комаду за следећих 100 комада. Ако продавац наручи 250 комада робе, робу наручује на трећем дисконтном нивоу, плаћа 80 динара по комаду за свих 250 комада по цени на трећем дисконтном

нивоу, плус $10 \times 100 + 10 \times 200 = 3000$ динара као додатну цену, па је у овом случају $K_{i3} = 3000$. Све вредности K_{id} се помоћу a_{id} и p_{id} могу израчунати на следећи начин:

$$K_{id} = K_{id-1} + (p_{id-1} - p_{id})a_{id-1}, \quad \forall\, d \in \mathscr{D}_1 \ \text{ где је } \ K_{i1} = 0.$$

Да би изложени проблем био записан као проблем мешовитог целобројног програмирања коришћене су следеће ненегативне реалне променљиве:
x^1_{ijl} = количина робе послате од стране добављача i продавцу j у периоду l
x^2_{jml} = количина робе послате од стране продавца j крајњем кориснику m у периоду l
q_{idl} = количина робе купљене од добављача i на дисконтном нивоу d ако је обављена куповина од добављача i на дисконтном нивоу d у периоду l
Q_{il} = укупна количина робе купљене од добављача i у периоду l

Такође, математички модел користи следеће бинарне променљиве:

$$y_{idl} = \begin{cases} 1, & \text{ако је куповина обављена од добављача } i \text{ на дисконтном нивоу } d \text{ у периоду } l \\ 0, & \text{иначе,} \end{cases}$$

$$t_{jl} = \begin{cases} 1, & \text{ако је продавница } j \text{ отворена у периоду } l \\ 0, & \text{иначе.} \end{cases}$$

Како више продавница може поручити робу од једног добављача у неком периоду, да би се сачувала информација о сваком поручивању робе користи се променљива x^1_{ijl}, а укупна количина робе која је поручена од i-тог добављача у l-том периоду се налази у променљивој Q_{il}. Слично више крајњих корисника може купити робу у истој продавници па се за чување информације о продаји робе сваком крајњем кориснику m у сваком периоду l користи променљива x^2_{jml}.

Користећи горе наведену нотацију, разматрани проблем UDFLP-QD се може представити као проблем мешовитог целобројног програмирања на следећи начин:

$$\min \sum_{i\in\mathscr{I}}\sum_{l\in\mathscr{L}}\sum_{d\in\mathscr{D}} p_{id}q_{idl} + \sum_{i\in\mathscr{I}}\sum_{l\in\mathscr{L}}\sum_{d\in\mathscr{D}_1} p_{id}y_{idl}a_{id-1} + \sum_{i\in\mathscr{I}}\sum_{l\in\mathscr{L}}\sum_{d\in\mathscr{D}} K_{id}y_{idl} + \sum_{j\in\mathscr{J}}\sum_{l\in\mathscr{L}} F_j t_{jl} + \sum_{j\in\mathscr{J}}\sum_{l\in\mathscr{L}} o_j(L-l+1)t_{jl} + \sum_{i\in\mathscr{I}}\sum_{j\in\mathscr{J}}\sum_{l\in\mathscr{L}} c^1_{ij}x^1_{ijl} + \sum_{j\in\mathscr{J}}\sum_{m\in\mathscr{M}}\sum_{l\in\mathscr{L}} c^2_{jm}x^2_{jml} \tag{2.1}$$

при ограничењима:

$$\sum_{j\in\mathscr{J}} x^2_{jml} = \lambda_{ml} \quad \forall\, m \in \mathscr{M}, \quad l \in \mathscr{L} \tag{2.2}$$

$$\sum_{i\in\mathscr{I}} x^1_{ijl} = \sum_{m\in\mathscr{M}} x^2_{jml} \quad \forall\, j \in \mathscr{J}, \quad l \in \mathscr{L} \tag{2.3}$$

$$Q_{il} = \sum_{j\in\mathscr{J}} x^1_{ijl} \quad \forall\, i \in \mathscr{I}, \quad l \in \mathscr{L} \tag{2.4}$$

$$Q_{il} = \sum_{d\in\mathscr{D}} q_{idl} + \sum_{d\in\mathscr{D}_1} y_{idl}a_{id-1} \quad \forall\, i \in \mathscr{I}, \quad l \in \mathscr{L} \tag{2.5}$$

$$q_{i1l} \le a_{i1}y_{i1l} \quad \forall\, i \in \mathscr{I}, \quad l \in \mathscr{L} \tag{2.6}$$

$$q_{idl} \le (a_{id} - a_{id-1})y_{idl} \quad \forall\, i \in \mathscr{I}, \quad l \in \mathscr{L}, \quad d \in \mathscr{D}_1 \tag{2.7}$$

$$\sum_{d\in\mathscr{D}} y_{idl} \le 1 \quad \forall\, i \in \mathscr{I}, \quad l \in \mathscr{L} \tag{2.8}$$

$$\sum_{i\in\mathscr{I}} x^1_{ijl} \le Ct_{jl} \quad \forall\, j \in \mathscr{J}, \quad l \in \mathscr{L} \tag{2.9}$$

$$\sum_{l\in\mathscr{L}} t_{jl} \le 1 \quad \forall\, j \in \mathscr{J} \tag{2.10}$$

$$x^1_{ijl} \ge 0 \quad \forall\, i \in \mathscr{I}, \quad j \in \mathscr{J}, \quad l \in \mathscr{L} \tag{2.11}$$

$$x^2_{jml} \ge 0 \quad \forall\, j \in \mathscr{J}, \quad m \in \mathscr{M}, \quad l \in \mathscr{L} \tag{2.12}$$

$$q_{idl} \geq 0 \quad \forall\, i \in \mathscr{I}, \quad d \in \mathscr{D}, \quad l \in \mathscr{L} \tag{2.13}$$

$$y_{idl} \in \{0,1\} \quad \forall\, i \in \mathscr{I}, \quad d \in \mathscr{D}, \quad l \in \mathscr{L} \tag{2.14}$$

$$t_{jl} \in \{0,1\} \quad \forall\, j \in \mathscr{J}, \quad l \in \mathscr{L} \tag{2.15}$$

Функција укупних трошкова 2.1 укључује (i) трошкове куповине, (ii) трошкове отварања продавница, (iii) трошкове рада продавница, и (iv) транспортне трошкове. Прве три суме у 2.1 одређују укупне трошкове куповине узимајући у обзир дисконтне нивое. Четврта сума представља трошкове отварања продавница, док пета сума представља трошкове одржавања продавница. На крају последње две суме представљају трошкове транспорта робе од добављача до продавница, као и од продавница до крајњих купаца респективно. Ограничење 2.2 означава да је захтев сваког корисника задовољен у сваком периоду. Ограничење 2.3 означава да је задовољен баланс протока робе кроз сваку продавницу у сваком периоду тј. количина робе која је набављена од добављача једнака је количини робе која је продата крајњим корисницима. Ограничење 2.4 одређује укупну количину робе која је наручена од сваког добављача у сваком периоду. Помоћу ограничења 2.5−2.8 одређује се дисконтни ниво за робу наручену од сваког добављача у сваком периоду. Ограничење 2.9 осигурава да поручивање робе од добављача могу да врше само успостављене продавнице. Ограничење 2.10 одређује у ком периоду је успостављена продавница на локацији j ако је уопште успостављена. Ограничења 2.11−2.15 одређују тип и знак променљивих.

Математички модел за UDFLP-QD је добијен тако што је ограничење

$$\sum_{i \in \mathscr{I}} x^1_{ijl} \leq C_j t_{jl} \quad \forall\, j \in \mathscr{J}, \quad l \in \mathscr{L}$$

(C_j капацитет j-те продавнице) из рада [18] релаксирано и добијено ново ограничење које не садржи ограничене капацитете продавница:

$$\sum_{i \in \mathscr{I}} x^1_{ijl} \leq C t_{jl} \quad \forall\, j \in \mathscr{J}, \quad l \in \mathscr{L}$$

где је C велика позитивна константа.

2.3 Преглед релевантне литературе

С обзиром на практични значај локацијских проблема, у литератури постоји велики број различитих метода за њихово решавање. Преглед метода за решавање се може наћи у [1] и [14]. У овом поглављу биће дат кратак осврт на методе из литературе које су предложене за решавање четири класе локацијских проблема којима припада разматрани проблем UDFLP-QD, а то су: двонивоски локацијски проблеми, динамички локацијски проблеми, локацијски проблеми са количинским попустом на цену робе и локацијски проблеми са избором добављача.

Двонивоски локацијски проблеми се у пракси највише срећу у планирању изградње здравствених установа и планирању и изградњи објаката за производњу и дистрибуцију [55]. Најчешћи начини решавања ових проблема су засновани на Лагранжевој[1] хеуристици [52] и методи гранања и ограничавања [24], [42]. У [53] за решавање варијанте двонивовског проблема са једним снабдевачем и ограничењима капацитета коришћена је метода гранања и ограничавања која је заснована на Лагранжевој методи релаксације. Што се хеуристичких метода тиче у [34] је представљен генетски алгоритам укључујући имплементацију са шемом динамичког програмирања за налажење успостављених ресурса како би се задовољиле потребе крајњих корисника. Према аутору [34], компонента динамичког програмирања је најзаслужнија за решавање инстанци великих димензија (до 2000 потенцијалних ресурса) за кратко време. Касније у [30] и [35] уведена су унапређења генетског алгоритма (унапређење компоненте динамичког програмирања и убацивање локалне претраге у генетски алгоритам што је у ствари меметски алгоритам). Други меметски алгоритам је представљен у раду [37] за решавање проблема изложеног у раду [32]. У раду [20] је развијена метахеуристика за двонивовски локацијски проблем са неограниченим капацитетима са ограничењем да се може успоставити само један ресурс на другом нивоу за сваки успостављени ресурс на првом нивоу. Аутори су развили вишеслојну претрагу са променљивим околинама. Термин вишеслојна долази од партиционисања структура околина у неколико нивоа, где је за сваки слој

[1] Joseph-Louis, comte de Lagrange 1736 – 1813 - француски математичар

примењена претрага са променљивим околинама.

У раду [8] је предложен двофазни алгоритам за решавање динамичког локацијског проблема тако што се у првој фази методом гранања и ограничавања у сваком временском периоду одређује скуп кандидата за решења, а затим у другој фази решење бива конструисано користећи динамичко програмирање. У раду [26] је такође представљен двофазни алгоритам за решавање двонивоског динамичког локацијског проблема. У првој фази предложеног алгоритма коришћен је алгоритам груписања да би се одређени крајњи корисници груписали у скупове, а затим је у другој фази проблем решаван за ове скупове. У раду [3] динамички локацијски проблем је решен методом симулираног каљења, у раду [41] методом табу претраге, а у раду [28] генетским алгоритмом. У раду [45] је решавана варијанта динамичког локацијског проблема са модуларним капацитетима хибридизацијом методе променљивог спуста и генетског алгоритма.

У раду [46] је први пут изложен локацијски проблем где су у обзир узете дисконтна цена робе, а цене отварања објеката, одржавања објеката, као и цене транспорта робе су конкавне функције количине наручене робе. Проблем је решаван методом гранања и ограничавања. Сличан проблем је представљен у раду [25], с тим што је цена одржавања посматрана као конкавна функција количине наручене робе. За решавање овог проблема у [25] предложена је итеративна метода. Итеративна метода дефинише и решава низ транспортних проблема како би на крају пронашла решење проблема. У раду [15] разматран је сличан проблем где је претпостављено да су све функције трошкова осим трошкова превоза конкавне функције количине наручене робе. Примећено је да се са повећањем количине наручене робе не може достићи дисконтни ниво па конкавне функције трошкова мењају облик и постају линеарне. За решавање проблема у [15] је имплементирана Лагранжева хеуристика.

Проблеми са избором добављача су подељени на две категорије[21], са једним извором тј. где крајњи корисник добављање робе врши само од једог добављача и више извора, где је дозвољено да продавнице врше добављање робе од више добављача да би се у потпуности задовољили захтеви крајњих корисника. Добављање робе од више добављача је понекад потребно због

ограничених капацитета добављача [54]. У раду [6] је при пут разматран локацијски проблем са избором добављача. За решавање проблема изложеног у раду [6] је предложена хеуристичка метода. Двонивоски проблем са више временских периода и више различитих производа је предложен први пут у [47] где су у разматрање узете цене са попустом на сваки производ као и на укупну количину свих наручених производа. У овом раду је први пут и предложен модел мешовитог целобројног програмирања.

Поред ових једнонивовских проблема постоје и радови у којима се изучавају двонивоски проблеми са избором добављача, али ниједан од њих не разматра динамичку локацију објеката. Проблем сличан проблему UDFLP-QD који се проучава у овом мастер раду је разматран у раду [44]. У раду [44] је анализиран двонивоски проблем, али без избора добављача јер су они унапред познати. Такође је представљен математички модел мешовитог целобројног програмирања, а за решавање је имплементиран напредни метод одсецања равни. У раду [18] је разматрана варијанта проблема из овог мастер рада са ограниченим капацитетима. Решен је на два начина. На први начин је решен тако што је подељен на два потпроблема, где први потпорблем разматра само продавнице и добављаче, а други крајње кориснике и продавнице. Како су ова два потпроблема повезана они су решавани итеративно. На други начин је решен тако што је посматран сваки период засебно и такви потпроблеми решени, а онда је динамичким програмирањем решен цео проблем.

Глава 3

Метода променљивих околина за решавање проблема UDFLP-QD

3.1 Основне поставке методе променљивих околина

Метода променљивих околина (енгл. Variable Neighborhood Search-VNS) је први пут изложена 1997. године у раду [38]. Метода је развијена за решавање проблема дискретне оптимизације, и због своје релативне једноставности (у погледу имплементације и уштеде на меморијским и временским ресурсима) је погодна за широку употребу. Основна идеја методе је у систематској промени околина у потрази за оптималним решењем и заснована је на следећим чињеницама [23], [27]:

(i) Локални минимум у једној околини није обавезно, и истовремено локални минимум за другу околину.

(ii) Глобални минимум је локални минимум за све околине.

(iii) Емпиријски је показано да су за већину проблема, сви или скоро сви локални минимуми близу једни другима.

Прва чињеница се користи за проналажење минимума у свакој од унапред дефинисаних оклина. Друга чињеница указује на то да треба користити неколико околина, ако је локални минимум који је пронађен лошег квалитета. Трећа чињеница указује на то да претрагу треба интензивно вршити у близини већ пронађеног минимума. Ове три чињенице се могу користити детерминистички, стохастички или комбиновано у циљу добијања различитих варијанти методе промењљивих околина.

При конструкцији методе промењљивих околина најпре треба дефинисати коначан скуп околина $\mathscr{N} = \{N_1, \ldots, N_{k_{max}}\}$ простора S допустивих решења, где $N_k(x)$ представља k-ту околину решења x. Уобичајен начин дефинисања околина је да оне буду угњеждене тј. $N_1 \subset N_2 \subset N_3 \subset \ldots \subset N_{k_{max}}$, али не и обавезно. Околине могу бити дефинисане у односу на различите метрике. Истовремено са дефинисањем скупа околина дефинише се и њихов редослед.

Основна варијанта VNS методе састоји се из следећих корака: *фаза побољшања* (енгл. Improvement phase), која се користи за могуће побољшање тренутног решења, као и такозвана *фаза размрдавања* (енгл. Shaking phase) у којој се постиже диверзификација решења у циљу изласка из локалних минимума. Ове две фазе заједно са *фазом померања* (енгл. Move or Not) се извршавају сукцесивно (једна за другом), најпре *фаза размрдавања*, затим *фаза побољшања* и на крају *фаза померања* док се не испуни неки унапред задати критеријум заустављања.

3.1.1 Фаза размрдавања

Циљ *фазе размрдавања* је да се потенцијално избегне упадање у локални минимум. У фази размрдавања се у k-тој околини $N_k(x)$ решења x на случајан начин бира решење x'.

Алгоритам 1 Размрдавање

1: Учитај x, k;
2: изабери произвољно $x' \in N_k(x)$;
3: **врати** x';

3.1.2 Фаза побољшања

У *фази побољшања* методе промељивих околина најчешће се користи локална претрага или нека друга метода заснована на локалном претраживању.

Локална претрага се базира на претраживању околине $\mathcal{N}(x)$ тренутног решења x у свакој итерацији. У општем случају околина $\mathcal{N}(x)$ решења x, која се претражује у локалној претрази не мора бити и најчешће није иста као околине које су дефинисане у VNS методи. Полазећи од почетног решења x у свакој итерацији се тражи боље решење x' (ако постоји). Локална претрага се зауставља када је пронађено решење x локални минимум у околини $\mathcal{N}(x)$, или је испуњен неки други критеријум заустављања.

Најчешће стратегије претраге које се корисрте у локалној претрази су:

- *Прво побољшање* (енгл. *first improvement*): Овом стратегијом се у околини $\mathcal{N}(x)$ текућег решења x тражи први сусед x' који побољшава вредност функције циља. Затим се тај сусед узима за текуће решење. Овом стратегијом се не претражује цела околина $\mathcal{N}(x)$ решења x већ њен део. У најгорем случају (тј. када није пронађено побољшање) претражује се цела околина $\mathcal{N}(x)$ решења x.

- *$r-$то побољшање* (енгл. *$r-$th improvement*): У овој стратегији се у околини $\mathcal{N}(x)$ текућег решења x тражи $r-$ти по реду сусед који побољшава вредност функције циља, где је r параметар који се задаје.

- *Најбоље побољшање* (енгл. *best improvement*): Овом стратегијом се се тражи сусед у околини $\mathcal{N}(x)$ текућег решења x који највише побољшава вредност функције циља. Сусед се одређује потпуно детерминистички тј. претрагом целе околине текућег решења. Ова врста претраге може бити временски захтевна за велике околине.

Алгоритам 2 Локална претрага са првим побољшањем

Учитај x;
понављај
    У околини $\mathcal{N}(x)$ текућег решења x пронаћи првог суседа x' који побољшава вредност функције циља;
    $x \leftarrow x'$;
док није задовољен критеријум заустављања;
врати x;

Алгоритам 3 Локална претрага са $r-$тим побољшањем

Учитај x, r;
понављај
    У околини $\mathcal{N}(x)$ текућег решења x пронаћи r-тог по реду суседа x' који побољшава вредност функције циља;
    $x \leftarrow x'$;
док није задовољен критеријум заустављања;
врати x;

Алгоритам 4 Локална претрага са најбољим побољшањем

Учитај x;
понављај
    $x' \leftarrow x$;
    $x \leftarrow argmin_{y \in \mathcal{N}(x')} f(y)$;
док је $f(x') < f(x)$;
врати x';

3.1.3 Фаза помераја

Сврха *фазе ūомераја* је усмеравање методе променљивих околина у претраживању простора допустивих решења. Тј. у *фази ūомераја* се одлучује да ли ће новопронађено решење бити прихваћено или одбачено, а затим се врши избор околине у којој се наставља даља претрага. Приступи у фази помераја који се најчешће користе у VNS имплементацијама су наведени и укратко описани.

Секвенцијална ūромена околине (енгл. *Sequential neighborhood change step*)[51]: Ако је дошло до унапређења текућег решења у *фази ūобољшања* у некој од околина онда се прихвата унапређено решење и претрага се наставља од прве околине тог решења, иначе се ново решење одбацује, а

претрага се наставља у следећој околини текућег решења.

Алгоритам 5 Секвенцијална промена околине

```
1: Учитај x, x', k;
2: ако је f(x') < f(x) онда
3:     x ← x';
4:     k = 1;
5: иначе
6:     k = k + 1;
7: врати x;
```

Циклична промена околине (енгл. *Cyclic neighborhood change step*)[49]: У овом начину промене околине без обзира да ли је дошло до побољшања текућег решења претрага се наставља у следећој околини, а решење се прихвата или одбацује у зависности да ли је дошло до побољшања или није.

Алгоритам 6 Циклична промена околине

```
1: Учитај x, x', k;
2:     k = k + 1;
3: ако је f(x') < f(x) онда
4:     x ← x';
5: врати x;
```

Цеваста промена околине (енгл. *Pipe neighborhood change step*)[50]: Ако је дошло до побољшања текућег решења у некој околини прихвата се унапређено решење и наставља са даљом претрагом те околине. У супротном се одбацује ново решење и претрага се наставља у следећој околини текућег решења.

Алгоритам 7 Цеваста промена околине

```
1: Учитај x, x', k;
2: ако је f(x') < f(x) онда
3:     x ← x';
4: иначе
5:     k = k + 1;
6: врати x;
```

Искошена промена околине (енгл. *Skewed neighborhood change step*)[7]: У овој методи промене околине може доћи до прихватања и оног решења које не унапређује текуће решење. Сврха овакве промене околине је претрага оних долина које су удаљене од текућег решења. При евалуацији решења чије се прихватање разматра у обзир се поред вредности функције циља у новом решењу као и у тренутном решењу узима и раздаљина између та два решења. Функција евалуације решења у овој методи која се користи за минимизацију функције циља f може бити дефинсана на следећи начин[23]:

$$g(x, y) = f(x) - f(y) - \alpha d(x, y)$$

где је $\alpha > 0$ реални параметар, а $d(x, y)$ представља растојање између решења x и y. Параметар α се бира тако да могу да се претражују долине које су далеко од текућег решења x када је $f(y)$ веће од $f(x)$, али не превише веће. Задовољавајуће вредности се траже експериментално сваки пут. Да би се избегло померање у неко решење које је блиско текућем решењу x за параметар α се узимају веће вредности када је $d(x, y)$ мало. Поред комбинације са секвенцијалном променом околине изложене у алгоритму 5 ова метода се може комбиновати и са цикличном и цевастом променом околине.

Алгоритам 8 Искошена промена околине

```
1: Учитај x, x', k, α
2: ако је f(x') − f(x) < αd(x, x') онда
3:     x ← x';
4:     k = 1;
5: иначе
6:     k = k + 1;
7: врати x;
```

3.2 Варијанте методе променљивих околина

Комбиновањем изложених алгоритама за *фазу размрдавања*, *фазу побољшања*, и *фазу помераја* добијају се различите варијанте методе променљивих околина.

3.2.1 Основна метода променљивих околина

Основна метода променљивих околина се састоји од наизменичног извршавања процедуре размрдавања 1, процедуре локалног претраживања 2, као и процедуре секвенцијалне промене околине 5, све до испуњења неког унапред задатог критеријума заустављања. Унапред је дефинисано k_{max} околина у којима се врши претрага. Критеријум заустављања је најчешће максималан број итерација, поред овог критеријума заустављања могу се дефинисати и другачији критеријуми, максимално време извршавања, максималан број понављања решења, или нека комбинација два или више критеријума заустављања.

Алгоритам 9 Основна метода променљивих околина

```
1: Учитај x, k_max;
2: понављај
3:     k = 1;
4:     понављај
5:         x' ←Размрдавање(x, k);
6:         x'' ←Локална пртерага(x');
7:         x ←Секвенцијална_промена_околине(x, x'', k);
8:     док је k ≤ k_max
9: док није задовољен критеријум заустављања
10: врати x;
```

3.2.2 Редукована метода променљивих околина

Из основне методе променљивих околина могу се добити различите варијанте те методе, једна од њих је и редукована метода променљивих околина (енгл. Reduced variable neighborhood search − RVNS). Ова метода се састоји само од фазе размрдавања и фазе помераја док је изостављена фаза побољшања. Веома је погодна за решавање инстанци великих димензија, јер се остварује уштеда у времену, а добијају се квалитетна решења. Претрага се и у овој методи врши у k_{max} околина које су унапред дефинисане. Могу се користити исти критеријуми заустављања као код основне методе променљивих околина.

Алгоритам 10 Редукована метода променљивих околина

```
1: Учитај x, k_max;
2: понављај
3:     k = 1;
4:     понављај
5:         x' ←Размрдавање(x, k);
6:         x ←Секвенцијална_промена_околине(x, x', k);
7:     док је k ≤ k_max;
8: док није испуњен критеријум заустављања;
9: врати x;
```

3.2.3 Метода променљивог спуста

Метода променљивог спуста (енгл. Variable Neighborhood Descent − VND) се базира на чињеници да решење које је локални минимум за већину околина има веће шансе да буде глобални минимум, од решења које је локални минимум у само једној околини, као и на чињеници да локални минимум у једној околини није локални минимум у другој околини. Да би се могла користити метода променљивог спуста најпре треба да буде дефнисан скуп околина $N_l(l = 1, \ldots, l_{max})$. Од почетног решења x метода променљивог спуста са секвенцијалном променом околина итеративно претражује околине N_l, $1 \leq l \leq l_{max}$, једну за другом по утврђеном редоследу. Док год се проналази побољшање тренутног решења у некој околини, то решење се узима за тренутно и даља претрага се наставља у првој околини тог новог тренутног решења. Цео процес претраге се зауставља када се за тренутно решење не може наћи побољшање ни у једној од l_{max} околина. Заменом процедуре секвенцијалне промене околине другим процедурама промене околине добијају се другачије варијанте методе променљивог спуста, а за проналажење бољег решења може користити и стратегија првог побољшања решења, а метода променљивих околина која користи стратегију најбољег побољшања и процедуру секвенцијалне промене околине је дата у алгоритму 11.

Алгоритам 11 Метода променљивог спуста

Учитај x, l_{max};
$l = 1$;
понављај
    $x' \leftarrow argmin_{y \in N_l(x)} f(y)$;
    $x \leftarrow$Секвенцијална_промена_околине(x, x', l);
док је $l \leq l_{max}$
врати x;

3.2.4 Општа метода променљивих околина

У општој методи променљивих околина процедура локалне претраге из фазе побољшања се мења методом променљивог спуста 11 или неком варијантом VND методе. Овде ваља напоменути да структура околина која се јавља у методи променљивог спуста не мора бити иста као структура околина која се јавља у самој методи променљивих околина. У општој методи околина дефинисано је k_{max} околина у којима се врши претрага, а у VND методи у оквиру опште методе променљивих околина је дефинисано l_{max} околина у којима се врши претрага. Могу бити коришћени различити критеријуми заустављања, као и у случају основне методе променљивих околина.

Алгоритам 12 Општа метода променљивих околина

Учитај x, k_{max}, l_{max};
понављај
    $k = 1$;
    понављај
        $x' \leftarrow$Размрдавање(x, k);
        $x'' \leftarrow$Метода_променљивог_спуста(x', l_{max});
        $x \leftarrow$Секвенцијална_промена_околине(x, x'', k);
    док је $k \leq k_{max}$
док није задовољен критеријум заустављања
врати x;

3.2.5 Метода променљивих околина заснована на декомпозицији проблема

Основна идеја методе променљивих околина засноване на декомпозицији проблема (енгл. Variable neighborhood decomposition search − VNDS) је да се систематски мења величина потпроблема који се јављају при декомпозицији почетног проблема од мањих ка већим. Штавише и тако добијени потпроблеми се решавају помоћу VNS-а, што води до двоструког VNS приступа. За разлику од других варијанти VNS методе, у VNDS методи се процедура побољшања не извршава у целом претраживачком простору, него у редукованом претраживачком простору који одговара потпроблему који је редукован из почетног проблема. У овом алгоритму решење y одговара решењу потпроблема. Уобичајно се генерише тако да има k различитих атрибута од тренутног решења x целог проблема. У фази побољшања се побољшава само решење y потпроблема у редукованом претраживачком простору. Затим се решење које је добијено кроз процедуру побољшања користи да би се креирало ново решење почетног проблема тако што се решење y' које је добијено помоћу процедуре побољшања спаја са парцијалним решењем $x' \setminus y$ и заједно дају цело решење полазног проблема. Као процедура побољшања у VNDS може се користити нека процедура локалне претраге, VND метода, или нека од варијанти VNS методе. Критеријум заустављања као и код осталих варијанти VNS методе може бити максималан број итерација, максимално време извршавања методе, максималан број понављања решења, а могу се и комбиновати два или више критеријума заустављања.

Алгоритам 13 Метода променљивих околина заснована на декомпозицији проблема

```
1: Учитај x, k_max;
2: понављај
3:     k = 1;
4:     понављај
5:         x' ← Размрдавање(x, k);
6:         y ← x' \ x;
7:         y' ← Процедура_побољшања(y);
8:         x'' ← (x' \ y) ∪ y';
9:         x ←Секвенцијална_промена_околине(x, x'', k);
10:    док је k ≤ k_max;
11: док није испуњен критеријум заустављања;
12: врати x;
```

Поред описаних варијанти у овом раду постоје и друге варијанте VNS методе: закошена метода променљивих околина (енгл. Skewed variable neighborhood search − S-VNS), угњеждена метода променљивих околина (енгл. Nested VNS). Поред ових варијанти који су прилагођене за решавање проблема комбинаторне оптимизације постоје и варијанте које су прилагођене за решавање проблема континуалне оптимизације. Више о тим методама се може наћи у [23].

3.3 Заједнички елементи имплементираних варијанти VNS методе за решавање UDFLP-QD

3.3.1 Кодирање и генерисање решења

Разматрани проблем UDFLP-QD не укључује ограничења капацитета на успостављене објекте. Ограничење 2.10 каже да једна продавница може бити успостављена у највише једном периоду. У складу са тим, почетно решење VNS методе је генерисано као двосегментни бинарни низ дужине $I \times L + J \times L$. Решење може да се замисли и као једна велика матрица која је састављена

од две мање матрице чије су димензије респективно $I \times L$ и $J \times L$. Цела матрица има L колона које представљају периоде. Први део велике матрице, тј. подматрица димензија $I \times L$ у свакој колони која представља један период има бар један бит са вредношћу 1 и он означава успостављеног добављача, тј. локацију са које може да се поручи роба, сви остали битови имају вредност 0 (промењљиве y_{idl} у 2.1). Елемент прве подматрице са индексима i и l је члан низа чији је индекс $i \cdot L + l$. Дисконтни ниво се одређује при рачунању вреености функције циља. Други део матрице тј. друга подматрица димензија $J \times L$ у свакој колони (која у ствари представља један период) има тачно један бит са вредношћу 1 који означава отворену продавницу у том периоду, а у свакој врсти може да има највише један бит са вредношћу 1 због ограничења 2.10. Елемент друге подматрице са индексима j и l је члан низа са индексом $I \cdot L + j \cdot L + l$. Решење генерисано на овај начин је допустиво решење.

На слици 3.1 дат је пример допустивог решења које је представљено као две подматрице од којих свака има по осам колона које представљају периоде. Свака подматрица има по 25 врста. У првој подматрици врсте представљају потенцијалне добављаче, а у другој подматрици врсте представљају потенцијалне продавнице. У првом периоду је отворена продавница на локацији 1 и добављачи на локацијама 3, 7, 10, 16, 19 и 25. У другом периоду је отворена продавница на локацији 24 и добављачи на локацијама 1, 3, 6, 12, 21 и 22. У трећем периоду је отворена продавница на локацији 7 и добављачи на локацијама 1, 5, 7, 11 и 18. У четвртом периоду је отворена продавница на локацији 23 и добављачи на локацијама 2, 8, 9, 13, 16, 19 и 20. У петом периоду је отворена продавница на локацији 18 и добављачи на локацијама 2, 11, 18, 23 и 24. У шестом периоду је отворена продавница на локацији 13 и добављачи на локацијама 1, 2, 9, 12, 13 и 19. У седмом периоду је отворена продавница на локацији 15 и добављачи на локацијама 1, 3, 7, 13, 14, 17, 19 и 24. У осмом периоду је отворена продавница на локацији 3 и добављачи на локацијама 7, 17 и 20.

	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
1	0	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0
2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
12	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
18	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0
19	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
20	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
21	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
24	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0
25	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Слика 3.1: Пример допустивог решења

3.3.2 Рачунање вредности функције циља

Како генерисано решење има тачно једну успостављену продавницу у сваком периоду сви крајњи купци купују од отворене продавнице. Сви они могу да подмире своје захтеве од те успостављене продавнице, јер не постоје ограничења капацитета. Стога је укупна количина робе купљена у отвореној продавници једнака збиру захтева крајњих корисника у посматраном периоду. Даље продавница бира најближег добављача који је отворен и од њега наручује сву робу. У зависности од количине наручене робе и граница дисконтних нивоа добављача од ког је поручена роба, одређује се дисконтни ниво имајући у виду ограничења 2.4−2.7. Тиме су израчунате прве три суме у изразу за фунцију циља 2.1. На крају се додаје цена отварања продавнице и цена одржавања продавнице што су респективно четврта и пета сума у изразу за функцију циља 2.1. Цео овај поступак се понавља за сваки период

да би се на крају добила коначна вредност функције циља за посматрано решење.

3.3.3 Дефиниција околина

Приликом решавања проблема различитим варијантама методе променљивих околина, дефинисана су три типа околина N_1, N_2 и N_3. Начин на који су дефинисане околине гарантује да су сви суседи допустивог решења x такође допустива решења.

- N_1: У првом типу околина се мења други део решења који одређује које продавнице и у ком периоду ће бити отворене, јер њихово одређивање представља најбитнији део проблема, као и због природе методе променљивих околина тј. да се претрага врши најчешће у првој околини неког решења. Нека је x решење, тада се сусед x' генерише мењајући други део решења тако што се на случајан начин бирају две продавнице и мењају се периоди у којима су оне отворене, затим се на случајан начин бирају два периода и мењају се продавнице које су у тим периодима отворене. Како је други део решења дужине $J \times L$ интерпретиран као матрица са J врста и L колона, најпре се на случајан начин изаберу две колоне у тој матрици које међусобно замењују места, а затим се на случајан начин бирају две врсте које међусобно замењују места. Сви овако генерисани суседи x' решења x чине околину првог типа, у ознаци $N_1(x)$.

- N_2: У другом типу околина се мења први део решења који одређује од којих добављача и у ком периоду може бити поручена роба. Приликом генерисања суседа x' решења x мења се први део решења тако што се на цлучајан начин бирају два добављача и мењају се периоди у којима је од њих наручена роба, а затим се бирају два периода и мењају се добављачи од којих је наручена роба. Слично као и у првом типу околине, ако се први део решења дужине $I \times L$ интерпретира као матрица са I врста и L колона, онда се најпре на случајан начин изаберу две

колоне које међусобно замењују места, а затим се на случајан начин бирају две врсте које међусобно замењују места. Сви овако генерисани суседи x' решења x чине његову околину другог типа, у ознаци $N_2(x)$.

- N_3: У трећем типу околина који чине унија прва два типа околина мењају се истовремено оба дела решења, тј. мењају се истовремено први део решења који одређује од којих добављача и у којим периодима може бити поручена роба, као и други део решења који одређује које продавнице и у ком периоду ће бити отворене. Периоди у којима се мењају продавнице исти као и периоди у којима се мењају добављачи, тј. ти периоди се бирају једном на случајан начин. То значи да се за одабране периоде истовремено мењају добављачи од којих може да се поручи роба и продавнице које су отворене у тим периодима. Сви овако генерисани суседи x' решења x чине околину трећег типа, у ознаци $N_3(x)$.

3.4 BVNS метода за решавање UDFLP-QD

Основни кораци BVNS методе за решавање UDFLP-QD дати су у алгоритму 14. Параметар k_{max} означава број околина у којима се врши претрага. Као критеријум заустављања BVNS методе је узета комбинација два критеријума заустављања. Први критеријум заустављања је максимално време извршавања (параметар T_{max}), а други критеријум заустављања је максималан број понављања најбољег решења (параметар R_{max}). У фази размрдавања и у фази помераја тј. у локалној претрази се користе дефиниције околина из 3.3.3. Начин на који су дефинисане околине и избор суседа, гарантује да су сви суседи допустивог решења и сами допустива решења. На почетку се учитавају решење проблема које је генерисано на случајан начин, као и вредности параметара, а затим се извршавају фаза размрдавања, фаза побољшања и фаза помераја једна за другом док се не испуни критеријум заустављања.

- У фази размрдавања приликом имплементирања основне методе променљивих околина, имплементација процедуре размрдавања дате у ал-

Алгоритам 14 BVNS метода за решавање UDFLP-QD

Улаз: решење $x, k_{max}, I_{max}, T_{max}, R_{max}$;
    $t = 0$; /* *Време* */
    $r = 0$; /* *Број понављања најбољег решења* */
понављај
    $k = 1$;
    понављај
/* *Фаза размрдавања* */
        изабери произвољно $x' \in N_k(x)$;
/* *Фаза побољшања (локална претрага са првим побољшањем)*/
        $it = 0$;
        понављај
            изабери произвољно $x'' \in N_k(x')$;
            ако је $f(x'') < f(x')$ **онда**
                $x' \leftarrow x''$;
            $it = it + 1$;
        док је $it < I_{max}$;
        $s \leftarrow x'$;
/* *Фаза помераја* */
        ако је $f(s) < f(x)$ **онда**
            $x \leftarrow s$;
            $k = 1$;
            $r = 1$;
        иначе
            $k = k + 1$;
            $r = r + 1$;
    док је $k \leq k_{max}$
    $t =$ CPUtime();
док је $(t < T_{max}) \wedge (r < R_{max})$
врати x;

горитму 1 има три случаја у зависности од тога у којој околини се размрдавање врши, тј. на случајан начин се бира сусед x' решења x у околини у којој се врши претрага. Сусед x' решења x који је добијен из процедуре размрдавања служи као полазно решење за локалну претрагу које је имплементирана у фази побољшања.

- Приликом имплементирања фазе побољшања код основне методе променљивих околина имплементирана је локална претрага са првим побољшањем дата у алгоритму 2. Најпре се врши иницијализација бројача итерација тј. вредност it се поставља на 0. Затим се бира произвољно решење x'' у k-тој околини решења x'. Ако је вредност функције циља за решење x'' мања од вредности функције циља за решење x' онда се даља локална претрага врши у k-тој околини тог новог решења. Ако вредност функције циља за решење x'' није мања од вредности функције циља за решење x' онда се оно одбацује и бира се нови сусед x'' решења x'. У оба случаја се вредност бројача итерација it увећава за 1. Овај поступак се понавља док се не постигне максималан број итерација за фазу побољшања (параметар I_{max}). Приликом извршавања локалне претраге тражи се прво побољшање тренутног решења у оној околини за коју је извршена фаза размрдавања.

- У фази помераја имплементирана је метода у којој се секвенцијално мења околина у којој се врши даља претрага дата у алгоритму 5. Ако је решење добијено локалном претрагом боље од тренутног најбољег решења онда оно постаје тренутно најбоље решење, даља претрага се наставља у првој околини N_1 новог тренутног решења, и почиње се са бројењем понављања најбољег тренутног решења испочетка. У супротном, тј. ако решење добијено локалном претрагом није боље од тренутног најбољег решења, даља претрага се наставља у следећој околини тренутно најбољег решења, а бројач понављања најбољег решења се увећава за 1.

Алгоритам 15 VND метода за решавање UDFLP-QD

```
1: Улаз: решење x, l_max;
2: l = 1;
3: понављај
4:     x' ← argmin_{y∈N_l(x)} f(y);
5:     ако је f(x') < f(x) онда
6:         x ← x';
7:         l = 1;
8:     иначе
9:         l = l + 1;
10: док је l ≤ l_max
11: врати x;
```

3.5 VND метода за решавање UDFLP-QD

Основни кораци VND методе за решавање UDFLP-QD дати су у алгоритму 15. Параметар l_{max} означава број околина у којима се врши претрага. У имплементацији VND методе коришћене су дефиниције типова околина описане у 3.3.3.

Најпре се учитавају почетно решење проблема x генерисано на случаја начин, и параметар l_{max}. Иницијализује се претрага која почиње од прве околине N_1 решења x, тако што се тражи оно решење x' које је најбоље тј. које има најмању вредност функције циља. У циљу проналажења најбољег решења у свакој околини су имплементиране три процедуре претраге по једна за систематску претрагу једног од три типа околина у потрази за најбољим допустивим решењем у посматраној околини тренутног решења. Ако тако пронађено решење x' има мању вредност функције циља од тренутног решења x оно постаје ново тренутно решење и претрага се наставља у првој околини тог новог тренутног решења. Ако тако пронађено решење x' има већу вредност функције циља од тренутног решења x онда се оно одбацује, а даља претрага се наставља у следећој околини тренутног решења x. Овај поступак се понавља све док постоји боље решење од тренутног решења у некој његовој околини. У тренутку када не постоји боље решење ни у једној околини тренутног решења стаје се са даљом претрагом и за најбоље решење се проглашава тренутно решење.

3.6 GVNS метода за решавање UDFLP-QD

Алгоритам 16 GVNS метода за решавање UDFLP-QD

```
1: Улаз: решење x, k_max, l_max, T_max, R_max;
2:     t = 0; /* Време */
3:     r = 0; /* Број понављања најбољег решења */
4: понављај
5:     k = 1;
6:     понављај
7: /* Фаза размрдавања */
8:         изабери произвољно x' ∈ N_k(x);
9: /* Фаза побољшања (метода променљивог спуста) */
10:        l = 1;
11:        понављај
12:            x'' ← argmin_{y∈N_l(x')} f(y);
13:            ако је f(x'') < f(x') онда
14:                x' ← x'';
15:                l = 1;
16:            иначе
17:                l = l + 1;
18:        док је l ≤ l_max
19:        s ← x';
20: /* Фаза помераја */
21:        ако је f(s) < f(x) онда
22:            x ← s;
23:            k = 1;
24:            r = 1;
25:        иначе
26:            k = k + 1;
27:            r = r + 1;
28:    док је k ≤ k_max
29:    t = CPUtime();
30: док је (t < T_max) ∧ (r < R_max)
31: врати x;
```

Основни кораци GVNS методе за решавање UDFLP-QD дати су у алгоритму 16. Параметар k_{max} означава број околина у GVNS методи, параметар l_{max} означава број околина у VND методи која је имплементирана у фази побољшања. Приликом имплементирања VND методе дефиниција околина

је иста као и дефиниција околина које се користе у GVNS методи описана у 3.3.3. Начин на који су дефинисане околине гарантује да су суседи допустивог решења допустива решења. Критеријум заустављања GVNS методе је комбинација два критеријума заустављања. Први критеријум заустављања је максимално време извршавања (параметар T_{max}). Други критеријум заустављања је максималан број понављања решења (параметар R_{max}). На почетку се учитавају решење проблема x, које је генерисано на случајан начин, као и вредности параметара, а затим се извршавају фаза размрдавања, фаза побољшања и фаза помераја једна за другом док се не испуни критеријум заустављања.

- У фази размрдавања, као и код основне методе, имплеменирана је процедура размрдавања описана у алгоритму 1, којом се врши размрдавање почетног решења у зависности у којој околини се тражи његов сусед, тј. на случајан начин се бира сусед x' решења x у околини у којој се врши претрага. Сусед x' решења x који је добијен из процедуре размрдавања служи као полазно решење за методу променљивог спуста које је имплементирана у фази побољшања.

- У фази побољшања, је имплементирана метода променљивог спуста дата у алгоритму 11. Поступак претраге је идентичан поступку претраге описаном у 3.5.

- У фази помераја, је као и код основне методе промењивих околина имплементирана секвенцијална промена околине дата у алгоритму 5. Поступак је исти као и код BVNS методе, стим што се овде са тренутним набољим решењем упоређује решење добијено VND методом у фази побољшања.

Глава 4

Експериментални резултати

У овој глави представљени су резултати предложених метахеуристичких метода на тест инстанцама из литературе. Добијени резултати метахеуристика су међусобно упоређени, а поређење је извршено и са оптималним или најбољим решењима егзактног решавача CPLEX. У циљу бољег сагледавања перформанси предложених метахеуристика, извршена је и статистичка анализа добијених резултата.

4.1 Тест инстанце

У раду су коришћене тест инстанце малих, средњих и великих димензија, по 10 инстанци из сваке групе. Тест инстанце малих димензија имају $M = 100$ крајњих купаца, $J = 25$ продавница и $I = 25$ добављача. Тест инстанце средњих димензија имају $M = 300$ крајњих купаца, $J = 25$ продавница и $I = 20$ добављача. Тест инстанце великих димензија имају $M = 1000$ крајњих купаца, $J = 100$ продавница и $I = 20$ добављача.

Инстанце малих и великих димензија су добијене од стране аутора рада [18]. Код ових инстанци су изостављени капацитети. Инстанце средњих димензија су генерисане према упутству из рада [18]. Тест инстанце средњих димензија су генерисане на следећи начин. Све координате добављача, продавница и крајњих купаца су генерисане униформно у квадрату 500×500. За рачунање растојања између добављача и продавница, као и продавница

и крајњих купаца је коришћена Еуклидска метрика изражена формулом 4.1

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

где су $(x_1, y_1), (x_2, y_2)$ координате тачака.

У пракси се запажа да цена набављања робе има велики удео у укупној цени [4]. У [21] и [31] се запажа да цена набављања робе није мања од $50\% - 60\%$ вредности укупне цене робе. Приликом генерисања инстанци вођено је рачуна о овој чињеници избором граница за параметре и међусобним везама вредности параметара.

У свим инстанцама се претпоставља да постоји $L = 8$ периода у којима се мењају захтеви крајњих корисника. Захтеви крајњих корисника $\lambda_{m1}, m = 1, \ldots, M$ у првом периоду су генерисани униформно између 200 и 600. За сваки од осталих периода захтеви $\lambda_{ml}, m = 1, \ldots, M, l = 2, \ldots, L$ крајњих корисника су генерисани униформно између 80% и 160% вредности у претходном периоду водећи се Басовим дифузионим моделом [5].

За све добављаче се претпоставља да робу нуде у $D = 4$ дисконтна нивоа. Почетна јединична цена $p_{i1}, i = 1, \ldots, I$ по којој сваки добављач нуди робу је генерисана униформно између 190 и 210, док су цене на осталим нивоима $p_{id}, i = 1, \ldots, I, d = 2, \ldots, D$ генерисане униформно између 90% и 95% вредности у односу на претходни дисконтни ниво. Граница првог дисконтног нивоа $a_{i1}, i = 1, \ldots, I$ је генерисана униформно између 6% и 12% од укупног захтева свих крајњих корисника у последњем периоду. Границе за остале дисконтне нивое $a_{id}, i = 1, \ldots, I, d = 2, \ldots, D$ су генерисане униформно између 140% и 160% вредности у претходном дисконтном нивоу. Цене отварања продавница $F_j, j = 1, \ldots, J$ су у корелацији са укупним потраживањима крајњих крисника у свим периодима. Цене одржавања продавница $o_j, j = 1, \ldots, J$ по периоду су генерисане униформно у интервалу $[\frac{1}{6}, \frac{1}{3}]$ од цене отварања продавнице.

4.2 Експериментални резултати и поређења

Све тест инстанце су тестиране на рачунару са Intel Celeron Dual-Core 2840 процесором са радним тактом од 2.16 GHz и 4 GB RAM меморије.

Решења свих тест инстанци добијена имплементираним метахеуристичким методама су упоређена међусобно, а решења инстанци малих димензија су упоређена и са оптималним решењима добијеним помоћу CPLEX-а, верзија 20.1. Време извршавања CPLEX-а је за тест инстанце средњих димензија ограничено на 1h, а за за тест инстанце великих димензија на 2h. Све метахеуристике као и CPLEX су имплементирани у програмском језику C.

Вредности параметара који су коришћени у имплементираним метахеуристикама су дати у табели 4.1. Вредности параметара T_{max} су дате у секундама.

Табела 4.1: Вредности параметара

Параметри	VND	BVNS	GVNS
k_{max}	–	3	3
l_{max}	3	–	3
T_{max}	–	3600	7200
R_{max}	–	50	5
I_{max}	–	500	–

Све три имплементиране варијанте методе променљивих околина су извршаване по 10 пута на инстанцама малих и средњих димензија, а по 5 пута на инстанцама великих димензија.

Резултати тестирања су представљени у табелама 4.2-4.4. Инстанце малих димензија CPLEX може да реши до оптималности, па се у табели 4.2 у којој су представљени резултати за инстанце малих димензија друга, трећа и четврта колона се односе на резултате CPLEX-а. У другој колони је представљена вредност функције циља opt_{sol} оптималног решења. У трећој колони је представљено време у секундама t за које је CPLEX успешно решио те инстанце. У четвртој колони је представљен број чворова претраживачког дрвета које CPLEX формира да би решио проблем, у ознаци *nodes*.

За инстанце средњих димензија CPLEX је после $1h$ рада успео да пронађе само допустива решења, па се у табели 4.3 у којој су представљени резултати над инстанцама средњих димензија друга, трећа и четврта колона се односе на CPLEX. У другој колони је представљена минимална вредност

функције циља у свим преосталим отвореним чворовима, у ознаци $best_{sol}$. У трећој колони је представљен број чворова претраживачког дрвета које CPLEX формира да би решио проблем, у ознаци *nodes*. У четвртој колони је представљено одступање *gap* између доње и горње границе изражено у процентима. Доња граница представља минималну вредност функције циља у свим преосталим отвореним чворовима дрвета претраживања. Горња граница представља најбоље допустиво решење до кога је дошао CPLEX решавајући проблем док није достигао временско ограничење.

Како CPLEX за инстанце великих димензија после $2h$ рада није могао да да ни допустива решења у табели 4.4 су изостављене колоне које се односе на CPLEX.

У другој колони табеле 4.4 представљена су најбоља решења до којих су дошле метахеуристике.

У свим табелама преостале колоне се односе на метахеуристике, редом на основну методу променљивих околина (BVNS), затим методу променљивог спуста (VND), и на крају последње четири колоне се односе на општу методу променљивих околина (GVNS).

Група колона која се односи на једну од наведених метахеуристика садржи следеће податке:

i) Просечно време (аритметичка средина) за које је пронађено најбоље решење t_{best} изражено у секундама.

ii) Просечно укупно време (аритметичка средина) t_{tot} извршавања метахеуристике изражено у секундама.

iii) Просечно одступање од најбољег познатог решења *agap* изражено у процентима.

iv) Стандардна девијација одступања σ изражена у процентима.

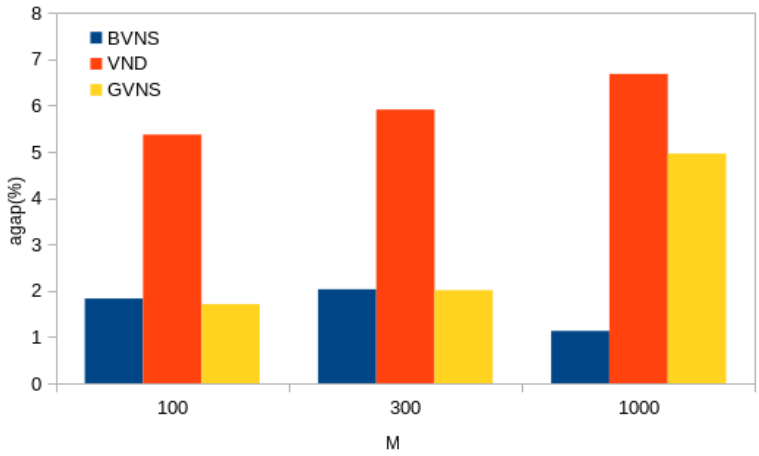
Вредности *agap* и σ су рачунате по формулама $agap = \frac{\sum_{i=1}^{N} gap_i}{N}$ и $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(gap_i - agap)^2}$ где је N број извршавања метахеуристике над неком инстанцом.

Нека је са $best_{sol}$ означено оптимално или најбоље познато решење за неку

инстанцу, и нека је sol_i решење које добијено у i-том извршавању метахеуристике над неком инстанцом, тада су вредности gap_i добијене по формули $gap_i = 100 \cdot \frac{|sol_i - best_{sol}|}{|best_{sol}|} \quad i = 1, \ldots N$, и оне представљају одступање у процентима решења sol_i од најбољег познатог решења $best_{sol}$.

Приликом тестирања метахеуристика за вредности параметара које су дате у табели 4.1 постигнути су резултати приказани у табелама 4.2, 4.3 и 4.4.

У табели 4.2 приказани су резултати добијени на инстанцама малих димензија које имају 100 крајњих корисника, 25 потенцијалних продавница и 25 добављача. Све инстанце је CPLEX успео да реши до оптималности, па је могуће поређење тако добијених решења са решењима добијеним помоћу метахеуристика, где се види да метахеуристике дају довољно квалитетна решења за много мање времена.



Слика 4.1: Просечно одступање (agap) у зависности од броја крајњих корисника (M)

Инстанце мањих димензија је CPLEX успео да реши до оптималности иако му за то треба знатно више времена од имплементираних метахеуристика. Са друге стране метехеуристике проналазе више него задовољавајућа решења са просечном вредношћу gap-а 1.83% за BVNS, 5.37% за VND, и 1.71% за GVNS за мале инстанце. На слици 4.1 је представљена промена просечних

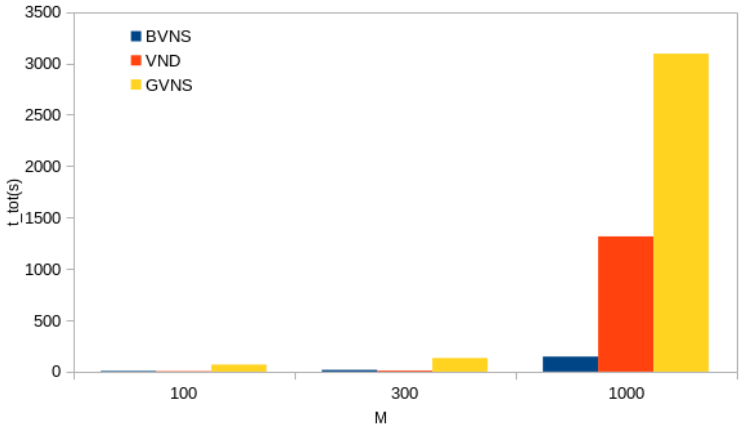
вредности gap-а у зависности од броја крајњих корисника. Из просечних вредности gap-а се види да најбоља решења за мале инстанце проналази GVNS, одмах затим следи BVNS, на крају VND даје најгора решења. Просечно укупно време извршавања за мале инстанце је редом 6.69 секунди за BVNS, 4.72 секунди за VND, и 68.92 секунди за GVNS. Најуспешнија метахеуристика над инстанцама малих димензија по брзини је VND. Просечно време првог проналаска најбољег решења за мале инстанце је редом 1.30 секунди за BVNS, 3.38 секунди за VND, и 29.00 секунди за GVNS. Просечне вредности стандардне девијације одступања су 0.46% за BVNS, 1.80% за VND, 0.82% за GVNS. Из ових вредности се види да је BVNS најстабилнија.

Табела 4.2: Резултати на инстанцама малих димензија ($M = 100, J = 25, I = 25$)

	CPLEX			BVNS				VND				GVNS			
Инстанца	opt_{sol}	$t(s)$	$nodes$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$
Data1	335889861.300	2341.81	4441	1.42	6.77	1.59	0.37	3.62	4.95	4.52	1.93	42.27	81.14	1.50	0.93
Data2	307172207.284	1470.42	1734	3.63	9.13	2.70	0.72	3.06	4.42	7.80	2.42	30.96	76.46	3.08	1.10
Data3	301628600.778	965.24	1131	1.75	7.07	1.68	0.42	4.18	5.50	3.46	1.48	34.63	76.60	1.19	0.49
Data4	333712956.733	1169.81	2481	1.45	6.76	1.20	0.26	4.86	6.25	4.16	1.42	25.78	62.74	1.66	0.68
Data5	325071751.788	3600.07	5302	0.64	6.14	1.75	0.48	0.60	1.94	5.04	1.67	44.92	88.41	0.79	0.59
Data6	322958202.168	1278.86	2420	0.39	5.68	1.55	0.45	3.62	4.96	5.94	2.10	23.86	73.10	1.91	1.25
Data7	337195756.271	857.31	1774	0.74	6.04	1.61	0.49	3.53	4.82	5.65	1.35	19.98	54.93	1.64	1.02
Data8	337200882.131	1697.45	1920	0.61	6.13	1.91	0.44	4.22	5.57	6.84	2.34	8.18	41.50	2.30	0.86
Data9	321462546.301	908.03	2655	0.88	6.21	2.08	0.52	3.05	4.40	3.90	1.52	24.04	69.52	1.47	0.66
Data10	319149605.134	1613.48	1866	1.49	6.97	2.24	0.44	3.02	4.37	6.38	1.76	35.42	64.79	1.56	0.58

У табели 4.3 дати су резултати добијени на инстанцама средњих димензија које имају 300 крајњих купаца, 25 потенцијалних продавница и 20 добављача. За ове инстанце је CPLEX успео да нађе само допустива решења после 1h рада, па су у овој табели вредности gap упоређене са вредностима gap добијеним помоћу метахеуристика. Овде се такође види да метахеуристике дају задовољавајућа решења у разумном времену.

Просечне вредности одступања за инстанце средњих димензија су 2.03% за BVNS, 5.91% за VND, 2.01% за GVNS, у овом случају су ове вредности нешто веће јер је као решење у односу на које је рачуната ова вредност узимана доња граница добијена помоћу CPLEX-a. Просечна укупна времена извршавања за инстанце средњих димензија су редом 16.63 секунди за BVNS, 10.17 секунди за VND, и 132.18 секунди за GVNS. Просечна времена првог проналаска најбољег решења за инстанце средњих димензија су редом 4.17 секунди за BVNS, 7.62 секунди за VND, и 51.98 секунди за GVNS. Просечне вредности стандардне девијације одступања су 0.44% за BVNS, 1.76% за VND, 1.02% за GVNS. Из ових вредности се види да је BVNS најстабилнија. Промена просечних укупних времена извршавања метахеуристика у зависности од броја крајњих корисника је дата на слици 4.2.



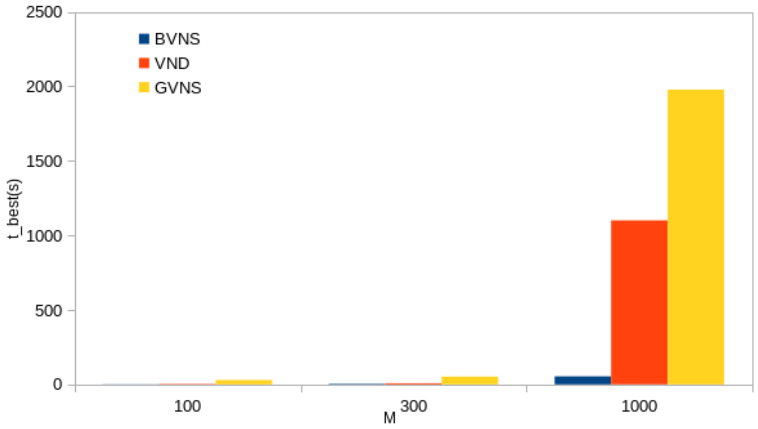
Слика 4.2: Просечно укупно време извршавања (t_{tot}) у зависности од броја крајњих корисника (M)

Табела 4.3: Резултати на инстанцама средњих димензија ($M = 300, J = 25, I = 20$)

	CPLEX			BVNS				VND				GVNS			
Инстанца	$best_{sol}$	$nodes$	$gap(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$
Data11	1143055510.671	289	0.49	3.56	16.29	2.12	0.42	7.78	10.30	5.18	2.06	54.29	120.51	2.13	1.16
Data12	1174475625.968	1404	8.54	6.08	18.28	1.89	0.37	9.04	11.58	6.63	1.74	33.55	117.88	2.39	0.92
Data13	1156674197.216	583	10.60	1.69	14.04	2.38	0.39	6.54	9.20	7.85	1.98	37.31	108.61	2.18	1.28
Data14	1134994702.570	491	15.62	1.50	13.89	2.27	0.46	7.68	10.22	8.06	1.47	98.21	179.09	2.06	0.71
Data15	1138451609.983	444	5.94	0.64	6.14	1.75	0.48	0.60	1.94	5.04	1.67	44.92	88.41	0.79	0.59
Data16	1200560919.527	113	1.81	1.34	14.28	2.65	0.40	6.29	8.79	5.02	1.66	79.92	165.35	2.55	1.03
Data17	1102607735.075	386	0.19	9.57	21.81	1.91	0.51	9.83	12.36	5.90	1.69	34.45	108.15	2.15	1.24
Data18	1062757522.013	971	9.03	3.52	16.03	2.20	0.60	7.64	10.18	7.08	1.61	36.23	130.94	2.01	1.10
Data19	1154792837.449	339	0.11	1.68	14.27	1.76	0.46	4.17	6.72	4.09	1.85	78.49	162.93	1.59	0.98
Data20	1127091282.611	0	16.08	8.89	21.29	1.47	0.26	8.65	11.23	5.47	2.18	41.69	122.01	1.64	1.00

У табели 4.4 су приказани резултати добијени на инстанцама великих димензија које имају 1000 крајњих купаца, 100 потенцијалних продавница и 20 добављача. За ове инстанце CPLEX после $2h$ рада није успео да нађе чак ни допустива решења, тако да су колоне које се односе на CPLEX у овој табели изостављене. Уместо њих је колона у којој су представљена најбоља решења до којих су дошле метахеуристике.

Просечна вредност gap-а за инстанце великих димензија је редом 1.13% за BVNS, 6.68% за VND, и 4.06% за GVNS. Просечна укупна времена извршавања за инстанце великих димензија су 146.53 секунди за BVNS, 1315.90 секунди за VND, и 3094.00 секунди за GVNS. Просечна времена првог проналаска најбољег решења за инстанце великих димензија су 53.79 секунди за BVNS, 1100.66 секунди за VND, и 1978.37 секунди за GVNS. Просечне вредности стандардне девијације одступања су 0.69% за BVNS, 2.33% за VND, 2.56% за GVNS. Из ових вредности се види да је BVNS најстабилнија. Промена просечног времена првог проналаска најбољег решења у зависности од броја крајњих купаца је дата на слици 4.3.



Слика 4.3: Просечно време првог проналажења најбољег решења (t_{best}) у зависности од броја крајњих корисника (M)

Табела 4.4: Резултати на инстанцама великих димензија ($M = 1000, J = 100, I = 20$)

		BVNS				VND				GVNS			
Инстанца	$best_{sol}$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$	$t_{best}(s)$	$t_{tot}(s)$	$agap(\%)$	$\sigma(\%)$
Data21	2361480777.665	42.97	136.04	2.34	0.59	1196.88	1410.62	5.15	2.85	1765.54	3161.18	1.88	1.78
Data22	2296711442.542	56.05	149.06	1.18	0.96	1039.65	1255.41	6.95	0.79	1795.43	3289.78	5.79	2.74
Data23	2296711442.542	74.40	167.46	1.16	1.08	689.89	905.77	10.49	2.40	2064.39	2453.17	8.56	4.60
Data24	2389242966.407	26.50	118.84	0.69	0.41	689.66	905.54	4.81	1.32	2461.52	3162.19	2.18	1.46
Data25	2293029541.748	71.11	163.66	0.60	0.43	1193.60	1407.85	9.67	2.47	1750.62	3058.01	3.20	2.01
Data26	2347191409.896	65.15	157.59	0.88	0.62	1212.24	1426.11	6.55	4.21	1810.32	2800.80	4.77	2.81
Data27	2316116857.633	40.02	132.28	1.87	1.08	1203.24	1419.57	3.67	2.06	1592.72	3267.77	3.00	1.79
Data28	2361803341.653	61.36	153.43	0.76	0.51	1543.70	1758.89	5.54	3.38	2106.66	3512.03	3.29	1.58
Data29	2310016084.832	41.37	135.37	1.06	0.79	1207.01	1423.73	6.63	2.30	1938.88	2930.29	4.19	3.17
Data30	2319476197.919	58.98	151.55	0.76	0.42	1030.72	1245.53	7.35	1.55	2497.65	3304.80	3.70	3.66

4.3 Статистичка анализа резултата

Рачунање само просечног одступања од најбољег решења и стандардне девијације одступања од најбољег решења понекад није довољно за поређење имплементираних метахеуристика. У том случају се тестира статистичка значајност разлика у перформансама између имплементираних метахеуристика.

По Демшару, аутору рада [12] најчешћи статистички непараметарски тест за тестирање перформанси између више од две метахеуристике је Фридманов[1] тест (енгл. Friedman test), и њему одговарајући Немењи[2] пост хок тест (енгл. Nemenyi post hoc test).

Хипотезе које се тестирају су:
H_0: имплементиране метахеуристике имају једнаке средње перформансе и уочене разлике су углавном случајне,
и алтернативна хипотеза:
H_1: средње перформансе бар једне имплементиране метахеуристике су статистички значајно различите од осталих.

Приликом тестирања ових хипотеза препорука је да се за степен значајности узима $\alpha = 0.05$. Да би се извршили Фридманов и Немењијев тест најпре се врши рангирање имплементираних метахеуристика на свакој инстанци посебно тако што се ранг 1 придружује оној метахеуристици која је постигла најбоље решење над том инстанцом, ранг 2 другој најбољој и на крају ранг 3 трећој најбољој, што се може видети у табели 4.5. Средња вредности ранга за BVNS је 1.70. Средња вредност ранга за VND је 2.90. Средња вредност ранга за GVNS је 1.40. Из овога се види да GVNS има најбоље перформансе, одмах затим следи BVNS, док на крају VND има најлошије перформансе.

Нека је са $U = 30$ означен укупан број узорака, у овом случају број инстанци, са $B = 3$ број посматрања, у овом случају број метахеуристика. Нека је са r_u^b означен ранг b-те метахеуристике од укупно B метахеуристика на

[1]Milton Friedman 1912 – 2006 - амерички економиста и статистичар
[2]Peter Björn Nemenyi 1927 – 2002 - амерички математичар

Табела 4.5: Поређење ранга на свим инстанцама

Вредности			Ранг		
BVNS	VND	GVNS	BVNS	VND	GVNS
339734570.937	342637951.215	336411498.069	2	3	1
310929258.933	318379022.368	308043137.868	2	3	1
304628254.879	306201512.266	303318407.245	2	3	1
336205987.632	339003484.368	336619100.910	1	3	2
329032087.261	333862960.785	325071597.137	2	3	1
325709569.046	326571977.583	325363207.039	2	3	1
340587920.807	352252142.719	339298583.479	2	3	1
342109173.234	347848219.016	339524676.621	2	3	1
325886396.647	327596567.303	322895570.646	2	3	1
323338930.557	330610506.590	322160493.640	2	3	1
1159453222.579	1166069394.905	1152864204.342	2	3	1
1189687336.492	1219447714.793	1188082870.824	2	3	1
1178067139.264	1223229729.494	1166913797.453	2	3	1
1153485194.458	1198760876.743	1139225463.136	2	3	1
1150415178.052	1152890104.808	1141134293.224	2	3	1
1224847698.854	1229765514.423	1217382265.533	2	3	1
1115992539.069	1133865442.322	1106026196.719	2	3	1
1073066155.124	1113823582.838	1067840469.186	2	3	1
1166286450.545	1180244585.214	1157305476.104	2	3	1
1138153532.390	1155982470.461	1133154163.187	2	3	1
2397469251.814	2402819870.453	2361480777.665	2	3	1
2296941911.004	2437614621.336	2380539961.206	1	3	2
2296711442.542	2477835117.154	2382248359.905	1	3	2
2389242966.407	2460738367.574	2390748799.725	1	3	2
2293029541.748	2416472820.155	2318125385.312	1	3	2
2347191409.896	2351039723.544	2376246695.129	1	2	3
2316116857.633	2335173801.283	2354144439.237	1	2	3
2361803341.653	2388387205.864	2400854650.744	1	2	3
2310016084.832	2401882177.500	2354685769.928	1	3	2
2321797650.487	2425731338.287	2319476197.919	2	3	1
		Укупни ранг	51	87	42
		Средњи ранг	1.70	2.90	1.40

u-тој од укупно U инстанци. Фридманов тест пореди средње рангове метахеуристика, $R_b = \frac{1}{U}\sum_u r_u^b$.
Користећи формулу [12]

$$\chi_F^2 = \frac{12U}{B(B+1)}\left[\sum_b R_b^2 - \frac{B(B+1)^2}{4}\right] \tag{4.2}$$

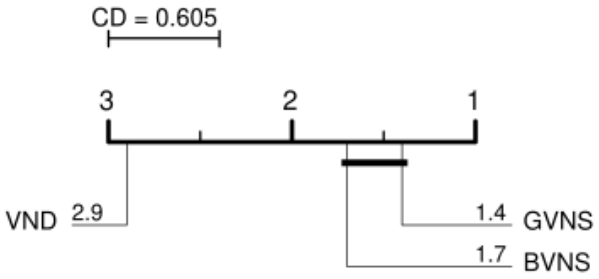
добија се да је $\chi_F^2 = 37.8$ за разматрани проблем, што је веће од критичне вредности[33] 6.2 са $df = B - 1 = 2$ степена слободе, па се одбацује нулта хипотеза H_0. Дакле, по Фридмановом тесту постоје статистички значајне разлике у перформансама међу метахеуристикама.

Фридманов тест даје само информацију да постоје статистички значајне разлике између имплементираних метахеуристика, али не и између којих, да би се то открило спроводи се Немењијев тест. Немењијев тест тестира парове имплементираних метахеуристика да би се открило која од имплементираних метахеуристика има перформансе које се статистички значајно разликују од перформанси осталих имплементираних метахеуристика. Критична вредност је [12] $q_\alpha = 2.343$, док се критична разлика рачуна по формули 4.3 и за разматрани проблем је $CD = 0.605$.

$$CD = q_\alpha \cdot \sqrt{\frac{B(B+1)}{6U}} \tag{4.3}$$

Уколико је разлика између вредности средњих рангова имплементираних метахеуристика већа од критичне вредности CD онда су разлике у њиховим перформансама статистички значајне. У разматраном проблему, а што се може видети и на дијаграму 4.4, разлика између средњих рангова BVNS и GVNS је мања од критичне вредности па је разлика у њиховим перформансама статистички безначајна. Разлика између средњих рангова VND са једне стране и BVNS и GVNS са друге стране је већа од критичне вредности CD, па се из овога закључује да се перформансе VND статистички значајно разликују од перформанси BVNS и GVNS.

Када се анализира квалитет добијених решења помоћу CPLEX-а за мале инстанце, за које је пронашао оптимална решења са оним добијеним помоћу



Слика 4.4: Дијаграм Немењи теста

имплементираних метахеуристика види се да метахеуристике за мале инстанце дају скоро оптимална решења за много краће време. Са друге стране дају и довољно квалитетна решења за средње и велике инстанце за које CPLEX даје само допустива решења или није у могућности да добије чак ни допустиво решење. Из овога се закључује да је нарочито за средње и велике инстанце погодно развијање метахеуристичких метода за њихово решавање јер дају довољно добра решења у разумном времену извршавања.

Глава 5

Закључак

У овом раду је проучаван проблем динамичке локације објеката са избором добављача који нуде попуст у зависности од количине наручене робе. Циљ је да се пронађу најбоље од понуђених локација за отварање продавница, као и одабир добављача да би се испунили захтеви крајњих корисника, тако да укупна цена буде минимална. Претпоставља се да добављачи нуде попуст у зависности од количине наручене робе, као и да се захтеви крајњих купаца мењају кроз периоде. Посматрајући све периоде, одлучује се када и где ће бити отворене продавнице у складу са промењеним захтевима крајњих корисника, као и од којих добављача ће бити наручена роба. Иако описани проблем приказује реална стања у одређеним ситуацијама, због великог броја ограничења веома је изазован за решавање.

Приликом доношења одлуке у обзир треба узети неколико чиниоца, промена захтева крајњих корисника у различитим периодима, локације добављача, потенцијалне локације продавница, трошкове превоза од добављача до продавница, трошкове превоза од продавница до крајњих купаца, трошкове отварања и одржавања продавница, да би се донеле најбоље могуће одлуке. Проблем је формулисан као проблем мешовитог целобројног програмирања. Посматрани проблем је NP-тежак као генерализација класичног локацијског проблема са неограниченим капацитетима.

Имајући у виду сложеност проблема, коришћење метахеуристичких метода за његово решавање представља природан избор. У овом раду је импле-

ментирана метода променљивих околина у три своје варијанте, као основна метода променљивих околина, метода променљивог спуста и општа метода променљивих околина.

На основу добијених резултата, поређене су перформансе метахеуристика међусобно, као и са IBM ILOG CPLEX егзактним решавачем. Да би се боље оцениле перформансе имплементираних метахеуристика извршена је и статистичка анализа добијених резултата. Анализа перформанси имплементираних метахеуристичких метода је показала да је GVNS успешнија од преостале две имплементиране методе у погледу квалитета добијених решења иако јој треба више времена да до тих решења дође. Анализом резултата дошло се до закључка да BVNS методи треба најмање времена за добијање решења.

Будући рад би могао да се концентрише на избор и имплементацију метахеуристика за решавање оригиналног проблема из рада [18] са ограниченим капацитетима.

Литература

[1] Aikens, C. (1985). Facility location models for distribution planning. European Journal of Operational Research, 22(3), 263-279. doi:10.1016/0377-2217(85)90246-2

[2] Alumur, S. A., Kara, B. Y., & Melo, M. T. (2015). Location and logistics. Location Science, 419-441. doi:10.13140/RG.2.1.3284.5285

[3] Antunes, A., & Peeters, D. (2001). On solving complex multi-period location models using simulated annealing. European Journal of Operational Research, 130(1), 190–201. https://doi.org/10.1016/s0377-2217(00)00051-5

[4] Ayhan, M. B., & Kilic, H. S. (2015). A two stage approach for supplier selection problem in multi-item/multi-supplier environment with quantity discounts. Computers & Industrial Engineering, 85, 1-12. doi:10.1016/j.cie.2015.02.026

[5] Bass, F. M. (1969). A new product growth for model consumer durables. Management Science, 15(5), 215-227. doi:10.1287/mnsc.15.5.215

[6] Benton, W. C. (1991). Quantity discount decisions under conditions of multiple items, multiple suppliers and resource limitations. International Journal of Production Research, 29(10), 1953-1961. doi:10.1080/00207549108948060

[7] Brimberg, J., Mladenović, N., & Urošević, D. (2015). Solving the maximally diverse grouping problem by skewed General Variable Neighborhood Search. Information Sciences, 295, 650-675. doi:10.1016/j.ins.2014.10.043

[8] Canel, C., Khumawala, B. M., Law, J., & Loh, A. (2001). An algorithm for the capacitated, multi-commodity multi-period facility location pro-

blem. Computers & Operations Research, 28(5), 411-427. doi:10.1016/s0305-0548(99)00126-4

[9] Correia, I., & Melo, T. (2017). A multi-period facility location problem with modular capacity adjustments and flexible demand fulfillment. Computers & Industrial Engineering, 110, 307-321. doi:10.1016/j.cie.2017.06.003

[10] Cvetković, D., Čangalović, M., Dugošija, Đ, Kovačević-Vujčić, V., Simić, S., Vuleta, J., & Cvetković, D. (1996). Kombinatorna Optimizacija. Beograd, Jugoslavija: Društvo operacionih istraživača Jugoslavije.

[11] Daskin, M. S. (1995). Network and discrete location: Models, algorithms, and applications. New York: Wiley.

[12] Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research. 7 1-30. https://doi.org/10.5555/1248547.1248548

[13] Dréo, J., Pétrowski, A., & Siarry, P. (2006). Metaheuristics for hard optimization: Methods and case studies. Berlin: Springer.

[14] Drezner, Z., & Hamacher, H. (1995). Facility location: Applications and theory. Berlin: Springer.

[15] Dupont, L. (2008). Branch and bound algorithm for a facility location problem with concave site dependent costs. International Journal of Production Economics, 112(1), 245-254. doi:10.1016/j.ijpe.2007.04.001

[16] Ebrahim, R. M., Razmi, J., & Haleh, H. (2009). Scatter search algorithm for supplier selection and order lot sizing under multiple Price Discount Environment. Advances in Engineering Software, 40(9), 766-776. doi:10.1016/j.advengsoft.2009.02.003

[17] Eiselt, H. A., & Marianov, V. (2011). Foundations of location analysis. New York: Springer.

[18] Emirhüseyinoğlu, G., & Ekici, A. (2019). Dynamic facility location with supplier selection under Quantity Discount. Computers & Industrial Engineering, 134, 64-74. doi:10.1016/j.cie.2019.05.023

[19] Farahani, R. Z., & Hekmatfar, M. (2009). Facility location concepts, Models, algorithms and case studies. Heidelberg: Physica-Verlag HD.

[20] Gendron, B., Khuong, P.-V., & Semet, F. (2015). Multilayer Variable Neighborhood Search for two-level uncapacitated facility location problems with single assignment. Networks, 66(3), 214–234. https://doi.org/10.1002/net.21626

[21] Ghodsypour, S., & O'Brien, C. (1998). A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming. International Journal of Production Economics, 56-57, 199-212. doi:10.1016/s0925-5273(97)00009-1

[22] Hansen, P., Mladenović, N., & Moreno Pérez, J. A. (2009). Variable Neighbourhood Search: Methods and applications. Annals of Operations Research, 175(1), 367-407. doi:10.1007/s10479-009-0657-6

[23] Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: Basics and variants. EURO Journal on Computational Optimization, 5(3), 423-454. doi:10.1007/s13675-016-0075-x

[24] Kaufman, L., Eede, M. V., & Hansen, P. (1977). A plant and warehouse location problem. Journal of the Operational Research Society, 28(3), 547-554. doi:10.1057/jors.1977.104

[25] Kelly, D. L., & Khumawala, B. M. (1982). Capacitated warehouse location with concave costs. Journal of the Operational Research Society, 33(9), 817-826. doi:10.1057/jors.1982.177

[26] Kchaou Boujelben, M., Gicquel, C., & Minoux, M. (2016). A MILP model and heuristic approach for facility location under multiple operational constraints. Computers & Industrial Engineering, 98, 446-461. doi:10.1016/j.cie.2016.06.022

[27] Kirkpatrick, S., & Toulouse, G. (1985). Configuration space analysis of travelling salesman problems. Journal De Physique, 46(8), 1277-1292. doi:10.1051/jphys:019850046080127700

[28] Ko, H. J., & Evans, G. W. (2007). A genetic algorithm-based heuristic for the dynamic integrated forward/Reverse Logistics Network for 3PLs. Computers & Operations Research, 34(2), 346–366. https://doi.org/10.1016/j.cor.2005.03.004

[29] Klose, A., & Drexl, A. (2005). Facility location models for distribution system design. European Journal of Operational Research, 162(1), 4-29. doi:10.1016/j.ejor.2003.10.031

[30] Korać, V. M., Kratica, J., & Savić, A. (2013). An improved genetic algorithm for the multi level uncapacitated facility location problem. International Journal of Computers Communications & Control, 8(6), 845. https://doi.org/10.15837/ijccc.2013.6.134

[31] Krajewski, J. L., & Ritzman, P. L. (2002). Operations management strategy and analysis. Upper Saddle River, NJ: Prentice-Hall.

[32] Landete, M., & Marín, A. (2008). New facets for the two-stage uncapacitated facility location polytope. Computational Optimization and Applications, 44(3), 487–519. https://doi.org/10.1007/s10589-008-9165-x

[33] Lindley, D. V., & Scott, W. F. (2000). New Cambridge statistical tables. Cambridge: Cambridge University Press.

[34] Marić, M. (2010). An Efficient Genetic Algorithm for Solving the Multi-Level Uncapacitated Facility Location Problem. Computing & Informatics, 29, 183-201.

[35] Marić, M., Stanimirović, Z., Djenić, A., & Stanojević, P. (2014). Memetic algorithm for solving the multilevel uncapacitated facility location problem. Informatica, 25(3), 439–466. https://doi.org/10.15388/informatica.2014.23

[36] Melkote, S., & Daskin, M. S. (2001). Capacitated facility location/network design problems. European Journal of Operational Research, 129(3), 481-495. doi:10.1016/s0377-2217(99)00464-6

[37] Mišković, S., & Stanimirović, Z. (2013). A memetic algorithm for solving two variants of the two-stage uncapacitated facility location problem. Information Technology And Control, 42(2). https://doi.org/10.5755/j01.itc.42.2.1768

[38] Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & Operations Research, 24(11), 1097-1100. doi:10.1016/s0305-0548(97)00031-2

[39] Munson, C. L., & Rosenblatt, M. J. (2001). Coordinating a three-level supply chain with quantity discounts. IIE Transactions, 33(5), 371-384. doi:10.1080/07408170108936836

[40] Neapolitan, R. E. (2015). Foundations of algorithms. Jones & Bartlett Learning.

[41] Rajagopalan, H. K., Saydam, C., & Xiao, J. (2008). A multiperiod set covering location model for dynamic redeployment of ambulances. Computers & Operations Research, 35(3), 814–826. https://doi.org/10.1016/j.cor.2006.04.003

[42] Ro, H., & Tcha, D. (1984). A branch and bound algorithm for the two-level uncapacitated facility location problem with some side constraints. European Journal of Operational Research, 18(3), 349-358. doi:10.1016/0377-2217(84)90156-5

[43] Sadjady, H., & Davoudpour, H. (2012). Two-echelon, multi-commodity supply chain network design with Mode Selection, lead-times and inventory costs. Computers & Operations Research, 39(7), 1345-1354. doi:10.1016/j.cor.2011.08.003

[44] Shu, J., Wu, T., & Zhang, K. (2014). Warehouse location and two-echelon inventory management with concave operating cost. International Journal of Production Research, 53(9), 2718-2729. doi:10.1080/00207543.2014.977456

[45] Silva, A., Aloise, D., Coelho, L. C., & Rocha, C. (2021). Heuristics for the dynamic facility location problem with modular capacities. European Journal of Operational Research, 290(2), 435–452. https://doi.org/10.1016/j.ejor.2020.08.018

[46] Soland, R. M. (1974). Optimal facility location with concave costs. Operations Research, 22(2), 373-382. doi:10.1287/opre.22.2.373

[47] Stadtler, H. (2006). A general quantity discount and supplier selection mixed integer programming model. OR Spectrum, 29(4), 723-744. doi:10.1007/s00291-006-0066-z

[48] Talbi, E. (2009). Metaheuristics: From design to implementation. Oxford: Wiley.

[49] Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., & Gendron, B. (2014). A general variable neighborhood search variants for the travelling salesman problem with draft limits. Optimization Letters, 11(6), 1047-1056. doi:10.1007/s11590-014-0788-9

[50] Todosijević, R., Mladenović, M., Hanafi, S., Mladenović, N., & Crévits, I. (2016). Adaptive general variable neighborhood search heuristics for solving the Unit Commitment Problem. International Journal of Electrical Power & Energy Systems, 78, 873-883. doi:10.1016/j.ijepes.2015.12.031

[51] Toksarı, M. D., & Güner, E. (2007). Solving the unconstrained optimization problem by a variable neighborhood search. Journal of Mathematical Analysis and Applications, 328(2), 1178-1187. doi:10.1016/j.jmaa.2006.06.025

[52] Tragantalerngsak, S., Holt, J., & Rönnqvist, M. (1997). Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem. European Journal of Operational Research, 102(3), 611-625. doi:10.1016/s0377-2217(96)00227-5

[53] Tragantalerngsak, S., Holt, J., & Rönnqvist, M. (2000). An exact method for the two-echelon, single-source, capacitated facility location problem. Eu-

ropean Journal of Operational Research, 123(3), 473-489. doi:10.1016/s0377-2217(99)00105-8

[54] Xia, W., & Wu, Z. (2007). Supplier selection with multiple criteria in volume discount environments. Omega, 35(5), 494-504. doi:10.1016/j.omega.2005.09.002

[55] Şahin, G., & Süral, H. (2007). A review of hierarchical facility location models. Computers & Operations Research, 34(8), 2310-2331. doi:10.1016/j.cor.2005.09.005