УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Ана З. Анокић

МАТЕМАТИЧКИ МОДЕЛИ И МЕТОДЕ РЕШАВАЊА НОВОГ ПРОБЛЕМА РАСПОРЕЂИВАЊА ВОЗИЛА ПРИ ОПТИМИЗАЦИЈИ ТРАНСПОРТА ПОЉОПРИВРЕДНИХ СИРОВИНА

докторска дисертација

Београд, 2017.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Ana Z. Anokić

MATHEMATICAL MODELS AND SOLUTION METHODS FOR THE NOVEL VEHICLE SCHEDULING PROBLEM FOR OPTIMIZING THE TRANSPORT OF AGRICULTURAL RAW MATERIALS

Doctoral Dissertation

Belgrade, 2017.

Ментор:

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

Чланови комисије:

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

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

др Татјана Давидовић, виши научни сарадник
Математички институт САНУ, Београд

Датум одбране:

Mojoj baki

Захвалнице

Желела бих да се захвалим свом ментору проф. др Зорици Станимировић што је прихватила ту улогу, веровала у мене и усмеравала мој рад на прави начин, дајући ми велику мотивацију да учим и стварам после више изгубљених година. Знања и искуства која сам стекла овим трогодишњим радом вреде заиста много, али морам да истакнем да сам имала велику срећу и част да на том путу радим са пре свега дивним људима и изузетним умовима. Хвала бескрајно др Татјани Давидовић за коју оправдано сматрам да је права ризница знања, а њено искуство, литература и идеје су биле драгоцене за квалитет овог истраживања. Уз њих две, напоран рад је био увек пријатан и уместо умора доносио задовољство и жељу да се ради још више и боље. Проф. др Милану Дражићу хвала на корисним саветима и сугестијама, пажљивом слушању мог излагања и вољи да помогне у сваком тренутку. Велику захвалност дугујем свом колеги Ђорђу Стакићу, студенту докторских студија из области рачунарства који је проналазио решење за сваки програмерски проблем. Његов ведар дух, идеје и спремност за сарадњу често су били моја покретачка снага.

Хвала проф. др Ненаду Младеновићу и проф. др Мирјани Чангаловић за сугестије и мени веома значајну подршку на конференцији у Малаги 2016. године и проф. др Kennet Sorensen-у на лепим вестима и похвалама кроз преписку у процесу рецензије рада у којем је представљен део овог истраживања.

Колегама са Катедре за статистику, Пољопривредног факултета у Београду, неизмерно хвала на великој моралној подршци кроз све моје успоне и падове. Посебну захвалност проф. др Ивани и Небојши Ралевић, проф. др Нади Лакић и мојој колегиници из канцеларије, доценту др Свјетлани Јанковић Шоја који су ме штитили од додатних обавеза и проблема омогућивши ми неометан рад током израде дисертације.

Захвална сам породицама Адамовић и Анокић за подршку и помоћ. Мом супругу Владимиру хвала за уговорен први састанак у фабрици шећера, хвала на дивним путовањима, изласцима, концертима и дружењу чиме је увек успевао да ме разоноди и улије ново самопоуздање током докторских студија, а посебно хвала на изузетној подршци на конференцији у Малаги. Нашој деци, Мини и Драгану захвални смо за сваки осмех, загрљај и велику животну радост.

Овај рад је посвећен мојој баки, Мари Адамовић, која је наша вечна породична инспирација. Од свих друштвених вредности највише је ценила образовање, а било јој је ускраћено у суровим условима Другог светског рата и патријахалног васпитања. Ипак није одустала, целог живота је учила и читала радујући се сваком новом сазнању. Хвала јој што је оставила такав траг. Кажу да сам од ње наследила благу нарав, стрпљење, унитрашњи мир, способност игнорисања негативних утицаја и веру да се све увек заврши на најбољи начин, а без тих особина не бих истрајала на овом путу.

Наслов дисертације: Математички модели и методе решавања новог проблема распоређивања возила при оптимизацији транспорта пољопривредних сировина

Резиме: Проблеми оптимизације налазе примену у свим сферама реалног живота. Адекватно моделирање проблема оптимизације и развој адекватних математичких метода за њихово решавање од великог су значаја за функционисање многих реалних система. Предмет ове докторске дисертације је нов проблем распоређивања возила (енгл. *Vehicle Scheduling Problem - VSP*) при оптимизацији транспорта пољопривредних сировина. Оптимизација транспорта сировина које се користе за прераду до финалног производа је од великог значаја у почетној фази производње. То је посебно изражено код прераде пољопривредних сировина, јер је њихова цена на тржишту ниска, па трошкови транспорта имају највећи удео у укупним трошковима производње финалног прозвода. Из тог разлога, смањење неопходног времена и уложених финансијских средстава у фази транспорта сировина, директно утиче на повећање профитабилности предузећа.

Варијанта проблема VSP која је разматрана у дисертацији настала је из потребе оптимизације транспорта шећерне репе у једној фабрици за производњу шећера у Србији, али се може применити и у ширем контексту, за оптимизацију транспорта сировина или робе који се организује у великим компанијама при истим или сличним условима. Проблем укључује низ специфичности по којима се разликује од постојећих варијанти проблема распоређивања возила, те математички модели који су до сада предложени у литератури не описују адекватно разматрани проблем.

У дисертацији је анализирана сложеност разматраног проблема. Релаксација предложеног проблема поређена је са проблемом паралелног распоређивања машина (енгл. *Parallel Machine Scheduling Problem - PMSP*). На основу еквивалентности са проблемом партиције (енгл. *Partitioning problem*), познато је да PMSP НП-тежак. У дисертацији је доказана аналогија између релаксације разматраног проблема и PMSP и закључено је да је предложени проблем НП-тежак.

Развијени су нови адекватни математички модели који обухватају све карактеристике разматраног проблема. Предложени математички модели су упоређени у смислу ефикасности коришћењем егзактних решавача Lingo 17 и CPLEX

MIP 12.6.2. Експериментални резултати су показали да оба егзактна решавача проналазе оптимална или допустива решења само на реалним инстанцама проблема малих димензија, што је и очекивано, имајући у виду сложеност проблема. Из тог разлога, коришћење хеуристичких и метахеуристичких метода представља адекватан приступ за решавање инстанци проблема већих димензија. С обзиром на специфичности разматраног проблема, постојеће имплементације хеуристичких и метахеуристичких метода за проблеме распоређивања или рутирања возила из литературе не могу бити директно примењене на овај конкретан проблем. Из тог разлога су, у овој дисертацији, дизајниране и имплементиране различите варијанте метахеуристике под називом Метода променљивих околина (енгл. *Variable Neighborhood Search - VNS*), као и једна варијанта Похлепне стохастичко-адаптивне процедуре претраге (енгл. *Greedy Randomized Adaptive Search Procedure - GRASP*), чији су елементи прилагођени карактеристикама разматраног проблема распоређивања возила.

Прво је разматран проблем VSP-P који представља потпроблем предложене варијанте проблема распоређивања возила. VSP-P је добијен искључивањем ограничења која се односе на усклађивање времена стизања камиона на сваку од локација и у круг фабрике. За решавање VSP-P дизајнирана су два алгоритма: варијанта Основне методе променљивих околина (енгл. *Basic Variable Neighborhood Search - BVNS*) и Похлепна стохастичко-адаптивна процедура претраге (енгл. *Greedy Randomized Adaptive Search Procedure - GRASP*). Обе методе су тестиране на инстанцама добијеним на основу података из праксе и на скупу генерисаних инстанци већих димензија. Експериментални резултати показују да и BVNS и GRASP метода на реалним инстанцама малих димензија достижу сва позната оптимална решења, добијена егзактним решавачима. На скупу реалних инстанци средњих димензија, BVNS метода је достигла или побољшала горње границе вредности функције циља добијених CPLEX решавачем за ограничено време извршавања од 5h. Такође, BVNS даје решења бољег квалитета у односу на GRASP методу на скупу реалних инстанци средњих димензија и генерисаних инстанци већих димензија. Анализом свих експерименталних резултата, закључено је да обе методе представљају адекватне приступе за решавање разматраног проблема. Ипак, уочава се да је BVNS боља када је у питању квалитет добијених решења, док је GRASP ефикаснији од BVNS у погледу времена извршавања.

За полазни проблем распоређивања возила (VSP) који укључује све карак-

теристике реалног проблема, дизајниране су и имплементиране три варијанте VNS методе: Основна метода променљивих околина (енгл. *Basic Variable Neighborhood Search - BVNS*), Адаптивна метода променљивих околина (енгл. *Skewed Variable Neighborhood Search - SVNS*) и Побољшана основна метода променљивих околина (енгл. *Improved Basic Variable Neighborhood Search - BVNSi*). Прве две методе користе исте структуре околина, при чему BVNS користи стратегију најбољег побољшања (енгл. *Best improvement strategy*), а SVNS стратегију првог побољшања (енгл. *First improvement strategy*) у фази локалне претраге. Све три методе су тестиране на реалним и генерисаним инстанцама. Експериментални резултати на скупу реалних и генерисаних инстанци су показали да BVNS даје решења бољег квалитета у односу на SVNS, али је време рада SVNS методе значајно краће. Трећи дизајнирани алгоритам (BVNSi) представља побољшану варијанту BVNS методе, која користи општије структуре околина у односу на BVNS и SVNS, што је допринело једноставности BVNSi алгоритма и брзини извршавања. Тестиране су две варијанте BVNSi методе које користе различите стратегије у оквиру процедуре локалног претраживања: BVNSi_F са стратегијом првог и BVNSi_B са стратегијом најбољег побољшања. Анализом и поређењем резултата закључено је да у погледу квалитета добијених решења као и стабилности, најбоље перформансе има BVNS метода, али је SVNS најбржа. Варијанте побољшане методе BVNSi успевају да пронађу нова најбоља решења за две реалне инстанце средњих димензија и да за значајно краће време реше инстанце већих димензија, у поређењу са BVNS и SVNS, редом. Са друге стране, варијанте BVNSi методе нису тако стабилне као BVNS и SVNS методе на скупу реалних и генерисаних инстанци. У случају једне генерисане инстанце већих димензија, обе BVNSi методе су имале значајно лошије перфоермансе у односу на остале методе, што је негативно утицало на просечне вредности функције циља и времена извршвања.

Предложени проблем распоређивања возила је од великог практичног значаја за оптимизацију транспорта пољопривредних сировина. Планирано је да добијени резултати буду примењени у пракси, делимично или у потпуности. Са теоријског аспекта, развијени математички модели представљају научни допринос областима оптимизације и математичког моделирања. Развијене варијанте VNS метода прилагођених проблему и поређења њихових перформанси представљају научни допринос области метахеуристичких метода за решавање проблема оптимизације.

Кључне речи: математичко моделирање, оптимизација трошкова транспорта, проблем распоређивања возила, метахеуристике, метода променљивих околина, похлепна стохастичко-адаптивна процедура претраге

Научна област: Математика

Ужа научна област: Оптимизација, Операциона истраживања

УДК број:

Dissertation title: Mathematical models and solution methods for the novel vehicle scheduling problem for optimizing the transport of agricultural raw materials

Abstract: Optimization problems arise from many real-life situations. The development of adequate mathematical models of optimization problems and appropriate solution methods are of great importance for performance of real systems. The subject of this doctoral dissertation is a novel vehicle scheduling problem (VSP) that arises from optimizing the transport of agricultural raw materials. The organization of transport of raw materials is of great importance in the initial phase of production. This is particularly evident in the case of agricultural raw materials, because their price in the market is very low, and therefore, the costs of their transport represent the largest part of the total production cost. For this reason, any reduction of time and money spent in this early production stage directly increases the company's profitability.

The considered variant of VSP arises from optimizing the transport of sugar beet in a factory for sugar production in Serbia, but it can also be applied in a wider context, i.e., to optimize the transport of raw materials or goods in large companies under the same or similar conditions. The considered problem involves a number of specific constraints that distinguish it from existing variants of the vehicle scheduling problem. Therefore, mathematical models proposed in the literature for other variants of VSP do not describe adequately the considered problem.

The complexity of the newly introduced VSP is analyzed. It is proven that the introduced VSP belongs to the class of NP-hard problems by comparing its relaxation with the Parallel Machine Scheduling Problem (PMSP). PMSP is known to be NP-hard, as it is equivalent to the Partitioning problem. From the established analogy between the relaxation of the considered VSP and PMSP, it is concluded that the VSP introduced in this dissertation is NP-hard.

New mathematical models of the considered problem that involve all problem specific properties, are developed. The proposed mathematical models are compared in sense of efficiency by using Lingo 17 and CPLEX MIP 12.6.2 solvers. Experimental results showed that both exact solvers provided optimal or feasible solutions only for small-size real-life problem instances. However, this was expectable, having in mind the NP-hardness of the considered problem. Therefore, heuristic and metaheuristic method seem to be appropriate approaches for solving problem instances of larger dimension. Due to specific properties of the considered problem, the ex-

isting implementations of heuristic and metaheuristic methods for vehicle routing and scheduling problems can not be directly applied. For this reason, different variants of well-known Variable Neighborhood Search (VNS) metaheuristic, as well as Greedy Randomized Adaptive Search Procedure (GRASP), are designed. The constructive elements of the proposed VNS and GRASP implementations are adapted to the characteristics of the considered vehicle scheduling problem.

A subproblem of the proposed variant of vehicle scheduling problem, denoted as VSP-P is considered first. VSP-P is obtained from the initial VSP by excluding problem specific constraints regarding vehicle arriving times to each location and to the factory area. Two metaheuristic algorithms are designed as solution methods for this subproblem: Basic Variable Neighborhood Search - BVNS, and Greedy Randomized Adaptive Search Procedure - GRASP. Both proposed approaches were tested on instances based on real-life data and on the set of generated instances of lager dimensions. Experimental results show that BVNS and GRASP reached all optimal solutions obtained by exact solvers on small-size real-life problem instances. On medium-size real-life instances, BVNS reached or improved upper bounds obtained by CPLEX solver under time limit of 5 hours. BVNS showed to be superior compared to GRASP in the sense of solution quality on medium size real-life instances, as well as on generated large-size problem instances. However, general conclusion is that both proposed methods represent adequate solution approaches for the subproblem VSP-P. BVNS provides solutions of better quality compared to GRASP, while GRASP outperforms BVNS regarding the average CPU time required to produce its best solutions.

For the initial vehicle scheduling problem (VSP) that includes all problem specific constraints, three VNS-based metaheuristic methods are designed and implemented: Basic Variable Neighborhood Search - BVNS, Skewed Variable Neighborhood Search - SVNS, and Improved Basic Variable Neighborhood Search - BVNSi. BVNS and SVNS use the same neighborhood structures, but different search strategies in local search phase: BVNS uses Best improvement strategy, while SVNS uses First improvement strategy. All three VNS-based methods are tested on real-life and generated problem instances. As it was expected, experimental results showed that BVNS outperformed SVNS regarding solution quality, while SVNS running time was significantly shorter compared to BVNS. The third designed algorithm BVNSi represents a variant of BVNS that uses more general neighborhood structures compared to the ones used in BVNS and SVNS. The use of such neighborhood structures

lead to the simplicity of BVNSi and shorter running times compared to BVNS. Two variants of BVNSi method that exploit different strategies in Local search phase are designed: BVNSi$_B$ with best improvement strategy and BVNSi$_F$ with First improvement strategy. The results of computational experiments for all proposed VNS-based methods for VSP are analyzed and compared. Regarding the quality of the obtained solutions, BVNS method shows the best performance, while SVNS needed the shortest average running times to produce its best solutions. Two variants of BVNSi method succeeded to find new best solutions on two medium size real life instances and to solve large size instances in shorter running time compared to BVNS and SVNS, respectively. However, both BVNSi$_B$ and BVNSi$_F$ turn out to be less stabile than BVNS and SVNS on real-life and generated inatances. In the case of one large-size generated instance, both BVNSi variants had significantly worse performance compared to BVNS and SVNS, which had negative impact on their average objective values and average running times.

The proposed vehicle scheduling problem is of great practical importance for optimizing the transport of agricultural raw materials. It is planned to use the obtained results in practice (partially or completely), as a support to decision makers who organize transportation of in the early production phase. From the theoretical point of view, the developed mathematical models represent a scientific contribution to the fields of optimization and mathematical modeling. The variants of VNS methods that are developed and adapted to the problem, as well as comparison of their performances, represent a scientific contribution to the field of metaheuristic methods for solving NP-hard optimization problems.

Keywords: mathematical modeling, optimization, vehicle scheduling problem, metaheuristics, variable neighborhood search method, greedy randomised adaptive search procedure

Research area: Mathematics

Research sub-area: Optimization, Operational research

UDC number:

Садржај

Глава 1

Увод

Бројни проблеми који се срећу у савременом животу могу се формулисати као проблеми оптимизације. Њихово решавање је од великог значаја за ефикасније функционисање реалних система. Када је у питању привреда, уштеда ресурса или смањење трошкова директно утиче на повећање профитабилности предузећа. Из тог разлога, развој модела и адекватних метода за решавање широког спектра проблема оптимизације је од великог практичног значаја, а интересовање истраживача за ову област константно расте, што се огледа у броју публикованих радова који се баве проблемима оптимизације.

Оптимизација транспорта сировина у производњи или транспорта финалних производа у ланцима снабдевања великих предузећа је изузетно важна. У свакој привредној грани, организација транспорта сировина има своје специфичности које су одређене типом и карактеристикама возила, расположивим ресурсима предузећа и компанија са којима посматрано предузеће сарађује, али и својствима саме сировине као што су, на пример, неопходни услови за одрживост робе који диктирају време и начин транспорта. Из тог разлога, сваком проблему оптимизације који се односи на организацију транспота у различитим сегментима привреде мора се приступити индивидулно, узимајући у обзир све специфичности посматраног проблема. Добро разумевање свих аспеката реалног проблема, укључујући циљ оптимизације, предуслов је за развој адекватних математичких модела који прецизно описују посматрани проблем. Након формулисања математичких модела, наредни корак је избор одговарајућих метода за решавање. Савремена литература из области оптимизације обилује примерима реалних транспортних проблема и методама решавања које представљају инспирацију за креирање нових и прилагођавање постојећих тех-

ника како би се омогућило ефикасно решавање новог транспортног проблема. Егзактне методе гарантују оптималност, али су доста захтевне када је у питању време извршавања. У случају проблема веће сложености, оне често већ код инстанци средњих димензија, не успевају да пронађу оптимално решење у разумном временском интервалу. Стога се, при решавању сложених проблема, егзактне методе користе у комбинацији са хеуристичким и метахеуристичким методама у виду хибридизованих хеуристика. Такође, решења на инстанцама малих димензија, добијених егзактним методама могу послужити као репер за оцену ефикасности дизајнираних хеуристика. Поред других хеуристичких метода, као што су генетски алгоритми и алгоритми инспирисани природним појавама, Метода променљивих околина се издваја као најчешће примењиван метода решавања проблема рутирања и распоређивања возила. Захваљујући једноставној имплементацији која води ка доста добрим резултатима у поређењу са другим методама. Развијене су бројне варијанте методе променљивих околина и примењиване као врло ефикасне методе решавања за различите проблеме овог типа. Такође, у литератури се могу пронаћи бројни примери решавања проблема рутирања и распоређивања возила применом једноставне метахеуристичке методе под називом Похлепна стохастичко-адаптивна процедура претраге. Стога се може сматрати да су метода променљивих околина и похлепна стохастичко-адаптивна процедура претраге приступи који су адекватни и перспективни приликом решавања проблема рутирања и распоређивања возила.

1.1 Проблеми оптимизације

Проблеми оптимизације налазе примену у свим сферама реалног живота (индустрији, пољопривреди, транспортним и телекомуникационим системима, логистици, медицини,...). Иако се ови проблеми у општем случају врло једноставно дефинишу и имају јасну формулацију, за њихово решавање улажу се изузетно велики напори. Адекватно моделирање проблема оптимизације и развој одговарајућих математичких метода за њихово решавање од великог су значаја за функционисање многих реалних система. Више о проблемима оптимизације може се наћи у [22] и [81].

За дефинисање проблема оптимизације у општој форми, неопходно је увести основне појмове.

Нека су дати *простор решења* S и *функција циља* или *критеријумска функција* $f : S \to R$. Решење проблема оптимизације се најчешће означава са $x \in S$. У општем случају, проблеми оптимизације се састоје у проналажењу оптималног решења $x^* \in S$, односно оног решења за које функција циља има најбољу вредност (најмању или највећу). Према томе, уколико је реч о проблему минимизације, општа математичка формулација проблема оптимизације је:

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

Проблеми код којих се тражи максимална вредност функције циља своде се на проблеме минимизације, с обзиром да важи релација:

$$\max_{x \in S} f(x) = -\min_{x \in S} -f(x). \tag{1.2}$$

Проблеме оптимизације је могуће класификовати према различитим критеријумима. У зависности од тога да ли се захтева да променљиве задовољавају одређене услове или не, разликујемо проблеме:

- *условне* оптмизације, у присуству ограничења
- *безусловне* оптимизације, без ограничења на променљивима.

Такође, могуће је захтевати да променљиве узимају само одређене вредности неког дискретног скупа реалних бројева, нпр. скуп целих или бинарних бројева, или пак, оне могу узети све вредности скупа реалних бројева. У том смислу разликујемо:

- *континуалне*
- *дискретне*

проблеме оптимизације. Такође, могуће је да проблем укључује два скупа променљивих: у првом, променљиве узимају све реалне вредности или реалне вредности из интервала реалних бројева, док променљиве другог скупа узимају вредности неког дискретног скупа реалних бројева. Тада је у питању *мешовити* проблем оптимизације.

Проблеми непрекидне оптимизације, у зависности од типа функције циља и постављених ограничења, могу бити: линеарни, конвексни, неконвексни, квадратни... Проблем линеарног програмирања (енгл. *Linear Programming - LP*) је проблем оптимизације у којем функција циља као и изрази који се јављају

у ограничењима линеарно зависе од променљивих. Општа формулација LP проблема у коначно димензионим просторима је:

$$\min \quad c^T x \tag{1.3}$$

при ограничењима

$$Ax \leq b, \tag{1.4}$$

где је $x \in R^n$ вектор променљивих, док матрица $A \in R^{m \times n}$ и вектори-колоне $b \in R^m$ и $c \in R^n$ садрже одговарајуће коефицијенте проблема.

Проблем квадратног програмирања (енгл. *Quadratic Programming-QP*) подразумева да постоји квадратна зависност функције циља од променљивих у моделу. Општа формулација проблема QP је:

$$\min \quad xQ^T x + c^T x \tag{1.5}$$

при ограничењима

$$Ax \leq b, \tag{1.6}$$

где је, уз претходно уведене ознаке, матрица $Q \in R^{n \times n}$.

Уколико се код проблема QP укључе и ограничења квадратног типа, добија се проблем са квадратним ограничењима (енгл. *Quadratically Constrained Programming-QCP*) који може имати линеарну или квадратну функцију циља.

$$\min \quad xQ^T x + c^T x \tag{1.7}$$

при ограничењима

$$xA^T x + d^T x \leq b, \tag{1.8}$$

где је, уз претходно уведене ознаке, низ $d \in R^n$. У случају када је функција циља линеарна важи $Q = 0$.

Параметри који се појављују у проблемима оптимизације могу бити одређени, случајног карактера тј. узимају вредности које су одређене расподелом вероватноћа, или неодређени. Према типу параметара проблеми оптимизације могу бити:

- *детерминистички*
- *стохастички* проблеми
- проблеми у условима *неодређености*.

Многи проблеми оптимизације спадају у НП-тешке проблеме. Каже се да је проблем оптимизације полиномске сложености, односно да припада П-класи, уколико за његово решавање постоји алгоритам полиномске сложености. Ако се за дату вредност може утврдити, у оквиру полиномског времена, да ли она представља решење проблема, сматра се да посметрани проблем припада класи НП. Проблем који је НП-тежак има својство да се сваки проблем класе НП може свести на њега у полиномском времену. Уколико НП-тежак проблем и сам припада класи НП, каже се да је такав проблем НП-комплетан. Како за многе проблеме оптимизације не постоје егзактни алгоритми полиномске сложености који омогућавају решавање проблемских инстанци већих димензија у разумном временском року, постоји стална потреба за креирањем алгоритама који дају задовољавајућа приближна решења. Бројне хеуристичке и метахеуристичке методе се успешно користе за решавање различитих проблема оптимизације.

1.2 Проблеми рутирања и распоређивања возила

Значајно место међу проблемима оптимизације заузимају проблеми рутирања и распоређивања возила. Захваљујући применама у разним сферама реалног живота они су веома заступљени у литератури и константно привлаче пажњу истраживача са теоријског и практичног аспекта. До сада је у литератури предложен велики број варијанти проблема рутирања и распоређивања возила и ефикасних метода за њихово решавање. И поред тога, нове реалне ситуације и услови који се јављају у пракси приликом организације транспорта представљају непресушни извор нових идеја и варијанти ових проблема, као и непрекидне мотивације за истраживање ове области.

Класичан проблем рутирања возила (енгл. *Vehicle routing problem - VRP*) представља генерализацију познатог НП-тешког проблема трговачког путника (енгл. *Traveling salesmen problem - TSP*) [44], [8], те VRP такође припада класи НП-тешких проблема. Класичан VRP се дефинише на усмереном или неусмереном графу, при чему сваки чвор представља клијента са одређеним потребама које треба задовољити, а свакој грани је додељена ненегативна реална вредност, која представља трошкове транспорта при коришћењу посматране гране. Циљ VRP-а је да се за свако од m идентичних возила одреди рута тако да укупни трошкови буду минимални, а притом треба да буду задовољени следећи услови:

- сваки клијент се посећује тачно једанпут,
- свака рута почиње и завршава се у депоу,
- укупне потребе сваког клијента нису веће од капацитета возила,
- дужина руте не прелази унапред дефинисану горњу границу.

Различите варијанте VRP-а, њихове математичке формулације и преглед метода из литературе које се користе за решавање ових проблема могу се наћи у [24].

Сложенији типови VRP-а укључују ограничења која се односе на капацитете и дужине рута. Када су присутна само ограничења капацитета, посматрани VRP се назива проблем рутирања возила са ограниченим капацитетом (енгл. *Capacitated Vehicle Routing Problem - CVRP*) и он је доста изучаван у литератури. Augerat и сар. [10] су за решавање CVRP користили методу гранања и ограничавања (енгл. *Brunch-and-Bound - BnB*) са три различите процедуре раздвајања: конструктивни алгоритам, похлепан алгоритам на случајан начин и једноставна процедура заснована на табу претраживању, чиме су дизајниране три варијанте методе гранања и одсецања (енгл. *Brunch-and-Cut - BnC*). Предложене BnC методе су достигле оптимална решења на познатим CVRP тест примерима из литературе и на генерисаним инстанцама већих димензија. Duhamel и сар. [31] су разматрали варијанту проблема CVRP, означену са 2L-CVRP, код које се потребе клијената састоје у достављању предмета окарактерисаних двема димензијама. Аутори су развили еволутивну локалну претрагу са вишеструким покретањем (енгл. *Multi-start evolutionary local search - MS-ELS*) за решавање разматраног 2L-CVRP. Експериментални резултати, добијени на класичним CVRP и на 2L-CVRP инстанцама из литературе, показали су да је метода MS-ELS предложена у Duhamel и сар. [31] супериорнија у односу на постојеће методе из литературе примењене на истим CVRP и 2L-CVRP тест примерима. Динамичку варијанту проблема CVRP која укључује случајна појављивања саобраћајних гужви чији су интензитет и дужина унапред одређени дистрибуцијом вероватноћа, разматрали су Mandziuk и сар. [66]. Предложене су горње границе поверења и примењене на методу претраге дрвета (енгл. *Upper confidence bounds applied to trees - UCT*), која представља проширење алгоритма претраге дрвета заснован на Monte Carlo методи. Експериментални резултати приказани у [66], показују да приступ заснован на

UCT алгоритму даје обећавајуће резултате у поређењу са осталим методама решавања динамичког CVRP-а.

Значајно место у литератури заузима и проблем рутирања возила са вишеструким путовањима (енгл. *Vehicle routing problem with multiple trips - VRPMT*). VRPMT представља варијанту VRP-а, код које возила могу да направе неколико рута током посматраног временског периода. Проблем је увео Fleischmann [36], а затим су се у литератури појавили и бројни радови који разматрају VRPMT или неку његову варијанту. Olivera и Viera [73] су за решавање разматраног VRPMT предложили процедуру са адаптивном меморијом (енгл. *Adaptive memory procedure - AMP*), која се заснива на комбиновању компоненти вискоквалитетних решења. AMP се састоји из три основна корака: конструкције решења коришћењем меморије, локалне претраге и освежавања меморије, који се смењују до задовољења критеријума заустављања. Резултати тестирања из [73] показују да је AMP алгоритам успео је да пронађе више оптималних решења на инстанцама из литературе од других приступа решавања истог проблема. VRP са вишеструким путовањима су разматрали и Cattaruzza и сар. [18] који су предложили меметски алгоритам са оператором локалне претраге који комбинује стандардне VRP потезе са потезима размене између путовања. Експериментални резултати из [18] су показали да је дизајниран меметски алгоритам надмашио постојеће алгоритме за решавање VRPMT, у смислу просечног квалитета добијених решења. Реалан проблем компаније за кућну доставу, који представља VRP варијанту са временским оквирима и вишеструким путовањима (енгл. *VRP with Time Windows and Multi-trips - VRPM-TW*), разматрали су Martinez и сар. у [68]. Проблем подразумева ограничени број возила и укључује просторна ограничења специфична за утоваривање објеката кружног облика. Предложен је модел линеарног програмирања који је коришћен за егзактно решавање инстанци малих димензија, а затим и хеуристичка метода за решавање већих тест примера. Хеуристика предложена у [68] се састоји из два корака: у првом кораку гради се иницијално решење помоћу модификованог *Solomon II* секвенцијалног метода уметања (енгл. *Solomon II sequential insertion method*), док други корак побољшава конструисано иницијално решење користећи табу претраживање. Battarra и сар. [12] су применили приступ адаптивног вођења на варијанту VRPMT, под именом VRP са минималним вишеструким путовањима (енгл. *Minimum multiple trip VRP - MMTVRP*). Овај проблем подразумева дате временске оквире за опслуживање клијената, свако

возило прави више од једне руте током радног дана, са циљем минимизације укупног броја вишеструких путовања. Експериментални резултати приступа адаптивног вођења на скупу реалних инстанци, показују његов потенцијал за решавање MMTVRP.

У новије време, при истраживању проблема рутирања возила, доста пажње се посвећује проблемима рутирања са прикупљањем и доставом (енгл. *Vehicle routing problems with pickup and delivery*). Madankumar и Rajendran [65] су предложили два модела мешовитог целобројног линеарног програмирања (енгл. *Mixed Integer Linear Programming - MILP*) за VRP са прикупљањем и доставом у ланцу снабдевања. Прецизније, проблем разматран у [65] је зелени проблем распоређивања возила са прикупљањем и доставом у полупропусном ланцу снабдевања (енгл. *Green vehicle routing problem with pickup and delivery in a semiconductor supply chain - G-VRPPD-SSC*). Код ове веријанте VRP транспорт се врши од тачке прикупљања до тачке доставе, без повратка у депо, при чему се води рачуна о минимизацији емисије штетних гасова приликом транспорта. Циљ првог предложеног MILP модела је одредити скуп рута и распоред возила са минималним трошковима које задовољавају постављене услове. Други MILP модел представља проширење првог модела, јер укључује различите цене горива на различитим бензинским станицама, а за циљ има минимизацију збира трошкова рутирања и трошкова горива. Неупарени проблем распоређивања возила са прикупљањем и доставом и вишеструким посетама разматрали су Xu и сар. [92]. Проблем се састоји у доношењу две интегрисане одлуке: упаривање снабдевача са захтевом корисника којег треба опслужити и одређивање рута за возила. Табу претраживање је примењено за решавање посматраног проблема у [92], при чему су добијена решења високог квалитета.

Значајно место међу проблемима оптимизације заузимају проблеми распоређивања (енгл.*Scheduling Problem-SP*). Међу њима, највећа пажња је посвећена проблемима распоређивања возила (енгл. *Vehicle Scheduling Problems – VSP*) који су природно повезани са проблемима рутирања возила. Ови проблеми се често јављају при оптимизацији транспортних система, као што су систем јавног саобраћаја (градски и међуградски превоз путника), транспорт разних врста пошиљки, транспорт новца и драгоцености са обезбеђењем, транспорт пољопривредних и других сировина, итд. У литератури постоји више различитих варијанти ових проблема, али генерално, њихов циљ је одређивање низа путовања које возила треба да обаве током одређеног временског периода како

би била задовољена сва постављена ограничења уз минималне трошкове.

Dantzig и Fulkerson (1954) [27] су формулисали модел линеарног програмирања за проблем одређивања минималног броја танкера који су неопходни како би унапред познат распоред транспорта горива био задовољен. Од излагања њихове идеје, интересовање за проблеме распоређивања је почело да расте. Преглед ових проблема, који укључује око 500 научних радова са различитим варијантама проблема распоређивања и методама за њихово решавање, из периода од 2006. до 2014. године дао је Allahverdi (2015) [3].

Проблеми распоређивања се, у општем случају, дефинишу на следећи начин. Дати су:

- скуп $A = (J_1, J_2, \ldots, J_n)$ задатака (послова),
- релација поретка $\prec$ на скупу A,
- *тежинска функција* W која пресликава скуп A на скуп целих ненегативних бројева, придружујући сваком задатку из скупа A време које је неопходно за његово извршавање,
- одређени број машина (m).

Претпоставка је да се у сваком временском интервалу $t = 0, 1, \ldots, t_{max}$ може извршити највише m послова. Притом, t_{max} представља последњи интервал у коме се извршава бар један посао. Ако је $J \prec J'$ тада извршавање задатка J' може да почне тек после $W(J)$ јединица од почетка извршавања задатка J. Циљ проблема распоређивања је да се минимизује t_{max}.

Најчешће се проблеми распоређивања возила везују за планирање транспорта у системима јавног саобраћаја. Прво се, на основу прикупљених података одреде потребе путника које дефинишу руте и путничке станице, а затим се приступа оптимизацији коришћења расположивих ресурса. Више о проблемима овог типа, њиховим класификацијама и методама решавања може се наћи у [13], [16] и [78].

Проблем распоређивања возила за транспортне системе у јавном саобраћају се дефинише на следећи начин. Нека је дат скуп путовања са фиксним временима поласка и повратка, познатим почетним и крајњим одредиштем и трајањем путовања између сваке две крајње станице. Циљ је придружити путовања возилима тако да важи:

- свако путовање се обавља тачно једном,
- свако возило обавља допустив низ путовања,
- укупни трошкови су минимални.

Укупни трошкови могу бити подељени на *фиксне* трошкове, који се користе за инвестиције и одржавање и *оперативне* трошкове који покривају гориво и поправке возила.

У општем случају, код проблема распоређивања возила, свако путовање карактерише време и локација поласка, као и време и локација повратка тј. завршетка путовања. Уколико возила крећу са истог места и враћају се по завршетку путовања на то место реч је о проблемима са једним депоом (енгл.*Single-depot VSP*), а проблем се значајно усложњава увођењем више депоа (енгл. *Multi-depot VSP*). Такође, у општем случају проблема распоређивања, возила у једном путовању посећују више локација, при чему су позната времена путовања од депоа до сваке локације и назад као и између локација, уколико су повезане тј. уколико је могуће путовати на посматраној релацији. Основни циљ је направити распоред путовања и придружити свако путовање једном и само једном возилу како би укупни трошкови били минимални. Поред инимизације укупних трошкова, код проблема распоређивања возила циљ може бити и минимизација радног времена, укупног времена транспорта, укупног пређеног растојања и др. У новије време, интензивно се ради на тзв. *зеленим* проблемима рутирања и распоређивања возила (енгл. *Green Vehicle Routing and Scheduling Problems-GVRSP*) код којих је циљ минимизовати емитовање штетних издувних гасова, посебно CO_2. Већина проблема распоређивања возила припада класи НП-тешких проблема.

Бројна истраживања се врше у циљу унапређења друмских и железничког саобраћајних система у урбаним срединама. Baita и сар. су [11] разматрали реалан проблем распоређивања возила за аутобуски превоз у граду Местре у Италији. Испоставило се да је посматрани проблем тежак за моделирање због специфичних захтева који произилазе из праксе и вишекритеријумске природе проблема. Предложена су три алгоритма за решавање разматраног проблема: традиционални који се заснива на скаларизацији различитих критеријума, лексикографски који је заснован на принципима логичког програмирања и *Pareto* оптимизација имплементирана коришћењем генетских алгоритама. Експериментални резултати и поређења показали су да су прве две методе надмашиле

трећу односно употреба гентских алгоритама у [11] резултовала је методом лоших перформанси. Haghani и сар. [45] су представили анализу три модела распоређивања возила за аутобуски превоз у граду Baltimore, SAD. Први модел укључује више депоа, док су друга два модела изведена из првог као специјални случајеви са једним депоом. Експериментални резултати су показали да један од модела са једним депоом има боље перформансе од преостала два предложена модела. Wang и Shen [89] су разматрали проблем распоређивања електричних аутобуса, који припада класи VSP са рутама и ограничењима времена пуњења батерија (енгл. *VSP with route and fueling time constraints - VSPRFTC*). За решавање разматраног проблема у [89] предложен је алгоритам заснован на методи оптимизације мрављим колонијома.

Проблеми распоређивања возила се често јављају и при транспорту добара у компанијама које се баве производњом. Fliedner и сар. [38] су предложили VSP са стратегијом складиштења на точковима између једног дистрибутивног центра и једне производне локације (енгл. *VSP under the warehouse-on-wheels policy - VSP-WOW*). Прецизније, код овог проблема, хомогена возила са ограниченим капацитетом која достављају материјале служе истовремено и као покретна складишта. Време стизања возила је одређено датумом испоруке који се повећава идући од најхитнијег контејнера ка најмање хитном. У раду [38] разматране су различите функције циља и анализирана је сложеност проблема.

У литератури постоје примери успешне примене проблема распоређивања возила при оптимизацији шећерне трске. Higgins [51] је разматрао проблем распоређивања возила при транспорту шећерне трске у аустралијској компанији и за његово решавање дизајнирао две метахеуристичке методе, табу претраживање и методу променљивих околина. Резултати представљени у [51] показују да обе методе проналазе решења која у просеку смањују време чекања за око 90% у поређењу са познатим решењима која су се примењивала у пракси. Milan и сар. [69] су проучавали проблем транспорта шећерне трске на Куби у циљу смањења транспортних трошкова интегрисањем друмског и железничког транспорта. Предложили су MILP модел који адекватно описује разматрани проблем и применили комерцијалне HyperLINDO и ILOG CPLEX решаваче како би пронашли оптимална решења на реалним тест примерима. Студију која потиче из тајландске индустрије прераде шећерне трске представили су Thuank и сар. [87]. У питању је проблем максимизације оцењених приноса шећерне трске, под једнаким условима за све произвођаче. За разматрани про-

блем предложен је математички модел и хеуристички алгоритам заснован на табу претраживању.

У литератури се могу пронаћи и радови у којима су разматрани проблеми транспорта неких других пољопривредних сировина. На пример, Sethanan и Pitakaso [82] су предложили пет различитих еволутивних алгоритмима за проблем оптимизације транспорта сировог млека. Проблем разматран у [82] састоји се у одређивању рута за прикупљање сировог млека из сакупљачких центара и транспорт до фабрика за прераду млека, са циљем минимизације укупних трошкова који се састоје из трошкова горива и трошкова чишћења танкера на возилима. Варијанта VRP која укључује возила са више преграда за оптимизацију прикупљања маслиновог уља у Тунису, разматрали су Lahyani и сар.[60]. Аутори су предложили формулацију целобројног линеарног програмирања и имплементирали алгоритам гранања и ограничавања за решавање разматраног проблема.

Неретко се проблеми распоређивања возила посматрају заједно са проблемима рутирања возила, при чему се прави распоред по којем возила праве руте, а истовремено се врши и оптималан избор локација које ће се наћи на свакој од рута. Реч је о интегрисаним проблемима рутирања и распоређивања. На значај оваквог приступа указали су Moons и сар. у [72]. Према [72], одвојено посматрање аспеката рутирања и распоређивања води ка субоптималним решењима проблема. Agustina и сар. [2] су интегрисали проблем рутирања и распоређивања возила при организацији транспорта хране. Проблем предложен у [2] назван је проблемом рутирања и распоређивања возила са концептом потрошачких зона и крутим временским оквирима (енгл. *Vehicle routing and scheduling with the concept of costumer zones and hard time windows - VRSP-CZHTW*). Циљ разматраног проблема је минимизација укупних трошкова, укључујући транспортне трошкове и трошкове пенала у случају ране или касне доставе, при чему се користе крути уместо флексибилних временских оквира, јер се достава хране мора извршити на време. Методу симулираног каљења применили су Xiao и сар. [91] за решавање зеленог проблема рутирања и распоређивања возила (енгл. *Green vehicle routing and scheduling problem - GVRSP*) са хијерархијским циљевима и пондерисаним кашњењима у [91]. Предложена је нова математичка формулација у коју су укључене три функције циља које се, редом, односе на: минимизацију укупне CO_2 емисије, укупно пређено растојање и укупно време путовања. Експериментални резултати представљени у [91] по-

казују да је дизајнирана метода ефикасна при решавању тест примера већих димензија. Androutsopoulos и Zografos [4] су интегрисали VRP и VSP формулишући и решавајући VRP са двоструким циљем и временским оквирима, (енгл. *Bi-objective time, load and path-dependent VRP with time windows - BTL-VRPTW*). Циљ предложеног BTL-VRPTW је одредити руте опслуживања и распоред путовања који минимизује укупне трошкове транспорта и потрошње горива. Прецизније, проблем подразумева одређивање времена путовања укључујући реалне временски промењиве услове у саобраћају. Предложена метода решавања интегрише аспекте рутирања и распоређивања у алгоритму заснованом на оптимизацији мрављим колонијама.

Оптимизација транспорта робе и сировина у реалним проблемима, укључује и низ специфичних проблемских захтева, што често води ка новим формулацијама, које се не могу пронаћи у литертури. Формулисање нових проблема може проузроковати недоумице у вези са сврставањем посматраног проблема у постојеће оквире и категорије. Стога је потребно узети у обзир више типова сродних проблема који у основи имају сличне карактеристике.

Поред проблема распоређивања возила, са практичног аспекта су значајни и проблеми распоређивања машина (*Machine Scheduling Problem* - MSP), који су доста проучавани у литератури. У општем случају, ови проблеми се могу дефинисати на следећи начин. За дати скуп од n послова и m расположивих машина, неопходно је придружити послове машинама тако да се минимизује укупно време рада. Притом, треба да буду задовољени следећи услови:

- свака машина у сваком тренутку може да обавља само један посао,
- сваки посао може да се обавља на само једној машини у било ком тренутку,
- када машина почне неки посао наставља са радом све до његовог завршетка.

Постоје проблеми распоређивања послова на само једној машини (енгл. *Single-machine Problems*) или на више машина (енгл. *Multi-machine Problems*). Оба типа могу укључити распоређивање машина на једном или више нивоа (енгл. *Single-stage Problems* и енгл. *Multi-stage Problems*). Класификације и опис проблема MSP у производњи, као и примери примене могу се наћи у [1].

Један од најпознатијих проблема распоређивања је *Job Shop Scheduling Problem-JSP* који припада класи проблема распоређивања на више машина и са више

нивоа (енгл. *Multi-stage Multi-machine Problem*) [23] и [39]. Тачније, сваки посао се састоји од низа задатака-операција који се морају одрадити по датом редоследу, при чему сваки од њих мора бити обављен на одређеној машини. Проблем се састоји у додељивању задатака машинама са циљем да се минимизује укупно време рада, које представља дужину временског интервала од почетка првог посла па до завршетка свих послова.

1.3 Егзактне методе решавања

За разлику од хеуристичких и метахеуристичких метода које се користе за приближно решавање проблема оптимизације, егзактне методе гарантују оптималност добијеног решења. Егзактне методе се користе у случајевима када је могуће добити оптимална решења у прихватљивом рачунском времену са становишта корисника, што је углавном случај при решавању проблема полиномијалне сложености или инстанци НП-тешких проблема мањих димензија.

Постоји више метода за егзактно решавање и избор адеквантне методе се врши у складу са типом и карактеристикама посматраног проблема. Проблем оптимизације без ограничења на променљивима назива се проблемом безусловне оптимизације, чији је општи облик следећи:

$$\min_{x \in R^n} f(x) \tag{1.9}$$

Градијент функције f се дефинише као колона вектор дужине n чији елементи представљају парцијалне изводе функције f:

$$\nabla f(x) = [\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \dots \frac{\partial f}{\partial x_n}(x)]^T \tag{1.10}$$

Све тачке локалног минимума или максимума задовољавају услов $\nabla f(x) = 0$. Решења једначине $\nabla f(x) = 0$ називају се *стационарне* тачке, што значи да су стационарне тачке кандидати за тачке локалног минимума или максимума. Хесијан матрица (енгл. *Hessian*) функције f је дефинисана изразом:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}. \tag{1.11}$$

Квадратна матрица $A \in R^n \times R^n$ је позитивно семи-дефинитна ако за произвољан вектор $x \in R^n$ важи релација $x^T Ax \geq 0$, а негативно семи-дефинитна када је $x^T Ax \leq 0$ за свако $x \in R^n$. Стационарна тачка x је тачка локалног минимума функције f уколико је Хесијан матрица у тачки x позитивно семи-дефинитна, а тачка локалног максимума уколико је Хесијан матрица у тачки x негативно семи-дефинитна.

За решавање проблема који су описани моделом линеарног програмирања (LP), са линеарном функцијом циља и линеарним везама међу променљивима у ограничењима, (1.3)-(1.4), користи се *Симплекс метод*, предложен од стране George B. Dantzig-a, 1947. године [26], [25]. Без обзира на, експоненцијалну комплексност (у најгорем случају), ова метода се показала врло ефикасном у пракси. Основна идеја се заснива на чињеници да свако линеарно ограничење дели простор решења на два полупростора, од којих један садржи само она решења која задовољавају посматрано ограничење. Ако се узму о обзир сва постављена ограничења, добија се низ полупростора чији пресек представља допустив скуп (*симплекс*), који је конвексан. Једно од темена симплекса представља оптимално решење, уколико постоји.

Метода гранања и ограничавања (енгл. *Brunch-and-Bound - BnB*) користи се за решавање проблема чији је простор решења коначан или пребројив. То су углавном дискретни проблеми, код којих је могуће извршити систематско набрајање кандидата решења у виду посебне структуре- *дрвета* (енгл. *tree*). Сваки *чвор* (енгл. *node*) дрвета представља парцијално решење проблема, а гранање дрвета одговара подели домена променљивих на дисјунктне подскупове. *Корен* дрвета (енгл. *root*) је чвор који садржи читав скуп решења. Листови дрвета (енгл. *leaf node*) су крајњи чворови дрвета, односно чворови у којима нема даљег гранања. Алгоритам истражује *гране* дрвета које представљају подскупове скупа решења. Гране се испитују у односу на горње и доње границе које су постављене на оптимално решење и одбацују уколико се утврди да се на њима не налази решење боље од тренутно најбољег пронађеног. Алгоритам зависи од стратегија за постављање граница с обзиром да ефикасне границе оптималног решења у великој мери утичу на време потребно за добијање оптималног решења. BnB методу је могуће користити и код континуалних проблема, разлагањем домена реалних променљивих на дисјунктне интервале.

За решавање линеарних проблема користи се такође и *Метода гранања и одсецања* (енгл. *Brunch-and-Cut - BnC*) која представља комбинацију BnB методе

и методе одсецајућих равни. *Метода одсецајућих равни* (енгл. *Cutting plane*) се користи када је циљ одредити скуп C, за који је познато да је конвексан без додатних информација о његовој структури. Тражени скуп може бити скуп оптималних или субоптималних решења у случају решавања проблема оптимизације. За случајно изабрану тачку x постоје две могућности: $x \in C$, чиме је добијена нова информација о скупу C или је могуће пронаћи *одсецајућу раван* која раздваја x и C. Одсецајућа раван је дефинисана изразом $a^T z = b$, где су a и b вектори одговарајућих димензија, при чему је $a \neq 0$ и важе релације: $a^T z \leq b$, $\forall z \in C$ и $a^T x \geq b$. На тај начин додаје се ново линеарно ограничење које задовољавају све тачке посматраног скупа C. Метода BnC користи одсецајуће равни како би се пронашла боља граница и на тај начин алгоритам постао бржи.

Метода гранања и оцењивања (енгл. *Brunch-and-Price - BnP*) је у основи врло блиска BnC методи. Једина разлика је у томе што се код BnP приликом формулисања релаксације проблема, користи *Метода генерисања колона* (енгл. *Column generation*). Основна идеја се занива на чињеници да је већина променљивих у решењу небазична и има оптималну вредност једнаку нули. Из тог разлога, потребно је посматрати само подскуп скупа свих променљивих. Алгоритам BnP почиње решавањем полазног проблема на формираном потскупу променљивих. Тако формирани проблем назива се *главни проблем* (енгл. *master problem*). На основу добијеног решења главног проблема одређују се вредности дуалних променљивих за свако од ограничења. Затим се, помоћу тих вредности, формирају потпроблеми чија вредност функције циља, у случају проблема минимизације, одговара смањењу вредности функције циља главног проблема. Решавањем потпроблема индентификује се променљива коју треба додати посматраном подскупу променњивих и поново решити главни проблем. Поступак се понавља све док постоје потпроблеми са негативном вредношћу функције циља, односно док постоји побољшање.

Метода гранања, оцењивања и одсецања (енгл. *Branch-and-Price-and-Cut - BnPnC*) укључује обе основне идеје које се јављају у BnC и BnP методама. Када се у сваком чвору дрвета реши релаксација методом генерисања колона, примењује се и метода одсецајућих равни како би се појачала релаксација проблема. Користе се и посебне технике које спречавају колизију ова два приступа како би методе BnC и BnP биле ефикасно интегрисане у BnPnC метод.

1.4 Метахеуристичке методе

Како егзактне методе у случају НП-тешких проблема и њихових инстанци већих димензија обично не могу да обезбеде оптимална решења у прихватљивом временском року, постоји потреба за коришћењем различитих приближних метода. Ове методе не гарантују оптималност, али дају решења доста доброг квалитета у задовољавајућем временском року. Постоје две класе приближних метода: апроксимативни алгоритми (енгл.approximation algorithms) и хеуристички алгоритми (енгл.heuristic algorithms). За разлику од хеуристика, код апроксимативних алгоритама могуће је евалуирати квалитет добијених решења са доказаном, односно гарантованом величином одступања од оптималног решења и израчунати границе рачунског времена.

Хеуристичке методе су методе које немају гарантовани квалитет добијенег приближног решења. Користе се за одређивање приближног решења конкретног проблема, засноване су на претпоставкама и техникама које воде ка добром решењу, али се у великој мери ослањају на специфичности самог проблема. Стога се оне могу применити улавном само на проблем за које су дизајниране, а не на ширу класу проблема оптимизације. Овај важан недостатак превазиђен је концептом метахеуристичких метода.

Метахеуристике су приближне методе на вишем, апстрактном нивоу и са општим правилима. Конципиране су тако да имају општу намену, па се могу применити на шири скуп проблема оптимизације. Приликом имплементације метахеуристике, потребно је прилагодити њене елементе конкретном проблему односно класи проблема, што је и главни задатак приликом дизајнирања успешне метахеуристичке методе.

Метахеуристике су настале средином прошлог века, а нарочито су популаризоване последње две деценије у бројним областима науке и инжењерства. Користе се за решавање проблема оптимизације у електроници, аеродинамици, динамици флуида, телекомуникацијама, роботици, симулацијама и идентификацијама у хемији, физици и биологији, у финансијама, логистици, транспорту, проблемима рутирања и распоређивања, менаџменту, економији итд. Њихова ефикасност је доказана решавањем сложених проблема оптимизације који потичу из готово свих области живота, а њихов развој се прати на бројним научним скуповима, конференцијама и у публикацијама светске истраживачке заједнице.

Бројне су класификације метахеуристичких метода. Имајући у виду различите критеријуме, метахеуристике могу бити:

- детерминистичке или стохастичке
- засноване на једном решењу или на популацији решења
- итеративне или конструктивне
- методе које памте информације током претраге или не
- инспирисане биолошким процесима или математички засноване.

Приликом истраживања простора решења, метахеуристике користе две основне стратегије *диверзификације* и *интензификације* претраге. Први принцип усмерен је ка посећивању делова простора решења који још увек није претражен, како претрага не би била фокусирана на само неколико области, што би водило завршетку алгоритма у локалном оптимуму. Интензификација или појачавање претраге односи се на детаљније истраживање обећавајућих делова простора решења, односно фаворизује се претрага у околини „добрих" решења. Приликом дизајнирања метахеуристичке методе за конкретни проблем, могуће је дати предност једном од ова два принципа. У екстремном случају, када се појачава диверзификација, метахеуристичка метода постаје претрага простора решења на потпуно случајан начин, док у супротном, при појачавању интензификације, метода се своди на обичну локалну претрагу, тј. трагање у околини иницијалног решења.

До сада је у литеаратури предложен велики број метахеуристичких метода и њихових модификација, а истраживање нових и модификација постојећих метахеуристика је процес који је још актуелан.

Метахеуристичке методе засноване на једном решењу

Метахеуристичке методе на бази једног решења (енгл. *Single-Solution Based Metaheuristics*) или S-метахеуристике у општем случају почињу процедуром која генерише иницијално решење. Затим се, кроз низ итерација, смењују две основне фазе: *фаза генерисања* и *фаза замене решења* све док се не задовољи критеријум заустављања. У фази генерисања формира се скуп кандидата $C(S)$ у околини текућег решења S, а у фази замене бира се кандидат $S' \in C(S)$ који

ће заменити текуће решење S. При томе је могуће памтити део претраге и користити добијене информације приликом избора новог решења. Постоји више различитих S-метахеуристика, а заједнички аспекти приликом њиховог дизајнирања су генерисање иницијалног решења и дефинисање структура околина које ће се користити током претраге простора решења.

Најпознатије S-метахеуристике су:

- Локално претраживање (енгл. *Local search - LS*)
- Симулирано каљење (енгл. *Simulated annealing - SA*)
- Табу претраживање (енгл. *Tabu search - TS*)
- Похлепна стохастичко-адаптивна процедура претраге (енгл. *Greedy randomised adaptive search procedure - GRASP*)
- Метода променљивих околина (енгл. *Variable neighborhood search - VNS*)

Локално претраживање је вероватно најједноставнија и најстарија метахеуристичка метода ([85], стр. 121). Сматра се да је Симплекс алгоритам који је предложио Dantzig 1947. године [26] прва варијанта локалне претраге. Локална претрага је једноставна процедура која полази од иницијалног решења и настоји да га побољша кроз низ локалних промена. При томе, свако ново решење које има бољу вредност функције циља у односу на текуће решење постаје ново текуће решење, све док се не пронађе локални оптимум. С обзиром да ново решење припада околини текућег решења, главни задатак је изабрати структуру околине која ће се користити и стратегију за претраживање дефинисане структуре околине. Три основне стратегије претраживања су:

- стратегија најбољег побољшања (енгл. *Best improvement strategy*)
- стратегија првог побољшања (енгл. *First improvement strategy*)
- стратегија случајаног избора (енгл. *Random selection*)

Првом стратегијом бира се најбоље решење из околине тренутног решења, што значи да је потребно претражити целу околину и евалуирати свако решење, а то је временски доста скупа операција. Стратегија првог побољшања прихвата прво решење које је боље од тренутног, што подразумева делимичну претрагу

околине тренутног решења, а у најгорем случају, када нема бољег суседа, претрагу комплетне околине. Код стратегије случајног избора, на случајан начин се бира једно од решења у околини текућег и прихвата као ново текуће решење, уколико је боље од тренутног.

Имајући у виду да је Локално претраживање метода која лако може завршити у локалном оптимуму, она није погодна у случају када простор решења има пуно различитих локалних оптимума. Из тог разлога, неопходно је развити методе засноване на локалном претраживању, а које имају механизме изласка из локалног оптимума.

Симулирано каљење је метахеуристичка метода, коју су предложили Kirkpatrick и сар. 1983. године [57]. Метода је заснована на принципима технолошког процеса каљења метала. Каљење је процес загревања супстанце до високе температуре, након којег се она постепено хлади при чему се добијају јаке кристалне структуре. Симулирано каљење користи аналогију физичког процеса и проблема оптимизације. Уведена је аналогија између оваквог физичког система и проблема оптимизације, по којој функција циља представља стање енергије система, решење одговара стању система, локални оптимум је еквивалент метастабилном стању, док глобални оптимум представља стање у коме систем има најмању енергију.

Симулирано каљење је стохастичка метода, заснована на локалном претраживању, а реализује се кроз низ итерација. У свакој итерацији генерише се решење из околине тренутног решења на случајан начин. Уколико је добијено решење боље од тренутног, прихвата се у потпуности и мења тренутно решење. У супротном, решење које је лошије од тренутног се прихвата са одређеном вероватноћом која зависи од тренутне температуре T и разлике $\triangle E$ између одговарајућих вредности функције циља случајно изабраног и тренутног решења. Вероватноће најчешће прате Болцманову (енгл. *Boltzmann*) дистрибуцију:

$$P(\triangle E, T) = e^{-\frac{\triangle E}{T}} \tag{1.12}$$

Како вредност контролног параметра T директно утиче на ток претраге, приликом дизајнирања алгоритма Симулираног каљења, неопходно је изабрати одговарајући распореда хлађења, односно скале по којој се посматрани параметар мења. Брзо смањивање параметра води ка брзој конвергенцији која може дати резултате лошијег квалитета, а сувише споро смањивање параметра T захтева пуно рачунског времена, па је потребно наћи адекватан начин промене

параметра током рада алгоритма који ће обезбедити компромис између квалитета решења и времена извршавања.

Табу претраживање је још једна метахеуристика заснована на локалном претраживању код које се механизам излажења из локалног оптимума састоји у прихватању лошијих решења када у посматраној околини не постоји побољшање. Ову метахеуристику предложио је Glover, 1986. године ([41]). За разлику од симулираног каљења, код табу претраживања околина текућег решења се претражује детерминистички. Када се достигне локални оптимум, односно када више нема побољшања у околини текућег решења, у следећем кораку бира се лошије решење. Оваква стратегија очигледно води ка појави циклуса током претраге, односно поново добијање решења која су већ била пронађена и настављање претраге идентичним путем. Да би се то спречило, уведен је концепт *табу листе*. Табу листе су меморијске структуре које „памте" карактеристике пронађених решења, односно путању претраге. Меморисање читаве претраге односно карктеристика свих посећених решења је превише захтевно, те је дужина табу листе ограничена. *Табу време* представља број итерација за време којих је решење које се налази у табу листи забрањено. Када се приликом претраживања околине текућег решења генерише неко решење, пре његове евалуације, проверава се да ли генерисано решење забрањено или не. Такође, после сваке итерације, табу листа се освежава додавањем карактеристика новог решења. У неким случајевима, могуће је, прихватити и решења која се налазе на табу листи, а задовољавају неки одређени услов. Тај корак се зове *Критеријум аспирације* и служи да умањи претерану рестрикцију коју табу листе могу произвести. Класичан критеријум аспирације који се користи је прихватање табу потеза онда када он води ка решењу које је боље од најбољег пронађеног. Према томе, у табу претраживању, допустиви су потези који нису забрањени или задовољавају критеријум аспирације.

Похлепна стохастичко-адаптивна процедура претраге је метахеуристичка метода са вишеструким покретањем (енгл. multi-start), која је такође заснована на локалном претраживању. GRASP су предложили Feo и Resende [32, 33] 1989. године. Свака итерација GRASP методе се састоји из две основне фазе: фаза похлепне стохастичке процедуре за конструкцију решења (енгл. *Greedy Randomised Construction - GRC*) и фазе локалне претраге (енгл. *Local Search - LS*). У GRC фази, формира се допустиво решење проблема чија се околина потом истражује у LS фази како би се пронашао локални оптимум.

Најбоље пронађено решење кроз све итерације представља решење добијено GRASP методом. Напредне верзије GRASP методе укључују адаптивне промене параметара у зависности од квалитета решења добијених у претходним итерацијама. Основна идеја је повећавање вероватноћа за избор оних вредности параметара које воде ка бољем решењу. Детаљније о GRASP методи објашњено је у поглављу 3.

Метода променљивих околина је метахеуристика заснована на локалном претраживању која се реализује систематским претраживањем унапред дефинисаног скупа околина. Претраживањем различитих околина алгоритам проналази различите локалне оптимуме и наставља претрагу излазећи из добијених оптимума. Дефинисане околине се претражују сукцесивно према утврђеном редоследу. Уколико се пронађе побољшање, оно се прихвата и претрага почиње од његове прве околине, у супротном прелази се на следећу околину. Метода су предложили Mladenović и Hansen 1997. године [71]. С обзиром да су метода променљивих околина и њене варијације изабране као методе решавања новог проблема распоређивања возила у овој докторској дисертацији, више детаља о овој методи налази се у поглављу 2.

Метахеуристике засноване на популацији решења

Метахеуристичке методе засноване на популацији решења (енгл. *Population Based Metaheuristics*) или P-метахеуристике су методе које користе скуп више решења који се мења током претраге водећи ка најбољем решењу. Полазећи од иницијалне популације, код P-метода се смењују две фазе: генерисање нове популације и замена текуће популације.

У првој фази формира се нова популација решења коришћењем једне од следеће две стратегије:

- стратегија заснована на принципима еволуције (енгл. *Evolutionary based*)
- стратегија преношења информација (енгл. *Blackboard based*).

У случају стратегије на бази еволуције врши се избор решења текуће популације и њихова репродукција коришћењем оператора мутације и рекомбинације који делују директно на репрезентацијама решења. Друга стратегија је заснована на памћењу својстава квалитетних решења која су током претраге прона-

ђена и преношењу информација о меморисаним својствима у циљу усмеравања претраге у правцу пожељнијих решења.

У другој фази P-метахеуристика врши се избор нове популације решења, што се постиже потпуном или делимичном заменом претходне популације новим решењима добијеним у фази генерисања.

Генетски алгоритми (енгл.*Genetic algorithm - GA*) припадају групи метахеуристичких метода које се базирају на популацији решења. Сматра се да их је увео Holland 1975. године [53]. GA се заснива на симулирању процеса природне еволуције једне популације јединки под дејством генетских оператора [83]. Иницијална популација се најчешће генерише на случајан начин, како би се обезбедила разноврсност генетског материјала. Свака јединка одговара неком решењу посматраног проблема и одговара јој низ карактера који представља њен *генетски код*. Елементи генетског кода називају се *гени* и могу имати вредности из одређеног скупа симбола који се назива *азбука* кодирања. Кодирање је изузетно важан корак GA и мора се пажљиво осмислити, с обзиром да неадекватан избор може довести до лоших резултата. Јединке се евалуирају коришћењем *функције прилагођености* која се дефинише на одређени начин и мери квалитет јединке тј. решења посматраног проблема. Почевши од иницијалне популације, добијају се нове генерације јединки помоћу генетских оператора: *селекције*, *укрштања* и *мутације*. Селекција јединки које чине нову популацију врши се на основу вредности функције прилагођености. На тај начин се издваја бољи генетски материјал. Оператор укрштања омогућава размену генетског кода две или више јединки - родитеља при чему се добијају нове једнинке - потомци. На крају, мутација представља промену дела генетског кода добијених јединки како би се појачала разноврсност при чему се са врло малим вероватноћама поједини симболи гентског кода јединке мењају другим симболима из азбуке кодирања. Генетски оператори се примењују у наведеном редоследу све док се не задовољи критеријум заустављања. Најчешће коришћени критеријуми су: достигнут максималан укупан број генерација, достигнут одређени унапред дат број генерација без побољшања решења, прекорачен лимит рачунског времена и сл.

Оптимизација ројевима честица (енгл.*Particle Swarm optimization - PSO*) је метахеуристика инспирисана интелигенцијом ројева, односно социјалним понашањем биолошких врста у заједницама као што су јата риба или птица и припада групи алгоритама који се једним именом зову *Алгоритми интели-*

генције ројева (енгл. *Swarm intelligance*). Предложили су је Kennedy и Eberhart 1995. године [55]. У основи PSO методе посматра се рој који садржи одређени број јединки. Свака јединка представља кандидата за решење посматраног проблема оптимизације и представљена је вектором одређене димензије у простору решења. Јединке имају промењљиве позиције и брзине при истраживању простора решења. Оне комуницирају и сарађују, што значи да успех једне јединке утиче на претрагу друге. Јединке мењају позиције и крећу се ка оптимуму, користећи следеће две информације: своју најбољу позицију, тј. своје најбоље пронађено решење и најбоље пронађено решење целог роја. При дизајнирању PSO методе, потребно је дефинисати околину сваке јединке која представља област социјалног утицаја других јединки на посматрану. У том циљу најчешће се користе две методе: метода глобалног побољшања (global best method - gbest) и метода локалног побољшања (local best method - lbest). Код прве методе околину чини цела популација, а при примени друге методе, околину неке јединке чине само оне јединке које су директно повезане са њом у придруженој топологији. У свакој итерацији PSO методе се, на основу најбољих решења у околини посматране јединке као и тренутне позиције и брзине, освежава брзина за сваку јединку по одређеној формули. Коришћењем израчунате брзине одређује се нова позиција јединке, која се после евалуације пореди са најбоље пронађеним решењем посматране јединке и целог роја. Уколико је боља, постаје ново најбоље решење. Итерације се смењују док се не задовољи критеријум заустављања.

Оптимизација мрављим колонијама (енгл.*Ant colony optimization - ACO*) представља методу инспирисану природним појавама из групе алгоритама интелигенције ројева. Инспирисана је колективним понашањем мрава, који користе једноставне механизме комуникације, како би заједно пронашли најкраћи пут до хране и транспортовали је до мравињака. Познато је да су мрави јединке слабог вида које чулом мириса бирају пут. Током транспорта, мрави остављају траг на путу, помоћу *феромона*, испарљиве супстанце коју луче. Овај траг води јединке ка извору хране, а количина супстанце која се налази на неком путу одређује вероватноћу избора посматраног пута. Алгоритам се састоји из два основна корака: *конструкција решења* и *освежавање феромонских трагова*. Конструкција решења се одвија за сваког мрава у виду похлепне процедуре која додаје једну по једну компоненту решења, поштујући вероватноће избора добијене на основу феромонског трага, који меморише ка-

рактеристике „добрих" решења. Други корак се односи на промену количине феромона на основу квалитета добијених решења и укључује две фазе: испаравање и појачавање. Испаравање се одвија аутоматски смањивањем нивоа феромона неком унапред одређеном стопом, док се појачавање феромона регулише у зависности од квалитета добијених решења помоћу различитих стратегија.

Оптимизација колонијом пчела (енгл.*Bee colony optimization - BCO*) је још једна од метода инспирисана интелигенцијом роја при обављању заједничких послова. Метода је предложена 2001. године првобитно под називом *Систем пчела* (енгл.*Bee system*) [63], а потом 2005. године под својим садашњим именом [86].

У заједници пчела постоји подела рада и свака пчела је специјализована за обављање одређене функције. Алгоритам на бази понашања пчела заснива се на следећем природним законитостима у колонији. Неколико пчела извиђача креће у потрагу за храном у близини кошнице. По повратку, у зависности од количине и квалитета пронађене хране, имају три могућности:

- да постану регрутери (обавесте друге пчеле о локацији, количини и квалитету нектара);
- да остану лојалне (настављају скупљање са исте локације, али не регрутују друге пчеле да их следе);
- да постану неопредељене (одустају од извора и чекају да буду регрутоване).

Регрутери износе информације о пронађеном нектару на плесном подијуму изводећи чувени плес у облику броја 8, познат као *Waggle dance*. Трајање једног плеса индицира удаљеност локације, угао под којим је плес усмерен у односу на Сунце показује правац у којем треба летети, а такође плесом се сазнаје количина и квалитет нектара. Неопредељене пчеле користе те информације да одаберу локацију на коју ће кренути да сакупљају нектар. Према томе, све одлуке зависе од квалитета и количине пронађене хране.

Метода оптимизације инспирисана понашењем пчела у порази за храном је метода на бази популације решења. Алгоритам користи сличност између начина на који пчеле врше потрагу за храном и начина на који алгоритми оптимизације врше претрагу простора решења [28]. Главни кораци BCO алгоритма су: *потрага за храном* и *плес*. У првом кораку врши се генерисање решења, а

у другом размена информација у циљу евалуирања добијених решења и усмеравања претраге у правцу решења бољег квалитета.

Осим наведених, постоје и друге метахеуристике инспирисане биолошким појавама и процесима, као што су понашање слепих мишева (енгл. *Bat-Inspired Optimization*) [94], кукавица (енгл. *Cuckoo Search Algorithm*) [95] свитаца (енгл. *Firefly Algorithm*) [93, 35], чапљи (енгл. *Green Heron Optimization Algorithm*) [84] или слонова (енгл. *Elephant Herding Optimization*) [88]. Такође, метода која се заснива на принципима биогеографије (енгл. *Biogeography-Based Optimization*) [96] успешно је коришћена за решавање проблема оптимизације.

У литератури се могу пронаћи и бројне хибридне метахеуристике. Могућности за хибридизацију су врло разноврсне. Метахеуристике се комбинују са другим метахеуристикама, најчешће са онима које имају комплементарна својства, па у комбинацији дају боље резултате. Такође, метахеуристике се успешно хибридизују са егзактним методама, као и са методама области вештачке интелигенције (енгл. artfical inteligence) а посебно са методама истраживања података (енгл. data mining) или машинског учења (енгл. machine learning). Хибридизацијом се често добијају методе које дају боље резултате од оних који су добијени применом само једне метахеуристичке методе. Детаљније о метахеуристикама, њиховим компонентама, карактеристикама и применама може се наћи у [85] или [43].

Глава 2

Метода променљивих околина

2.1 Опште карактеристике

Метода променљивих околина (енгл. *Variable neighborhood search-VNS*) је ефикасна метахеуристичка метода, која се већ две деценије успешно користи за решавање проблема комбинаторне и глобалне оптимизације. Основне поставке ове методе изнели су Младеновић и Hansen, 1997 године у свом раду [71]. Аутори VNS методе су приметили да, у случају неколико познатих проблема комбинаторне оптимизације, промена околина приликом локалне претраге простора свих решења омогућава добијање бољих резултата него при примени дотадашњих метода које нису користиле такав приступ (симулирано каљење, табу претрага). У раду [71] је разматран проблем трговачког путника (енгл. *Traveling Salesman Problem - TSP*) и први пут предложена метода VNS за његово решавање. Успешна примена методе VNS на TSP водила је ка закључку да би се метода променљивих околина уз адекватне модификације могла успешно применити и на друге проблеме распоређивања и рутирања. Већ 2001. године, Младеновић и Hansen у раду [46] представљају неколико модификација опште методе променљивих околина и неколико примера успешне примене за решавање проблема комбинаторне оптимизације.

Суштина методе променљивих околина је сукцесивно, систематско истраживање скупа унапред дефинисаних околина, како би се током претраге добили различити локални оптимуми. Промена околине такође омогућава излаз из локалних оптимума, што води ка претрази других области простора решења. VNS се ослања на три једноставне чињенице:

1. Локални оптимум у дносу на једну околину не мора бити локални оптимум у односу на неку другу,

2. Глобални оптимум је локални оптимум у односу на сваку околину,

3. За већину проблема локални оптимуми у односу на разне околине су релативно близу.

Систематска претрага околина се врши све док се не задовољи неки критеријум заустављања. Најчешће коришћени критеријуми заустављања су: максимално процесорско време рада, максималан број итерација, минимално побољшање у оквиру унапред дефинисаног броја итерација, максималан број итерација између два побољшања и други. Генерално, избор критеријума заустављања би требало да обезбеди компромис између квалитета добијених решења и утрошеног времена. Најчешће се одређује експерименталним путем, анализирањем понашања алгоритма на репрезентативном скупу инстанци.

До данас је у литератури предложено више варијанти методе променљивих околина:

- Основна метода променљивих околина (енгл. *Basic VNS - BVNS*)
- Метода променљивог спуста (енгл. *Variable neighborhood descent - VND*),
- Општа метода променљивих околина (енгл. *General VNS - GVNS*),
- Редукована метода променљивих околина (енгл. *Reduced VNS - RVNS*),
- Адаптивна метода променљивих околина (енгл. *Skewed VNS - SVNS*),
- Метода променљивих околина на бази декомпозиције (енгл. *Variable neighborhood decomposition search - VNDS*),
- Примално-дуална метода променљивих околина (енгл. *Primal-dual VNS - PD-VNS*),
- Паралелна метода променљивих околина (енгл. *Parallel VNS - PVNS*),
- Метода променљивих формулација (енгл. *Variable Neighborhood Formulation Space Search - VNFSS*).

и друге.

Различите варијанте VNS методе су веома популарне у многим областима математике и инжењерства уколико задовољавају већину од следећих особина [48]:

- *једноставност* - једноставан и јасан принцип који има широку примену,
- *кохерентност* - кораци су повезани и имају природан след, пратећи основни принцип,
- *ефикасност* - омогућава добијања оптималних решења или решења блиских оптималним на инстанцама разматраног проблема,
- *ефективност* - добијају се оптимална или висококвалитетна решења у прихватљивом временском року, чак и у случају инстанци проблема великих димензија,
- *робустност* - може се адаптирати за разне проблеме оптимизације тако да за сваки конкретан проблем даје квалитетна решења на већем скупу инстанци, а не само на посебно изабраним инстанцама,
- *прилагођеност корисницима* - једноставна, разумљива и лако применљива метода са само неколико параметара (неретко и са само једним параметром), чију вредност треба одредити за сваки конкретан проблем,
- *иновативност* - постоји широк спектар могућности за примене на нове проблеме или класе проблема, модификације и хибридизацију са другим методама, егзактним и хеуристичким.

2.2 Основна метода променљивих околина

Основна метода променљивих околина се састоји у претраживању простора решења користећи једну или више различитих структура околина, кроз две основне фазе: фазе *размрдавања* (енгл. *Shaking phase*) и фазе *локалне претраге* (енгл. *Local search phase*). Ови кораци осликавају два супротстављена принципа претраживања допустивог скупа решења: диверзификације и интензификације. Могуће је дати предност једном од ова два принципа појачавањем рада алгоритма у одговарајућој фази. Ако се алгоритму дозволи да више ради

на фази размрдавања, тада је процес диверсификације појачан, што у граничном случају води ка претраживању простора решења на потпуно случајан начин. Ако се појача друга фаза, алгоритам се приближава локалној претрази, односно тражењу решења у околини иницијалног решења. Потребно је направити добар компромис између фазе размрдавања и локалне претраге који ће водити алгоритам ка квалитетним решењима у кратком времену извршавања.

За примену Основне методе променљивих околина најпре је потребно дефинисати низ структура околина N_r, $r = 1, \ldots, r_{max}$, које ће се користити у фази размрдавања и низ околина N_l ,$l = 1, \ldots, l_{max}$ које ће бити употребљене током локалне претраге. Ова два низа могу бити идентична или делити само неке чланове. Могуће је и да се потпуно различите структуре користе у фазама размрдавања и локалне претраге. У сваком случају, веома је важно да структуре околина буду дефинисане у складу са карактеристикама проблема који се разматра. Неадекватан избор околина по правилу води ка лошим перформансама алгоритма. Алгоритам почиње генерисањем почетног решења S које постаје текуће решење и наставља фазом размрдавања у првој околини N_1 из одговарајућег низа околина. У фази размрдавања бира се, на случајан начин, решење у околини тренутног решења. Улога фазе размрдавања је да помогне алгоритму да избегне конвергенцију ка локалном оптимуму. У посматраној околини N_r, текућег решења S, бира се, на случајан начин, решење S'. Тиме је фаза размрдавања завршена. Затим се, локалном претрагом покушава побољшати S', при чему се добија ново решење S'', као резултат локалне претраге. Ако се не пронађе побољшање тренутног решења, односно ако S'' није боље од S, прелази се на следећу околину N_{r+1}. У супротном, када је $f(S'') < f(S)$, где је са $f(S)$ означена вредност функције циља посматраног решења S, уз претпоставку да се алгоритам односи на проблем минимизације, ново побољшано решење постаје текуће, а претрага креће поново од околине N_1. Ови кораци се понављају док се не задовољи критеријум заустављања. Структура Основне методе променљивих околина представљена је Алгоритмом 1.

2.3 Метода променљивог спуста

Метода променљивог спуста је детерминистичка варијанта методе променљивих околина. Полази се од иницијалног решења и дефинише се низ структура околина N_l, $l = 1, \ldots, l_{max}$, које се претражују. Околине се користе сукце-

Алгоритам 1 Основна метода променљивих околина

```
1: procedure BVNS
2:     Generate initial solution S;
3:     repeat
4:         r ← 1;
5:         while r ≤ r_max do
6:             S' ← Shaking(S, r);
7:             S'' ← Local search(S');
8:             if f(S'') < f(S) then
9:                 S ← S'';
10:                r ← 1;
11:            else
12:                r ← r + 1;
13:    until Stopping criterion satisfied
       return Best solution found
```

сивно при спуштању ка локалном оптимуму. Уколико у текућој околини није пронађено побољшање, прелази се на нову, у супротном, добијено побољшано решење замењује тренутно решење и претрага креће од прве околине. Ова стратегија је ефикасна ако су околине коплементарне у смислу да локални оптимум у једној околини неће бити и локални оптимум у некој другој околини. Алгоритмом 2 представљени су основни кораци методе променљивог спуста.

Алгоритам 2 Метода променљивог спуста

```
1: procedure VND
2:     Generate initial solution S;
3:     l ← 1;
4:     while l ≤ l_max do
5:         Find the best neighbor S' ∈ N_l(S);
6:         if f(S') < f(S) then
7:             S ← S';
8:             l ← 1;
9:         else
10:            l ← l + 1;
       return Best solution found
```

У литератури је предложена и VND метода са вишеструким покретањем (енгл. Multi-start VND method) [64]. У питању је Метода променљивог спуста која се понавља све док се не задовољи критеријум заустављања и при сваком понављању полази се од случајно генерисаног иницијалног решења.

2.4 Општа метода променљивих околина

Разлика између Опште методе променљивих околина и Основне методе променљивих околина је у фази локалног претраживања, у којој се врши интензификација претраге. Општа метода променљивих околина користи Методу променљивог спуста. Стога је потребно дефинисати два скупа околина: околине N_r, $r = 1, \ldots, r_{max}$, које ће бити претраживане у фази размрдавања и околине N_l, $l = 1, \ldots, l_{max}$, које ће бити претраживане у оквиру Методе променљивог спуста. Могуће је да се исте структуре околина користе у фази размрдавања и фази локалне претраге. Притом та два скупа околина могу да буду идентична или да имају заједничке само неке елементе. Алгоритам 3 представља основну структуру Опште методе променљивих околина.

Алгоритам 3 Општа метода променљивих околина

```
1: procedure GVNS
2:     Generate initial solution S;
3:     repeat
4:         r ← 1;
5:         while r ≤ r_max do
6:             S' ← Shaking(S, r);
7:             S'' ← VND(S');
8:             if f(S'') < f(S) then
9:                 S ← S'';
10:                r ← 1;
11:            else
12:                r ← r + 1;
13:    until Stopping criterion satisfied
       return Best solution found
```

2.5 Редукована метода променљивих околина

Фаза локалне претраге код претходних варијанти методе променљивих околина је често врло захтевна у погледу времена извршавања, односно потребно је уложити доста труда за претраживање околине посматраног решења, посебно приликом решавања инстанци проблема већих димензија. Из потребе за добијањем решења за мање утрошеног рачунског времена, настала је варијанта методе променљивих околина која потпуно искључује фазу локалне претраге. Тако добијена метода се назива Редукована метода променљивих околина (енгл.

Reduced VNS - RVNS). Показала се успешном при решавању инстанци проблема великих димензија, код којих је генерално локална претрага изузетно захтеван део алгоритма у погледу времена извршавања [49]. Основни кораци Редуковане методе променљивих околина представљени су Алгортимом 4.

Алгоритам 4 Редукована метода променљивих околина

```
1: procedure RVNS
2:     Generate initial solution S;
3:     repeat
4:         r ← 1;
5:         while r ≤ r_max do
6:             S' ← Shaking(S, r);
7:             if f(S') < f(S) then
8:                 S ← S';
9:                 r ← 1;
10:            else
11:                r ← r + 1;
12:    until Stopping criterion satisfied
       return Best solution found
```

2.6 Адаптивна метода променљивих околина

Адаптивна метода променљивих околина омогућава претраживање области које су удаљене од тренутног решења. Уколико алгоритам води ка областима простора решења која су неограничено далеко од тренутног решења, метода се своди на Методу вишеструког покретања (енгл. *Multistart method*) у којој се сваки пут претражује околина новог случајно изабраног решења, а познато је да таква метода, генерално, није увек ефикасна. Због тога се настоји да растојање између тренутног решења и области која се укључује у претрагу буде контролисано. Прецизније, прихвата се прво побољшање S'' размрданог решења S' чак и ако је горе од полазног решења S, под условом да разлика између вредности функција циља решења S'' и S, не буде већа од производа реалног параметра α и растојања $\rho(S'', S)$ између посматраних решења. Параметар α се експериментално одређује за сваки проблем који се разматра. Процедура *Keep Best* (S, S') једноставно бира боље од два решења, односно ако је S' бољи од S тада је $S \leftarrow S'$. Основни концепт Адаптивне методе променљивих околина представљен је Алгоритмом 5.

Алгоритам 5 Адаптивна метода променљивих околина

procedure SVNS
  Generate initial solution S;
  repeat
    $r \leftarrow 1$;
    $S_{best} \leftarrow S$;
    while $r \leq r_{max}$ **do**
      $S^{'} \leftarrow Shaking(S, r)$;
      $S^{''} \leftarrow$ *First improvement*$(S^{'})$;
      Keep Best (S_{best}, S);
      if $f(S^{''}) - \alpha\rho(S^{''}, S) < f(S)$ **then**
        $S \leftarrow S^{''}$;
        $r \leftarrow 1$;
      else
        $r \leftarrow r + 1$;
    $S \leftarrow S_{best}$;
  until Stopping criterion satisfied
  return Best solution found

2.7 Преглед примена метода променљивих околина на проблеме распоређивања и рутирања возила

У литератури постоји велики број примера успешне примене разних варијанти Методе променљивих околина на различите проблеме распоређивања и рутирања возила. Идеје које су изложене у тим радовима су значајне и са теоријског и са практичног аспекта. Од избора структура околина које се користе у фазама размрдавања и локалној претрази, вредности параметара и критеријума заустављања, зависе и перформансе примењене методе на конкретан проблем. Примена Методе променљивих околина за нови проблем који до сада није разматран у литератури или решаван VNS методом, захтева прилагођавање неких или свих елемената VNS методе датом проблему. Некада је могуће користити делове постојећих VNS имплементација за сличне проблеме, а нередко је неопходно адаптирати све делове VNS методе датом проблему, пратећи концепт одабране варијанте методе променљивих околина.

Стога је дизајнирање одговарајуће варијанте Методе променљивих околина за решавање конкретног проблема оптимизације сложен процес који се поред теоријских основа ослања на емпиријска тестирања великог броја инстанци,

како би се искристалисао најбољи приступ. У наставку су наведене неке од примена методе променљивих околина на проблеме распоређивања и рутирања возила.

Kytöjoki и сар. [59] су развили ефикасну методу променљивих околина која може бити примењена на широк спектар реалних проблема рутирања возила. Резултати добијени VNS методом предложеном у [59] су упоредиви са резултатима до тада познатих метода из литературе.

Метода променљивих околина је примењена на Периодични проблем рутирања возила (енгл. *Periodic Vehicle Routing Problem - PVRP*) без временских оквира (енгл. *Time Windows - TW*) у раду [50]. PVRP представља проширену варијанту класичног проблема рутирања возила (енгл. *Vehicle Routing Problem - VRP*). Код PVRP, посматра се период од неколико дана током којег се сваки од клијената мора опслужити одређени број пута, при чему постоји флексибилност при збору дана када ће локације бити посећене (распоред опслуживања није унапред дефинисан). Експериментални резултати приказани у раду [50] показују да VNS метода за решавање PVRP надмашује постојеће методе решавања посматраног проблема.

Fleszar и сар. [37] су разматрали Отворени проблем рутирања возила (енгл. *Open Vehicle Routing Problem - OVRP*) који за циљ има минимизацију броја коришћених возила, а затим укупно пређено растојање или укупно утрошено време транспорта. Тако формулисан циљ има за последицу да су трошкови укључивања додатног возила већи од уштеде која се добија смањењем укупног утрошеног времена за транспорт. Код проблема OVRP, у свакој рути возило полази из депоа, а завршава на некој од локација на којима се налазе клијенти, дакле возила се не враћају назад у депо. За разматрани проблем предложена је метода променљивих околина код које су структуре околина дефинисане потезима преусмеравања рута и размењивања сегмената рута. Резултати су показали да је VNS компаративан са најбољим дотадашљим методама решавања OVRP.

Chan и Liu [62] су развили методу променљивих околина за решавање проблема рутирања и распоређивања инвентара (енгл. *Inventory Routing and Scheduling Problem - IRSP*) у ланцима снабдевања. У раду [62] проблем рутирања и распоређивања возила су интегрисани једним моделом, а предложена метода променљивих околина се показала ефикаснијом од претходних метода решавања у смислу квалитета добијених решења, с обзиром да је остварено смањење

трошкова на дневном нивоу.

Khouadjia и сар. [56] су разматрали проблем рутирања возила са динамичким захтевима (енгл. *Dynamic Vehicle Routing Problem - DVRP*). Реч је о проблему рутирања код кога је могуће да током радног дана дође до нових захтева који изискују промену планираних рута. За решавање овог проблема, у раду [56] су предложене две хеуристике: метода променљивих околина и метода оптимизацијеројем честица (PSO). Коришћени су нови и познати тест примери из литературе и анализирани су резултати добијени применом обе хеуристике. Закључено је да VNS у просеку прави мање руте од PSO, док PSO даје решења која до тада нису била позната у литератури. VNS се показала значајно успешнијом од PSO при решавању тест инстанци већих димензија, у смислу квалитета добијених решења и времена извршавања, док је PSO дала решења бољег квалитета на инстанцама мањих димензија, али је спорија у односу на VNS методу. Општи закључак аутора рада [56] је да VNS метода даје квалитетнија решења од PSO за мање рачунског времена, док је PSO стабилнија од VNS.

De Armas и Melian-Batista [29] су развили VNS хеуристику за динамички проблем рутирања возила са временским оквирима (енгл. *Dynamic Rich Vehicle Routing Problem with Time Windows - DRVRPTW*). Као код DVRP, овај проблем укључује динамичко мењање захтева које усмерава претрагу простора решења током извршавања. Сем тога, код DRVRPTW су постављени и временски оквири у којима се клијенти могу опслужити без плаћања пенала. Резултати приказани у [29] указују да је метода променљивих околина врло ефикасна у поређењу са претходним резултатима из литературе, а развијени софтвер заснован на VNS методи је имплементиран у транспортном систему компаније у Шпанији.

Адаптивну методу променљивих околина (SVNS) применили су Macedo и сар. [64] решавајући интегрисани локацијски проблем рутирања (енгл. *Location Routing Problem - LRP*), који се састоји у избору локација за отварање депоа и генерисању низа рута из отворених депоа, како би се опслужио скуп клијената са минималним укупним трошковима. У разматраном проблему, претпоставка је да возило може да направи и више од једне руте током задатог периода. SVNS метода се показала ефикасном при решавању датог проблема: достигла је оптимална решења на значајном броју инстанци и надмашила *Multi-start VND* методу, предложену у истом раду, која користи исте структуре околина.

Cheikh и сар. [21] су применили VNS методу за проблем рутирања возила са вишеструким путовањима (енгл. *Vehicle Routing Problem with Multiple Trips - VRPMT*). VRPMT је варијанта проблема рутирања возила која дозвољава да се возила користе више пута током радног времена. Коришћене су четири структуре околина и алгоритам је тестиран на тест инстанцама из литературе. Применом методе променљивих околина остварени су бољи резултати од постојећих на истим инстанцама.

Методу променљивих околина на два нивоа развили су Wassan и сар. [90] за нову варијанту проблема рутирања возила са вишеструким путовањима и транспортом при повратку (енгл. *Multiple Trip Vehicle Routing Problem with Backhauls - MT-VRPB*). У разматраном проблему рутирања, возила праве више рута у току радног времена, а по завршеној испоруци на некој од локација могу да преузму терет који враћају назад у депо. Предложена метода је достигла сва најбоља позната решења на скупу тестираних инстанци из литературе са прилично малим временом извршавања и показала се адекватним приступом за решавање разматраног проблема.

Bula и сар. [15] су применили методу променљивих околина за решавање проблема рутирања хетерогених возила (енгл. *Heterogeneous Fleet Vehicle Routing Problem - HFVRP*) при транспорту опасних материјала. У предложеном моделу, функција циља минимизује очекивани ризик, а специфична ограничења се односе на изложеност становништва уколико дође до инцидента. Перформансе методе променљивих околина су додатно побољшане процедуром која је заснована на проблему партиције скупа (енгл. *Set Partitioning Problem - SP*). Прецизније, решења добијена у фази локалне претраге смештају се у скуп из којег се, решавањем SP проблема, бира комбинација рута која минимизује ризик.

Bortfeldt и сар. [14] су разматрали интегрисани проблем рутирања и паковања тродимензионалних предмета, под називом проблем рутирања са груписаним теретом при повратку и 3D ограничењима утовара (енгл. *VRP with clustered backhauls and 3D loading constraints - 3L-VRPCB*). Аутори су у [14] предложили два хибридна алгоритма, при чему се сваки од њих састоји из два корака: процедуре рутирања и паковања. Процедуре рутирања у предложеним алгоритмима су различите, једна је заснована на методи великих околина, а друга на методи променљивих околина, док је процедура паковања иста у обе методе и користи алгоритам претраге дрвета. Обе предложене методе дале су

обећавајуће резултате при решавању инстанци проблема 3L-VRPCB из литературе.

Представљени примери успешне примене методе променљивих околина на разне варијанте проблема распоређивања и рутирања возила, били су мотивација за дизајнирање и имплементацију неколико варијанти VNS методе за решавање реалног проблема распоређивања возила при оптимизацији транспорта пољопривредних сировина, који је предмет ове дисертације.

Глава 3

Похлепна стохастичко-адаптивна процедура претраге

3.1 Опште карактеристике

Похлепна стохастичко-адаптивна процедура претраге (енгл. *Greedy Randomised Adaptive Search Procedure - GRASP*) GRASP представља метахеуристичку методу чије су основне поставке изнели Feo и Resende [32, 33]. Ова метода се већ преко две деценије успешно користи за решавање различитих комбинаторних проблема, укључујући и проблеме рутирања и распоређивања возила. Преглед примена GRASP методе на разне проблеме оптимизације може се наћи у [34] и [79]. Прву примену GRASP методе предложили су управо њени аутори, Feo и Resende [32, 33], за решавања проблема покривања скупа (енгл. *Set Covering Problem*).

Свака итерација GRASP методе састоји се из две основне фазе: фаза похлепне стохастичке процедуре за конструкцију решења (енгл. *Greedy Randomised Construction - GRC*) и фазе локалне претраге (енгл. *Local Search - LS*). Улога GRC фазе је генерисање допустивог решења за разматрани проблем, док фаза локалне претраге истражује околину решења конструисаног у GRC фази у циљу пронашалажења локалног оптимума. Итерације се смењују све док се не задовољи критеријум заустављања. Постоји више могућности за избор критеријума заустављања: достигнут унапред задати максималан број итерација, достигнуто задато максимално време извршавања, достигнут одређени квалитет решења, итд. Најбоље пронађено решење кроз све GRASP итерације сматра се најбољим решењем добијеним GRASP методом. Основни кораци GRASP ме-

тоде представљени су Алгоритмом 6.

Алгоритам 6 Похлепна стохастичко-адаптивна процедура претраге

```
procedure GRASP(ProblemData, n_max, Seed)
    f(S_best) ← +∞;
    repeat
        S ← GRC(Seed);
        S ← LocalSearch(S);
        if f(S) < f(S_best) then
            S_best ← S;
    until Stopping criterion satisfied
     return S_best;
```

У GRC фази додаје се елемент по елемент парцијалном решењу, све док се не генерише комплетно решење. Елементи који се додају парцијалном решењу, изабрани су из тзв. *листе кандидата* (енгл. *candidate list - CL*). Листа кандидата се формира на основу вредности похлепне функције оцењивања (енгл. *greedy evaluation function*), која одражава промене у вредности функције циља при додавању сваког конкретног елемента из CL парцијалном решењу. Из листе кандидата се издваја подскуп који се назива *редукована листа кандидата* (енгл. *restrictive candidat list - RCL*). RCL чине они елементи из CL који имају нека унапред дефинисана позитивна својства, те се очекује да ће избор тих елемената водити ка квалитетном решењу. Пробабилистички аспект GRASP методе састоји се у избору кандидата из RCL који се врши на потпуно случајан начин, тј. изабрани кандидат не мора обавезно бити и најбољи. Алгоритам 7 приказује основне кораке GRC фазе.

Алгоритам 7 Похлепна стохастичка процедура за конструкцију решења

```
procedure GRC(Seed)
    S ← ∅;
    Evaluate the incremental costs of the candidate elements;
    while S is not complete do
        Build the restricted candidate list (RCL);
        Select an element j ∈ RCL;
        S ← S ∪ j;
        Reevaluate the incremental costs;
    return S;
```

При формирању редуковане листе кандидата, сваком елементу листе кандидата $j \in CL$ придружује се вредност похлепне функције оцењивања $ic(j)$, која

се назива *трошак повећања* (енгл. *incremental cost*). Ако су ic_{min} и ic_{max} редом најмањи и највећи трошак повећања на скупу свих елемената листе кандидата, односно: $ic_{min} = \min\{ic(j) \mid j \in CL\}$, $ic_{max} = \max\{ic(j) \mid j \in CL\}$, тада се редукована листа кандидата из оних елемената листе CL чији трошкови повећања нису већи од $ic_{min} + \alpha(ic_{max} - ic_{min})$, где је $\alpha \in [0, 1]$ реални параметар. Прецизније, за дату листу кандидата CL, редукована листа кандидата RCL се формира на следећи начин:

$$RCL = \{j \in CL \mid ic(j) \leq ic_{min} + \alpha(ic_{max} - ic_{min})\} \tag{3.1}$$

Уколико параметар α има вредност 0, GRC се своди на класични похлепни алгоритам, а у случају када је $\alpha = 1$ конструкција решења се врши на потпуно случајан начин. Дакле, неопходно је направити адекватан избор вредности за параметар α како би се постигао компромис између ових крајњих случајева.

Решење добијено у GRC фази представља иницијално решење за фазу локалне претраге. У овој фази истражује се његова околина и као резултат враћа се пронађени локални оптимум. Основни кораци фазе локалне претраге представљени су Алгоритмом 8.

Алгоритам 8 Локална претрага

```
procedure LS (Neighborhood structure N, Initial solution S)
    while S is not local optimum do
        Find S' ∈ N(S) with f(S') < f(S);
        S ← S';
    return S;
```

Два основна параметра у GRASP методи су: параметар који дефинише критеријум заустављања (максималан број итерација, максимално задато време извршавања, задати квалитет добијеног решења итд.) и параметар α који се користи при формирању RCL.

3.2 Проширења основног концепта GRASP методе

Поред основног концепта GRASP методе, постоје и њене модификације које се успешно користе за решавање различитих проблема оптимизације, као и

бројне хибридне методе у којима се GRASP метода комбинује са другим метахеуристикама.

Важну улогу у GRC фази има параметар α. У зависности од начина одабира параметра α током GRASP итерација, постоји неколико варијанти ове методе. Код основног концепта GRASP-а, параметар α има фиксну вредност из интервала $[0, 1]$ или може бити биран на случајан начин из неког дискретног подскупа $\{\alpha_1, \alpha_2, \ldots, \alpha_\nu\}$ посматраног интервала са одређеном дистрибуцијом вероватноћа. Prais и Ribeiro су 2000. године [76] предложили Реактивну похлепну стохастичко-адаптивну процедуру претраге (енгл. *Reactive Greedy Randomised Adaptive Search Procedure - Reactive GRASP*). Код овог концепта GRASP методе, параметар α има својство самоподешавања на основу квалитета добијених решења. Наиме, за сваку од ν дозвољених вредности α_i за $i = 1, \ldots, \nu$, одређује се вероватноћа њеног избора (p_i). Иницијално све вероватноће имају исте вредности, тј. $p_i = \frac{1}{\nu}$ за $i = 1, \ldots, \nu$. Нека је f^* вредност функције циља тренутног решења S^* и нека је f_i просечна вредност свих решења добијених GRASP методом коришћењем вредности параметра $\alpha = \alpha_i$ за $i = 1, \ldots, \nu$. Вероватноће p_i избора вредности α_i се периодично (после извесног броја GRASP итерација) прерачунавају по формули $p_i = q_i / \sum_{j=1}^{\nu} q_j$, где је $q_i = f^*/f_i$ за $i = 1, \ldots, \nu$. На овај начин, повећава се вероватноћа избора оних вредности α_i за које се добијају решења бољег квалитета. Реактивна GRASP метода представља побољшање основног концепта GRASP методе које води ка решењима бољег квалитета захваљујући самоподешавању парметра α.

У случају када алгоритам за конструкцију решења у оквиру основног концепта GRASP методе није довољно осетљив на случајан избор елемената, у литератури је предложена модификација GRASP методе која укључује тзв. пертурбацију трошкова (енгл. *cost perturbation*). Основна идеја оваквог приступа је увођење неке врсте „пометње" у скуп вредности који одговарају почетним трошковима, по угледу на методу зашумљавања (енгл. *Noising method*), коју су предложили Charon и Hudry [19]. За разлику од већине метахеуристичких метода које садрже у себи спуст ка локалном оптимуму, метода зашумљавања при посматрању произвољног решења додаје и потез пертурбације тог решења на случајан начин. При том се поставља услов да је разлика између вредности функције циља решења добијеног пертурбацијом и вредности функције циља посматраног решења, величина која се налази у интервалу $[-r, +r]$. Вредност параметра r се константно смањује како алгоритам напредује, па је потребно

дефинисати његову почетну вредност, као и начин (стопу) смањења. Како се пертурбације врше на случајан начин, при примени методе зашумљавања нема гаранције за добијање локалног оптимума, па се она често комбинује са другим методама. Више о овој методи може се наћи у [19] и [20].

Код основног концепта GRASP методе, у фази конструкције решења, сви елементи из RCL имају једнаке вероватноће избора. Међутим, при избору кандидата из RCL могу се увести функције пристрасности (енгл. *bias functions*). За дефинисање вероватноћа за избор кандидата посматране листе, може се користити било која дистрибуција вероватноће. Кандидати се рангирају према вредности похлепне функције. Ако је $r(e)$ ранг елемента e и $bias(r(e))$ вредност унапред дефинисане функције пристрасности која одговара рангу елемента e, тада се вероватноћа избора елемента e рачуна према формули $p(e) = bias(r(e))/\sum_{e' \in RCL} bias(r(e'))$. До сада је у литератури коришћено више функција пристрасности, а неке од њих су:

- функција пристрасности заснована на случајаном избору: $bias(r) = 1$,
- линеарна функција пристрасности: $bias(r) = 1/r$,
- логаритамска функција пристрасности: $bias(r) = log^{-1}(r+1)$,
- експоненцијална функција пристрасности: $bias(r) = e^{-r}$,
- полиномијална функција пристрасности реда n: $bias(r) = r^{-n}$.

Још једна модификација GRASP методе је метода поновног повезивања стаза (енгл. *Path relinking - GRASP-PR*), коју је 1997. године предложио Glover [42], 1997. године. Основна идеја GRASP-PR методе је да се током претраге истражују трајекторије које повезују текућа GRASP решења и најбоља (елитна) решења пронађена коришћењем неке друге методе претраживања, као што је нпр. табу претраживање. Алгоритам почиње од једног или више елитних решења и потом генерише стазе у простору претраге које повезују решења. У основном облику методе поновног повезивања стаза, постоји почетно и циљно решење између којих се формира путања бирањем оних потеза који уводе атрибуте циљног решења у текуће решење. После сваког потеза врши се евалуација добијеног решења, као и освежавање допустивих потеза. Алгоритам враћа најбоље пронађено решење на посматраној стази. Стратегијом фаворизовања

атрибута које поседују елитна решења, претрага се усмерава ка решењима високог квалитета.

Поред основног концепта GRASP методе и њених модификација, у литератури се може пронаћи велики број различитих хибридних метода у којима се GRASP метода комбинује са другим метахеуристикама: симулираним каљењем, табу претраживањем, методом променљивих околина, генетским алгоритмима и другима.

3.3 Преглед примена GRASP методе на проблеме распоређивања и рутирања возила

У литератури се могу пронаћи бројни примери примена GRASP методе на различите варијанте проблема рутирања и распоређивања возила. Kontoravdis и Bard [58] су предложили GRASP методу при решавању проблема рутирања возила са временским оквирима (енгл. *VRP with time windows - VRPTW*). Проблем распоређивања возила са врло тесним временским оквирима размотрао је Atkinson [9] и решио га применом GRASP метахеуристике са укљученим стратегијама локалне и глобалне адаптивне претраге. Carreto и Baker [17] су развили интерактивни приступ са елементима GRASP методе за решавање проблема рутирања возила са транспортом при повратку (енгл. *VRP with backhauls*). Хибридну методу која је настала комбинацијом GRASP метахеуристике и Еволутивне локалне претраге (енгл. *Evolutionary Local Search - ELS*) за VRP предложио је Prins [77]. Duhamel и сар. [30] су такође користили хибридизацију GRASP и ELS метахеуристика при решавању локацијског проблема рутирања возила са ограниченим капацитетом (енгл. *Capacitated Location-routing Problem*). Park и Seo [74] су развили GRASP методу за распоређивање и рутирање транспортера у бродоградилишту са циљем уједначавања оптерећености транспортера, уз захтев да се сви распоређени блокови морају транспортовати у оквиру датог временског периода. Модификовану варијанту GRASP методе, под називом GRASP метода са вишефазном претрагом (енгл. *Multiple Phase Neighborhood Search - GRASP*), предложио је Marinakis [67] при решавању проблема рутирања возила ограничених капацитета (енгл. *Capacitated Vehicle Routing Problem - CVRP*). Ho и Szeto [52] су развили GRASP метахеуристику са поновним повезивањем путања (енгл. *GRASP with Path Relinking*) за проблем селективног прикупљања и испоруке (енгл. *Selective Pickup and Delivery*

Problem - SPDP). Детаљан преглед радова који се односе на GRASP методу и њене варијанте може се наћи у [34], [79] и [80].

Бројне успешне примене GRASP методе на различите проблеме рутирања и распоређивања возила чине ове методу одговарајућим приступом за решавање једне варијанте проблема распоређивања возила у овој докторској дисертацији. Стога су наведени примери представљали мотивацију за дизајнирање GRASP методе, за дату варијанту VSP. Резултати добијени GRASP методом су анализирани и упоређени са резултатима методе променљивих околина за исту варијанту VSP.

Глава 4

Проблем распоређивања возила при оптимизацији транспорта пољопривредних сировина

4.1 Организација транспорта у фабрици шећера у Србији

Шећерна репа је пољопривредна култура која има изузетно ниску цену на тржишту. Из тог разлога, трошкови транспорта представљају највеће трошкове у производњи шећера. Уштеда која би се постигла ефикасном организацијом транспорта је веома значајна за повећање профита, а прецизан, поуздан и добро осмишљен план транспорта је вишеструко користан за компанију која организује транспорт и за произвођаче који јој указују поверење вишегодишњом сарадњом.

Код проблема који је разматран у овој докторској дисертацији, компанија за производњу шећера откупљује сировине од великог броја индивидуалних произвођача. С обзиром на ниску цену шећерне репе, произвођачи немају интереса да сами превозе робу до фабрике или неког сакупљачког центра, па компанија о свом трошку организује транспорт робе од сваког индивидуалног произвођача до фабрике за прераду. То се најчешће реализује у сарадњи са другим предузећима од којих компаније узима возила у најам и ангажује раднике који обављају послове утовара, истовара и транспорта репе.

Крајем пролећа, када произвођачи могу да процене количине које ће припре-

мити за откуп, састављају се уговори и дефинише се план по коме се за сваког индивидуалног произвођача одређује датум када ће његова роба бити преузета. Производна сезона, која се назива *кампања*, укључује симултани транспорт и прераду шећерне репе до финалног производа и почиње сваке године у септембру и траје до децембра, а понекад се продужи и на јануар следеће године. План транспорта се прави тако да за сваки датум у току кампање буду обезбеђене дневне потребе фабрике, односно да сваког дана од почетка кампање у фабрику стигне унапред дефинисана количина шећерне репе. Процес покретања фабричких машина је веома скуп, те је потребно обезбедити довољне количине робе у фабрици, како не би дошло до прекида производње. Стога се, за сваки датум у току кампање прави списак локација са предвиђеним количинама шећерне репе коју треба превести од сваке локације до фабрике. Под локацијом се подразумева уговорено место на коме најчешће један, а у ретким случајевима два произвођача сакупе шећерну репу. Обично је то адекватно уређен простор испред саме њиве који посматрани произвођач обрађује. Имајући у виду да сви пољопривредни послови у великој мери зависе од временских услова, а јесен је годишње доба када су честе временске непогоде, кише, а понекад и снег, план који је дефинисан у пролеће је подложан сталним променама. Најављене кише могу приморати произвођаче да изваде шећерну репу неколико дана пре договореног датума или их изненадно невреме може спречити да припреме робу на време. Из тог разлога, неопходно је да се план транспорта прави на дневном нивоу на основу тренутне ситуације. План транспорта са састоји од списка локација, количина које се налазе на свакој од њих као и броја дана колико роба стоји на отвореном на свакој од посматраних локација чекајући транспорт.

Најважнији услов при састављању плана транспорта је да се за сваки појединачни датум у току кампање фабрика снабде неопходном дневном количином робе која се превози са расположивих локација. Није обавезно превести сву репу која се посматраног дана налази на отвореном, али је приоритет обезбедити довољну количину за неометани рад постројења. Међутим, при избору локација које ће се опслужити, неопходно је водити рачуна о броју дана колико роба стоји на отвореном. Шећерна репа се ређа на чврстој подлози у облику призме и тако задржава квалитет више дана у зависности од временских услова. Не погодује јој ни сувише топло време, које је могуће у септембру, а ни сувише влажно време, које је карактеристично за јесен. Губитак квалитета услед дужег стајања доводи до смањења откупне цене, што произвођаче чини

незадовољним, а компанија остаје без квалитетне сировине. Стога се, приликом састављања плана транспорта за конкретни дан мора обезбедити да локације на којима репа стоји више од дозвољеног броја дана буду опслужене, односно репа која се налази на отвореном дуже него што је дозвољено мора бити превезена у току посматраног дана.

У кругу фабрике постоји паркинг за возила која су узета у најам за потребе транспорта. Током радног времена возила праве туре које подразумевају одлазак од фабрике до локације на којој се налази репа, утоваривање и повратак у фабрику. Према томе, фабрика је полазна и крајња тачка туре сваког возила, па се ради о проблему транспорта са једним депоом (енгл. *single depot*). Возила се користе више пута у току дана, односно по завршеном истовару, свако возило може започети нову туру што значи да је у питању је проблем распоређивања возила са више путовања (енгл. *multiple trips*). У свакој тури, возила одлазе до једне од локација и враћају се у фабрику. Возила не посећују више од једне локације пре повратка у фабрику, јер није дозвољено мешање сировина са различитих локација, због процене квалитета чији резултат одређује цену довезене репе. Количине на свакој од локација значајно су веће од капацитета возила, па је потребно посетити сваку локацију неколико пута да би се превезла сва припремљена репа. Стога, у највећем броју транспорта, возила са пуним капацитетом натоварене репе са неке од локација улазе у круг фабрике. Само возило које последње по реду посећује неку локацију може бити делимично пуно.

Код организације транспорта за посматрани проблем, веома је важно направити распоред возила који елиминише редове чекања на свакој од локација и испред саме фабрике, где возила фреквентно пристижу током целог дана. Камиони који стоје у колони чекајући на ред за истовар или утовар представљају највећи показатељ неефикасне организације транспорта, с обзиром да се најам камиона и радника плаћа по сату рада. Такође, у случају конкретног реалног проблема, расположиви ресурси компаније и транспортних предузећа са којима сарађује су ограничени јер се на свакој локацији која је предвиђене за опслуживање одређеног дана, налази само једна машина за утовар. Из тог разлога, за време утоваривања репе у једно возило на датој локацији, непотребно је да на исту локацију стижу друга возила. Са друге стране, у кругу фабрике могуће је истовремено опслужити и више возила чији је број унапред ограничен. Када возила стигну са шећерном репом у фабрику, потребно је неколико минута за

анализирање узорка робе која је довезена и истовар. Да се не би стварали редови чекања потребно је направити распоред возила који ће омогућити да у току радног времена, у сваком тренутку у фабрици не буде више од унапред ограниченог броја возила са робом, а да се на свакој локацији у сваком тренутку налази само једно возило. Позната су просечна времена трајања утовара на локацијама и задржавања возила у фабричком кругу приликом истовара.

Имајући у виду горе наведено, циљ посматраног проблема је направити план транспорта за сваки радни дан који задовољава сва наведена ограничења, а тако да време транспорта буде минимално. Дакле, потребно је минимизовати временски тренутак када се сва возила врате у фабрику и заврше истовар у својим последњим турама одређеног дана. У случају разматраног реалног проблема може се сматрати да је минимизација времена транспорта еквивалентна са минимизацијом трошкова транспорта. Заиста, у току целе кампање мора се превести целокупна количина репе која је те године припремљена, тако да се трошкови горива могу сматрати фиксним, па је циљ умањити време рада које утиче на трошкове радне снаге.

Према томе, преглед карактеристика посматраног проблема је:

- Транспорт шећерне репе се организује на дневном нивоу;
- Возила су хомогена и користе се више пута у току дана;
- Фабрика је почетна и крајња тачка сваке туре;
- Свако возило у свакој тури посећује само једну локацију;
- Дневне потребе фабрике (количина довезене робе у току дана) морају бити задовољене;
- Постоји ограничење у броју дана колико роба сме да стоји на отвореном;
- Свака локација на којој роба стоји дуже од дозвољеног броја дана, мора бити опслужена;
- Два возила се не могу истовремено утоварити на једној локацији;
- Ограничен број возила се може истовемено истоварити у фабрици;
- Редови чекања нису дозвољени ни на локацијама, нити испред фабрике;
- Циљ је минимизовати временски тренутак када сва возила заврше истовар у својим последњим турама.

Табела 4.1: Нотација

J:	Скуп локација;
I:	Скуп возила;
K:	Скуп тура;
n:	Укупан број локација;
m:	Укупан број возила;
k_{max}:	Максималан број тура које возило може да направи у току дана;
c_j:	Количина сакупљене сировине (у тонама) на локацији $j \in J$;
d_j:	Растојање од фабрике до локације $j \in J$ (у km);
CV:	Капацитет возила (у тонама);
C:	Дневне потребе фабрикеу (у тонама);
v:	Просечна брзина возила (у km/h);
u:	Просечно време задржавања возила у фабрици по завршетку туре потребно за истовар и анализирање узорака (у h);
w:	Просечно време трајања утовара (у h);
t_j:	Број дана колико роба стоји на отвореном на локацији $j \in J$;
t_0:	Максималан број дана колико роба може да стоји на отвореном без губитка квалитета;
T_j:	Бинарна величина додељена локацији $j \in J$, дефинисана са: $T_j = 1$ ако је $t_j > t_0$ и $T_j = 0$ у супротном;
p:	Број возила који може бити истовремено опслужен у кругу фабрике;
ε:	Мала позитивна константа за коју важи $\varepsilon < CV$ и $\varepsilon < c_j/CV$ за свако $j \in J$;
M:	Велика позитивна константа;
t_{start}:	Почетак радног времена;
t_{end}:	Крај радног времена.

4.2 Ознаке

У циљу представљања математичке формулације описаног проблема и његових варијанти, неопходно је увести одговарајуће ознаке. Нотација која ће бити коришћена приказана је у Табели 4.1.

4.3 Математичка формулација потпроблема - VSP-P

Прво је посматран потпроблем описаног проблема распоређивања возила при транспорту шећерне репе. Овај потпроблем, означен са VSP-P, добијен је релаксацијом посматраног проблема, тако што се претпоставља да на свакој локацији и у кругу фабрике има довољно ресурса да се истовремено опслужи

неограничен број возила. Другим речима, код VSP-P су дозвољени сусрети произвољног броја возила на локацијама и у кругу фабрике.

Виртуалне туре

Максималан број тура које свако возило може да реализује током дана означен је са k_{max}. Међутим, не морају сва возила током дана имати исти број тура. Број тура који ће возило направити током једног радног дана зависи од удаљености изабраних локација од фабрике. Удаљеност локације која се опслужује директно утиче на време трајања туре, па стога возила која одлазе на ближе локације могу имати више тура од оних која посећују локације на већој удаљености од фабрике, током радног дана. Ради једноставности дефинисања и планирања броја управљачких променљивих у моделу, уведен је концепт *виртуалне* туре. Број тура за свако возило се проширује до k_{max}, чиме се оставља могућност да возило има једну или више виртуалних тура. Сматра се да током виртуалне туре возило остаје у кругу фабрике. Међутим, виртуалне туре не утичу на рачунања времена завршетка рада сваког возила, с обзиром да је њихово трајање једнако нули. Стога овај концепт не утиче на вредност функције циља, а технички омогућава изједначавање броја тура за сва возила.

Променљиве

За формулисање математичког модела који адекватно описује проблем VSP-P, потребно је увести три групе променљивих и једну променљиву која се користи у функцији циља:

- Бинарне променљиве x_{ik}^{j}, $i \in I$, $k \in K$, $j \in J$, одређују локацију коју возило посећује у одговарајућој тури. Наиме, x_{ik}^{j} узима вредност 1 ако возило i посећује локацију j у тури k, у супротном је $x_{ik}^{j} = 0$. Ако је тура $k \in K$ возила $i \in I$ виртуална, тада важи $\sum_{j \in J} x_{ik}^{j} = 0$;

- Реалне променљиве t_{ik}, $i \in I$, $k \in K$ дефинишу време поласка сваког возила у одговарајућој тури из фабрике;

- Бинарне променљиве y_j, $j \in J$ су индикатори који одређују да ли је локација j испражњена или не. Ако је количина робе припремљена на локацији j превезена, тада важи $y_j = 0$, док је, у супротном, $y_j = 1$. Употреба

ових промењивих омогућава увид у укупну количину робе која стиже у фабрику. Наиме, ако је локација j испражњена, количина робе која је транспортована са ове локације једнака је c_j. У супротном, превезена количина са посматране локације представља производ капацитета возила CV и броја посета локације j од стране свих возила која су опслужила дату локацију у току радног дана.

- Промењљива T, која узима реалне вредности, се користи у функцији циља. Она дефинише последњи временски тренутак када сва возила заврше истовар у својој последњој тури. Циљ проблема је минимизовати вредност промењљиве T.

Предложени математички модел и реформулација

За проблем VSP-P предложен је следећи модел мешовитог целобројног програмирања са квадратним ограничењима (енгл. *Mixed Integer Quadratically Constrained Programming - MIQCP*):

$$\min \quad T \tag{4.1}$$

при условима

$$\sum_{j\in J} x_{ik}^j \leq 1 \quad \forall i \in I, \quad \forall k \in K, \tag{4.2}$$

$$CV \sum_{(i,k)\in I\times K} x_{ik}^j - CV + \varepsilon \leq c_j \quad \forall j \in J, \tag{4.3}$$

$$T_j + y_j \leq 1 \quad \forall j \in J, \tag{4.4}$$

$$\sum_{j\in J} (1 - y_j) \cdot c_j + CV \cdot \sum_{(i,j,k)\in I\times J\times K} y_j x_{ik}^j \geq C, \tag{4.5}$$

$$\sum_{(i,k)\in I\times K} x_{ik}^j \geq (c_j/CV) \cdot (1 - y_j) \quad \forall j \in J, \tag{4.6}$$

$$\sum_{(i,k)\in I\times K} y_j x_{ik}^j + \varepsilon \leq c_j/CV \quad \forall j \in J, \tag{4.7}$$

$$t_{i,k+1} \geq t_{ik} + \frac{2}{v} \sum_{j\in J} d_j x_{ik}^j + (u + w) \sum_{j\in J} x_{ik}^j \quad \forall i \in I, \quad \forall k \in K \setminus \{k_{max}\}, \tag{4.8}$$

$$t_{i,k_{max}} + \frac{2}{v}\sum_{j\in J} d_j x^j_{ik_{max}} + (u+w)\sum_{j\in J} x^j_{ik_{max}} \leq T \quad \forall i \in I, \tag{4.9}$$

$$x^j_{ik} \in \{0,1\} \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K, \tag{4.10}$$

$$y_j \in \{0,1\} \quad \forall j \in J, \tag{4.11}$$

$$t_{ik} \in [t_{start}, t_{end}] \quad \forall i \in I, \quad \forall k \in K. \tag{4.12}$$

Функција циља (4.1) заједно са ограничењем (4.9) минимизује временски тренутак у којем се завршава дневни транспорт за сва возила, односно тренутак у којем последње пристигло возило заврши истовар робе у својој последњој тури.

Ограничење (4.2) обезбеђује да свако возило у свакој тури може да опслужи највише једну локацију. Дакле, свако возило у свакој тури има могућност да посети неку локацију или направи виртуалну туру. Укупна количина робе која се превезе са локације $j \in J$ у фабрику не може бити већа од припремљене количине c_j на датој локацији, што је обезбеђено ограничењем (4.3). Умањилац CV који се налази на левој страни ограничења (4.3) одражава чињеницу да последње возило које посећује неку локацију не мора бити пуно робе. Како би количина која се транспортује са локације $j \in J$ умањену за величину капацитета возила CV била строго мања од прикупљене количине c_j на локацији $j \in J$, на левој страни ограничења (4.3) је додата мала позитивна константа ε. То је неопходно у случају када је c_j једнако производу природног броја и капацитета возила CV тј. целобројном умношку величине CV. Без строге неједнакости, било би могуће да возила непотребно направе туру више до локације $j \in J$. На пример, ако је $c_j = 120$ тона и $CV = 30$ тона, локацију j је потребно посетити највише 4 пута, међутим, без константе ε у ограничењу (4.3), постоји могућност да се локација j посети 5 пута, како је $5 \cdot 30 - 30 \leq 120$.

Ограничење (4.4) обезбеђује услов да се све количине са локације $j \in J$ морају превести током радног дана уколико на овој локацији роба стоји на отвореном дуже од t_0 дана. Имајући у виду дефиницију бинарних величина T_j (видети Табелу 4.1), у случају када је $t_j \leq t_0$, важи $T_j = 0$, а десна страна ограничења (4.4) је једнака 1. Стога је, у овом случају, ограничење (4.4) задовољено независно од транспортоване количине са локације j. Са друге стране,

ако је $t_j > t_0$, тада је $T_j = 1$ и локација j мора бити испражњена током радног дана, што значи да y_j мора имати вредност 0.

Ограничење (4.5) обезбеђује задовољење дневних потреба фабрике. Прва сума на левој страни релације (4.5) односи се на количине које су довезене са локација које су испражњене, а друга представља количину робе, довезене са неиспражњених локација.

Вредности променљивих y_j су дефинисане ограничењима (4.6) и (4.7). Ако је локација $j \in J$ испражњена, y_j узима вредност 0 и релација (4.6) обезбеђује да је локација j посећена не мање од c_j/CV пута. У случају када локација $j \in J$ није испражњена, тада је $y_j = 1$ и ограничење (4.6) је задовољено. Ограничење (4.7) обезбеђује да је свака неиспражњена локација $j \in J$ посећена строго мање од c_j/CV пута. У случају да се ради о испражњеној локацији, ограничење (4.7) није активно.

Ограничење (4.8) одражава чињеницу да свако возило $i \in I$ не може почети нову туру док се претходна не заврши. Време поласка сваке туре не сме бити мање од времена поласка претходне туре, увећаном за време трајања вожње у оба смера и збиром дужине трајања утовара и истовара. Ако је тура $k \in K \setminus \{k_{max}\}$ возила $i \in I$ виртуална, обе суме на десној страни ограничења (4.8) су једнаке 0. На крају, ограничења (4.9)–(4.12) дефинишу тип променљивих које су коришћене у предложеном моделу.

MIQCP модел (4.1)–(4.12) је преформулисан у еквивалентан проблем мешовитог целобројног линеарног програмирања (енгл. *Mixed Integer Linear Programming - MILP*). Да би се извршила линеаризација, неопходно је трансформисати производ бинарних променљивих y_j и x_{ik}^j који се јавља у ограничењима (4.5) и (4.7). У том циљу, уводи се нови скуп бинарних променљивих r_{ik}^j, $i \in I, j \in J, k \in K$, који, у поменутим ограничењима, представља замену за производ $y_j x_{ik}^j$. Променљиве r_{ik}^j морају да задовољавају следећа ограничења:

$$r_{ik}^j - \frac{1}{2}(y_j + x_{ik}^j) \leq 0 \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K, \tag{4.13}$$

$$r_{ik}^j - y_j - x_{ik}^j + 1 \geq 0 \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K, \tag{4.14}$$

$$r_{ik}^j \in \{0, 1\} \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K. \tag{4.15}$$

Ограничење (4.5) је замењено ограничењем (4.5')

$$\sum_{j\in J}(1-y_j)\cdot c_j + CV\cdot\sum_{j\in J}\sum_{(i,k)\in I\times K} r_{ik}^j \geq C, \tag{4.5'}$$

а ограничење (4.7) замењено је ограничењем (4.7')

$$\sum_{(i,k)\in I\times K} r_{ik}^j + \varepsilon \leq c_j/CV \qquad \forall j \in J. \tag{4.7'}$$

На овај начин, предложени MIQCP модел (4.1)–(4.12) је трансформисан у еквивалентан MILP модел (4.1)-(4.4), (4.5'), (4.6), (4.7'), (4.8)– (4.12), (4.13)–(4.15).

При линеаризацији MIQCP модела, додато је nmk_{max} нових променљивих бинарног типа и $2nmk_{max}$ нових ограничења. Како повећање броја променљивих и ограничења не утиче значајно на ефикасност формулације, при тестирању егзактним решавачем коришћена је само MILP формулација проблема VSP-P.

За проблем VSP-P формулисан је и нелинеаран математички модел који је представљен у раду [5]. Међутим, имајући у виду да су егзактни решавачи коришћењем MILP формулације достигли највећи број оптималних решења на реалним инстанцама за значајно краће просечно време извршавања, нелинеаран модел формулисан у [5] је изостављен из ове докторске дисертације.

4.4 Математичка формулација комплетног проблема-VSP

Проблем распоређивања возила, описан у секцији 4.1 који укључује све наведене услове означен је са VSP. Овај проблем подразумева сва ограничења наведена у математичкој формулацији потпроблема VSP-P, као и додатна ограничења која се односе на следећа два услова:

- два возила не могу бити истовремено утоварена ни на једној од локција на којима се сировина налази,
- ограничен број возила може бити истовремено опслужен у кругу фабрике.

Да би ова два услова представили адекватним математичким изразима, неопходно је увести неколико група нових променљивих и значајан број нових ограничења, што у великој мери утиче на сложеност математичке формулације проблема VSP.

Променљиве

Поред променљивих x_{ik}^{j}, t_{ik}, y_j и променљиве T које су уведене при формулацији модела за проблем VSP-P, за формулисање математичког модела проблема VSP потребно је увести још неколико група променљивих:

- Бинарне променљиве b_{ik}^{ls}, $i, l \in I$, $k, s \in K$, се користе за формулисање ограничења које обезбеђује да два возила не могу у исто време да утоваре сировину на било којој од локација. Прецизније, улога ових променљивих је да се избегну апсолутне заграде око израза који представља разлику између времена стизања два различита возила на исту локацију. Наиме, ако је $t_{ik} - t_{ls} < 0$, тада је $b_{ik}^{ls} = 1$, у супротном је $b_{ik}^{ls} = 0$.

- Три скупа променљивих су коришћена како би се математички формулисао услов да највише p возила може бити опслужено у кругу фабрике у исто време. Бинарне променљиве z_{ik}^{ls}, $i, l \in I$, $k, s \in K$ се дефинишу на следећи начин: $z_{ik}^{ls} = 1$ ако разлика између времена стизања у фабрику возила i и l у њиховим турама k и s, редом, није већа од дужине трајања истовара u; у супротном је $z_{ik}^{ls} = 0$. Бинарне променљиве g_{ik}^{ls}, $i, l \in I$, $k, s \in K$ су уведене како би се формулисао овај услов без апсолутних заграда, на сличан начин као и променљиве b_{ik}^{ls}. Коначно, реалне ненегативне променљиве a_{ik}^{ls}, $i, l \in I$, $k, s \in K$ представљају апсолутне вредности разлике између времена стизања у фабрику возила i and l у њиховим турама k и s, редом.

Предложени математички модел

За проблем распоређивања возила VSP најпре је предложен следећи модел мешовитог целобројног програмирања са квадратним ограничењима (MIQCP):

$$\min \quad T \tag{4.16}$$

при условима

$$\sum_{j \in J} x_{ik}^{j} \leq 1 \quad \forall i \in I, \quad \forall k \in K, \tag{4.17}$$

$$CV \sum_{(i,k) \in I \times K} x_{ik}^{j} - CV + \varepsilon \leq c_j \quad \forall j \in J, \tag{4.18}$$

$$T_j + y_j \leq 1 \quad \forall j \in J, \tag{4.19}$$

$$\sum_{j \in J} (1 - y_j) \cdot c_j + CV \cdot \sum_{(i,j,k) \in I \times J \times K} y_j x_{ik}^j \geq C, \tag{4.20}$$

$$\sum_{(i,k) \in I \times K} x_{ik}^j \geq (c_j / CV) \cdot (1 - y_j) \quad \forall j \in J, \tag{4.21}$$

$$\sum_{(i,k) \in I \times K} y_j x_{ik}^j + \varepsilon \leq c_j / CV \quad \forall j \in J, \tag{4.22}$$

$$t_{i,k+1} \geq t_{ik} + \frac{2}{v} \sum_{j \in J} d_j x_{ik}^j + (u + w) \sum_{j \in J} x_{ik}^j \quad \forall i \in I, \quad \forall k \in K \setminus \{k_{max}\}, \tag{4.23}$$

$$t_{i,k_{max}} + \frac{2}{v} \sum_{j \in J} d_j x_{ik_{max}}^j + (u + w) \sum_{j \in J} x_{ik_{max}}^j \leq T \quad \forall i \in I, \tag{4.24}$$

$$t_{ik} - t_{ls} + M b_{ik}^{ls} \geq w x_{ik}^j x_{ls}^j \quad \forall j \in J, \quad \forall i \neq l \in I \quad \forall k, s \in K, \tag{4.25}$$

$$-t_{ik} + t_{ls} + M(1 - b_{ik}^{ls}) \geq w x_{ik}^j x_{ls}^j \quad \forall j \in J, \quad \forall i \neq l \in I \quad \forall k, s \in K, \tag{4.26}$$

$$z_{ik}^{is} = 0 \qquad \forall i \in I \quad \forall k, s \in K, \tag{4.27}$$

$$z_{ik}^{ls} \leq \sum_{j \in J} x_{ik}^j \sum_{j \in J} x_{ls}^j \quad \forall i, l \in I \quad \forall k, s \in K, \tag{4.28}$$

$$2 z_{ik}^{ls} + (1 - 2 z_{ik}^{ls})(\frac{1}{u} a_{ik}^{ls} + \varepsilon) \geq \sum_{j \in J} x_{ik}^j \sum_{j \in J} x_{ls}^j \quad \forall i \neq l \in I \quad \forall k, s \in K, \tag{4.29}$$

$$t_{ik} - t_{ls} + \frac{2}{v} \sum_{j \in J} d_j (x_{ik}^j - x_{ls}^j) \leq a_{ik}^{ls} \quad \forall i, l \in I \quad \forall k, s \in K, \tag{4.30}$$

$$-t_{ik} + t_{ls} - \frac{2}{v} \sum_{j \in J} d_j (x_{ik}^j - x_{ls}^j) \leq a_{ik}^{ls} \quad \forall i, l \in I \quad \forall k, s \in K, \tag{4.31}$$

$$a_{ik}^{ls} \leq t_{ik} - t_{ls} + \frac{2}{v}\sum_{j\in J} d_j(x_{ik}^j - x_{ls}^j) + 2(t_{end} - t_{start})(1 - g_{ik}^{ls}) \quad \forall i,l \in I \ \ \forall k,s \in K, \tag{4.32}$$

$$a_{ik}^{ls} \leq -t_{ik} + t_{ls} - \frac{2}{v}\sum_{j\in J} d_j(x_{ik}^j - x_{ls}^j) + 2(t_{end} - t_{start})g_{ik}^{ls} \quad \forall i,l \in I \ \ \forall k,s \in K, \tag{4.33}$$

$$\sum_{(l,s)\in I\times K} z_{ik}^{ls} \leq p - 1, \quad \forall i \in I \quad \forall k \in K, \tag{4.34}$$

$$x_{ik}^j \in \{0,1\} \quad \forall i \in I, \quad \forall j \in J, \quad \forall k \in K, \tag{4.35}$$

$$y_j \in \{0,1\} \quad \forall j \in J, \tag{4.36}$$

$$z_{ik}^{ls} \in \{0,1\} \quad \forall i,l \in I, \quad \forall k,s \in K, \tag{4.37}$$

$$t_{ik} \in [t_{start}, t_{end}] \quad \forall i \in I, \quad \forall k \in K, \tag{4.38}$$

$$a_{ik}^{ls} \geq 0 \quad \forall i,l \in I, \quad \forall k,s \in K, \tag{4.39}$$

$$g_{ik}^{ls} \in \{0,1\} \quad \forall i,l \in I, \quad \forall k,s \in K, \tag{4.40}$$

$$b_{ik}^{ls} \in \{0,1\} \quad \forall i,l \in I, \quad \forall k,s \in K. \tag{4.41}$$

Ограничења (4.16)-(4.24) су идентична ограничењима (4.1)-(4.9) у моделу VSP-P, и њихова значења су детаљно објашњена у претходном одељку.

Ограничења (4.25) и (4.26) обезбеђују да се два возила не могу истовремено утоваривати на истој локацији. Наиме, ако два различита возила посећује исту локацију, апсолутна вредност разлике њихових времена кретања из фабрике рачуна се као апсолутна вредност разлике између њихових времена стизања на дату локацију, с обзиром да возила имају исте техничке карактеристике и прелазе исти пут. Израчуната апсолутна вредност мора бити већа или једнака од времена трајања утовара w, што је обезбеђено ограничењима (4.25) и (4.26).

Скуп ограничења (4.27)-(4.34) обезбеђује да највише p натоварених возила може истовремено бити опслужено у кругу фабрике, по повратку са посећених локација. Ограничењем (4.27) се игнорише случај када посматране туре припадају истом возилу. Улога неједнакости (4.28) је да се искључи случај када бар једно од посматраних возила има виртуалну туру. Ограничењем (4.29) одређује се вредност бинарних променљивих z_{ik}^{ls} на следећи начин: $z_{ik}^{ls} = 1$ ако апсолутна вредност разлике између времена стизања у круг фабрике, возила i и l, у њиховим турама k и s, респективно, није већа од времена трајања истовара u, у супротном је $z_{ik}^{ls} = 0$. Улога производа сума на десној страни ограничења (4.29) је изузимање случаја када је тура k возила i или тура s возила l виртуална. Ограничења (4.30)-(4.33) обезбеђују да свака од променљивих a_{ik}^{ls} има адекватну вредност која представља апсолутну вредност разлике између времена стизања у фабрику возила i и l у њиховим турама k и s, респективно. Коначно, ограничење (4.34) одражава чињеницу да највише $p-1$ других возила може бити истоварено у исто време заједно са возилом i у његовој тури k, за свако $i \in I$ и $k \in K$. Типови променљивих коришћених у моделу су дефинисани ограничењима (4.35)-(4.41).

Реформулација модела

Да би се предложени MIQCP модел који описује проблем VSP трансформисао у модел мешовитог целобројног линеарног програмирања, најпре се уводе исти скупови променљивих који су коришћени при линеаризацији модела проблема VSP-P. Наиме, неопходно је заменити производ бинарних променљивих y_j и x_{ik}^j у ограничењима (4.20) и (4.22) променљивом r_{ik}^j. При томе се додају ограничења која дефинишу зависност ове три групе променљивих:

$$r_{ik}^j - \frac{1}{2}(y_j + x_{ik}^j) \leq 0 \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K, \tag{4.42}$$

$$r_{ik}^j - y_j - x_{ik}^j + 1 \geq 0 \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K, \tag{4.43}$$

$$r_{ik}^j \in \{0, 1\} \qquad \forall i \in I,\ \ \forall j \in J,\ \ \forall k \in K. \tag{4.44}$$

Заменом производа $y_j x_{ik}^j$ променљивом r_{ik}^j, ограничење (4.20) се трансформише у

$$\sum_{j\in J}(1-y_j)\cdot c_j + CV \cdot \sum_{(i,j,k)\in I\times J\times K} r_{ik}^j \geq C, \tag{4.20'}$$

а ограничење (4.22) постаје

$$\sum_{(i,k)\in I\times K} r_{ik}^j + \varepsilon \leq c_j/CV \qquad \forall j \in J. \tag{4.22'}$$

За разлику од MIQCP формулације проблема VSP-P за чију линеаризацију је потребно увести само променљиве r_{ik}^j, у циљу добијања линеарног модела који описује проблем VSP, потребно је елиминисати и преостале квадратне изразе у ограничењима (4.25), (4.26), (4.28) и (4.29). Из тог разлога, уводи се скуп реалних променљивих π_{ik}^{ls} $i,l \in I$, $k,s \in K$, дефинисаних са $\pi_{ik}^{ls} = z_{ik}^{ls} a_{ik}^{ls}$. Променљиве π_{ik}^{ls} омогућавају да се избегне производ бинарних променљивих z_{ik}^{ls} и реалних променљивих a_{ik}^{ls}. Користећи променљиве π_{ik}^{ls}, ограничење (4.29) се трансформише у (4.29'):

$$2z_{ik}^{ls} + \frac{1}{u}a_{ik}^{ls} - \frac{2}{u}\pi_{ik}^{ls} - 2\varepsilon z_{ik}^{ls} + \varepsilon \geq \sum_{j\in J} x_{ik}^j \sum_{j\in J} x_{ls}^j \quad \forall i \neq l \in I \quad \forall k,s \in K. \tag{4.29'}$$

Поред тога, новоуведене променљиве π_{ik}^{ls} морају да задовољавају следећа ограничења:

$$\pi_{ik}^{ls} \leq (t_{end} - t_{start}) z_{ik}^{ls} \quad \forall i,l \in I \quad \forall k,s \in K, \tag{4.45}$$

$$\pi_{ik}^{ls} \geq a_{ik}^{ls} - (t_{end} - t_{start})(1 - z_{ik}^{ls}) \quad \forall i,l \in I \quad \forall k,s \in K, \tag{4.46}$$

$$\pi_{ik}^{ls} \leq a_{ik}^{ls} + (t_{end} - t_{start})(1 - z_{ik}^{ls}) \quad \forall i,l \in I \quad \forall k,s \in K, \tag{4.47}$$

$$\pi_{ik}^{ls} \geq 0 \qquad \forall i,l \in I, \qquad \forall k,s \in K, \tag{4.48}$$

Улога ограничења (4.45)-(4.48) је да обезбеде коректну дефиницију променљивих π_{ik}^{ls} као производа бинарних променљивих z_{ik}^{ls} и реалних променљивих a_{ik}^{ls} за $i,l \in I$, $k,s \in K$. Константа $t_{end} - t_{start}$ се користи као горња граница вредности реалних променљивих a_{ik}^{ls}, имајући у виду да ове променљиве представљају апсолутну вредност разлике времена пристизања два различита возила у круг

фабрике. Заиста, ако је $z_{ik}^{ls} = 0$, из ограничења (4.45) и (4.48) произилази да је $\pi_{ik}^{ls} = 0$. У супротном, када је $z_{ik}^{ls} = 1$, из ограничења (4.46) и (4.47) следи да је $\pi_{ik}^{ls} = a_{ik}^{ls}$.

За комплетну линеаризацију предложеног MIQCP модела, неопходно је још заменити производ бинарних променљивих x_{ik}^{j} и x_{ls}^{h} новом бинарном променљивом ρ_{ikls}^{jh}, за све $j \in J$, $i, l \in I$, $k, s \in K$. Ова трансформација ће очигледно утицати на трансформацију ограничења (4.25), (4.26), (4.28) и (4.29').

Заменом производа $x_{ik}^{j} x_{ls}^{h}$ променљивом ρ_{ikls}^{jh}, ограничења (4.25) и (4.26), редом постају:

$$t_{ik} - t_{ls} + M b_{ik}^{ls} \geq w \rho_{ikls}^{jj} \quad \forall j \in J, \quad \forall i \neq l \in I \quad \forall k, s \in K, \tag{4.25'}$$

$$-t_{ik} + t_{ls} + M(1 - b_{ik}^{ls}) \geq w \rho_{ikls}^{jj} \quad \forall j \in J, \quad \forall i \neq l \in I \quad \forall k, s \in K. \tag{4.26'}$$

С обзиром да важи $\sum_{j \in J} x_{ik}^{j} \sum_{j \in J} x_{ls}^{j} = \sum_{j \in J} \sum_{h \in J} x_{ik}^{j} x_{ls}^{h}$, заменом $x_{ik}^{j} x_{ls}^{h}$ са ρ_{ikls}^{jh}, ограничења (4.28) и (4.29') се, редом, трансформишу у:

$$z_{ik}^{ls} \leq \sum_{j \in J} \sum_{h \in J} \rho_{ikls}^{jh} \qquad \forall i, l \in I \qquad \forall k, s \in K, \tag{4.28'}$$

$$2 z_{ik}^{ls} + \frac{1}{u} a_{ik}^{ls} - \frac{2}{u} \pi_{ik}^{ls} - 2\varepsilon z_{ik}^{ls} + \varepsilon \geq \sum_{j \in J} \sum_{h \in J} \rho_{ikls}^{jh} \quad \forall i \neq l \in I \quad \forall k, s \in K. \tag{4.29"}$$

У модел се укључују и нова ограничења која ће обезбедити да променљиве ρ_{ikls}^{jh} буду коректно дефинисане као производ бинарних променљивих x_{ik}^{j} и x_{ls}^{h}:

$$\rho_{ikls}^{jh} - \frac{1}{2}(x_{ik}^{j} + x_{ls}^{h}) \leq 0 \qquad \forall i, l \in I, \ \ \forall j, h \in J, \ \ \forall k, s \in K, \tag{4.49}$$

$$\rho_{ikls}^{jh} - x_{ik}^{j} - x_{ls}^{h} + 1 \geq 0 \qquad \forall i, l \in I, \ \ \forall j, h \in J, \ \ \forall k, s \in K, \tag{4.50}$$

$$\rho_{ikls}^{jh} \in \{0, 1\} \qquad \forall i, l \in I, \ \ \forall j, h \in J, \ \ \forall k, s \in K. \tag{4.51}$$

На крају, MILP модел добијен линеаризацијом MIQCP модела посматраног VSP проблема је представљен изразима (4.16)-(4.19), (4.20'), (4.21), (4.22'), (4.23)-(4.24), (4.25'), (4.26'), (4.27), (4.28'), (4.29"), (4.30)-(4.51).

Да би се предложени MIQCP модел проблема VSP успешно трансформисао у еквивалентну MILP формулацију, било је неопходно увести значајан број нових променљивих и нових ограничења. Приликом линеаризације број променљивих се повећао за $n^2m^2k_{max}^2 + m^2k_{max}^2 + nmk_{max}$ ($n^2m^2k_{max}^2 + m^2k_{max}^2$ бинарних и nmk_{max} реалних), а број ограничења за $2n^2m^2k_{max}^2 + 3m^2k_{max}^2 + 2nmk_{max}$. Без обзира на повећање сложености модела гледајући са аспекта броја променљивих и ограничења, MILP формулација се показала успешнијом при решавању егзактним солвером, обезбедивши више оптималних решења на инстанцама малих димензија за знатно краће време извршавања (видети одељак 7.2).

4.5 Сложеност проблема

Да би се испитала сложеност описаног проблема, имајући у виду да се ради о новом проблему који још није разматран у литератури, потребно је пронаћи аналогију разматраног проблема или неког његовог потпроблема са проблемом познате сложености.

У овом одељку биће показано да је предложена варијанта проблема распоређивања возила НП-тежак проблем. Познато је да проблеми распоређивања возила са једним депоом припадају класи НП-тешких проблема [61]. Генерално, код проблема распоређивања возила, свако возило посећује неколико локација у свакој тури пре повратка у депо. Ово није случај у разматраном проблему распоређивања возила при траспорту шећерне репе, с обзиром да није дозвољено мешање сировина са различитих локација због могућности добијања мешавине незадовољавајућег квалитета. Из тог разлога није могуће пронаћи релаксацију разматраног проблема која би била једнаке или веће сложености од класичног проблема распоређивања возила са једним депоом из [61]. Међутим, може се испитати еквиваленција разматраног VSP са проблемом распоређивања машина, што води ка следећој теореми.

Теорема 4.5.1. Разматрани проблем распоређивања возила при транспорту шећерне репе (VSP) је НП-тежак проблем оптимизације.

Доказ. Доказ теореме се заснива на студији комплексности различитих варијанти проблема распоређивања машина коју је дао Pinedo у [75]. Посматра се проблем паралелног распоређивања машина, који је у студији [75] означен са $Pm||C_{max}$. Доказано је да је $Pm||C_{max}$ НП-тежак у класичном смислу, због

еквиваленције са проблемом партиције (енгл. *Partitioning Problem*) [40], [75]. Код проблема $Pm||C_{max}$, потребно је распоредити N независних задатака на m идентичних машина које паралелно раде (Pm). Време процесирања задатка $j \in \{1 \dots N\}$ је $p(j)$. Сваки задатак треба да буде додељен тачно једној машини. Са друге стране, свака машина може да обавља само један задатак у исто време и распоређивање једног задатка на две или више машина није дозвољено. Циљ је минимизовати коначно време завршетка (енгл. *makespan*), тј. максимум свих времена рада посматраних машина, који се рачуна по формули $C_{max} = max_{1 \leq i \leq m} C_i$, при чему C_i одговара времену завршетка рада машине i.

Аналогија између разматраног проблема распоређивања возила и проблема паралелног распоређивања машина може се увести на следећи начин. Машине се идентификују са возилима и сваки задатак одговара једној тури. Задатак се може означити са τ_{ij}^k, где је i индекс возила, k представља редни број туре, док је j индекс локације коју посећује возило i у тури k. Очигледно је број задатака једнак укупном броју тура, односно важи $N = m \cdot k_{max}$. Време процесирања задатка τ_{ij}^k може се посматрати као функција која зависи од индекса локације j, тј. $p(\tau_{ij}^k) = \varphi(j)$, $j \in J \cup \{0\}$, односно време процесирања задатка зависи од растојања локације j и фабрике. У том случају, $j = 0$ означава локацију фабрике што одговара виртуалној тури, па важи $\varphi(0) = 0$, с обзиром да је трајање виртуалне туре једнако 0. На крају, минимизација коначног времена завршетка C_{max} код проблема паралелног распоређивања машина еквивалентна је минимизацији вредности променљиве T, где $T = max_{1 \leq n \leq m} T_{i,k_{max}}$ представља последњи тренутак када сва возила завршавају своје последње туре.

На описани начин, $Pm||C_{max}$ се може полиномски свести на потпроблем разматраног проблема распоређивања возила, који се добија занемаривањем свих осталих специфичних ограничења. Имајући у виду да је $Pm||C_{max}$ НП-тежак ([75], Пример D.3.2, стр. 595), може се закључити да разматрани проблем распоређивања возила за транспорт шећерне репе представља НП-тежак проблем оптимизације. □

Имајући у виду да је потпроблем VSP-P добијен из VSP уклањањем два ограничења која се односе на сусрете возила на локацијама и у кругу фабрике, а која нису укључена у релаксацију проблема VSP из доказа Теореме 4.5.1, може се закључити да је и потпроблем VSP-P НП-тежак.

4.6 Специфичности предложених проблема VSP и VSP-P у односу на постојеће варијанте проблема VSP и VRP из литературе

У оквиру одељка 1.2 дат је преглед различитих варијанти проблема рутирања и распоређивања возила из литературе. Као што је наведено, постоје бројне варијанте ових проблема који се односе на јавни саобраћај у урбаним срединама, транспорт робе у ланцима снабдевања, разне типове достава као и транспорт различитих сировина. Неколико радова се односи на транспорт пољопривредних сировина као што су сирово млеко, шећерна трска, маслиново уље и др. Циљ овог одељка је да се истакну сличности и разлике између проблема који су предмет ове докторске дисертације и сличних проблема који су до сада изучавани у литератури.

Проблеми који су разматрани у овој докторској дисертацији, VSP и VSP-P имају специфична ограничења које произилазе из конкретне реалне ситуације. Као прво, при организацији транспорта шећерне репе у посматраној компанији, важно је спречити мешање сировина са различитих локација, због могућности добијања мешавине лошијег квалитета, при чему се не може извршити прецизна процена квалитета сировина са више локација. Овај услов директно имплицира да возилима није дозвољено да праве руте, односно свако возило у свакој тури може да посети само једну локацију. То ограничење важи чак и када се локација последњи пут посећује, без обзира што возило које прави ту туру не мора бити натоварено до пуног капацитета. Из тог разлога, уколико би се разматрани проблеми посматрали као варијанте проблема CVRP са вишеструким путовањима, прављење рута би се морало избећи. Идеја да се свака локација посматра као више клијената чији су захтеви мањи или једнаки од капацита возила би могла да спречи возила да праве руте. Међутим, у том случају, како би свака рута била типа фабрика-локација-фабрика, неопходно је да фабрика има на располагању возила адекватних различитих капацитета. То не одговара реалној ситуацији јер транспортне компаније обично имају ограничен број возила чији су капацитети фиксни и не обавезно једнаки. Уз то, неопходно је претпоставити да ће се свака локација бити посећена неколико пута док се не изврши транспорт целокупне количине која је на њој припремљена. Како су капацитети расположивих возила унапред познати, сваку локацију мора посетити више од

једног возила, али у различитим рутама, при чему свака рута мора имати облик фабрика-локација-фабрика. Ако возило последње посећује локацију пре њеног испражњења, то возило након посете датој локацији мора отићи право у фабрику без обзира колико је напуњено. Возилима се може дозволити да посећују више од једне локације у оквиру исте руте само под условом да је на свакој од локација унапред познат квалитет припремљене репе сваког радног дана. Тада би исто возило могло да посећује локације које имају једнак квалитет робе. Ово би значило да би квалитет робе требало испитивати на самој локацији, што захтева додатне трошкове и тешко је изводљиво у пракси. Осим тога, квалитет робе се може променити током дана због временских услова, нпр. јаке кише или током транспорта до фабрике. Такође, ако се посматрани проблем своди на проблем рутирања, неопходно би било укључити и растојања између сваке две локације. Такав концепт би повећао комплексност разматраних проблема, имајући у виду специфична ограничења у вези са временима полазака возила из фабрике (два возила се не могу истовремено утоварити на истој локацији и само p возила могу да се опслуже истовремено у кругу фабрике). Повећана сложеност проблема би значајно утицала на решавање и било би тешко обезбедити допустива решења.

Из наведених разлога, имајући у виду карактеристике разматраних проблема VSP и VSP-P, може се закључити да они припадају класи проблема VSP, а не VRP. Поред тога, анализирајући сложеност проблема, доказано је да је проблем НП-тежак управо свођењем на проблем распоређивања машина (одељак 4.5), што је такође један од индикатора који показује да разматрани проблеми припадају класи VSP.

Имајући у виду студије из литературе које се односе на транспорт пољопривредних сировина ([51], [60], [69], [72], [82] и [87]), може се закључити да се постојећи проблеми оптимизације транспорта пољпривредних сировина из литературе значајно разликују од проблема који су представљени у овој докторској дисертацији. Узрок представљају разлике међу типовима сировина које треба транспортовати (постојаност, начин утовара и истовара, контрола квалитета), разлике у типу и карактеристикама возила која се користе приликом транспорта и разлике у расположивој инфраструктури. Такође, функције циља у проблемима из литературе најчешће односе на минимизацију укупних трошкова који укључују трошкове транспорта и друге специфичне трошкове карактеристичне за разматране проблеме, док је циљ посматраних проблема

VSP и VSP-P минимизовати радно време возила која се користе.

Из анализе и поређења са радовима из литературе, може се закључити да разматрани проблеми VSP и VSP-P који су предмет ове докторске дисертације представљају нове проблеме који припадају класи проблема распоређивања возила, имајући у виду функцију циља и специфична ограничења из праксе.

Глава 5

Метахеуристички приступ решавању VSP-P

За решавање потпроблема распоређивања возила при транспорту шећерне репе VSP-P, дизајниране су две методе. Једна је заснована на основној методи променљивих околина (BVNS), а друга је типа похлепне стохастичко-адаптивне процедуре претраге (GRASP). У овом поглављу изложене су предложене методе са свим компонентама. Детаљи предложених BVNS и GRASP имплементација описани су у наредним одељцима.

5.1 Основна метода променљивих околина за решавање VSP-P

Основна метода променљивих околина (BVNS) истражује простор претраге користећи једну или више структура околине и састоји се из три основна корака: фазе размрдавања, локалне претраге и помераја. У фази размрдавања бира се на случајан начин решење из околине текућег решења, како би се појачала диверсификација и помогло претрази да изађе из локалног оптимума. У фази локалне претраге, алгоритам почиње од решења које је добијено као резултат фазе размрдавања и покушава да побољша тренутно решење истражујући једну или више његових околина. Уколико се пронађе боље решење од тренутно најбољег, добија се ново најбоље решење, што представља фазу помераја. Ови кораци се смењују све док се не задовољи критеријум заустављања.

Сви елементи предложене (BVNS) методе за решавање VSP-P изложени су

у следећим одељцима.

Репрезентација решења

Решење проблема $S = (s, aq)$ је представљено као уређен пар кога чине матрица s, чији су елементи цели ненегативни бројеви и низ ненегативних реалних бројева (aq). Матрица s се састоји из m врста дужине k_{max}, при чему i-та врста одговара возилу $i \in \{1, \ldots, m\}$ и садржи целе ненегативне бројеве $s_{ik} \in \{0, 1, \ldots, n\}$. Елемент s_{ik} означава индекс локације које возило i посећује у тури $k \in \{1, \ldots, k_{max}\}$. Прецизније, ако возило i посећује локацију $j \in \{1, \ldots, n\}$ у тури k, тада је $s_{ik} = j$. У случају када је k-та тура возила i виртуална, s_{ik} има вредност 0. Због једноставности, свако решење се генерише тако да се виртуалне туре, ако постоје, налазе увек на крају врста у матрици решења s.

Друга компонента решења S је низ (aq) ненегативних реалних бројева, који се назива *низ преосталих количина*. Низ (aq) чува информацију о преосталим количинама за транспорт на свакој од локација. Прецизније, дужина низа (aq) је $n + 1$, при чему елемент $aq(j)$ одговара локацији $j \in \{0, \ldots, n\}$, где $j = 0$ означава виртуалну туру. Количине робе које су преостале на локацији $j \in \{1, \ldots, n\}$ смештене су у $aq(j)$. На пример, $aq(5) = 30$ означава да се на локацији 5 налази 30 тона робе за транспорт. Како је тура $j = 0$ виртуална, поставља се $aq(0) = 0$. Вредности $aq(j)$ се освежавају током рада алгоритма, сваки пут када се добије ново решење.

Рачунање вредности функције циља

Свако решење $S = (s, aq)$ које се генерише током рада алгоритма, мора се евалуирати како би се донела одлука да ли се решење прихвата или не. Евалуација решења се врши рачунањем вредности функције циља на следећи начин. За свако возило $i \in I$, распоред тура је одређен вредностима s_{ik}, $k \in K$ у i-том реду матрице решења s. На основу вредности s_{ik}, рачуна се време поласка t_{ik} сваког возила $i \in I$ у свакој тури $k \in K$. У фази предпроцесирања, прво се израчуна неопходно време за опслуживање локације $j \in J$ као дужина трајања вожње од фабрике до локације j и назад, увећана за време трајања утовара и истовара робе. Како сва возила почињу своју прву туру у исто време (почетак радног дана), t_{i1} има вредност t_{start} за свако возило $i \in I$. Време

поласка t_{ik} возила $i \in I$ у тури $k \in \{2, \ldots, k_{max}\}$, добија се увећањем $t_{i,k-1}$ за време које је потребно за опслуживање локације коју посматрано возило i посећује у тури $k-1$. На овај начин, обезбеђено је да возило почиње следећу туру по завршетку претходне. Израчунате вредности времена полазака t_{ik} се уносе у матрицу T_d. Време завршетка рада за свако возило се добија увећањем времена поласка посматраног возила у његовој последњој невиртуалној тури ка одређеној локацији за време које је потребно за опслуживање те локације. Имајући у виду ограничења (4.1) и (4.9), вредност функције циља $T = f(S)$ се одређеује као максимална вредност међу временима завршетака рада свих посматраних возила:

$$T = \max_{i \in I} t_f(i), \tag{5.1}$$

где су

$$t_f(i) = t_{ik_{max}} + \frac{2}{v} \sum_{j \in J} d_j x^j_{ik_{max}} + (u+v) \sum_{j \in J} x^j_{ik_{max}}, \qquad \forall i \in I. \tag{5.2}$$

За виртуалне туре k, вредности t_{ik} су међусобно једнаке и представљају време завршетка последње невиртуалне туре возила i. У случају када је $T > t_{end}$, одговарајуће решење је недопустиво.

Пример 1. Посматрајмо инстанцу $T_{4,4,4}$ са следећим подацима: $m = 4$ локације, $n = 4$ возила, $k_{max} = 4$ туре, $c_1 = 120$t, $c_2 = 100$t, $c_3 = 50$t, $c_4 = 80$t, $C = 330$t, $CV = 27$t , $t_0 = 7$ дана, $t_1 = 4$ дана, $t_2 = 8$ дана, $t_3 = 5$ дана, $t_4 = 9$ дана, $d_1 = 55$km , $d_2 = 60$km, $d_3 = 40$km, $d_4 = 50$km, $v = 35$km/h, $u = 0.17$h, $w = 0.12$h, $t_{start} = 6$h и $t_{end} = 24$h. За посматрану инстанцу, матрица s_{opt} оптималног решења и одговарајућа времена полазака T_d су

$$s_{opt} = \begin{bmatrix} 4 & 4 & 3 & 3 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 2 & 4 & 0 \end{bmatrix} \quad \text{и} \quad T_d = \begin{bmatrix} 6,000 & 9,147 & 12,294 & 14,870 \\ 6,000 & 9,719 & 13,152 & 16,585 \\ 6,000 & 9,719 & 13,152 & 16,585 \\ 6,000 & 9,719 & 13,438 & 16,585 \end{bmatrix}.$$

Елементи низа преосталих количина за ово решење су $aq = (0, 12, 0, 0, 0)$.

Из матрице s_{opt}, може се видети да у оптималном решењу постоје три виртуалне туре. Оне се појављују код другог, трећег и четвртог возила и смештене су на крајевима одговарајућих редова матрице s_{opt}. Анализирањем улазних података, јасно је да су локације 2 и 4 прве које треба опслужити, јер на њима

роба стоји на отвореном $t_2 = 8$ и $t_4 = 9$ дана, редом. Из низа (aq), очигледно је да је сва роба прикупљена на локацијама 2 и 4 транспортована до фабрике, с обзиром да је $aq(2) = aq(4) = 0$. Превезене количине са локација 1, 2, 3 и 4 су 108, 100, 50 и 80 тона, редом, што у збиру износи 338 тона, што значи да су дневне потребе фабрике у износу од 330 тона задовољене. Времена завршетака за возила 2, 3 и 4 могу се директно прочитати из матрице времена полазака T_d, с обзиром да сва три возила имају виртуалну туру као последњу. Време завршетка возила 1 добија се увећањем последњег елемента у првом реду матрице T_d за време које је потребно да се опслужи локација 3. Дакле, прво возило је последње које завршава своје туре, те је време кад сва возила завршавају своје туре $T = 17,446$ часова, што је приближно 17 часова 26 минута и 46 секунди. Та вредност представља вредност функције циља оптималног решења.

Генерисање иницијалног решења

У предложеној имплементацији методе поменљивих околина, примењена је похлепна процедура за генерисање иницијалног решења $S = (s, aq)$. У првом кораку, свака локација се означи као хитна или не, у зависности од броја дана колико роба стоји на отвореном на тој локацији. Уколико роба на локацији $j \in J$ стоји на отвореном t_j дана и притом је $t_j > t_0$, посматрана локација се сматра хитном. За хитну локацију, није важно колико време стајања робе на отвореном премашује унапред одређен број дана t_0. Хитне локације имају висок приоритет и морају бити опслужене током радног дана. После идентификовања хитних локација, процедура наставља са испитивањем да ли се количине робе на хитним локацијама могу транспортовати коришћењем датог скупа возила у оквиру датог броја тура и радног времена. Уколико то није могуће, не постоји допустиво решење за дате улазне податке и BVNS алгоритам завршава са радом. У супротном, процедура за рачунање вредности функције циља сортира локације за опслуживање према два критеријума: хитности и удаљености од фабрике. Сортирана листа локација има следећу структуру: на врху листе налазе се локације које су хитне сортиране у неопадајући низ према растојању од фабрике, за којима следе нехитне локације, такође сортиране у неопадајући низ према удаљености од фабрике.

Иницијално, елементи матрице s иницијалног решења $S = (s, aq)$ су постављени на 0, док елементи низа (aq) имају иницијалне вредности једнаке количинама сакупљене робе на свакој од локација, тј. $aq(j) = c(j)$, $j = 1, \ldots, n$ и

$aq(0) = 0$. Затим се узима једна по једна локација сортиране листе и попуњава се матрица s, тако што се најпре попуни прва колона која одговара првим турама свих возила. Процедура наставља попуњавањем друге колоне матрице s, затим треће, итд., све док се не испразне све хитне локације и не задовоље потребе фабрике. Сваком возилу је дозвољено да започне невиртуалну туру уколико се она може завршити у оквиру радног времена. У случају да то није могуће, локација из сортиране листе која је на реду додељује се наредном возилу. Током генерисања матрице решења, преостале количине на локацијама које су смештене у низу (aq) се освежавају. Када је матрица решења формирана, процедура сортира елементе сваке од врста матрице у нерастући низ према растојању од фабрике. Оваквим сортирањем врста се обезбеђује да свако од возила најпре дуже, а потом краће туре из листе тура која је додељена датом возилу. На крају, на основу вредности s_{ik} сортиране матрице иницијалног решења, рачунају се вредности t_{ik} времена полазака сваког возила $i \in I$ у свакој тури $k \in K$.

Пример 2. Посматрајмо улазне податке инстанце $T_{4,4,4}$ разматране у примеру 1. Листа локација сортираних по приоритету је $(4, 2, 3, 1)$. Локације 2 и 4 су хитне локације, с обзиром да роба на њима стоји на отвореном дуже од $t_0 = 7$ дана. Приметимо да ће локација 4 бити опслужена пре локације 2, јер је она ближа фабрици у односу на локацију 2 ($d_4 = 50\text{km}$, $d_2 = 60\text{km}$). За хитним локацијама 4 и 2 следе нехитне локације 3 и 1, такође сортиране у неопадајући низ према растојању од фабрике ($d_3 = 40\text{km}$, $d_1 = 55\text{km}$). Овако формирана матрица s је:

$$s = \begin{bmatrix} 4 & 2 & 3 & 1 \\ 4 & 2 & 1 & 0 \\ 4 & 2 & 1 & 0 \\ 2 & 3 & 1 & 0 \end{bmatrix}.$$

Свака врста матрице се сортира у нерастући низ према удаљености посећених локација од фабрике. Сортирана матрица иницијалног решења s је:

$$s = \begin{bmatrix} 2 & 1 & 4 & 3 \\ 2 & 1 & 4 & 0 \\ 2 & 1 & 4 & 0 \\ 2 & 1 & 3 & 0 \end{bmatrix},$$

док је $aq = (0, 12, 0, 0, 0)$.

Како локација 4 има приоритет и на њој се налази 80t сакупљене робе, она мора бити посећена 3 пута. Када се сва роба са локације 4 превезе, локација 2 се посећује 4 пута. Локације 3 и 1 се посећују 2 и 4 пута, редом, пре него што се задовоље потребе фабрике.

Почевши од генерисане матрице решења s, елементи одговарајуће матрице времена полазака T_d се добијају на следећи начин. У фази претпроцесирања, прво се израчунају времена потребна за опслуживање сваке од локација, која су једнака времену трајања путовања у оба смера увећаном за време трајања утовара и истовара. У посматраном примеру, времена потребна за опслуживање локација 1, 2, 3, и 4 су: 3,433h, 3,719h, 2,576h и 3,147h, редом. Елементи прве колоне матрице T_d су једнаки времену које одговара почетку радног дана: $t_{i1} = t_{start} = 6\text{h}$, за свако $i \in I$. Елемент t_{id} у тури $d = 2, \ldots, k_{max}$ возила $i \in I$ једнак је времену поласка у претходној тури возила $i \in I$ увећаном за време које је потребно да се опслужи локација посећена у претходној тури. Из матрице решења s, може се приметити да сва возила, изузев првог, имају виртуалну последњу туру, чије је трајање једнако нули. Према томе, времена полазака у последњој тури возила 2, 3 и 4 једнака су временима завршетака за ова возила. Одговарајућа матрица времена полазака T_d, формирана на описани начин је:

$$T_d = \begin{bmatrix} 6,000 & 9,719 & 13,152 & 16,299 \\ 6,000 & 9,719 & 13,152 & 16,299 \\ 6,000 & 9,719 & 13,152 & 16,299 \\ 6,000 & 9,719 & 13,152 & 15,728 \end{bmatrix}.$$

На основу генерисане матрице T_d, може се видети да је возило 1 последње које завршава своје туре, с обзиром да је његово време поласка у последњој тури ка локацији 1 једнако 15,442h и оно завршава истовар у последњој тури у 18,875h. Стога је вредност функције циља генерисаног решења $T = 18,875\text{h}$, односно приближно 18h 52min и 30s.

Структуре околина

Предложена метода променљивих околина користи две структуре околина, означених са N^I и N^{II}. Околина N^I решења S добија се заменом локација у турама возила i, $i \in \{1, \ldots, m\}$ и неиспражњених локација које одговарају индексима ненула елемената низа преосталих количина. Прецизније, сусед S' из околине N^I решења S добија се заменом туре возила ка нехитној локацији

неком од тура ка неиспражњеној локацији, чији се индекс налази међу индексима ненула елемената низа aq. Такође, N^I-сусед се може добити и заменом виртуалне туре туром ка неиспражњеној локацији. Сложеност ове околине је, у најгорем случају, $m \cdot k_{max} \cdot (n+1)$. Околина N^{II} решења S садржи сва решења S' добијена тако што пар возила у S размењује по једну туру. Комплексност ове околине је у најгорем случају $\binom{m}{2} \cdot k_{max}^2$. Треба приметити да околина N^{II} чува допустивост решења. Са друге стране, трансформације којима је дефинисна околина N^I могу нарушити допустивост решења, јер постоји могућност да код N^I-суседа S' допустивог решења S потребе фабрике не буду задовољене. Међутим, следећи потез од решења S' у оквиру истог типа околине може произвести допустиво решење.

Може се доказати да су описане околине коректно дефинисане за посматрани проблем VSP-P. Прецизније, важи следећа теорема.

Теорема 5.1.1. Оптимално решење посматраног проблема распоређивања возила се може добити из било ког допустивог решења применом коначног броја горе описаних трансформација којима су дефинисане структуре околина N^I и N^{II}.

Доказ. Нека је $S = (s, aq)$ произвољно допустиво решење посматраног проблема VSP-P. Имајући у виду коришћену репрезентацију решења, матрица иницијалног решења s се састоји из m редова од којих сваки одговара по једном возилу, док низ преосталих количина aq са елементима $aq(j)$, $j = 0, \ldots, n$ чува информацију о расположивим количинама на локацијама $j \in \{1, \ldots, n\}$. Елемент $aq(0)$ се поставља на нулу, с обзиром да индекс 0 одговара виртуалној тури. Нека је $S^* = (s^*, aq^*)$ оптимално решење проблема VSP-P.

Да би се решење S трансформисало у оптимално решење S^*, потребно је извршити следеће кораке. Почевши од тура које одговарају првом возилу у решењу S, разматра се једна по једна тура из прве врсте матрице решења s и локација у посматраној тури се размењује са другом локацијом која се налази у тури неког другог возила (укључујући виртуалне туре и индексе низа aq). Локација се може заменити и нулом, што одговара виртуалној тури у s^*. Циљ је добити листу тура првог возила у матрици оптималног решења s^*. Процедура се наставља док све локације у листи тура сваког возила матрице s не буду смештене на „коректне" позиције, тј. на одговарајуће позиције из матрице оптималног решења s^*.

Приликом разматрања сваке од тура возила i, могућа су четири случаја:

Случај 1. Локација се већ налази на одговарајућој позицији у листи тура које одговарају возилу i. У овом случају, нема промене и поступак се наставља од следеће туре.

Случај 2. Локација у тури k возила i у решењу S може заменити своју позицију са локацијом $s(i_1, k_1)$ где је $i_1 > i$, тако да важи $s(i,k) = s^*(i,k)$. У овом случају, обе локације које мењају своја места су већ присутне у матрици решења s и једна од њих ће се преместити на одговарајућу позицију. Замена овог типа неће угрозити допустивост новодобијеног решења, с обзиром да потребе фабрике (у смислу количине транспортоване робе) остају непромењене и све хитне локације остају опслужене.

Случај 3. Може се десити да је потребно укључити нову локацију у матрицу решења s или да се нека локација која се већ налази у s мора посетити још једном. У овом случају, посматрана локација у листи тура возила i, уколико није хитна, се замењује траженом локацијом, односно индексом локације одговарајућег ненула елемента низа преосталих количина (aq).

Случај 4. Локација $s(i,k)$ из случаја 3. је хитна, а потребно је заменити ову локацију индексом локације одговарајућег ненула елемента низа преосталих количина (aq). Ова замена се може реализовати применом два потеза, при чему први потез, описан у случајевима 2. и 3. има за циљ да на позицију (i,k) матрице s постави локацију која није хитна, а затим се примењује други потез који је описан у случају 3.

Случај 5. У случају када локацију $s(i,k)$ треба заменити локацијом $s(i,k_1)$ за $k_1 > k$, потребно је узастопно применити два потеза описана у случају 3.

Наведене трансформације могу обезбедити смештање свих елемената матрице решења на коректне позиције, применом највише $2mk_{max}$ трансформација. □

Имплементација Основне методе променљивих околина за VSP-P

Структура предложене BVNS методе представљена је Алгоритмом 9. BVNS почиње процедуром за генерисање почетног решења S која је детаљно описана у одељку 5.1. Свака BVNS итерација почиње постављањем реда околине r на вредност 1. Затим се наставља, извршавањем три главна корака: *размрдавање*,

Алгоритам 9 Предложена BVNS метода за VSP-P

```
procedure VNS(Problem Data, r_max, t_max)
    Generate initial solution S;
    repeat
        r ← 1;
        while r ≤ r_max do
            S' ← ShakeN^I(S, r):                          //Shaking in N^I
            if S' is feasible then
                                                          //Local Search in N^II
                S' ← Sort_1(S');
                i_1 ← 1;  i_2 ← m;
                k_1 ← 1; k_2 ← k_max;
                imp ← 0;
                S'' ← S';
            while i_1 < i_2 do
                if max(t̂_f(i_1), t̂_f(i_2)) < max(t'_f(i_1), t'_f(i_2)) then
                    S' ← Exchange(i_1, i_2, k_1, k_2, S');
                    S' ← Sort_2(S');
                    k_1 ← k_max; k_2 ← 1;
                    imp ← 1;
                else
                    if (k_1 = k_max) ∧ (k_2 = 1) then
                        if (imp=1) then
                            if f(S') < f(S'') then
                                S'' ← S';
                            imp ← 0;
                            k_1 ← 1; k_2 ← k_max;
                        else
                            i_2 ← i_2 − 1;
                            k_1 ← 1; k_2 ← k_max;
                    else
                        if (k_2 > 1) then
                            k_2 ← k_2 − 1;
                        else
                            if (k_1 < k_max) then
                                k_1 ← k_1 + 1; k_2 ← k_max;
            if f(S'') < f(S) then                         //Move or Not step
                S ← S'';
                r ← 1;
            else
                r ← r + 1;
    until SessionTime ≥ t_max
```

локална претрага и *померај* у оквиру петље са променама околина, све док је $r \leq r_{max}$. BVNS итерације се понављају док се не достигне лимит времена

извршавања од t_{max}.

Фаза размрдавања се реализује у околини N^I реда r, $r = 1, \ldots, r_{max}$, извршавањем r узастопних потеза којим је ова околина дефинисана, при чему се потези бирају на случајан начин. После сваког потеза у околини N^I, елементи низа преосталих количина се ажурирају. Треба приметити да је дозвољена замена само неиспражњених локација које одговарају индексима ненула елемената у низу преосталих количина и нехитних локације из тура посматраног возила. На овај начин, све хитне локације ће остати опслужене и количина робе превезена у фабрику са сваке од локација неће бити већа од припремљене количине на посматраној локацији. Међутим, овај потез може променити укупну количину робе која стиже у фабрику, што се може одразити на допустивост решења. Из тог разлога, решење добијено у фази размрдавања ће бити прослеђено фази локалне претраге само уколико овај услов допустивости није нарушен тј. ако су задовољене дневне потребе фабрике. У супротном, фаза размрдавања BVNS имплементације се понавља за исту вредност r.

У фази локалне претраге предложене BVNS методе, истражује се околина N^{II} решења $S' = (s', aq')$ на систематичан начин. Прво се елементи у свакој врсти матрице решења s' сортирају у нерастућем поретку према удаљености посећених локација од фабрике. Потом се возила сортирају у нерастући низ на основу вредности времена завршетака своје последње туре, што се реализује процедуром $Sort_1$. Фаза локалне претраге почиње од првог ($i_1 = 1$) и последњег ($i_2 = m$) возила из сортиране листе и покушава да размени локације прве туре ($k_1 = 1$) возила i_1 и последње туре ($k_2 = k_{max}$) возила i_2. На овај начин покушава се размена најдуже и најкраће туре возила i_1 и i_2, редом. Рачунају се две помоћне вредности $\hat{t}_f(i_1)$ и $\hat{t}_f(i_2)$ које представљају времена завршетка рада возила i_1 и i_2, редом, под претпоставком да је извршена замена тура. Израчунате вредности $\hat{t}_f(i_1)$ и $\hat{t}_f(i_2)$ се користе у циљу евалуације одговарајућег потеза. Уколико потез води ка смањењу максималног завршног времена возила i_1 и i_2, замена тура k_1 и k_2 се реализује процедуром $Exchange$. Након тога, на новодобијено решење се примењује процедура $Sort_2$, која сортира туре два посматрана возила i_1 и i_2 у нерастући низ према удаљености посећених локација од фабрике. Такође, процедура $Sort_2$ сортира и листу возила у нерастући низ према времену завршетка. Ако размена тура k_1 и k_2 не доводи до побољшања максималног од времена завршетка возила i_1 и i_2, процедура локалне претраге наставља кроз сортирану листу тура возила i_2 покушавајући да размени, једну

по једну, туру k_2, $k_2 < k_{max}$ и туру k_1 возила i_1, све док се не добије побољшање или не стигне до прве туре возила i_2, односно док k_2 не добије вредност 1, без побољшања. Ако се не добије побољшање ни за $k_2 = 1$ процедура посматра сортиране туре возила i_1 и покушава да размени, једну по једну туру k_1, $k_1 \leq k_{max}$ туром k_2 возила i_2 све док се не оствари побољшање или се достигне $k_1 = k_{max}$ без побољшања. Уколико не дође до побољшања разменом тура возила i_1 и i_2, описани кораци се понављају над возилом i_1 и возилима $i_2 - 1, i_2 - 2, \ldots, i_1 - 1$. Најбоље пронађено решење у фази локалне претраге се чува у S". Ако је локални оптимум S" бољи од текућег решења S, у односу на вредност функције циља, врши се замена S са S" и алгоритам наставља претрагу у оквиру околине N^I реда $r = 1$. У супротном, ред r се повећава на $r + 1$. Тиме је завршена и фаза помераја. Алгоритам се завршава када се достигне задати лимит времена извршавања.

5.2 Похлепна стохастичко-адаптивна процедура претраге за решавање VSP-P

Поред основне методе променљивих околина која је описана у претходном одељку, за решавање проблема VSP-P дизајнирана је и похлепна стохастичко-адаптивна процедуре претраге. GRASP метода је итеративни поступак чија се свака итерација састоји из два корака: фазе похлепне стохастичке процедуре за конструкцију решења и локалне претраге. Итерације се смењују док се не задовољи критеријум заустављања. Због међусобног поређења резултата предложених BVNS и GRASP метода, коришћен је исти критеријум заустављања, односно GRASP алгоритам такође завршава са радом када се достигне лимит времена извршавања од t_{max}. Елементи предложене методе су детаљно описани у наредним одељцима.

Фаза предпроцесирања

Предложена GRASP метахеуристика за посматрани проблем VSP-P почиње са фазом предпроцесирања која се извршава пре фазе похлепне стохастичке процедуре за конструкцију решења (енгл. *Greedy Randomised Construction - GRC*) и фазе локалне претраге (енгл. *Local Search - LS*). Фаза предпроцесирања се састоји из два корака. Прво се локације на којима се налази сакупљена роба

сортирају на исти начин као при примени основне методе променљивих околина: на врху листе налазе се хитне локације које су сортиране у неопадајући низ према удаљености од фабрике, а за њима долазе локације које нису хитне, сортиране на исти начин према удаљености од фабрике (видети одељак 5.1). У другом кораку фазе предпроцесирања, за сваку локацију j у сортираној листи, рачунају се трошкови повећања (енгл. *incremental costs*), у ознаци $ic(j)$ на следећи начин:

$ic(j) = \frac{2}{v}d_j + u + w$, ако локација j није хитна и

$ic(j) = 0$, уколико је j хитна локација.

Постављање $ic(j)$ на вредност 0 за хитне локације води ка већој вероватноћи избора хитних локација приликом конструкције решења у GRC фази, а самим тим се вероватноћа генерисања недопустивих решења у GRC фази значајно смањује.

Похлепна стохастичка процедура конструкције решења

GRC фаза је процес који додаје један по један елемент парцијалном решењу док се не формира комплетно решење. За разлику од BVNS методе, репрезентација решења код дизајниране GRASP методе не користи низ преосталих количина (aq), па стога репрезентација решења подразумева само матрицу решења, односно важи $S \equiv s$. Елементи који се додају парцијалном решењу изабрани су из листе кандидата (енгл. *candidate list - cl*). Листа кандидата се формира на основу вредности похлепне функције оцењивања. Похлепна функција оцењивања одражава промене у вредности функције циља које ће додати елемент проузроковати када се укључи у парцијално решење. Стохастички аспект GRASP методе се одражава у избору једног од кандидата са листе на случајан начин, који не мора бити обавезно најбољи. Иницијално, вероватноће избора елемената листе кандидата су једнаке, а затим се периодично мењају после сваких γ итерација, у зависности од квалитета најбољих решења пронађених у претходним GRASP итерацијама. Имајући у виду начин промене вероватноћа, предложена варијанта GRASP методе представља Реактивну GRASP методу (енгл. *Reactive GRASP*), која детаљније објашњена у одељку 3.2.

На почетку GRC фазе, за кандидат листу cl узима се сортирана листа локација, која је добијена у фази предпроцесирања. Иницијално, дужина листе

кандидата је једнака броју локација, тј. $lenghth(cl) = n$, али како GRC фаза напредује, $lenghth(cl)$ се може смањивати, с обзиром да се неке локације могу испразнити током радног дана. За сваку локацију j листе cl, чува се информација о преосталој количини робе на тој локацији и та информација се освежава сваки пут када се приликом конструкције решења одабере локација j.

Имајући у виду структуру листе кандидата и начин рачунања трошкова повећања, доња и горња граница интервала трошкова повећања на следећи начин: $ic_{min} = ic(cl(1))$ и $ic_{max} = ic(cl(length(cl)))$ се рачунају. Рестриктивна кандидат листа (енгл. *restricted candidate list - rcl*) садржи локације из cl чије су вредности трошкова повећања мање од задате горње границе. Прецизније, кандидат локација j ће припадати rcl уколико важи $ic(j) \in [ic_{min}, ic_{min} + \alpha(ic_{max} - ic_{min})]$, где је α параметар из интервала $(0, 1)$.

Код основне варијанте GRASP методе, α има фиксну вредност у свим итерацијама, при чему $\alpha = 0$ одговара детерминистичкој хеуристици, јер се у том случају GRC фаза се своди на класични похлепни алгоритам, док се за веће вредности параметра α конструкцији решења врши на потпуно случајан начин. Основна идеја реактивне GRASP методе је да се вредност параметра α самоподешава током итерација алгоритма у зависности од резултата из претходних итерација (видети [76]). Код реактивне GRASP методе, адекватна вредност параметра α се обично бира из релативно малог (дискретног) скупа допустивих вредности. Вероватноће избора једне од допустивих вредности за α се периодично освежавају током извршавања алгоритма према вредностима функције циља које одговарају најбољим решењима, пронађеним у претходним итерацијама.

У GRC фази предложене GRASP методе, коришћен је скуп од ν допустивих вредности за α, тј., $\alpha_i = \frac{i}{\nu}$, $i = 1, \ldots, \nu$. Нека је p_i вероватноћа избора α_i, $i = 1, \ldots, \nu$. Иницијално, све допустиве вредности параметра α имају исте вероватноће избора: $p_i = \frac{1}{\nu}$, $i = 1, \ldots, \nu$. Нека је f^* вредност функције циља тренутно најбољег решења S^* и нека је f_i просечна вредност функције циља свих GRASP решења која су добијена коришћењем вредности α_i за $i = 1, \ldots, \nu$ током претходних итерација алгоритма. Вероватноће p_i избора α_i се периодично освежавају на следећи начин: $p_i = q_i / \sum_{j=1}^{\nu} q_j$, где је $q_i = f^*/f_i$, $i = 1, \ldots, \nu$. У свакој од првих ν итерација, GRASP се извршава са једном од вредности $\alpha_i = \frac{i}{\nu}$, $i = 1, \ldots, \nu$, како би се иницијализовале вредности f_i. Прерачунавање вероватноћа према описаној стратегији врши се после сваких γ итерација, по-

чевши од $(\nu+1)$. итерације. У дизајнираној GRASP имплементацији, вредност параметра ν је 10.

Алгоритам 10 Похлепна стохастичка процедура конструкције решења

```
procedure GRC(Sorted list of locations, ic, α, ind)
    cl ← Sorted list of locations;
    k ← 1;
    i ← 1;
    S ← 0;
    ind ← 1;
    repeat
        ic_min ← ic(cl(1));
        ic_max ← ic(cl(length(cl)));
        rcl ← ∅;
        for each j ∈ cl do
            if ic(j) ≥ ic_min) ∧ (ic(j) ≤ ic_min + α(ic_max − ic_min) then
                rcl ← rcl ⋃ {j};
        Randomly select j from rcl;
        if vehicle i can serve location j in tour k within working time then
            S(i, k) ← j;
        else
            if i < m then
                i ← i + 1;
            else
                if k < k_max then
                    k ← k + 1;
                    i ← 1;
                else
                    ind ← 0 ;
    until (feasible solution is obtained or ind=0)
    return (S, ind);
```

Након формирања rcl, на случајан начин се бирају локације и придружују матрици решења S. Иницијално, сваки елемент матрице S има вредност 0, и током GRC фазе, S се попуњава вертикално, колона по колона. Најпре се формира прва колона матрице решења S, тј. одређују се локације које ће свако од возила посетити у првој тури, затим се попуњава друга колона матрице S која одговара другој тури, итд. Локација $j \in rcl$, изабрана на случајан начин, се додељује возилу i у тури k, само ако се опслуживање локације j од стране возила i у тури k може завршити у оквиру радног времена. Уколико то није случај, бира се следеће возило $i+1$ као кандидат за опслуживање локације j. Када се локација j придружи неком од возила у датој тури, преостале количине на

локацији j се освежавају, а сума количина које стижу у фабрику се повећава за количину која је довезена са локације j у посматраној тури. Када се локација испразни, искључује се из листе кандидата. Наведени кораци се понављају све док се не задовоље потребе фабрике и не испразне све хитне локације. Ако конструисано решење није допустиво тј. хитне локације се нису испражњене или потребе фабрике нису задовољене, GRC фаза се понавља. Псеудокод предложене GRC процедуре представљен је Алгоритмом 10.

Локално претраживање

Решење S које је генерисано у GRC фази прослеђује се фази локалне претраге (LS). У предложеној GRASP методи имплементирана је иста LS процедура која је коришћена код BVNS и детаљно описана у одељку 5.1. Једина разлика је у томе што локална претрага код GRASP методе не користи низ преосталих количина, имајући у виду репрезентацију решења. Примењена локална претрага систематски истражује околину N^{II} текућег решења S и враћа најбоље пронађеног суседа S'.

Алгоритам 11 Предложена GRASP метода за VSP-P

```
procedure GRASP(ProblemData, γ, t_max)
    n_iter ← 0;
    Initialize probabilities;
    Create Sorted list of locations;
    Calculate ic;
    repeat
        ind ← 0;
        while ind = 0 do
            (S, ind)←GRC(Sorted list of locations, ic, α, ind); //Solution construction
        S' ← LocalSearchN^II(S):                                //Local Search step
        if f(S') < f(S) then                                    //Update solution
            S ← S';
        n_iter ← n_iter + 1;
        if n_it mod γ = 0 then
            Update probabilities
    until SessionTime ≥ t_max
```

Имплементација Похлепне стохастичко-адаптивне процедуре претраге за VSP-P

Након фазе предпроцесирања, у предложеној GRASP методи, следи фаза конструкције решења која се понавља све док се не добије допустиво решење, што је регулисано параметром ind. Решење S конструисано у GRC фази прослеђује се фази локалне претраге. Локални оптимум S' добијен у овој фази представља резултат једне GRASP итерације. Итерације се смењују док се не задовољи критеријум заустављања, односно док се не достигне унапред одређена горња граница времена извршавања. Коначно решење GRASP методе представља најбоље решење међу резултатима свих GRASP итерација. Псеудокод имплементиране GRASP методе за проблем VSP-P представљен је Алгоритмом 11.

Глава 6

Метахеуристички приступ решавању VSP

За решавање тест примера проблема VSP које су већих димензија, дизајниране су три варијанте метахеуристичких метода променљивих околина: BVNS, побољшани BVNS и SVNS. У наредним одељцима објашњени су сви детаљи предложених метахеуристичких приступа решавању проблема VSP.

6.1 Основна и Адаптивна метода променљивих околина за решавање VSP

Иако се предложене BVNS и SVNS методе дизајниране за VSP разликују по структури, оне имају неколико заједничких елемената: репрезентацију решења, начин рачунања вредности функције циља, стратегију генерисања иницијалног решења и коришћене структуре околина. У наредним одељцима су најпре описани заједнички елементи, а затим изложене структуре BVNS и SVNS имплементација, респективно.

Репрезентација решења

У имплементацијама предложених BVNS и SVNS метода, решење S је представљено у виду матрице целих ненегативних бројева, која се састоји из m врста дужине k_{max}, где m представља број возила а k_{max} максималан број тура које возило може направити током радног дана. i-ти ред матрице S одговара i-том возилу $i \in \{1, \ldots, m\}$ и садржи целе ненегативне бројеве $s_{ik} \in \{0, 1, \ldots, n\}$, где

s_{ik} означава индекс локације коју возило i посећује у тури $k \in \{1, \ldots, k_{max}\}$. Прецизније, ако возило i посећује локацију $j \in \{1, \ldots, n\}$ у тури k тада је $s_{ik} = j$. Такође, s_{ik} може имати и вредност нула, што означава случај када је тура k возила i виртуална. Као што је објашњено у одељку 4.3, коришћењем концепта виртуалних тура, постиже се да у репрезентацији решења сва возила имају једнак број тура. Због једноставности, виртуалне туре (уколико постоје) су увек смештене на крају врста које одговарају возилима.

Рачунање вредности функције циља

За дату матрицу решења S, вредност функције циља се рачуна на следећи начин. Вредности s_{ik}, $i \in I, k \in K$ матрице S дефинишу низ тура за свако возило $i \in I$. На основу елемената s_{ik}, рачунају се вредности t_{ik} које представљају времена полазака возила i у тури k. Најпре се рачунају времена која су неопходна за опслуживање локација $j \in J$ као време трајања вожње од фабрике до локације j и назад, увећано за време трајања истовара и утовара. Како сва возила започињу своју прву туру у исто време, t_{i1} се поставља на вредност која одговара почетку радног времена t_{start} за свако возило $i \in I$. Време поласка t_{ik} возила $i \in I$ у тури $k \in \{2, \ldots, k_{max}\}$ једнако је времену $t_{i,k-1}$ увећаном за време које је неопходно да се опслужи локација посећена у тури $k-1$. Овим је обезбеђено да возило не може почети нову туру док се не заврши претходна. Када се, на описани начин, израчунају времена полазака t_{ik}, процедура наставља испитивањем задовољености следећа два услова: не може се истовремено вршити утовара два возила на истој локацији и највише p возила се може истоварити у исто време у кругу фабрике. У случају када је разлика између времена стизања два возила на исту локацију мања од времена трајања утовара, време поласка возила које је стигло касније се повећава за најмању вредност која разрешава конфликт. Уколико, у неком тренутку, постоји више од p возила за истовар у кругу фабрике, p времена полазака која имају најмању вредност остају непромењена, док се преостала времена полазака повећавају за најмању вредност, која разрешава конфликт. Ова два корака се смењују све док постоје наведени конфликти који нарушавају допустивст решења. Добијена времена полазака t_{ik} се смештају у матрицу T_d. Време завршетка за свако возило се рачуна као време поласка у његовој последњој непразној тури увећено за време опслуживања локације која је посећена у тој тури. Имајући у виду ограничења (4.1) и (4.9), вредност функције циља T је једнака максималној од свих вредности

времена завршетака радног дана коришћених возила, тј. важи релација:

$$T = \max_{i \in I}\{t_{ik_{max}} + \frac{2}{v}\sum_{j \in J} d_j x^j_{ik_{max}} + (u+v)\sum_{j \in J} x^j_{ik_{max}}\}. \tag{6.1}$$

У случају када је $T > t_{end}$ добијено решење се одбацује због недопустивости.

Пример 3. Нека је дата инстанца $T_{5,2,4}$ са $m = 5$ локација, $n = 2$ возила, $k_{max} = 4$ тура, $c_1 = 50$t, $c_2 = 50$t, $c_3 = 30$t, $c_4 = 80$t, $c_5 = 20$t, $C = 180$t, $CV = 27$t, $t_0 = 7$ дана, $t_1 = 4$ дана, $t_2 = 9$ дана, $t_3 = 5$ дана, $t_4 = 8$ дана, $t_5 = 3$ дана, $d_1 = 50$km, $d_2 = 30$km, $d_3 = 40$km, $d_4 = 20$km, $d_5 = 35$km, $v = 35$ km/h, $u = 0.17$h, $w = 0.12$h, $t_{start} = 6$h и $t_{end} = 24$h, $p = 1$. Времена неопходна за опслуживање локација 1, 2, 3, 4 и 5, рачуната на основу улазних параметара су: 3,147h, 2,004h, 2,576h, 1,433h и 2,290h, редом. За ову инстанцу, оптимално решење S_{opt} и одговарајућа матрица времена полазака T_d су:

$$S_{opt} = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 3 & 4 & 4 & 4 \end{bmatrix} \quad \text{и} \quad T_d = \begin{bmatrix} 6.000 & 8.004 & 11.151 & 13.155 \\ 6.000 & 8.576 & 10.009 & 11.442 \end{bmatrix}.$$

Из решења S_{opt}, може се видети да постоји само једна виртуална тура у оптималном решењу, на крају првог реда матрице S_{opt}, што значи да прво возило у последњој тури остаје у кругу фабрике. Анализирањем улазних података, јасно је да локације 2 и 4 морају бити опслужене, с обзиром да роба стоји на отвореном $t_2 = 9$ и $t_4 = 8$ дана, што прелази лимит од $t_0 = 7$ дана. Анализирајући решење S_{opt}, очигледно је да су локације 2 и 4 испражњене у току радног дана, што је и био захтев, јер су у питању хитне локације. Локације 2 и 4 су посећене 2 и 3 пута, редом, што је довољно за транспорт $c_2 = 50$t и $c_4 = 80$t, имајући у виду капацитет возила од $CV = 27$t. Транспортоване количине робе са локација 1, 2, 3, 4 и 5 су: 27t, 50t, 27t, 80t и 0t, редом. Дакле, укупно је транспортовано 184t, чиме су дневне потребе фабрике од 180t задовољене. Време завршетка рада првог возила (13.155h) може се прочитати директно из матрице времена полазака T_d, с обзиром да је његова последња тура виртуална. Време завршетка другог возила (12.855h) добија се додавањем времена које је потребно за опслуживање локације 4 последњем елементу другог реда матрице T_d. Очигледно, прво возило је последње које завршава своје туре. Према томе, вредност функције циља је $T = 13.155$h (приближно 13h 9min и 18s).

Генерисање иницијалног решења

При генерисању иницијалног решења, предложене BVNS и SVNS методе такође користе сортирање локација према два критеријума, као и у случају BVNS методе за проблем VSP-P (видети одељак 5.1). Први критеријум је хитност, јер хитне локације имају приоритет и морају бити опслужене током радног дана. Други критеријум за сортирање локација је њихова удаљеност од фабрике. Према томе, локације су сортиране у листу на следећи начин. На почетку листе налазе се хитне локације, сортиране у неопадајући низ према удаљености од фабрике. Затим следе локације које нису хитне сортиране на исти начин.

Процедура за генерисање иницијалног решења S_{init} узима једну по једну локацију из сортиране листе и попуњава прве колоне метрице решења. Када се формирају прве туре свих возила (прва колона матрице попуњена) процедура наставља попуњавањем друге колоне матрице решења, затим треће итд. За свако возило, последња невиртуална тура ка некој локацији може почети уколико се може уклопити у радно време. Уколико то није могуће, посматрана тура и све наредне туре за то возило постају виртуалне и његов распоред је формиран. Коначно, када се задовоље потребе фабрике и испразне хитне локације, преостале туре до k_{max} (уколико их има) постају виртуалне, за свако возило.

Пример 4. Нека је дата инстанца са подацима као у Примеру 3. Листа локација сортираних према приоритету је $(4, 2, 5, 3, 1)$. Локације 4 и 2 су хитне, а локација 4 је ближа фабрици него локација 2 ($d_4 = 20$ километара, $d_2 = 30$ километара). Затим следе локације које нису хитне: 1, 3 и 5, сортиране такође у неопадајући низ према удаљености од фабрике ($d_5 = 35$ километара, $d_3 = 40$ километара, and $d_1 = 50$ километара). Дакле, иницијално решење S_{init} је

$$S_{init} = \begin{bmatrix} 4 & 4 & 2 & 3 \\ 4 & 2 & 5 & 3 \end{bmatrix}.$$

Како је локација 4 прва у сортираној листи и на њој се налази 80t прикупљене робе, она мора бити посећена 3 пута. Затим, хитна локација 2, која је друга по реду у сортираној листи, мора бити посећена 2 пута. Након тога, разматрају се нехитне локације 5,3 и 1 према редоследу који одговара њиховим позицијама у сортираној листи локација.

На основу елемената генерисане матрице S_{init} рачунају се времена t_{ik} полазака возила из фабрике. Као што је наведено у овом одељку приликом опи-

сивања начина рачунања вредности функције циља, најпре се време поласка у првој тури за сва возила поставе на вредност која је једнака почетку радног времена. Затим се, за сваку наредну туру време поласка рачуна тако што се време поласка у претходној тури увећа за време потребно за опслуживање локације која је посећена у претходној тури посматраног возила. На крају, проверавају се услови допустивости који се односе на сусрете возила на свакој од локација и ограниченом броју возила који се могу истовремено опслужити у кругу фабрике. Све док постоји конфликт врше се неопходна повећавања вредности времена полазака, на начин описан у подсецији која се односи на рачунања вредности функције циља. Добијене вредности уносе се у одговарајућу матрицу времена полазака T_d:

$$T_d = \begin{bmatrix} 6,000 & 7,433 & 8,866 & 10,870 \\ 6,170 & 7,603 & 9,607 & 11,897 \end{bmatrix}.$$

Из генерисане матрице T_d, може се приметити да је друго возило последње које завршава своје туре, с обзиром да је време поласка другог возила у његовој последњој тури (ка локацији 3) 11,897h а завршетак истовара је у 14,473h. Крај радног времена првог возила је 13,446h. Према томе, вредност функције циља генерисаног решења S_{init} је $T = 14,473$h (проближно 14h 28min и 23s). Иницијално решење генерисано описаном процедуром је допустиво, али не и оптимално.

Структуре околина

Предложени BVNS и SVNS алгоритми користе три типа структура околина, означених са N^I, N^{II} и N^{III}. Околина N^I решења S се дефинише на следећи начин: N^I-сусед S' од S се добија избором пара возила који размењује пар својих невиртуалних тура. Сложеност ове околине је у најгорем случају је $\binom{m}{2} \cdot k_{max}^2$, па се ова околина користи само у фази размрдавања. Околина N^{II} се састоји из свих решења која се добијају разменом позиција две невиртуалне туре једног возила у посматраног решења S. Комплексност ове околине у најгорем случају је $m \cdot \binom{k_{max}}{2}$. Слабија сложеност околине N^{II} у односу на околину N^I представља њену предност због чега је ова околина је коришћена и у фази размрдавања и у фази локалне претраге. Коначно, околина N^{III} је дефинисана на следећи начин: за свако возило, свака нехитна локација се замени са локацијом која није испражњена. С обзиром да остали услови допустивости решења

нису угрожени оваквом трансформацијом, потребно је само проверити да ли су потребе фабрике задовољене. У случају да је тај услов испуњен, решење се сматра допустивим. Сва допустива решења се потом додатно побољшавају, заменом последње невиртуалне туре сваког возила, осим у случају када оно посећује хитну локацију, виртуалном туром. Поново, прихватају се само она решења код којих су задовољене потребе фабрике. Сложеност околине N^{III} у најгорем случају је nm^2k_{max}. Ова околина се користи у локалној претрази само у случају када побољшање није пронађено у оквиру околине N^{II}.

Може се доказати да су описане околине коректно дефинисане за посматрани проблем VSP. Прецизније, важи следеће тврђење.

Теорема 6.1.1. Оптимално решење проблема распоређивања возила VSP се може добити из било ког допустивог решења применом коначног броја горе описаних трансформација којима су дефинисане структуре околина N^I, N^{II} и N^{III}.

Доказ. Нека је S било које иницијално решење посматраног проблема VSP. Имајући у виду описани начин репрезентације решења, матрица S се састоји из m редова од којих сваки одговара по једном возилу. Нека је S^* оптимално решење проблема VSP.

Да би се решење S трансформисало у оптимално решење S^*, потребно је извршити следеће кораке. Почевши од тура које одговарају првом возилу матрице S, испитује се једна по једна тура и локација у посматраној тури се размењује са другом локацијом која се налази у другој тури истог или произвољној тури неког другог возила. Локација се може заменити и нулом, што одговара виртуалној тури у S^*. Циљ је добити листу тура првог возила у матици оптималног решења S^*. Процедура се наставља док све локације у листи тура сваког возила матрице S не буду смештене на „коректне" позиције, тј. на одговарајуће позиције у матрици оптималног решења S^*. Приликом разматрања сваке од тура возила i, могући су следећи случајеви:

Случај 1. Локација се већ налази на коректној позицији у листи тура које одговарају возилу i. У овом случају, нема промене и поступак се наставља од следеће туре.

Случај 2. Локација у тури k возила i у решењу S може заменити своју позицију са локацијом $S(i, k_1)$ где је $k_1 > k$, тако да важи $S(i, k) = S^*(i, k)$. У овом случају, обе локације које мењају своја места су већ присутне у матрици

решења S међу турама истог возила , тако да ће се једна од посметраних локација преместити на одговарајућу позицију. Замена овог типа неће угрозити допустивост новодобијеног решења, с обзиром да потребе фабрике у смислу количине транспортоване робе остају непромењене и хитне локације остају опслужене. Времена полазака возила i се рачунају на раније описани начин тако да задовољавају преостала ограничења.

Случај 3. Локација у тури k возила i у решењу S може заменити своју позицију са локацијом $S(i_1, k_1)$ где је $i_1 > i$, тако да важи $S(i,k) = S^*(i,k)$. И у овом случају, обе локације које мењају своја места су већ присутне у матрици решења S, па решење остаје допустиво из истих разлога као и у претходном случају.

Случај 4. Може се десити да је потребно укључити нову локацију у матрицу решења S (укључујући и локацију 0 која означава виртуалну туру) или да се нека неиспражњена локација која се већ налази у S мора посетити још једном. У овом случају, ако посматрана локација у листи тура посматраног возила i није хитна она се замењује траженом локацијом. Такође, посматрана локација се може заменити нулом, ако је потребно, али само ако се она налази у последњој невиртуалној тури посматраног возила.

Случај 5. Ако је посматрана локација $S(i,k)$ из случаја 4. хитна а треба је заменити неком неиспражњеном локацијом, неопходно је применити два узастопна потеза. Један од потеза је описан случајевима 2. или 3., како би се на позицију (i,k) матрице S поставила локација која није хитна, док је други потез описан случајем 4.

Случај 6. Уколико се локација у случају 4. налази у тури посматраног возила која није последња од невиртуалних тура, а треба је заменити нулом како би одговарала оптималном решењу, то значи да су у оптималном решењу посматрана тура и све преостале туре посматраног возила виртуалне. У том случају над посматраним решењем неопходно је коначан број пута применити трансформацију којом се последња тура посматраног возила замењује виртуалном туром. Случај када је нека од тих локација хитна решава се потезима описаним у случају 5.

Применом трансформација описаних у случајевима 1-6 могуће је сместити све елементе матрице произвољног решења S на одговарајуће позиције у матрици оптималног решења S^*. Број трансформација које треба реализовати је највише $2 \cdot mk_{max}$. □

6.2 Имплементација Основне методе променљивих околина

Структура предложене BVNS методе представљена је Алгоритмом 12. BVNS алгоритам почиње генерисањем иницијалног решења S, постављањем $r = 1$, и иницијализацијом бројача итерација које нису довеле до побољшања n_{it} на вредност 1. Затим се три основна корака *фаза размрдавања*, *локална претрага* and *померај* понављају унутар петље у којој се мењају околине док год је $r \leq r_{max}$. Фаза размрдавања користи околине N^I и N^{II}, при чему се прво истражује околина N^I за све вредности $r = 1, \ldots, r_{max}$, а затим се размрдавање врши у околини N^{II}, такође за све вредности $r = 1, \ldots, r_{max}$. Промена околина је контролисана параметром λ. Размрдавање у околинама N^I и N^{II} реда r се изводи понављањем потеза у одговарјућој околини r пута, при шему се потези бирају на случајан начин.

У фази локалне претраге предложене BVNS методе, прво се истражује околина N^{II} реда 1 користећи стратегију најбољег побољшања. Ако побољшање у овој околини није пронађено, локална претрага се завршава враћајући најбоље побољшање текућег решења. У супротном, претрага наставља у околини N^{III} реда 1. Након истраживања околине N^{III}, фаза локалне претраге у BVNS методи се завршава и најбоље пронађено решење сачувано је у S". Ако је локални оптимум S" бољи од тренутног решења S (у односу на вредност функције циља $f(\cdot)$), врши се замена S са S" и алгоритам наставља претрагу у оквиру околине N^I реда $r = 1$, у супротном r се поставља на $r + 1$, што представља корак помераја.

Описани кораци се понављају све док се не задовољи бар један од критеријума заустављања: прекорачена горња граница времена извршавања t_{max} или максималан број итерација без побољшања $iter_{max}$. На овај начин, постиже се компромис између квалитета добијених решења и времена извршавања алгоритма. Вредности пареметара који се користе за проверу критеријума заустављања t_{max} и $iter_{max}$ обично зависе од димензије инстанце проблема.

Алгоритам 12 Предложена BVNS метода за VSP

procedure BVNS($ProblemData, r_{max}, t_{max}, iter_{max}$)
  Generate initial solution S;
  repeat
    $r \leftarrow 1$;
    $n_{it} \leftarrow 1$;
    $\lambda \leftarrow 1$;
    while $r \leq r_{max}$ **do**
      if λ **then** //*Shaking step*
        $S' \leftarrow ShakeN^{I}(S, r)$;
        if $r = r_{max}$ **then**
          $r \leftarrow 0$;
          $\lambda \leftarrow 0$;
      else
        $S' \leftarrow ShakeN^{II}(S, r)$;
        if $r = r_{max}$ **then**
          $\lambda \leftarrow 1$;
      $S'' \leftarrow LocalSearchN^{II}(S')$; //*Local Search step*
      if $f(S'') \geq f(S')$ **then**
        $S'' \leftarrow LocalSearchN^{III}(S'')$;
      if $f(S'') < f(S)$ **then** //*Move or Not step*
        $S \leftarrow S''$;
        $r \leftarrow 1$;
        $n_{it} \leftarrow 1$;
        $\lambda \leftarrow 1$;
      else
        $r \leftarrow r + 1$;
        $n_{it} \leftarrow n_{it} + 1$;
  until $SessionTime \geq t_{max}$ || $n_{it} > iter_{max}$

6.3 Имплементација Адаптивне методе променљивих околина

SVNS је варијанта методе променљивих околина која омогућава претрагу делова простора решења која су далеко од текућег решења [49, 47]. Структура методе SVNS је слична структури BVNS, с изузетком трећег корака (фазе помераја) и стратегији првог побољшања која се користи у фази локалне претраге. Генерално, код SVNS методе, локални оптимум S" се прихвата чак и када је лошији од S, при чему је ниво прихватљиве деградације квалитета решења регулисан параметром α. Притом је неопходно изабрати одговарајућу вредност параметра α, како би се избегло дегенерисање SVNS методе у хеуристику са

вишеструким покретањем. Кораци предложене SVNS методе за разматрани проблем VSP представљени су Алгоритмом 13.

Алгоритам 13 Предложена SVNS метода за VSP

procedure SVNS($ProblemData, r_{max}, \alpha, t_{max}, iter_{max}$)
  Generate initial solution S;
  repeat
    $r \leftarrow 1$;
    $S_{best} \leftarrow S$;
    $n_{it} \leftarrow 1$;
    $\lambda \leftarrow 1$;
    while $r \leq r_{max}$ **do**
      if λ **then**
        $S' \leftarrow ShakeN^{I}(S, r)$; *//Shaking step*
        if $r = r_{max}$ **then**
          $r \leftarrow 0$;
          $\lambda \leftarrow 0$;
      else
        $S' \leftarrow ShakeN^{II}(S, r)$;
        if $r = r_{max}$ **then**
          $\lambda \leftarrow 1$;
      $S'' \leftarrow LocalSearchN^{II}(S')$; *//Local Search step*
      if $f(S'') \geq f(S')$ **then**
        $S'' \leftarrow LocalSearchN^{III}(S'')$;
      if $f(S) < f(S_{best})$ **then** *//Move or Not step*
        $S_{best} \leftarrow S$;
      if $f(S^{''}) - \alpha \cdot |f(S'') - f(S)| < f(S)$ **then**
        $S \leftarrow S''$;
        $r \leftarrow 1$;
        $n_{it} \leftarrow 1$;
        $\lambda \leftarrow 1$;
      else
        $r \leftarrow r + 1$;
        $n_{it} \leftarrow n_{it} + 1$;
    $S \leftarrow S_{best}$;
  until $SessionTime \geq t_{max}$ || $n_{it} > iter_{max}$

Потребно је направити избор адекватне вредности параметра α који могућава истраживање делова простора решења удаљених од S, у случајевима када је S" лошије од S, али не превише. Неадеквантним избором вредности параметра α, постоји могућност да се метода дегенерише у методу са вишестуким покретањем, која је позната као слабо ефикасна, јер у свакој итерацији креће из решења генерисаних на случајан начин [49]. У предложеној SVNS имплемен-

тацији вредност параметра α је изабрана на основу резултата прелиминарних експеримената (видети одељак 7.1).

6.4 Побољшана основна метода променљивих околина за решавање VSP

У циљу ефикаснијег решавања проблема VSP, дизајнирана је још једна варијанта Основне методе променљивих околина под називом побољшана Основна метода променљивих околина (енгл. *Improved Basic Variable Neighborhood Search - BVNSi*). BVNSi је креирана у циљу смањења времена извршавања, али постизања истог или побољшања квалитета решења у односу на претходно описану BVNS за VSP. Предложена BVNSi метода користи концепт низа преосталих количина у фази размрдавања, који је уведен приликом дизајнирања Основне методе променљивих околина за решавање потпроблема VSP-P (видети одељак 5.1). Потпроблем VSP-P није укључивао два типа врло сложених ограничења, што је проузроковало значајне промене при адаптирању овог концепта за решавање VSP.

Репрезентација решења

Решења проблема VSP код дизајниране BVNSi методе кодирана су на исти начин као код методе BVNS за проблем VSP-P (видети одељак 5.1). Наиме, решење проблема $S = (s, aq)$ је представљено уређеним паром кога чине матрица s, димензија $m \times k_{max}$ чији елементи представљају посећене локације или виртуалне туре и низ преосталих количина (aq) димензије $n + 1$ (видети одељак 5.1).

Рачунање вредности функције циља

Предложена BVNSi метода користи исти начин рачунања вредности функције циља као и претходно описане две методе BVNS и SVNS (видети одељак 6.1, с обзиром да су све три методе дизајниране за решавање истог проблема VSP. Према томе, вредност функције циља код предложене BVNSi методе рачуна се према формули 6.1.

Генерисање почетног решења

Почетно решење методе BVNSi се генерише на сличан начин као код методе BVNS за проблем VSP-P, што је детаљно описано у одељку 5.1. Међутим, за разлику од процедуре која генерише иницијално решење у методи BVNS за проблем VSP-P, не врше се сортирања тура за свако возило. Наиме, код проблема VSP редослед тура сваког појединачног возила је веома битан, односно редослед тура утиче на сусрете возила на локацијама и у кругу фабрике, док је код проблема VSP-P свеједно којим редом ће неко возило обављати своје дневне туре. Због једноставности и потреба алгоритма BVNS, код проблема VSP-P туре сваког возила се након формирању сортирају од најдужих до најкраћих. У случају BVNSi методе за VSP, туре остају у редоследу у којем су формиране, с обзиром да се времена полазака возила у свакој тури одређују тако да буду задовољена ограничења у вези са сусретима возила на локацијама и у кругу фабрике. Промена редоследа тура неког возила би очигледно могла нарушити испуњеност поменутих услова.

Структуре околина

Побољшана Основна метода променљивих околина, BVNSi, користи два типа околина, означених са N^I и N^{II}. Околина N^I решења S добија се заменом локација у турама два различита возила i_1 и i_2, $i_1, i_2 \in \{1, \ldots, m\}$ или заменом локације у тури једног возила i и неиспражњене локације која одговара индексу ненула елемента низа преосталих количина. Прецизније, N^I-сусед S' решења S добија се тако што два различита возила размењују посећене локације у по једној својој тури или се нехитна локација у тури једног возила замењује локацијом која није испражњена. Дакле, потез у овако дефинисаној околини се реализује избором два различита цела броја, i_1 и i_2 из скупа $\{0, \ldots, m\}$, при чему 0 означава низ преосталих количина, а потом избором њихових тура за размену. Такође, N^I-сусед S' решења S се може добити и заменом прве виртуалне туре једног возила решења S туром ка неиспражњеној локацији, као и последње невиртулане туре једног возила виртуалном туром која одговара индексу 0 низа aq. Сложеност ове околине је, у најгорем случају, $\binom{m}{2} \cdot k_{max}^2 + m \cdot k_{max} \cdot (n+1)$. Околина N^{II} садржи сва решења S' добијена тако што једно возило у S размењује пар својих непразних тура. Комплексност ове околине је у најгорем случају $m \cdot \binom{k_{max}}{2}$. Околина N^{II} чува допустивост

решења, док трансформације којима је дефинисна околина N^I могу нарушити допустивост решења, тј. може десити да код новог решења S' потребе фабрике не буду задовољене. Међутим, неки од наредних потеза у оквиру исте околине може поново произвести допустиво решење.

Као у случају околина које су коришћене у претходним предложеним методама, овде се може доказати да су описане околине коректно дефинисане за посматрани проблем VSP. Прецизније, важи следеће тврђење:

Теорема 6.4.1. Оптимално решење посматраног проблема распоређивања возила се може добити из било ког допустивог решења применом коначног броја трансформација којима су дефинисане структуре околина N^I и N^{II}.

Доказ. Нека је $S = (s, aq)$ произвољно допустиво решење посматраног проблема VSP, при чему се матрица решења s се састоји из m редова од којих сваки одговара по једном возилу, док низ преосталих количина aq са елементима $aq(j)$, $j = 0, \ldots, n$ чува информацију о расположивим количинама на локацијама $j \in \{1, \ldots, n\}$. При томе, важи $aq(0) = 0$, с обзиром да индекс 0 одговара виртуалној тури. Нека је $S^* = (s^*, aq^*)$ оптимално решење проблема VSP.

Да би се решење S трансформисало у оптимално решење S^*, потребно је, почевши од тура које одговарају првом возилу матрице решења s, испитати једну по једну туру и разменити по потреби локацију у њој са другом локацијом која се налази у тури неког другог возила (укључујући виртуалне туре и индексе низа aq). Локација се може заменити и нулом, што одговара виртуалној тури у s^*. Циљ је добити листе тура возила у матици оптималног решења s^*. При испитивању сваке туре возила i, могућа су четири случаја:

Случај 1. Локација се већ налази на коректној позицији у листи тура које одговарају возилу i. У овом случају, нема промене и поступак се наставља од следеће туре.

Случај 2. Локација у тури k возила i у решењу S може да замени своју позицију са локацијом $s(i, k_1)$ за $k_1 > k$. У овом случају, обе локације које мењају своја места су већ присутне у матрици решења s и једна од њих ће се преместити на одговарајућу позицију. Замена овог типа неће угрозити допустивост новодобијеног решења, с обзиром да потребе фабрике у смислу количине транспортоване робе, остају непромењене, а такође и хитне локације остају опслужене.

Случај 3. У случају када локација $s(i,k)$ треба да се замени са локацијом $s(i_1,k_1)$ где је $i_1 > i$, тако да важи $s(i,k) = s^*(i,k)$, такође се неће угрозити допустивост решења из истих разлога.

Случај 4. Може се десити да је потребно укључити нову локацију у матрицу решења s или да се нека локација која се већ налази у s мора посетити још једном. У овом случају, посматрана локација, уколико није хитна, у листи тура посматраног возила i се замењује траженом локацијом, тј. мења се индексом локације одговарајућег ненула елемента низа преосталих количина aq.

Случај 5. Да би се хитна локација $s(i,k)$ из случаја 4., уколико је то потребно, заменила индексом низа aq, примењују се два узастопна потеза, од којих је један описан случајевима 2. или 3., а други случајем 4.

Применом трансформације описаних случајевима 1 - 5 могуће је сместити све елементе матрице произвољног решења S на одговарајуће позиције у матрици оптималног решења S^*. Број трансформација које треба реализовати је $2mk_{max}$. □

Имплементација BVNSi методе

Метода BVNSi је по својој структури варијанта Основне методе променљивих околина, са три основна корака: фазом размрдавања, локалном претрагом и померајем, који се смењују до задовољења критеријума заустављања. Основна разлика BVNSi методе у односу на BVNS и SVNS је општија структура околина које се користе. Наиме, околина N^I, која је коришћена у фази размрдавања BVNSi методе, обухвата својом структуром неколико различитих потеза који су код BVNS и SVNS садржани у различитим околинама. У фази локалне претраге методе BVNSi, примењене су две различите стратегије: стратегија најбољег и стратегија првог побољшања, чиме су добијене две варијанте ове методе, означене редом са $BVNSi_B$ и $BVNSi_F$. Структуре обе варијанте методе $BVNSi$ представљене су Алгоритмом 14, при чему вредност параметра str одређује варијанту BVNSi методе: $str = 1$ за $BVNSi_B$, а $str = 0$ за $BVNSi_F$. У обе варијанте фигуришу параметри p, r_{max} и $iter_{max}$, али се њихове вредности разликују у зависности од тога која је варијанта изанрана. У случају $str = 1$, изабрана је метода $BVNSi_B$ са стратегијом најбољег побољшања и одговарајућим вредностима параметара $p = p_B$, $r_{max} = r_{max_B}$ и $iter_{max} = it_B$. Уколико је $str = 0$ користи се метода $BVNSi_F$ са стратегијом првог побољшања у фази локалне претраге и вредностима параметара $p = p_F$,

$r_{max} = r_{max_F}$ и $iter_{max} = it_F$.

Алгоритам 14 Предложена BVNSi метода за VSP

```
1:  procedure BVNSi(ProblemData, r_max_B, p_B, r_max_F, p_F, str, t_max, it_B, it_F)
2:      Generate initial solution S;
3:      repeat
4:          r ← 1;
5:          while r ≤ r_max do
6:              S' ← Shaking(S, r);
7:              if str = 1 then
8:                  apply best improvement strategy;
9:                  r_max ← r_max_B;
10:                 p ← p_B;
11:                 iter_max ← it_B;
12:             else
13:                 apply first improvement strategy;
14:                 r_max ← r_max_F;
15:                 p ← p_F;
16:                 iter_max ← it_F;
17:             S'' ← Local search(S');
18:             if f(S'') < f(S) then
19:                 S ← S'';
20:                 r ← 1;
21:             else
22:                 r ← r + 1;
23:     until SessionTime ≥ t_max || n_it > iter_max
```

Глава 7

Експериментални резултати

Сви експериментални резултати при тестирању дизајнираних метахеуристичких метода су добијени на рачунару са Intel Core i7-2600 процесором на 3.40GHz и 12GB RAM меморије под Linux оперативим системом. У циљу добијања оптималних решења или граница вредности функције циља оптималних решења, коришћени су решавачи CPLEX 12.6.2 [70] и Lingo 17 [54]. Експериментални резултати коришћењем CPLEX решавача добијени су на истој рачунарској конфигурацији на којој су тестиране и метахеуристике, док је оптимизација Lingo 17 решавачем извршавана на рачунару са Intel Core i5 - 3320M процесором на 2.60 GHz и 4GB RAM меморије под Windows 7 оперативим системом.

Комерцијални CPLEX решавач је најпознатији и најчешће коришћен егзактни решавач проблема оптимизације који подржава решавање линеарних и квадратних проблема са целобројним и реалним променљивима. Овај решавач омогућава добијање решења у случајевима конвексне и неконвексне квадратне функције циља, као и за конвексна квадратна ограничења. Прецизније, CPLEX се може користити за решавање проблема Мешовитог целобројног квадратног програмирања (енгл. *Mixed Integer Quadratic Problem - MIQP*) као и проблеме типа MIQCP под следећим претпоставкама [70]:

- Функција циља је линеарна или је функција циља конвексна и садржи квадратне изразе или функција циља садржи квадратне изразе који су производ бинарних променљивих. У последњем случају, функција циља не мора бити обавезно конвексна;
- Ограничења садрже квадратне изразе, али се могу представити у конусној

форми другог реда (енгл. *Second order cone*) или квадратни изрази у ограничењу укључују само производе бинарних променљивих.

Проблем конуса другог реда (енгл. *Second order cone program - SOCP*) представља конвексан проблем оптимизације изражен следећим релацијама:

$$\min \qquad f^T x \tag{7.1}$$

при условима

$$||A_i x + b_i|| \leq c_i^T x + d_i \qquad \forall i = 1, \ldots, N \tag{7.2}$$

где променљива $x \in R^n$, док су $f \in R^n$, $A_i \in R^{n_i \times n}$, $b_i \in R^{n_i}$, $c_i \in R^n$, а $d_i \in R$ параметри проблема. Норма која се појављује код ограничења је стандардна еуклидска норма, дефинисана са $||u|| = (u^T u)^{1/2}$. Ограничења (7.2) називају се ограничењима конуса другог реда.

При тестирању дизајнираних метахеуристичких метода за решавање проблема VSP-P и VSP коришћена су два скупа инстанци:

- реалне инстанце које садрже податке добијене из посматране фабрике шећера у Србији и
- генерисане инстанце проблема већих димензија које прате структуру реалних инстанци, креиране за тестирање перформанси дизајнираних метода.

Проблем VSP-P је потпроблем проблема VSP добијен искључивањем два врло сложена ограничења, па је стога могуће тестирати метахеуристичке методе за решавање VSP-P на инстанцама већих димензија и добити решења у разумном времену извршавања. Експериментални резултати који се односе на проблем VSP-P добијени су коришћењем следећа два скупа инстанци:

- Скуп реалних инстанци, који укључује 44 инстанце са до 15 локација, 40 возила и максималног броја од 20 тура током радног дана.
- Скуп генерисаних инстанци, које обухватају до 150 локација, 100 возила и максимално 100 тура током радног дана. Овај скуп садржи 17 тест примера, који су генерисани пратећи структуру реалних инстанци.

За тестирање предложених MIQCP и MILP формулација и метахеуристичких метода које су дизајниране за решавање проблема VSP, коришћен је исти

скуп реалних инстанци као и у случају проблема VSP-P, док скуп генерисаних инстанци за проблем VSP које су већих димензија, представља подскуп скупа генерисаних инстанци проблема VSP-P. Проблем VSP укључује два врло захтевна ограничења која са порастом димензије тест примера доводе до значајног повећања времена извршавања дизајнираних алгоритама. Из тог разлога, скуп генерисаних инстанци проблема VSP укључује до 30 локација, 60 возила и 25 тура током радног дана. Овај скуп садржи 10 тест примера.

7.1 Експериментални резултати примене метахеуристичких метода на проблем VSP-P

Математичке формулације које описују проблем VSP-P изложене су у одељку 4.3, а за добијање оптималних решења посматраног проблема коришћен је комерцијални CPLEX решавач. VSP-P је прво формулисан као проблем мешовитог целобројног програмирања са квадратним ограничењима (MIQCP), а потом реформулисан у проблем мешовитог целобројног линеарног програмирања (MILP). MILP формулација проблема VSP-P је тестирана коришћењем CPLEX решавача.

Анализа параметара

Метахеуристичке методе најчешће зависе од једног или више параметара. За дизајнирање методе са добрим перформансама неопходно је подесити вредности параметара. Ове вредности могу утицати у великој мери на процес решавања и квалитет добијених решења при примени метахеуристичких метода оптимизације.

Извршен је низ прелиминарних експеримената како би се одредиле адекватне вредности параметара r_{max} и p предложене BVNS методе, као и параметра γ предложене GRASP методе. Тестирање параметара је извршено на подскупу од 12 тест примера који се састоји из: 4 инстанце малих димензија, 4 реалне инстанце средњих димензија и 4 генерисане инстанце већих димензија. При испитивању сваке од посматраних вредности параметара, BVNS и GRASP методе су извршаване по 30 пута на свакој инстанци из посматраног скупа. Као критеријум заустављања у свим експериментима тестирања параметара коришћена је достигнута горња граница времена извршавања t_{max}, која је по-

стављена на 1s за инстанце малих димензија, 10s за средње и 100s за генерисане инстанце већих димензија.

Прво је тестиран параметар r_{max} у предложеној BVNS методи. С обзиром да r_{max} представља максималан ред околине која се истражује у BVNS, посматране вредности за r_{max} зависе од максималног броја тура k_{max} које возило може да направи током радног дана. Испитано је осам различитих формула које дефинишу везу између r_{max} и k_{max}. Параметар p је постављен на фиксну вредност $p = 1/(n+1)$ у експериментима са параметром r_{max}. Добијени резултати представљени су Табелом 7.1 на следећи начин. Прва колона, означена са $T_{n,m,k_{max}}$, садржи назив тестиране инстанце који укључује вредности основних параметара у инстанци: n (број локација), m (број возила) и k_{max} (максималан број тура). Генерисане инстанце су означене словом r у суперскрипту имена. Табела 7.1 садржи 8 делова, при чему сваки од њих одговара по једној од тестираних формула за r_{max}. Сваки део табеле састоји се из три колоне: најбоља вредност функције циља (*best*) добијена методом BVNS у 30 узастопних пуштања, просечно време извршавања (у секундама) за које је достигнуто најбоље добијено решење (t) и просечно процентуално одступање (gap) BVNS решења у односу на најбоље добијено у 30 пуштања. Најбоље вредности функције циља које су добијене у експериментима тестирања параметара r_{max} су истакнуте у целој Табели 7.1. У последњој врсти, означеној са **Просек**, дате су просечне вредности података из одговарајуће колоне и за сваку тестирану формулу за r_{max}. Најбоље просечне вредности су истакнуте. На основу резултата приказаних у Табели 7.1, може закључити да све тестиране формуле за r_{max} показују сличне перформансе у погледу квалитета добијених решења и времена извршавања, што указује на стабилност предложене BVNS методе. Међу тестираним вредностима, изабрана је формула $r_{max} = \lfloor k_{max}/2 \rfloor$, с обзиром да се њеним коришћењем добија најмања посечна вредност функције циља (46,388).

Када је формула за r_{max} изабрана, извршен је низ експеримената за подешавање параметра p, који представља вероватноћу избора виртуалне туре из скупа индекса низа преосталих количина у фази размрдавања. Разматрано је осам различитих формула које дефинишу зависност параметра p од броја локација n. Резултати су представљени у Табели 7.2 на исти начин као у Табели 7.1. У тестовима за избор вредности параметра p, коришћена је формула $r_{max} = \lfloor k_{max}/2 \rfloor$, с обзиром да се показала као најбоља у претходним тестовима.

Примењен је исти критеријум заустављања као код експеримената са r_{max}. Из резултата представљених у Табели 7.2, може се закључити да се просеци најбољих вредности функције циља, просечна времена извршавања и просечна одступања вредности функције циља од најбољих веома мало разликују када се користе различите формуле за p. Како је применом формуле $p = 3/(n+3)$ добијена најмања просечна вредност функције циља (46,373), та формула је изабрана за p у свим наредним тестовима.

При тестирању параметра γ код GRASP методе, примењена је слична стратегија као при тестирању параметра p и r_{max} методе BVNS. Табелом 7.3 представљени су резултати експеримената са осам различитих вредности за параметар γ рангираних од 100 до 20000. Структура Табеле 7.3 је иста као код Табела 7.1-7.2. Из Табеле 7.3, може се видети да три вредности параметра γ (1000, 2000 и 10000) дају исте просечне вредности функције циља (46,625). Ипак, вредност $\gamma = 10000$ је изабрана као најбоља, јер је у овом случају добијено краће просечно време извршавања у односу на друге две вредности, као и најмање просечно одступање у односу на све тестиране вредности параметра γ.

Табела 7.1: Тестирање параметра r_{max} у BVNS методи за проблем VSP-P

Инстанца	$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor}$			$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 1}$			$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 1}$			$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 2}$		
$T_{n,m,k_{max}}$	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,287**	0,003	0,336	**15,287**	0,050	0,015	**15,287**	0,070	0,000	**15,287**	0,071	0,000
$T_{5,2,4}$	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000
$T_{5,5,5}$	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000
$T_{5,20,20}$	**24,591**	0,224	0,003	**24,591**	0,290	0,007	**24,591**	0,283	0,007	**24,591**	0,480	0,003
$T_{8,40,10}$	**26,201**	2,640	0,285	26,206	3,650	0,261	26,206	3,612	0,261	26,234	2,674	0,189
$T_{10,10,10}$	**16,243**	0,017	0,000	**16,243**	0,226	0,000	**16,243**	0,215	0,000	**16,243**	0,085	0,000
$T_{15,20,15}$	25,990	2,640	0,285	25,990	3,650	0,261	**25,986**	4,787	0,025	**25,986**	4,791	0,025
$T^r_{10,50,10}$	**36,667**	9,301	0,005	**36,667**	4,773	0,012	**36,667**	4,836	0,012	**36,667**	9,323	0,007
$T^r_{30,60,10}$	**29,493**	29,238	0,041	**29,493**	35,791	0,018	**29,493**	36,702	0,042	**29,493**	36,524	0,042
$T^r_{50,70,70}$	**116,673**	74,658	0,107	116,730	78,080	0,079	116,730	77,771	0,082	116,726	64,688	0,085
$T^r_{120,90,90}$	216,607	81,711	0,066	**216,550**	77,050	0,095	**216,550**	77,293	0,096	216,664	79,666	0,038
Просек	**46,388**	16,928	0,071	46,389	17,021	0,052	46,389	17,131	0,044	46,400	**16,525**	**0,032**

Инстанца	$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 2}$			$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 3}$			$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 3}$			$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 4}$		
$T_{n,m,k_{max}}$	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,287**	0,058	0,000	**15,287**	0,060	0,000	**15,287**	0,075	0,000	**15,287**	0,078	0,000
$T_{5,2,4}$	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000
$T_{5,5,5}$	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,001	0,000	**18,879**	0,001	0,000
$T_{5,20,20}$	**24,591**	0,478	0,003	**24,591**	0,651	0,000	**24,591**	0,639	0,000	**24,591**	0,430	0,003
$T_{8,40,10}$	26,234	2,683	0,189	26,210	3,988	0,250	26,210	3,982	0,273	26,206	2,978	0,266
$T_{10,10,10}$	**16,243**	0,084	0,000	**16,243**	0,333	0,000	**16,243**	0,328	0,000	**16,243**	0,718	0,000
$T_{15,20,15}$	25,990	4,189	0,006	25,990	4,065	0,006	**25,986**	5,034	0,027	**25,986**	5,049	0,027
$T^r_{10,50,10}$	**36,667**	9,141	0,007	**36,667**	9,618	0,016	**36,667**	9,686	0,016	**36,667**	5,005	0,007
$T^r_{30,60,10}$	**29,493**	37,486	0,059	**29,493**	37,300	0,059	**29,493**	40,125	0,047	**29,493**	39,712	0,047
$T^r_{50,70,70}$	116,726	64,503	0,085	116,730	75,663	0,085	116,730	75,994	0,085	116,726	76,615	0,087
$T^r_{120,90,90}$	216,664	81,700	0,037	216,559	79,490	0,068	216,599	80,027	0,068	216,603	79,672	0,067
Просек	46,400	16,694	0,032	46,393	17,598	0,040	46,393	17,991	0,043	46,393	17,522	0,042

Табела 7.2: Тестирање параметра p у BVNS методи за проблем VSP-P

Инстанца	$\mathbf{p = 0,5/(n+0,5)}$			$\mathbf{p = 1/(n+1)}$			$\mathbf{p = 1,5/(n+1,5)}$			$\mathbf{p = 2/(n+2)}$		
$T_{n,m,k_{max}}$	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,287**	0,005	0,187	**15,287**	0,003	0,336	**15,287**	0,002	0,561	**15,287**	0,002	0,486
$T_{5,2,4}$	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000
$T_{5,5,5}$	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000
$T_{5,20,20}$	**24,591**	0,155	0,003	**24,591**	0,224	0,003	**24,591**	0,344	0,003	**24,591**	0,209	0,000
$T_{8,40,10}$	**26,201**	3,546	0,287	**26,201**	2,640	0,285	26,210	3,221	0,218	**26,201**	2,702	0,230
$T_{10,10,10}$	**16,243**	0,067	0,000	**16,243**	0,017	0,000	**16,243**	0,063	0,000	**16,243**	0,038	0,000
$T_{15,20,15}$	25,990	4,284	0,003	25,990	5,343	0,003	25,990	3,224	0,000	25,990	3,306	0,009
$T^r_{10,50,10}$	**36,667**	7,072	0,011	**36,667**	9,301	0,005	**36,667**	7,713	0,015	**36,667**	3,131	0,006
$T^r_{30,60,10}$	**29,493**	37,404	0,018	**29,493**	29,238	0,041	**29,493**	21,434	0,024	**29,493**	35,580	0,035
$T^r_{50,70,70}$	116,783	71,015	0,047	**116,673**	74,658	0,107	116,726	72,732	0,061	116,721	72,758	0,057
$T^r_{120,90,90}$	216,660	82,562	0,064	216,607	81,711	0,066	216,489	83,546	0,074	216,489	84,517	0,071
Просек	46,402	17,176	**0,052**	46,388	16,928	0,071	46,384	16,023	0,080	46,383	16,854	0,075

Инстанца	$\mathbf{p = 2,5/(n+2,5)}$			$\mathbf{p = 3/(n+3)}$			$\mathbf{p = 3,5/(n+3,5)}$			$\mathbf{p = 4/(n+4)}$		
$T_{n,m,k_{max}}$	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,287**	0,001	0,561	**15,287**	0,002	0,561	**15,287**	0,002	0,523	**15,287**	0,001	0,523
$T_{5,2,4}$	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,000	0,000
$T_{5,5,5}$	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000	**18,879**	0,000	0,000
$T_{5,20,20}$	**24,591**	0,313	0,000	**24,591**	0,448	0,003	**24,591**	0,169	0,018	**24,591**	0,102	0,003
$T_{8,40,10}$	**26,201**	3,631	0,223	**26,201**	2,980	0,179	26,206	3,524	0,209	**26,201**	2,610	0,209
$T_{10,10,10}$	**16,243**	0,456	0,000	**16,243**	0,330	0,000	**16,243**	0,095	0,000	**16,243**	0,225	0,000
$T_{15,20,15}$	25,990	4,979	0,005	**25,986**	3,946	0,022	**25,986**	4,282	0,030	25,990	3,665	0,003
$T^r_{10,50,10}$	**36,667**	9,842	0,010	**36,667**	6,206	0,012	**36,667**	4,167	0,013	**36,667**	6,340	0,006
$T^r_{30,60,10}$	**29,493**	30,079	0,048	**29,493**	34,094	0,029	**29,493**	27,101	0,018	**29,493**	21,327	0,012
$T^r_{50,70,70}$	116,721	74,589	0,041	**116,673**	71,515	0,073	116,726	62,163	0,032	**116,673**	71,065	0,059
$T^r_{120,90,90}$	216,546	80,556	0,046	**216,431**	87,819	0,080	**216,431**	81,364	0,067	216,436	83,371	0,052
Просек	46,387	17,037	0,078	**46,373**	17,278	0,080	46,378	**15,239**	0,076	46,374	15,726	0,072

Резултати и поређења

Предложени BVNS и GRASP алгоритми су тестирани на скупу свих разматраних реалних и генерисаних инстанци проблема VSP-P. Обе методе су извршаване по 30 пута на сваком тест примеру. Критеријум заустављања код предложених BVNS и GRASP метода је достигнут лимит времена извршавања (t_{max}). Вредности параметра t_{max} су постављене на: $t_{max} = 1$s за мале реалне инстанце, $t_{max} = 10$s за реалне примере средњих димензија и $t_{max} = 100$s за генерисане инстанце.

Експериментални резултати на реалним инстанцама малих димензија су представљени Табелом 7.4. Назив инстанце је садржан у колони $T_{n,m,k_{max}}$. Колоне које се односе на резултате добијене помоћу CPLEX решавача садрже вредност функције циља која одговара оптималном решењу *opt. sol.* и одговарајуће време извршавања *t(s)*. Прва колона која одговара предложеној BVNS методи представља вредност функције циља најбоље пронађеног BVNS решења *best*, при чему је коришћена ознака *opt* у случају када се та вредност поклапа

Табела 7.3: Тестирање параметра γ у GRASP методи за проблем VSP-P

Инстанца	$\gamma = 100$			$\gamma = 200$			$\gamma = 500$			$\gamma = 1000$		
$T_{n,m,k_{max}}$	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,397**	0,339	1,694	**15,397**	0,336	1,694	**15,397**	0,336	1,694	**15,397**	0,337	1,694
$T_{5,2,4}$	**13,156**	0,001	0,000	**13,156**	0,000	0,000	**13,156**	0,001	0,000	**13,156**	0,002	0,000
$T_{5,5,5}$	**18,879**	0,298	0,000	**18,879**	0,297	0,000	**18,879**	0,297	0,000	**18,879**	0,298	0,000
$T_{5,20,20}$	**24,616**	5,039	0,014	**24,616**	5,093	0,014	**24,616**	5,073	0,014	**24,616**	4,620	0,038
$T_{8,40,10}$	**26,487**	2,814	0,094	**26,487**	2,771	0,094	**26,487**	2,805	0,094	**26,487**	2,803	0,091
$T_{10,10,10}$	**16,386**	5,026	0,567	**16,386**	5,094	0,527	**16,386**	5,130	0,527	**16,386**	5,045	0,527
$T_{15,20,15}$	**26,329**	5,293	0,252	**26,329**	5,282	0,252	**26,329**	5,103	0,256	**26,329**	4,943	0,252
$T^r_{10,50,10}$	36,900	39,484	0,087	36,900	38,409	0,092	36,900	39,802	0,087	**36,839**	41,594	0,223
$T^r_{30,60,10}$	**29,950**	0,000	0,019	**29,950**	0,000	0,019	**29,950**	0,000	0,019	**29,950**	0,000	0,019
$T^r_{50,70,70}$	117,359	39,236	0,000	117,359	39,096	0,000	**117,354**	39,267	0,004	**117,354**	43,841	0,004
$T^r_{120,90,90}$	**217,236**	42,118	0,032	**217,236**	44,274	0,032	**217,236**	37,301	0,033	**217,236**	38,673	0,032
Просек	46,631	11,637	0,230	46,631	11,721	0,227	46,630	11,260	0,227	**46,625**	11,846	0,240
Инстанца	$\gamma = 2000$			$\gamma = 5000$			$\gamma = 10000$			$\gamma = 20000$		
$T_{n,m,k_{max}}$	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000	**16,874**	0,000	0,000
$T_{4,5,7}$	**15,397**	0,279	1,459	**15,397**	0,284	1,459	**15,397**	0,311	1,459	**15,397**	0,341	1,731
$T_{5,2,4}$	**13,156**	0,000	0,000	**13,156**	0,000	0,000	**13,156**	0,001	0,000	**13,156**	0,002	0,000
$T_{5,5,5}$	**18,879**	0,363	0,000	**18,879**	0,382	0,000	**18,879**	0,371	0,000	**18,879**	0,321	0,000
$T_{5,20,20}$	**24,616**	3,414	0,013	**24,616**	3,964	0,001	**24,616**	3,875	0,012	**24,616**	5,050	0,029
$T_{8,40,10}$	**26,487**	3,015	0,093	26,491	3,571	0,058	**26,487**	2,939	0,081	**26,487**	2,944	0,092
$T_{10,10,10}$	**16,386**	3,939	0,477	**16,386**	3,900	0,477	**16,386**	3,967	0,477	**16,386**	5,094	0,538
$T_{15,20,15}$	**26,329**	5,323	0,215	**26,329**	5,271	0,207	**26,329**	4,978	0,196	**26,329**	4,768	0,248
$T^r_{10,50,10}$	**36,839**	43,671	0,218	**36,839**	40,374	0,213	**36,839**	39,715	0,218	36,900	34,518	0,091
$T^r_{30,60,10}$	**29,950**	0,000	0,019	**29,950**	0,000	0,019	**29,950**	0,000	0,019	**29,950**	0,000	0,019
$T^r_{50,70,70}$	**117,354**	44,314	0,004	**117,354**	35,085	0,004	**117,354**	35,180	0,004	**117,354**	33,244	0,007
$T^r_{120,90,90}$	**217,236**	48,273	0,023	**217,236**	47,913	0,228	**217,236**	48,112	0,023	**217,236**	38,776	0,032
Просек	**46,625**	12,716	0,210	46,626	11,729	0,222	**46,625**	11,621	**0,207**	46,630	**10,422**	0,232

са *opt. sol*. Преостале колоне које се односе на BVNS методу садрже: просечно време извршавања *avg. t(s)* за које BVNS достиже решење *best*, просечно процентуално одступање *avg. gap (%)* BVNS решења у односу на *opt. sol.*. Све просечне вредности су израчунате на основу резултата добијених кроз 30 BVNS узастопних извршавања. Резултататати предожене GRASP методе су представљени у последње три колоне на исти начин као и BVNS резултати. Последњи ред Табеле 7.4, означен са **Просек**, садржи просечне вредности представљених резултата по одговарјућим колонама.

Анализом података приказаних у Табели 7.4, може се закључити да се предложеном BVNS методом достижу сва оптимална решења на инстанцама малих димензија које су решене до оптималности помоћу CPLEX решавача, док GRASP није успео да пронађе оптимална решења у случају две инстанце ($T_{4,5,7}$ и $T_{6,6,6}$). Просечно време извршавања BVNS и GRASP методе на тестираним инстанцама је 0,002 и 0,070s, редом, што је значајно мање у поређењу са просечним временом извршавања CPLEX решавача (24,102s). Такође, GRASP метода се показала стабилнијом од BVNS методе на овом скупу инстанци, с обзиром

Табела 7.4: Експериментални резултати на реалним инстанцама малих димензија решених до оптималности CPLEX решавачем

Инстанца	**CPLEX**		**BVNS**			**GRASP**		
$T_{n,m,k_{max}}$	*opt. sol.*	*t(s)*	*best*	*t(s)*	*gap(%)*	*best*	*t(s)*	*gap(%)*
$T_{3,2,4}$	**16,874**	1,13	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,3,2}$	**9,780**	0,64	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,3,3}$	**13,727**	0,89	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,3,4}$	**16,303**	0,37	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,3,5}$	**17,164**	0,83	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,4,2}$	**10,580**	0,69	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,4,3}$	**13,727**	1,34	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,4,4}$	**16,303**	0,95	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{3,5,2}$	**12,294**	1,37	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,2,3}$	**12,870**	0,75	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,2,4}$	**13,446**	1,55	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,3,2}$	**10,580**	1,68	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,3,3}$	**12,299**	1,91	**opt**	0,000	6,196	**opt**	0,000	0,000
$T_{4,4,2}$	**11,151**	0,25	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,5,2}$	**10,866**	1,64	**opt**	0,000	1,841	**opt**	0,000	0,000
$T_{4,4,3}$	**14,299**	0,93	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,4,4}$	**16,874**	2,68	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{4,5,7}$	**15,287**	296,17	**opt**	0,002	0,561	15,397	0,311	1,459
$T_{5,2,3}$	**10,584**	1,13	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{5,2,4}$	**13,156**	2,65	**opt**	0,000	0,000	**opt**	0,001	0,000
$T_{5,3,2}$	**10,580**	1,15	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{5,3,3}$	**13,156**	1,73	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{5,3,4}$	**16,589**	2,69	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{5,4,3}$	**13,156**	0,65	**opt**	0,000	1,303	**opt**	0,000	0,000
$T_{5,5,2}$	**11,437**	1,63	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{5,5,5}$	**18,879**	38,23	**opt**	0,000	0,000	**opt**	0,371	0,000
$T_{6,2,3}$	**11,727**	1,21	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,2,4}$	**13,160**	1,20	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,3,2}$	**10,294**	0,73	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,3,3}$	**12,870**	2,10	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,4,2}$	**10,009**	1,29	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,5,2}$	**10,866**	0,76	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{6,6,6}$	**16,569**	216,60	**opt**	0,017	0,149	16,597	0,401	0,675
$T_{6,7,6}$	**11,479**	53,65	**opt**	0,001	0,048	**opt**	0,062	0,000
$T_{7,3,3}$	**12,299**	2,00	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{7,5,2}$	**10,580**	2,94	**opt**	0,000	0,000	**opt**	0,000	0,000
$T_{7,5,6}$	**16,140**	25,55	**opt**	0,000	0,052	**opt**	0,425	0,255
$T_{8,6,5}$	**14,246**	241,85	**opt**	0,000	0,021	**opt**	0,003	0,255
Просек	13,622	24,102	**opt**	**0,002**	0,376	14,106	0,070	**0,080**

да је применом GRASP добијено мање просечно одступање рачунато у односу на оптимална решења.

Експериментални резултати на 6 реалних инстанци средњих димензија и свим генерисаним инстанцама су представљени Табелом 7.5. CPLEX решавач је пронашао допустива решења за све реалне инстанце и за само један генерисан тест пример. Добијена допустива решења су представљена у колони *best* у делу табеле који се односи на резултате CPLEX решавача. Код резултата добијених CPLEX решавачем, колона $t(s)$ је изостављена, с обзиром да је извршавање на овим инстанцама трајало до достигнуте граница времена извршавања или прекорачења меморија рачунара. Колоне које се односе на BVNS и GRASP решења, као и последњи ред означен са **Просек** имају исту структуру и значење као у Табели 7.4. Вредности функције циља најбољих решења су истакнуте у колонама *best*, као и најбоље вредности у последњем реду **Просек**.

На основу резултата приказаних у Табели 7.5, може се закључити да је BVNS по квалитету добијених решења надмашио и GRASP и CPLEX. Предложеном BVNS методом достигнуте су све горње границе добијене CPLEX решавачем а у случају три инстанце ($T_{5,20,20}$, $T_{8,40,10}$ и $T_{15,20,15}$) горње границе су поправљене. На свим инстанцама овог скупа, најбоља решења GRASP методе су, изузев $T_{5,5,10}$, лошија у односу на одговарајућа решења добијена BVNS методом и CPLEX решавачем. Међутим, како показује колона *gap(%)*, вредности функције циља најбољих решења пронађених GRASP методом су веома блиска решењима добијеним BVNS методом. Просечно одступање вредности функције циља решења добијена GRASP методом од одговарајућих вредности најбољих познатих решења износи 0,559%. BVNS је стабилна у генерисању најбољих познатих решења, с обзиром да просечно одступање има врло малу вредност (0,041%). Са аспекта времена извршавања, GRASP метода се показала бржом у поређењу са BVNS методом, имајући у виду да је њено просечно време извршавања 21,573s, док се BVNS метода просечно извршавала за време од 34,941s.

7.2 Експериментални резултати примене метахеуристичких метода на проблем VSP

За добијање оптималних решења проблема VSP коришћен је комерцијални решавач Lingo 17, који је дизајниран за решевање великог броја различитих типова математичких модела [54]. Математичке формулације посматраног про-

Табела 7.5: Експериментални резултати на реалним инстанцама већих димензија и генерисаним инстанцама за које је CPLEX решавач пронашао само допустиво решење

Инстанца	**CPLEX**	**BVNS**			**GRASP**		
$T_{n,m,k_{max}}$	*best*	*best*	$t(s)$	*gap(%)*	*best*	$t(s)$	*gap(%)*
$T_{5,5,10}$	**17,781**	**17,781**	0,706	0,002	**17,781**	1,084	0,000
$T_{5,10,10}$	**17,014**	**17,014**	0,727	0,019	17,039	3,562	0,279
$T_{5,20,20}$	24,648	**24,591**	0,448	0,003	24,616	3,875	0,112
$T_{8,40,10}$	26,267	**26,201**	2,980	0,179	26,487	2,939	1,174
$T_{10,10,10}$	**16,243**	**16,243**	0,330	0,000	16,386	3,967	1,360
$T_{15,20,15}$	26,214	**25,986**	3,946	0,022	26,329	4,978	1,521
$T^r_{10,20,20}$	**58,653**	**58,653**	16,861	0,038	59,000	0,000	0,638
$T^r_{10,30,15}$	/	**49,150**	1,190	0,112	49,264	0,000	0,573
$T^r_{10,50,10}$	/	**36,667**	6,206	0,012	36,839	39,715	0,687
$T^r_{15,30,25}$	/	**63,694**	9,371	0,018	63,813	45,349	0,252
$T^r_{15,40,20}$	/	**53,620**	16,715	0,015	53,739	38,618	0,221
$T^r_{20,30,10}$	/	**22,214**	4,503	0,000	22,614	0,000	1,879
$T^r_{20,40,20}$	/	**41,163**	3,697	0,000	41,510	0,000	1,347
$T^r_{25,50,15}$	/	**36,060**	7,440	0,010	36,179	0,000	0,675
$T^r_{30,30,25}$	/	**46,674**	55,386	0,025	47,017	0,000	0,101
$T^r_{30,60,15}$	/	**29,493**	34,094	0,029	29,950	0,000	0,019
$T^r_{40,60,55}$	/	**103,680**	52,475	0,030	104,199	30,703	0,548
$T^r_{45,65,70}$	/	**109,403**	57,609	0,052	109,979	43,953	0,533
$T^r_{50,70,70}$	/	**116,673**	71,515	0,073	117,354	35,180	0,004
$T^r_{60,80,80}$	/	**126,453**	64,394	0,023	127,029	47,016	0,481
$T^r_{70,60,60}$	/	**171,857**	79,355	0,051	172,543	50,999	0,436
$T^r_{80,65,65}$	/	**181,413**	85,149	0,059	182,274	34,384	0,535
$T^r_{90,80,80}$	/	**158,183**	78,465	0,091	158,873	29,077	0,436
$T^r_{100,85,85}$	/	**184,274**	85,980	0,056	185,017	46,497	0,407
$T^r_{120,90,90}$	/	**216,431**	87,819	0,080	217,236	48,112	0,023
$T^r_{150,100,100}$	/	**240,646**	81,103	0,065	241,336	50,896	0,298
Просек	/	**83,624**	34,941	**0,041**	84,016	**21,573**	0,559

блема су изнете у одељку 4.4. VSP је формулисан у облику проблема мешовитог целобројног програмирања са квадратним ограничењима (MIQCP), а потом реформулисан у проблем мешовитог целобројног линеарног програмирања (MILP) (видети одељак 4.4). С обзиром на значајан број нових променљивих и ограничења који су додати приликом реформулисања, оба предложена модела, MIQCP и MILP су тестирана коришћењем Lingo решавача. Постављена је горња граница времена извршавања Lingo решавачу од 10h. У оквиру тог времена, решавач је дао оптимална решења или горње границе вредности функ-

ције циља само за тест примере малих димензија. Притом се MILP показала успешнијом од MIQCP, с обзиром да је њеном применом добијено више оптималних решења за реалне инстанце мањих димензија. Добијена оптимална решења и горње границе вредности функције циља коришћене су за оцену перформанси три дизајниране метахеуристичке методе: BVNS, SVNS и BVNSi на реалним инстанцама мањих димензија. Развијене метахеуристичке методе су затим тестиране на реалним инстанцама средњих димензија и већим генерисаним инстанцама.

Анализа параметара

Пре евалуације перформанси предложених BVNS, SVNS и BVNSi метода на свим тест инстанцама, извршен је низ експеримената на подскупу тест инстанци како би се пронашла адекватна вредност параметра r_{max} у BVNS и SVNS и параметра α код SVNS методе. За обе варијанте BVNSi методе (BVNSi$_B$ и BVNSi$_F$), извршени су тестови подешавања параметара r_{max} и p. Експерименти подешавања параметара су извршени на подскупу од 12 изабраних инстанци који садржи: 6 реалних инстанци мањих димензија, 4 средње реалне инстанце и 2 генерисане инстанце већих димензија. Само неколико најмањих и неколико највећих инстанци је коришћено у експериментима за подешавање параметара, с обзиром да се за прве добијају оптимална решења независно од вредности параметара, а за друге је потребно доста времена како би се завршили сви тестови.

Прво су посматране различите формуле за r_{max} које зависе од максималног броја тура k_{max} код BVNS методе. Ове формуле су одабране на основу истраживања која се односе на VNS методу из литературе. У тестовима који су овде вршени параметар $iter_{max}$ има фиксну вредност 5. За сваку од посматраних формула за r_{max} и за сваку од инстанци, BVNS је извршаван 5 пута. Резултати су представљени Табелом 7.6 на исти начин као и при тестирању параметара у случају проблема VSP-P. Име инстанце је садржано у колони $T_{n,m,k_{max}}$, где је n број локација, m број возила и k_{max} максималан број тура које возило може да направи током радног дана. Сваки део Табеле 7.6 одговара једној од тестираних формула за r_{max} и састоји се из четири колоне: вредност функције циља најбољег решења ($best$) добијеног BVNS методом кроз 5 узастопних пуштања, просечно укупно време извршавања (t_{tot}), просечно време које је потребно за добијање најбољег решења (t) и просечно процентуално одступање (gap) BVNS

решења у односу на најбоље добијено у посматраних 5 пуштања. Сва времена која су пријављена изражена су у секундама. Најбоље добијене вредности кроз цео експеримент за r_{max} су истакнуте у Табели 7.6. Последњи ред, означен са **Просек**, садржи просечне вредности представљених података у свакој од колона и за свако тестирано r_{max}, а најбоље вредности су истакнуте. Примарни критеријум за анализу и подешавање параметара је квалитет добијених решења. На основу резултата представљених Табелом 7.6, изабрана је формула $r_{max} = [\frac{k_{max}+1}{2}] + 1$, с обзиром да је њеном применом добијена најмања просечна вредност функције циља (20,027).

У случају SVNS методе, примењена је иста стратегија за подешавање параметра r_{max} и резултати су представљени Табелом 7.7. При тестирању r_{max}, параметар α има фиксну вредност 0,05 а вредност $iter_{max} = 5$ је такође коришћена код критеријума заустављања. Из резултата представљених у Табели 7.7, може се закључити да су разлике у просечним вредностима функције циља, просечним временима и просечним одступањима прилично мале када се користе различите формуле за r_{max}, што указује на стабилност предложене SVNS методе. Формула $r_{max} = [\frac{k_{max}}{2}] + 3$ води ка најмањој просечној вредности функције циља (20,280). Иако просечно време извршавања при примени ове формуле није најкраће, она води ка решењима најбољег квалитета па је изабрана за даља тестирања дизајниране SVNS методе.

Табелом 7.8, представљени су резултати експеримената за одређивање најбоље вредности параметра α који се користи у SVNS методи. Посматрано је 8 различитих вредности за α, рангираних од $\alpha = 0,025$ до $\alpha = 0,900$. Код ових тестова, коришћена је формула $r_{max} = [\frac{k_{max}}{2}] + 3$, одабрана на основу претходних тестирања параметра r_{max}. За сваку тестирану вредност параметра α и сваку инстанцу посматраног подскупа, SVNS је извршаван 5 пута. Резултати су представљени на исти начин као у Табелама 7.6-7.7. Вредност $\alpha = 0,05$ је показала најбоље перформансе у смислу квалитета добијених решења, с обзиром да је њеним коришћењем добијена најмања просечна вредност функције циља (20,280), док просечно време извршавања прихватљиво веће у односу на друге тестиране вредности. Из тог разлога, вредност параметра α је постављена на 0,05 у даљим експериментима са SVNS методом.

За обе варијанте предложене BVNSi методе, BVNSi$_B$ и BVNSi$_F$, извршена су тестирања у циљу одређивања адекватних вредности параметара r_{max} и p. Најпре је разматрана метода BVNSi$_B$ и 8 различитих формула за r_{max} (истих

као код BVNS и SVNS метода) при чему је за параметар p узета формула $p = 1/(n+1)$. Добијени резултати су представљени Табелом 7.9, која има исту структуру као и Табела 7.6.

На основу резултата тестирања параметра r_{max} методе BVNSi$_B$ приказаних у Табели 7.9 може се видети да је најбоља вредност у смислу квалитета добијених решења на изабраном скупу истанци $r_{max} = \lfloor \frac{k_{max}+1}{2} \rfloor + 2$. У погледу просечних времена извршавања и просечних процентуалних одступања решења у односу на најбоља добијена у 5 узастопних извршавања, BVNSi$_B$ алгоритам је најуспешнији при коришћењу формуле $r_{max} = [\frac{k_{max}}{2}]$. Како је квалитет решења примарни критеријум у анализи и подешавању параметара, изабрана је формула $r_{max} = \lfloor \frac{k_{max}+1}{2} \rfloor + 2$ и она је коришћена у свим даљим тестирањима методе BVNSi$_B$.

Након избора одговарајуће формуле за r_{max} у методи BVNSi$_B$, извршен је низ експеримената за подешавање параметра p. Параметар p има исто значење као и у методи BVNS која је дизајнирана за решавање проблема VSP-P. Из тог разлога су тестиране исте формуле које одражавају зависност овог параметра од укупног броја локација n (видети одељак 7.1). Резултати тестирања параметра p у методи BVNSi$_B$ представљени су Табелом 7.10.

Имајући у виду просечне вредности резултата у Табели 7.10, међу тестираним формулама за избор адекватне зависности између параметра p и броја локација n, најбољи просечни квалитет решења остварује се коришћењем формуле $p = 1/(n+1)$. Наиме, просечна вредност функције циља на датом скупу инстанци (20,274) је најмања међу одговарајућим вредностима при примени осталих тестираних формула за p. Из тог разлога, формула $p = 1/(n+1)$ је изабрана за даља тестирања методе BVNSi$_B$, без обзира што просечно време извршавња и просечно одступање у односу на најбоље пронађено решење у 5 узастопних пуштања ове методе на датум скупу инстанци нису најмањи у поређењу са осталим тестираним формулама.

Исти тестови за одређивање адекватних избора вредности параметара r_{max} и p поновљени су за методу BVNSi$_F$, која користи стратегију првог побољшања у фази локалне претраге. Добијени резултати представљени су Табелама 7.11 - 7.12. При тестирању разматраних формула за r_{max}, вредност параметра p је такође постављена на вредност $p = 1/(n+1)$. На основу резултата приказаних у Табели 7.11 изабрана је формула $r_{max} = \lfloor \frac{k_{max}}{2} \rfloor + 3$, која је потом коришћена при тестирању 8 формула за избор параметра p. Како је формула $p = 3/(n+3)$

дала најмању просечну вредност функције циља најбољих пронађених решења (20,468) на посматраном скупу инстанци (видети Табелу 7.12), изабран је пар $r_{max} = \lfloor \frac{k_{max}}{2} \rfloor + 3$ и $p = 3/(n+3)$ у даљим испитивањима перформанси методе BVNSi_F.

Табела 7.6: Тестирање параметра r_{max} у BVNS методи за проблем VSP

Инстанца	$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 2}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,000	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	0,000	**13,156**	0,000	0,000	1,310	**13,156**	0,000	0,000	0,441	**13,156**	0,000	0,000	0,000
$T_{5,3,3}$	13,441	0,000	0,000	1,303	**13,156**	0,000	0,000	2,606	**13,156**	0,000	0,000	0,869	**13,156**	0,000	0,000	3,041
$T_{3,3,5}$	**17,164**	0,014	0,004	0,333	**17,164**	0,006	0,002	0,333	**17,164**	0,006	0,000	0,666	**17,164**	0,006	0,002	0,000
$T_{3,4,4}$	**16,303**	0,008	0,004	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,004	0,000	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	0,526
$T_{5,5,10}$	**17,843**	1,050	0,506	0,378	17,867	0,458	0,238	0,179	**17,843**	0,672	0,470	0,512	17,871	0,924	0,486	0,505
$T_{8,6,5}$	**14,336**	0,094	0,024	4,106	14,907	0,024	0,002	0,613	**14,336**	0,056	0,030	2,671	14,907	0,032	0,006	1,188
$T_{10,10,10}$	16,523	18,078	13,200	0,197	16,557	6,040	3,764	1,139	**16,496**	6,518	4,370	0,604	16,520	14,248	9,578	0,585
$T_{15,20,15}$	26,729	399,814	296,024	1,033	26,676	342,960	268,272	1,306	26,609	229,152	149,234	1,315	26,676	664,496	491,550	1,236
$T^r_{10,50,10}$	39,696	3500,168	2503,416	1,912	39,014	4021,308	3232,116	3,005	**38,729**	4371,002	3617,550	2,025	39,296	7747,924	6243,416	1,459
$T^r_{30,60,10}$	39,013	70298,860	48708,784	2,027	38,617	94130,978	69478,904	1,884	38,794	105957,986	74249,228	1,077	38,339	120613,934	91186,742	2,815
Просек	20,162	**6184,841**	**4293,497**	1,028	20,096	8208,481	6081,942	1,204	**20,027**	9213,783	6501,740	0,992	20,094	10753,464	8160,982	0,975

Инстанца	$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 2}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 4}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	0,000	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	1,310	**13,156**	0,000	0,000	1,310	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	1,310
$T_{5,3,3}$	**13,156**	0,000	0,000	0,434	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	0,434	**13,156**	0,000	0,000	1,303
$T_{3,3,5}$	**17,164**	0,006	0,004	0,333	**17,164**	0,010	0,002	0,333	**17,164**	0,012	0,002	0,333	**17,164**	0,008	0,000	0,333
$T_{3,4,4}$	**16,303**	0,004	0,000	0,209	**16,303**	0,008	0,002	0,000	**16,303**	0,010	0,002	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,578	**10,866**	0,002	0,002	0,526	**10,866**	0,002	0,002	0,000	**10,866**	0,002	0,000	0,000
$T_{5,5,10}$	17,871	0,556	0,332	0,571	17,871	0,856	0,342	0,639	17,871	0,590	0,328	0,320	17,871	0,920	0,366	0,544
$T_{8,6,5}$	14,907	0,036	0,006	0,613	**14,336**	0,060	0,028	3,029	14,907	0,042	0,004	0,920	**14,336**	0,064	0,026	3,508
$T_{10,10,10}$	16,524	8,318	5,742	0,360	16,521	9,952	7,322	0,173	16,524	8,474	5,260	0,712	16,526	14,376	8,478	0,975
$T_{15,20,15}$	**26,607**	671,996	465,492	1,791	26,733	269,332	164,692	1,875	26,690	631,570	453,026	1,196	26,640	360,360	270,200	0,467
$T^r_{10,50,10}$	38,839	6244,138	4421,340	4,231	39,067	4106,680	3179,078	1,165	38,959	5383,954	3692,170	2,794	39,010	4976,858	3767,992	2,295
$T^r_{30,60,10}$	38,856	112903,694	82813,904	0,853	38,661	88764,980	51945,300	2,587	**37,961**	125059,522	76751,050	3,328	38,867	123938,036	79394,934	1,164
Просек	20,094	9985,729	7308,902	1,052	20,059	7762,657	4608,064	0,970	20,045	10817,242	6699,063	**0,757**	20,064	10774,219	6953,500	1,020

Табела 7.7: Тестирање параметра r_{max} у SVNS методи за проблем VSP

Инстанца	$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 2}$			
$T_{3,2,4}$	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	2,619	**13,156**	0,000	0,000	1,310	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	0,000
$T_{5,3,3}$	13,441	0,000	0,000	2,976	13,441	0,000	0,000	0,425	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	1,303
$T_{3,3,5}$	**17,164**	0,002	0,002	1,332	**17,164**	0,002	0,000	0,666	**17,164**	0,002	0,000	0,333	**17,164**	0,002	0,002	0,666
$T_{3,4,4}$	**16,303**	0,002	0,000	0,209	**16,303**	0,000	0,000	0,209	**16,303**	0,002	0,000	0,147	**16,303**	0,002	0,000	0,209
$T_{6,5,2}$	11,151	0,000	0,000	0,000	**10,866**	0,000	0,000	2,104	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	1,052
$T_{5,5,10}$	17,900	0,206	0,086	0,953	17,900	0,154	0,128	0,479	17,957	0,150	0,086	0,732	17,929	0,136	0,064	0,988
$T_{8,6,5}$	15,136	0,200	0,002	0,000	15,107	0,032	0,008	0,151	14,907	0,022	0,004	0,920	14,907	0,024	0,006	0,920
$T_{10,10,10}$	16,586	3,038	2,006	0,722	16,584	1,786	1,246	0,179	16,551	1,978	1,362	0,716	16,580	1,614	0,858	0,574
$T_{15,20,15}$	26,900	91,920	69,060	0,562	26,711	87,882	62,902	0,959	26,733	52,688	35,472	1,236	**26,667**	83,990	48,632	2,186
$T^r_{10,50,10}$	43,289	53,910	48,826	1,845	40,539	67,026	62,956	5,607	42,910	23,158	19,340	2,571	43,324	18,450	15,220	1,727
$T^r_{30,60,10}$	40,716	54,066	38,644	1,243	40,596	196,038	142,204	1,864	40,630	127,680	102,188	0,844	40,886	56,812	38,122	0,750
Просек	20,718	16,945	13,219	1,123	20,437	29,410	22,454	1,191	20,601	17,140	13,204	0,926	20,651	**13,419**	**8,575**	**0,893**

Инстанца	$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 2}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 4}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	0,000	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	0,441	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	0,000
$T_{5,3,3}$	**13,156**	0,000	0,000	0,434	**13,156**	0,000	0,000	1,303	**13,156**	0,000	0,000	0,434	**13,156**	0,000	0,000	2,172
$T_{3,3,5}$	**17,164**	0,004	0,002	0,999	**17,164**	0,002	0,000	0,666	**17,164**	0,004	0,000	0,333	**17,164**	0,006	0,002	0,666
$T_{3,4,4}$	**16,303**	0,004	0,002	0,147	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,002	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,000	**10,866**	0,002	0,000	0,000
$T_{5,5,10}$	17,957	0,234	0,144	0,636	17,896	0,190	0,138	0,562	17,900	0,175	0,068	0,798	**17,871**	0,296	0,154	0,799
$T_{8,6,5}$	15,136	0,036	0,004	0,000	**14,336**	0,050	0,022	3,422	14,707	0,034	0,014	1,399	14,707	0,054	0,012	1,710
$T_{10,10,10}$	16,554	1,828	1,226	0,670	**16,550**	1,728	1,104	0,792	16,580	2,022	1,218	0,819	16,553	3,202	2,286	0,155
$T_{15,20,15}$	26,819	71,966	45,176	1,701	26,696	93,094	58,998	2,188	26,671	59,444	42,664	0,740	26,727	106,484	70,662	0,977
$T^r_{10,50,10}$	41,186	63,388	50,010	4,335	39,931	161,396	123,482	3,626	**39,696**	237,940	113,718	6,329	39,730	84,966	81,172	7,372
$T^r_{30,60,10}$	40,674	83,504	24,060	1,406	40,434	246,762	225,216	1,791	41,366	158,060	86,800	0,277	**40,313**	149,082	94,552	2,149
Просек	20,487	18,414	10,052	1,013	**20,280**	41,935	34,080	1,268	20,370	38,140	20,374	0,927	20,285	28,675	20,737	1,362

Табела 7.8: Тестирање параметра α у SVNS методи за проблем VSP

Инстанца	$\alpha = \mathbf{0,025}$				$\alpha = \mathbf{0,05}$				$\alpha = \mathbf{0,10}$				$\alpha = \mathbf{0,25}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,000
$T_{5,2,4}$	**13,156**	0,000	0,000	1,303	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	3,041
$T_{5,3,3}$	**13,156**	0,000	0,000	0,869	**13,156**	0,000	0,000	1,303	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	1,737
$T_{3,3,5}$	**17,164**	0,004	0,002	0,000	**17,164**	0,002	0,000	0,666	**17,164**	0,004	0,000	0,666	**17,164**	0,004	0,000	0,666
$T_{3,4,4}$	**16,303**	0,004	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,004	0,004	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,526	**10,866**	0,002	0,000	0,000
$T_{5,5,10}$	**17,863**	0,258	0,152	0,333	17,896	0,190	0,138	0,562	17,900	0,284	0,194	0,670	17,900	0,254	0,106	0,857
$T_{8,6,5}$	14,536	0,048	0,006	2,673	**14,336**	0,050	0,022	3,422	14,907	0,046	0,002	1,227	14,536	0,052	0,018	2,595
$T_{10,10,10}$	16,643	3,604	2,284	1,229	16,550	1,728	1,104	0,792	**16,494**	3,362	2,236	1,226	16,526	3,044	1,946	0,740
$T_{15,20,15}$	26,697	90,446	63,206	1,794	26,696	93,094	58,998	2,188	**26,671**	70,690	65,350	1,167	26,729	127,850	81,644	1,009
$T^r_{10,50,10}$	40,950	381,656	376,642	3,727	**39,931**	161,396	123,482	3,626	40,216	58,356	54,574	7,094	41,727	97,154	78,824	2,294
$T^r_{30,60,15}$	40,730	134,394	110,312	1,399	40,434	246,762	225,216	1,791	40,914	106,432	40,720	0,933	**40,404**	102,054	65,262	1,604
Просек	20,412	50,868	46,050	1,046	**20,280**	41,935	34,080	1,268	20,385	**19,932**	**13,590**	1,335	20,445	27,535	18,983	1,015
Инстанца	$\alpha = \mathbf{0,50}$				$\alpha = \mathbf{0,75}$				$\alpha = \mathbf{0,80}$				$\alpha = \mathbf{0,90}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,000	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	0,441	**13,156**	0,000	0,000	0,000	**13,156**	0,000	0,000	1,310	**13,156**	0,000	0,000	0,000
$T_{5,3,3}$	**13,156**	0,000	0,000	3,041	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	1,303
$T_{3,3,5}$	**17,164**	0,006	0,002	0,333	**17,164**	0,004	0,004	0,000	**17,164**	0,004	0,000	0,333	**17,164**	0,004	0,000	0,333
$T_{3,4,4}$	**16,303**	0,004	0,002	0,000	**16,303**	0,004	0,000	0,000	**16,303**	0,004	0,002	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	0,596	**10,866**	0,002	0,000	0,000	**10,866**	0,002	0,000	0,596	**10,866**	0,000	0,000	1,052
$T_{5,5,10}$	17,896	0,310	0,106	0,457	17,900	0,288	0,168	0,543	17,900	0,312	0,206	0,091	17,900	0,260	0,112	0,255
$T_{8,6,5}$	14,907	0,046	0,012	0,805	14,707	0,048	0,012	1,710	14,707	0,044	0,016	2,020	14,764	0,044	0,010	1,974
$T_{10,10,10}$	16,556	2,586	1,488	1,526	16,700	2,908	1,652	0,536	16,523	3,354	2,206	0,996	16,551	3,496	2,536	0,475
$T_{15,20,15}$	26,727	152,338	135,254	0,933	26,876	138,684	95,184	1,675	26,734	124,370	104,012	1,122	27,070	129,760	88,800	0,435
$T^r_{10,50,10}$	41,943	51,232	48,820	1,529	41,637	50,156	43,850	1,886	40,323	199,232	196,722	4,818	40,134	68,874	61,734	7,215
$T^r_{30,60,15}$	40,920	131,992	82,812	0,828	41,121	112,180	78,966	0,946	40,999	123,338	58,816	0,751	40,759	136,328	75,826	0,812
Просек	20,539	28,210	22,375	0,874	20,538	25,356	18,320	**0,809**	20,392	37,555	30,165	1,204	20,391	28,231	19,085	1,183

Табела 7.9: Тестирање параметра r_{max} у методи BVNSi$_B$ за проблем VSP

Инстанца	$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 2}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,016
$T_{5,2,4}$	14,017	0,000	0,000	0,408	14,017	0,000	0,000	0,815	**13,156**	0,000	0,000	5,673	**13,156**	0,000	0,000	4,363
$T_{5,3,3}$	13,727	0,000	0,000	1,249	**13,156**	0,000	0,000	3,909	**13,156**	0,000	0,000	4,344	**13,156**	0,000	0,000	2,172
$T_{3,3,5}$	**17,164**	0,000	0,000	1,332	17,450	0,000	0,000	0,000	**17,164**	0,002	0,000	1,530	**17,164**	0,002	0,000	0,666
$T_{3,4,4}$	**16,303**	0,000	0,000	0,417	**16,303**	0,002	0,002	0,000	**16,303**	0,000	0,000	0,626	**16,303**	0,002	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	2,104	**10,866**	0,000	0,000	2,104	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	2,104
$T_{5,5,10}$	18,100	0,230	0,146	0,852	18,239	0,238	0,130	0,238	**17,900**	0,414	0,284	1,304	17,984	0,340	0,220	0,764
$T_{8,6,5}$	14,907	0,006	0,002	1,227	14,764	0,008	0,002	2,013	14,907	0,014	0,006	0,613	**14,536**	0,012	0,008	3,184
$T_{10,10,10}$	16,956	3,008	2,018	1,001	16,963	3,372	2,156	1,324	17,014	3,260	2,100	0,705	16,929	8,576	6,110	0,776
$T_{15,20,15}$	27,560	119,132	81,388	2,406	27,876	105,252	58,338	1,127	27,368	267,356	229,282	1,160	27,504	155,316	109,380	1,940
$T^r_{10,50,10}$	40,101	1975,118	1648,796	2,123	39,854	2561,432	2192,128	1,462	39,981	1737,514	1368,960	1,998	40,694	3187,540	2405,374	1,620
$T^r_{30,60,10}$	40,033	12806,134	6178,408	0,788	**38,613**	19905,260	10256,210	4,277	39,406	30852,680	20009,482	1,407	40,279	27193,380	15186,612	0,875
Просек	20,551	**1241,969**	**659,230**	**1,215**	20,415	1881,297	1042,414	1,552	20,341	2738,437	1800,843	1,858	20,454	2545,431	1475,642	1,623

Инстанца	$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 2}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 4}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,339
$T_{5,2,4}$	**13,156**	0,002	0,000	6,107	**13,156**	0,000	0,000	4,363	**13,156**	0,000	0,000	5,238	**13,156**	0,000	0,000	5,238
$T_{5,3,3}$	**13,156**	0,000	0,000	2,606	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	1,303
$T_{3,3,5}$	**17,164**	0,002	0,000	1,332	**17,164**	0,002	0,000	1,728	**17,164**	0,002	0,000	1,530	**17,164**	0,004	0,002	1,197
$T_{3,4,4}$	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,004	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,000
$T_{5,5,10}$	18,157	0,338	0,216	0,441	17,929	0,738	0,476	1,275	18,070	0,360	0,216	0,860	17,957	0,428	0,268	1,239
$T_{8,6,5}$	14,707	0,014	0,004	2,020	14,707	0,026	0,006	2,331	14,707	0,026	0,010	1,160	14,907	0,018	0,006	1,112
$T_{10,10,10}$	16,897	4,718	3,398	0,891	**16,869**	11,150	8,530	0,593	16,929	5,020	3,570	0,641	17,014	5,026	3,176	0,840
$T_{15,20,15}$	**27,161**	249,974	198,262	2,332	27,304	188,636	142,906	1,360	27,419	194,284	139,870	1,876	27,590	238,930	147,158	1,624
$T^r_{10,50,10}$	40,037	2742,370	2143,452	1,830	40,387	3601,120	2783,164	1,380	40,100	1867,710	1322,622	3,182	**39,820**	3208,826	2605,224	2,598
$T^r_{30,60,10}$	38,806	29608,102	19486,994	2,302	39,043	55885,376	38059,562	1,628	39,650	23337,862	11941,614	1,190	39,920	39567,370	23897,698	1,039
Просек	**20,274**	2717,127	1819,361	1,799	20,313	4973,921	3416,220	1,482	20,366	2117,106	1117,325	1,616	20,394	3585,051	2221,128	1,377

Табела 7.10: Тестирање параметра p у методи BVNSi$_B$ за проблем VSP

Инстанца	**p = 0, 5/(n + 0, 5)**				**p = 1/(n + 1)**				**p = 1, 5/(n + 1, 5)**				**p = 2/(n + 2)**			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,000
$T_{5,2,4}$	14,017	0,000	0,000	6,897	14,017	0,000	0,000	6,107	**13,156**	0,000	0,000	4,804	**13,156**	0,000	0,000	5,673
$T_{5,3,3}$	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	2,606	**13,156**	0,000	0,000	1,303	**13,156**	0,000	0,000	2,172
$T_{3,3,5}$	**17,164**	0,002	0,002	1,196	17,450	0,002	0,000	1,332	**17,164**	0,002	0,000	1,294	**17,164**	0,002	0,002	1,332
$T_{3,4,4}$	**16,303**	0,002	0,000	0,209	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	0,526
$T_{5,5,10}$	18,214	0,276	0,148	0,125	18,157	0,338	0,216	0,441	18,157	0,342	0,208	0,472	18,071	0,288	0,172	0,531
$T_{8,6,5}$	**14,707**	0,016	0,006	2,020	**14,707**	0,014	0,004	2,020	**14,707**	0,016	0,006	2,331	**14,707**	0,024	0,014	1,826
$T_{10,10,10}$	16,986	3,882	2,646	0,297	16,897	4,718	3,398	0,891	16,871	5,356	4,156	0,676	**16,814**	7,474	5,190	1,115
$T_{15,20,15}$	27,533	203,094	151,544	1,494	27,161	249,974	198,262	2,332	27,103	183,326	134,054	2,323	27,390	174,460	123,988	0,835
$T^r_{10,50,10}$	39,854	3967,866	3199,890	2,221	40,037	2742,370	2143,452	1,830	40,157	2457,166	2045,816	0,372	40,334	2159,394	1728,358	0,635
$T^r_{30,60,10}$	38,986	55146,398	41421,184	2,423	**38,806**	29608,102	19486,994	2,302	39,679	25241,130	15776,810	1,377	39,884	27411,476	13376,776	1,668
Просек	20,317	4943,461	3731,285	1,776	**20,274**	2717,127	1819,361	1,799	20,349	2323,945	1496,754	1,471	20,393	2479,427	1269,542	1,359
Инстанца	**p = 2, 5/(n + 2, 5)**				**p = 3/(n + 3)**				**p = 3, 5/(n + 3, 5)**				**p = 4/(n + 4)**			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,677
$T_{5,2,4}$	**13,156**	0,000	0,000	5,673	**13,156**	0,000	0,000	4,363	**13,156**	0,000	0,000	5,673	**13,156**	0,000	0,000	3,929
$T_{5,3,3}$	**13,156**	0,000	0,000	1,303	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	3,041	**13,156**	0,000	0,000	1,797
$T_{3,3,5}$	**17,164**	0,002	0,000	1,197	**17,164**	0,002	0,000	0,666	**17,164**	0,004	0,004	1,197	**17,164**	0,002	0,000	0,666
$T_{3,4,4}$	**16,303**	0,002	0,002	0,000	**16,303**	0,002	0,002	0,000	**16,303**	0,004	0,002	0,000	**16,303**	0,002	0,002	0,209
$T_{6,5,2}$	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,526
$T_{5,5,10}$	**17,929**	0,606	0,396	1,466	17,951	0,558	0,330	1,203	18,129	0,662	0,434	0,347	17,957	0,418	0,294	1,241
$T_{8,6,5}$	**14,707**	0,014	0,004	2,020	**14,707**	0,014	0,004	2,331	14,907	0,024	0,004	1,227	14,907	0,016	0,004	1,227
$T_{10,10,10}$	16,840	5,524	3,834	0,864	16,957	7,570	5,014	0,775	16,929	6,946	4,396	0,658	16,839	3,968	2,600	1,072
$T_{15,20,15}$	27,390	319,888	229,912	1,481	27,590	140,334	82,964	2,076	**27,097**	402,386	317,082	2,445	27,533	177,424	127,226	0,892
$T^r_{10,50,10}$	40,049	2930,548	2202,994	2,329	**39,416**	2293,974	1866,326	3,068	39,593	2287,822	1794,254	3,978	40,667	1936,852	1405,418	1,769
$T^r_{30,60,10}$	39,334	32386,794	17646,182	1,899	39,553	21946,420	12570,620	2,016	39,599	18526,902	9126,980	2,287	40,007	22152,020	12371,850	0,659
Просек	20,314	2970,282	1673,610	1,676	20,308	2032,406	1210,438	1,648	20,314	**1768,729**	**936,930**	1,866	20,452	2022,559	1158,950	**1,217**

Табела 7.11: Тестирање параметра r_{max} у методи BVNSi$_F$ за проблем VSP

Инстанца	$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 1}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 2}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,016
$T_{5,2,4}$	14,017	0,000	0,000	1,223	14,017	0,000	0,000	0,815	14,017	0,000	0,000	0,815	**13,156**	0,000	0,000	5,673
$T_{5,3,3}$	13,441	0,000	0,000	2,976	13,441	0,000	0,000	1,275	13,441	0,000	0,000	1,275	13,441	0,000	0,000	1,275
$T_{3,3,5}$	**17,164**	0,000	0,000	1,332	**17,164**	0,000	0,000	1,332	**17,164**	0,002	0,002	0,999	**17,164**	0,002	0,002	0,999
$T_{3,4,4}$	**16,303**	0,000	0,000	0,209	**16,303**	0,000	0,000	0,000	**16,303**	0,000	0,000	0,000	**16,303**	0,000	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	1,578	**10,866**	0,000	0,000	0,526
$T_{5,5,10}$	18,129	0,088	0,042	0,851	18,043	0,092	0,048	1,210	18,043	0,092	0,048	1,210	**17,951**	0,148	0,090	1,375
$T_{8,6,5}$	14,907	0,004	0,000	1,227	14,907	0,006	0,002	0,613	**14,707**	0,008	0,000	2,020	**14,707**	0,008	0,000	2,020
$T_{10,10,10}$	16,929	0,844	0,584	1,181	16,871	0,740	0,458	1,233	16,871	0,738	0,454	1,233	16,929	0,934	0,612	0,810
$T_{15,20,15}$	**27,591**	25,188	18,550	1,963	27,904	33,906	25,044	0,962	27,647	29,076	21,740	1,550	27,647	29,550	22,096	1,550
$T^r_{10,50,10}$	42,729	73,382	55,162	2,589	42,843	81,304	67,220	1,534	42,843	80,094	66,190	1,534	43,580	36,226	33,184	0,905
$T^r_{30,60,10}$	**40,481**	68,866	52,816	1,510	40,836	59,108	43,316	0,826	40,679	47,470	32,000	1,610	40,679	47,580	32,074	1,610
Просек	20,786	14,031	10,596	1,500	20,839	14,596	11,341	**1,061**	20,788	13,123	10,036	1,265	20,775	**9,537**	**7,338**	1,480

Инстанца	$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 2}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor (k_{max}+1)/2 \rfloor + 3}$				$\mathbf{r_{max} = \lfloor k_{max}/2 \rfloor + 4}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	0,677
$T_{5,2,4}$	**13,156**	0,000	0,000	5,673	**13,156**	0,000	0,000	5,238	**13,156**	0,000	0,000	5,238	**13,156**	0,000	0,000	3,494
$T_{5,3,3}$	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	3,041	**13,156**	0,000	0,000	3,041
$T_{3,3,5}$	**17,164**	0,002	0,002	1,332	**17,164**	0,002	0,002	1,332	**17,164**	0,002	0,000	0,999	**17,164**	0,002	0,000	0,999
$T_{3,4,4}$	**16,303**	0,000	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,000	**10,866**	0,000	0,000	0,000	**10,866**	0,000	0,000	0,000
$T_{5,5,10}$	**17,951**	0,144	0,086	1,375	18,214	0,184	0,068	0,471	18,214	0,114	0,044	0,471	18,071	0,164	0,102	0,765
$T_{8,6,5}$	**14,707**	0,010	0,002	1,399	**14,707**	0,018	0,006	1,399	14,821	0,012	0,002	1,080	14,821	0,012	0,002	1,080
$T_{10,10,10}$	16,929	0,924	0,606	0,810	**16,814**	1,696	1,008	1,359	**16,814**	0,902	0,536	1,359	16,900	1,240	0,762	0,676
$T_{15,20,15}$	27,619	32,910	23,138	1,750	27,619	33,292	23,400	1,750	27,676	25,020	13,772	1,920	27,676	25,202	13,876	1,920
$T^r_{10,50,10}$	43,580	35,706	32,710	0,905	**42,234**	45,550	42,110	2,585	**42,234**	44,684	41,306	2,585	42,899	42,542	26,580	2,271
$T^r_{30,60,10}$	40,671	152,322	117,844	0,843	40,671	156,354	121,562	0,843	40,597	113,968	95,620	0,911	40,597	116,890	98,066	0,911
Просек	20,748	18,502	14,532	1,483	**20,648**	19,758	15,680	1,514	20,656	15,392	12,607	1,552	20,707	15,505	11,616	1,320

Табела 7.12: Тестирање параметра p у методи BVNSi$_F$ за проблем VSP

Инстанца	$\mathbf{p = 0,5/(n+0,5)}$				$\mathbf{p = 1/(n+1)}$				$\mathbf{p = 1,5/(n+1,5)}$				$\mathbf{p = 2/(n+2)}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	1,016	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	1,355
$T_{5,2,4}$	14,017	0,000	0,000	5,238	14,017	0,000	0,000	5,238	**13,156**	0,000	0,000	6,541	14,017	0,000	0,000	0,408
$T_{5,3,3}$	**13,156**	0,000	0,000	3,475	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	2,172	**13,156**	0,000	0,000	3,929
$T_{3,3,5}$	**17,164**	0,002	0,002	0,999	17,450	0,002	0,002	1,332	**17,164**	0,002	0,000	0,333	**17,164**	0,002	0,000	0,333
$T_{3,4,4}$	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	0,000	**10,866**	0,000	0,000	0,000
$T_{5,5,10}$	18,214	0,150	0,086	0,869	18,214	0,184	0,068	0,471	17,986	0,142	0,076	1,271	17,929	0,128	0,068	1,433
$T_{8,6,5}$	14,907	0,008	0,000	0,613	14,707	0,018	0,006	1,399	14,907	0,008	0,002	1,227	14,536	0,012	0,006	2,673
$T_{10,10,10}$	16,956	1,226	0,870	1,517	16,814	1,696	1,008	1,359	17,100	0,878	0,524	0,635	16,929	1,222	0,838	0,641
$T_{15,20,15}$	27,933	33,288	21,200	0,877	27,619	33,292	23,400	1,750	27,619	34,050	25,594	1,003	27,443	30,868	21,510	2,095
$T^r_{10,50,10}$	40,797	36,292	33,362	7,157	42,234	45,550	42,110	2,585	42,234	97,638	74,612	4,155	42,234	101,654	74,936	4,338
$T^r_{30,60,10}$	**40,671**	152,546	114,740	0,939	**40,671**	156,354	121,562	0,843	**40,671**	134,556	101,654	0,448	**40,671**	129,528	98,596	0,519
Просек	20,583	18,626	14,188	1,951	20,648	19,758	15,680	1,514	20,670	22,273	16,872	1,595	20,677	21,951	16,330	1,477

Инстанца	$\mathbf{p = 2,5/(n+2,5)}$				$\mathbf{p = 3/(n+3)}$				$\mathbf{p = 3,5/(n+3,5)}$				$\mathbf{p = 4/(n+4)}$			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	0,000	0,000	0,677	**16,874**	0,000	0,000	0,339	**16,874**	0,000	0,000	1,355	**16,874**	0,000	0,000	0,677
$T_{5,2,4}$	**13,156**	0,000	0,000	3,929	**13,156**	0,000	0,000	0,815	**13,156**	0,000	0,000	4,797	**13,156**	0,000	0,000	3,935
$T_{5,3,3}$	**13,156**	0,000	0,000	1,737	**13,156**	0,000	0,000	0,434	**13,156**	0,000	0,000	3,909	**13,156**	0,000	0,000	0,869
$T_{3,3,5}$	**17,164**	0,002	0,002	0,666	**17,164**	0,002	0,002	0,999	**17,164**	0,002	0,000	0,666	**17,164**	0,002	0,002	0,666
$T_{3,4,4}$	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,000	0,000	**16,303**	0,002	0,002	0,000	**16,303**	0,000	0,000	0,000
$T_{6,5,2}$	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	1,052	**10,866**	0,000	0,000	0,526	**10,866**	0,000	0,000	0,000
$T_{5,5,10}$	18,213	0,122	0,058	0,289	**17,924**	0,164	0,104	1,294	17,986	0,124	0,048	1,048	18,100	0,138	0,076	0,379
$T_{8,6,5}$	14,907	0,008	0,002	1,227	**14,336**	0,020	0,012	3,229	14,907	0,008	0,002	1,227	14,707	0,012	0,002	2,020
$T_{10,10,10}$	16,986	0,914	0,516	0,303	**16,729**	1,184	0,806	1,462	16,899	1,338	1,008	0,477	16,783	1,278	0,876	1,825
$T_{15,20,15}$	27,561	26,136	15,674	2,215	27,446	31,898	23,204	2,511	**27,277**	37,076	26,546	2,573	27,443	31,536	24,848	2,166
$T^r_{10,50,10}$	42,854	74,250	70,968	1,569	40,861	77,996	67,544	5,935	41,833	48,280	40,648	3,825	**40,671**	70,714	64,284	6,051
$T^r_{30,60,10}$	40,760	218,934	199,760	0,948	40,797	180,450	163,482	1,123	40,821	225,038	200,034	1,275	40,833	70,792	45,668	1,508
Просек	20,733	26,697	23,915	**1,218**	**20,468**	24,310	21,263	1,599	20,604	25,989	22,357	1,807	20,505	**14,539**	**11,313**	1,675

Резултати и поређења

Предложени BVNS, SVNS и BVNSi алгоритми су евалуирани на свим реалним и генерисаним тест примерима. Сваки од алгоритама је узастопно извршаван 30 пута на свакој инстанци. Све три предложене метахеуристичке методе користе комбинацију два критеријума заустављања: достигнут максималан број итерација ($iter_{max}$) и прекорачен лимит времена извршавања (t_{max}). Алгоритми се заустављају када се испуни један од та два услова. Вредности параметара који се користе код критеријума заустављања су:

- $iter_{max} = 5$, $t_{max} = 1$s за реалне инстанце малих димензија;
- $iter_{max} = 6$ код BVNS и BVNSi$_B$, $iter_{max} = 10$ код SVNS и BVNSi$_F$, $t_{max} = 500$s за реалне инстанце средњих димензија;
- $iter_{max} = 8$ код BVNS и BVNSi$_B$, $iter_{max} = 15$ код SVNS и BVNSi$_F$, $t_{max} = 20000$s за генарисане инстанце.

Изабране су различите вредности параметра $iter_{max}$, имајући у виду да су SVNS и BVNSi$_F$ алгоритми значајно бржи од BVNS и BVNSi$_B$, јер користе стратегију првог побољшања у фази локалне претраге.

Најпре су у Табели 7.13 представљени експериментални резултати на реалним инстанцама малих димензија коришћењем Lingo решавача, добијени у оквиру ограниченог времена извршавања од 10h. Прва колона $T_{n,m,k_{max}}$ садржи назив инстанце. У другој колони представљена је вредност функције циља која одговара оптималном решењу. Наредних шест колона представљају детаљније резултате тестирања MIQCP/MILP формулација Lingo решавачем и садрже: вредност функције циља најбољег пронађеног допустивог решења (UB), одговарајућу доњу границу вредности функције циља (LB) и време извршавања ($t(s)$). У колонама UB и LB је коришћена ознака *opt* у одговарајућим пољима у случају када се UB и LB поклапају са *opt.sol.*. Ако у оквиру датог времена извршавања није пронађено оптимално или допустиво решење, ознака / је уписана у одговарајуће поље.

Из резултата представљених у Табели 7.13 може се видети да је MILP формулација успешнија од MIQCP. Коришћењем MILP модела, Lingo решавач је нашао оптимална решења за сваку од 32 инстанце малих димензија, са изузетком инстанце $T_{4,4,4}$, за коју је пронађено допустиво решење. Користећи MIQCP

Табела 7.13: Експериментални резултати Lingo решавача на реалним инстанцама малих димензија

Инстанце		**MIQCP**			**MILP**		
$T_{n,m,k_{max}}$	**opt. sol.**	*UB*	*LB*	$t(s)$	*UB*	*LB*	$t(s)$
$T_{3,2,4}$	**16,874**	**opt**	**opt**	121,170	**opt**	**opt**	56,620
$T_{3,3,2}$	**9,780**	**opt**	**opt**	61,400	**opt**	**opt**	18,880
$T_{3,3,3}$	**13,727**	**opt**	**opt**	2719,230	**opt**	**opt**	209,530
$T_{3,3,4}$	**16,303**	21,2843	15,597	36000,000	**opt**	**opt**	1950,930
$T_{3,3,5}$	**17,164**	23,919	16,430	36000,000	**opt**	**opt**	4735,130
$T_{3,4,2}$	**10,750**	**opt**	**opt**	1385,230	**opt**	**opt**	368,970
$T_{3,4,3}$	**13,897**	18,216	13,727	36000,000	**opt**	**opt**	2851,73
$T_{3,4,4}$	**16,303**	/	/	36000,000	**opt**	**opt**	17687,21
$T_{3,5,2}$	**12,294**	**opt**	**opt**	8917,850	**opt**	**opt**	793,320
$T_{4,2,3}$	**13,040**	**opt**	**opt**	15,300	**opt**	**opt**	36,170
$T_{4,2,4}$	**13,446**	**opt**	**opt**	260,910	**opt**	**opt**	81,940
$T_{4,3,2}$	**10,580**	**opt**	**opt**	92,420	**opt**	**opt**	24,980
$T_{4,3,3}$	**12,299**	**opt**	**opt**	7903,180	**opt**	**opt**	737,060
$T_{4,4,2}$	**11,151**	**opt**	**opt**	848,820	**opt**	**opt**	160,110
$T_{4,4,3}$	**14,299**	16,813	13,934	36000,000	**opt**	**opt**	2941,23
$T_{4,4,4}$	/	22,0757	15,799	36000,000	**16,874**	16,589	36000,000
$T_{4,5,2}$	**10,866**	**opt**	**opt**	11146,910	**opt**	**opt**	1297,770
$T_{5,2,3}$	**10,754**	**opt**	**opt**	54,900	**opt**	**opt**	16,410
$T_{5,2,4}$	**13,156**	**opt**	**opt**	2454,280	**opt**	**opt**	1560,850
$T_{5,3,2}$	**10,750**	**opt**	**opt**	259,370	**opt**	**opt**	39,240
$T_{5,3,3}$	**13,156**	13,959	12,503	36000,000	**opt**	**opt**	1517,030
$T_{5,3,4}$	**16,589**	20,647	16,259	36000,000	**opt**	**opt**	5315,82
$T_{5,4,3}$	**13,156**	17,450	12,961	36000,000	**opt**	**opt**	3717,61
$T_{5,5,2}$	**11,437**	11,607	11,437	36000,000	**opt**	**opt**	3407,02
$T_{6,2,3}$	**11,727**	**opt**	**opt**	540,220	**opt**	**opt**	19,070
$T_{6,2,4}$	**13,326**	**opt**	**opt**	26915,830	**opt**	**opt**	883,280
$T_{6,3,2}$	**10,294**	**opt**	**opt**	172,550	**opt**	**opt**	78,760
$T_{6,3,3}$	**12,870**	**opt**	**opt**	26787,01	**opt**	**opt**	1729,44
$T_{6,4,2}$	**10,009**	**opt**	**opt**	5144,310	**opt**	**opt**	243,130
$T_{6,5,2}$	**10,866**	14,534	10,294	36000,000	**opt**	**opt**	5212,77
$T_{7,3,3}$	**12,299**	13,156	12,299	36000,000	**opt**	**opt**	608,950
$T_{7,5,2}$	**10,580**	13,843	10,253	36000,000	**opt**	**opt**	13751,400
Просек	/	/	/	17618,918	**12,832**	12,823	3376,636

формулацију, само 19 од 32 инстанце малих димензија су решене до оптималности у оквиру постављеног лимита времена извршавања. Допустива решења су добијена за 12 од преосталих 13 инстанци, док у случају једне инстанце ($T_{3,4,4}$), Lingo није успео да пронађе чак ни допустиво решење користећи предложен MIQCP модел. Просечно време извршавања Lingo решавача при тестирању MILP формулације износи 3376,636s, што је преко 5 пута брже од просечног времена извршавања коришћењем MIQCP модела (17618,918s).

У Табели 7.14 представљени су резултати тестирања предложених BVNS и SVNS метода на скупу малих инстанци. Прва колона садржи име инстанце, док се друга и трећа колона ове табеле односе на најбоље Lingo резултате добијене при тестирању предложених MIQCP и MILP формулација. У колони *best.sol.* налази се вредност функције циља која одговара најбољем решењу добијеном коришћењем Lingo решавача, а у колони $t(s)$ одговарајуће време извршавања.

Табела 7.14: Експериментални резултати BVNS и SVNS методе на реалним инстанцама малих димензија

Инстанце	**Lingo**		**BVNS**				**SVNS**			
$T_{n,m,k_{max}}$	**best. sol.**	$t(s)$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	56,620	**opt**	0,001	0,000	0,734	**opt**	0,001	0,000	0,282
$T_{3,3,2}$	**9,780**	18,880	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{3,3,3}$	**13,727**	209,530	**opt**	0,000	0,000	0,000	**opt**	0,001	0,000	0,000
$T_{3,3,4}$	**16,303**	1950,930	**opt**	0,001	0,000	0,000	**opt**	0,002	0,000	0,000
$T_{3,3,5}$	**17,164**	4735,130	**opt**	0,005	0,002	0,999	**opt**	0,002	0,001	0,610
$T_{3,4,2}$	**10,750**	368,970	**opt**	0,000	0,000	2,365	**opt**	0,001	0,001	0,622
$T_{3,4,3}$	**13,897**	2851,73	**opt**	0,002	0,000	2,696	**opt**	0,002	0,001	2,503
$T_{3,4,4}$	**16,303**	17687,21	**opt**	0,004	0,001	0,035	**opt**	0,002	0,000	0,000
$T_{3,5,2}$	**12,294**	793,320	**opt**	0,001	0,000	0,930	**opt**	0,001	0,000	0,000
$T_{4,2,3}$	**13,040**	15,300	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{4,2,4}$	**13,446**	81,940	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{4,3,2}$	**10,580**	24,980	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{4,3,3}$	**12,299**	737,060	**opt**	0,001	0,000	1,936	**opt**	0,000	0,000	2,013
$T_{4,4,2}$	**11,151**	160,110	**opt**	0,000	0,000	3,416	**opt**	0,000	0,000	3,587
$T_{4,4,3}$	**14,299**	2941,23	**opt**	0,002	0,001	0,666	**opt**	0,002	0,000	0,266
$T_{4,4,4}$	**16,874**	36000,000	**16,874**	0,004	0,001	1,129	**16,874**	0,005	0,001	0,452
$T_{4,5,2}$	**10,866**	1297,770	**opt**	0,001	0,000	2,191	**opt**	0,001	0,000	1,139
$T_{5,2,3}$	**10,754**	16,410	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{5,2,4}$	**13,156**	1560,850	**opt**	0,001	0,000	1,540	**opt**	0,001	0,000	0,873
$T_{5,3,2}$	**10,750**	39,240	**opt**	0,000	0,000	0,215	**opt**	0,000	0,000	0,036
$T_{5,3,3}$	**13,156**	1517,030	**opt**	0,001	0,000	1,375	**opt**	0,001	0,000	1,448
$T_{5,3,4}$	**16,589**	5315,82	**opt**	0,002	0,000	0,919	**opt**	0,001	0,000	0,115
$T_{5,4,3}$	**13,156**	3717,61	**opt**	0,003	0,001	1,231	**opt**	0,002	0,001	0,869
$T_{5,5,2}$	**11,437**	3407,02	**opt**	0,001	0,000	2,415	**opt**	0,001	0,001	1,499
$T_{6,2,3}$	**11,727**	19,070	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,2,4}$	**13,326**	883,280	**opt**	0,001	0,000	0,089	**opt**	0,001	0,000	0,088
$T_{6,3,2}$	**10,294**	78,760	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,3,3}$	**12,870**	1729,44	**opt**	0,001	0,000	0,000	**opt**	0,001	0,000	0,000
$T_{6,4,2}$	**10,009**	243,130	**opt**	0,000	0,000	0,000	**opt**	0,001	0,000	0,000
$T_{6,5,2}$	**10,866**	5212,77	**opt**	0,002	0,001	1,402	**opt**	0,002	0,000	0,701
$T_{7,3,3}$	**12,299**	608,950	**opt**	0,001	0,000	0,774	**opt**	0,001	0,001	0,929
$T_{7,5,2}$	**10,580**	13751,400	**opt**	0,001	0,000	2,430	**opt**	0,002	0,000	1,620
Просек	**12,832**	3375,984	**12,832**	**0,001**	**0,000**	0,921	**12,832**	**0,001**	**0,000**	**0,614**

Експериментални резултати предложених метахеуристичких метода BVNS и SVNS на реалним инстанцама малих димензија су представљени на следећи начин. За сваку методу, прва колона садржи вредност функције циља која одговара најбољем пронађеном решењу (*best*), при чему је са *opt* означен случај када се то решење поклапа са оптималним *opt.sol*. Следеће две колоне које се односе време извршавања посматране методе и садрже просечно укупно време извршавања (t_{tot}) и просечно време које је било неопходно да се достигне најбоље пронађено решење (t) применом те методе. На крају, у колони (*gap*) налази се просечно процентуално одступање добијених решења у односу на оптимална, рачунато на основу 30 узастопних пуштања. Сва времена извршавања су изражена у секундама. Последњи ред Табеле 7.14, означен са **Просек** садржи просечне вредности свих представљених резултата.

Из резултата приказаних у Табели 7.14, може се видети да обе предложене

Табела 7.15: Експериментални резултати BVNSi$_B$ и BVNSi$_F$ методе на реалним инстанцама малих димензија

Инстанце	**Lingo**		**BVNSi$_B$**				**BVNSi$_F$**			
$T_{n,m,k_{max}}$	**best. sol.**	$t(s)$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{3,2,4}$	**16,874**	56,620	**opt**	0,001	0,000	0,847	**opt**	0,001	0,000	0,564
$T_{3,3,2}$	**9,780**	18,880	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{3,3,3}$	**13,727**	209,530	**opt**	0,000	0,000	0,029	**opt**	0,001	0,000	0,000
$T_{3,3,4}$	**16,303**	1950,930	**opt**	0,001	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{3,3,5}$	**17,164**	4735,130	**opt**	0,003	0,000	1,131	**opt**	0,002	0,000	1,032
$T_{3,4,2}$	**10,750**	368,970	**opt**	0,000	0,000	1,867	**opt**	0,000	0,000	1,494
$T_{3,4,3}$	**13,897**	2851,73	**opt**	0,001	0,000	2,407	**opt**	0,001	0,000	2,792
$T_{3,4,4}$	**16,303**	17687,21	**opt**	0,003	0,000	0,070	**opt**	0,002	0,000	0,070
$T_{3,5,2}$	**12,294**	793,320	**opt**	0,000	0,000	0,465	**opt**	0,000	0,000	0,155
$T_{4,2,3}$	**13,040**	15,300	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{4,2,4}$	**13,446**	81,940	**opt**	0,000	0,000	8,363	**opt**	0,000	0,000	0,000
$T_{4,3,2}$	**10,580**	24,980	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{4,3,3}$	**12,299**	737,060	**opt**	0,000	0,000	8,363	**opt**	0,000	0,000	6,350
$T_{4,4,2}$	**11,151**	160,110	**opt**	0,000	0,000	5,160	**opt**	0,000	0,000	4,441
$T_{4,4,3}$	**14,299**	2941,23	**opt**	0,001	0,000	0,266	**opt**	0,001	0,000	0,000
$T_{4,4,4}$	**16,874**	36000,000	**16,874**	0,002	0,001	1,196	**16,874**	0,002	0,001	0,857
$T_{4,5,2}$	**10,866**	1297,770	**opt**	0,000	0,000	5,876	**opt**	0,000	0,000	4,120
$T_{5,2,3}$	**10,754**	16,410	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{5,2,4}$	**13,156**	1560,850	**opt**	0,000	0,000	6,183	**opt**	0,000	0,000	5,600
$T_{5,3,2}$	**10,750**	39,240	**opt**	0,000	0,000	0,359	**opt**	0,000	0,000	0,215
$T_{5,3,3}$	**13,156**	1517,030	**opt**	0,000	0,000	2,534	**opt**	0,000	0,000	2,389
$T_{5,3,4}$	**16,589**	5315,82	**opt**	0,001	0,001	0,689	**opt**	0,001	0,000	0,517
$T_{5,4,3}$	**13,156**	3717,61	**opt**	0,001	0,001	2,534	**opt**	0,001	0,000	2,244
$T_{5,5,2}$	**11,437**	3407,02	**opt**	0,000	0,000	1,999	**opt**	0,000	0,000	1,083
$T_{6,2,3}$	**11,727**	19,070	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,2,4}$	**13,326**	883,280	**opt**	0,000	0,000	6,776	**opt**	0,000	0,000	4,811
$T_{6,3,2}$	**10,294**	78,760	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,3,3}$	**12,870**	1729,44	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,4,2}$	**10,009**	243,130	**opt**	0,000	0,000	0,000	**opt**	0,000	0,000	0,000
$T_{6,5,2}$	**10,866**	5212,77	**opt**	0,000	0,000	0,789	**opt**	0,000	0,000	0,701
$T_{7,3,3}$	**12,299**	608,950	**opt**	0,001	0,000	1,936	**opt**	0,000	0,000	1,394
$T_{7,5,2}$	**10,580**	13751,400	**opt**	0,000	0,000	2,520	**opt**	0,000	0,000	2,160
Просек	**12,832**	3375,984	**12,832**	**0,000**	**0,000**	1,949	**12,832**	**0,000**	**0,000**	**1,343**

методе достижу сва оптимална решења на тест примерима малих димензија које је Lingo решио до оптималности, а у случају инстанце $T_{4,4,4}$, решења добијена дизајнираним метахеуристичким методама се поклапају са допустивим Lingo решењем. BVNS и SVNS просечно укупно време извршавања је значајно краће од просечног времена извршавања Lingo решавача. На основу просечних вредности колона t_{tot} и t_s, може се извести закључак да су BVNS и SVNS методе подједнако брзе на малим тест примерима, али је SVNS метода стабилнија у поређењу са BVNS, с обзиром да је добијено мање просечно одступање рачунато у односу на оптимална решења (0,614% за SVNS, а 0,921% за BVNS).

Резултати тестирања две варијанте методе BVNSi на скупу реалних инстанци малих димензија представљени су Табелом 7.15. Структура Табеле 7.15 је иста као код Табле 7.14, стим што последњих 8 колона садрже резултате тестирања метода BVNSi$_B$ и BVNSi$_F$.

Анализирањем просечних вредности резултата приказаних у Табели 7.15, може се закључити да обе варијанте BVNSi достижу сва позната оптимална решења на скупу реалних инстанци малих димензија за занемарљиво мало време извршавања. Поређењем резултата све четири методе (BVNS, SVNS, BVNSi$_B$ и BVNSi$_F$), односно просечних вредности приказаних у одговарајућим колонама Табела 7.14 - 7.15, примећује се да су BVNSi$_B$ и BVNSi$_F$ за нијансу брже од предложених BVNS и SVNS метода, али су и мање стабилне. Применом метода BVNSi$_B$ и BVNSi$_F$ на скупу малих инстанци добијена су просечна процентуална одступања у односу на оптимална решења у износу од 1,949% и 1,343%, што је, редом, два пута веће од одговарајућих вредности (0,921% и 0,614%) за методе BVNS и SVNS. Притом је коректно поредити BVNSi$_B$ са BVNS, као и BVNSi$_F$ са SVNS методом, с обзиром да користе исту стратегију у фази локалне претраге.

Експериментални резултати на скупу реалних инстанци средњих димензија и скупу генерисаних инстанци представљени су Табелом 7.16. На овим тест примерима, Lingo није успео да пронађе чак ни допустиво решење у оквиру 10h извршавања. Из тог разлога, Табела 7.5 садржи само резултате добијене применом предложених метахеуристичких метода. Резултати на реалним инстанцама су представљени у горњем делу Табеле 7.16, а на генериснаним инстанцама већих димензија налазе се у доњем делу, при чему су колоне које одговарају свакој од метода организоване на исти начин као у Табелама 7.14 - 7.15.

Табела 7.16: Експериментални резултати на реалним инстанцама средњих димензија и већим генерисаним инстанцама за које Lingo решавач није пронашао допустива решења

Инстанца	**BVNS**				**SVNS**				**BVNSi$_B$**				**BVNSi$_F$**			
$T_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T_{4,5,7}$	**15,459**	0,130	0,048	1,169	**15,459**	0,189	0,087	1,491	**15,459**	0,125	0,043	4,314	**15,459**	0,061	0,024	3,793
$T_{5,5,5}$	**18,879**	0,038	0,016	2,772	**18,879**	0,024	0,012	2,016	**18,879**	0,014	0,007	4,592	**18,879**	0,017	0,009	3,298
$T_{5,5,10}$	**17,843**	0,587	0,343	0,745	**17,843**	0,286	0,192	0,560	17,894	0,405	0,257	1,938	17,900	0,308	0,179	1,518
$T_{5,10,10}$	**17,327**	8,963	5,761	0,924	17,357	3,908	2,401	1,110	17,390	9,889	6,565	1,912	17,441	3,289	2,068	2,122
$T_{5,20,20}$	34,646	784,091	163,415	4,600	35,074	554,309	275,789	7,645	**34,247**	513,767	252,088	3,687	35,041	513,517	394,133	6,560
$T_{6,6,6}$	**16,797**	0,065	0,015	0,414	**16,797**	0,140	0,022	0,312	16,854	0,079	0,026	1,066	**16,797**	0,047	0,018	0,980
$T_{6,7,6}$	**11,631**	0,209	0,125	1,279	11,654	0,238	0,133	1,358	11,689	0,191	0,116	3,443	11,656	0,110	0,072	2,458
$T_{7,5,6}$	16,226	0,046	0,018	2,215	16,226	0,096	0,050	1,554	16,338	0,054	0,029	4,316	**16,224**	0,044	0,025	3,151
$T_{8,6,5}$	**14,336**	0,041	0,009	4,404	**14,336**	0,053	0,020	4,033	**14,336**	0,032	0,008	5,081	14,536	0,023	0,007	3,027
$T_{8,40,10}$	28,209	508,59	397,797	3,010	**28,040**	200,534	130,721	2,800	28,709	414,398	263,364	5,515	28,280	227,204	140,645	2,833
$T_{10,10,10}$	**16,497**	7,317	5,009	0,854	16,500	5,354	3,469	0,750	16,753	9,016	6,216	3,365	16,700	2,393	1,655	1,542
$T_{15,20,15}$	26,667	374,468	285,789	1,273	**26,617**	135,178	102,528	1,166	27,247	239,513	183,005	4,470	26,990	62,542	43,942	1,166
Просек	**26,057**	187,172	95,372	**2,629**	26,087	100,034	**57,269**	2,755	26,199	131,943	79,080	4,855	26,211	**89,951**	64,753	4,102

Инстанца	**BVNS**				**SVNS**				**BVNSi$_B$**				**BVNSi$_F$**			
$T^r_{n,m,k_{max}}$	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*	*best*	$t_{tot}(s)$	$t(s)$	*gap(%)*
$T^r_{10,20,20}$	**59,453**	10398,013	8039,529	1,565	59,743	180,291	125,092	2,995	60,207	6463,622	5237,878	2,696	60,549	263,152	183,928	3,545
$T^r_{10,30,15}$	**50,371**	15717,992	13224,902	3,304	51,436	189,799	155,731	10,495	50,921	9475,923	7857,364	6,619	51,167	260,638	227,54	10,005
$T^r_{10,50,10}$	**38,614**	5302,06	4075,727	3,004	39,071	273,812	177,605	6,037	39,969	3044,93	2347,391	4,711	39,700	449,524	309,412	6,830
$T^r_{15,30,25}$	69,741	22919,492	13934,652	5,443	**69,137**	358,025	287,308	6,682	70,916	20146,577	14119,823	5,767	70,687	484,678	411,965	7,526
$T^r_{15,40,20}$	**63,684**	23005,872	9516,957	2,273	64,739	380,752	263,891	4,519	64,439	20185,318	14281,634	3,078	65,311	448,132	363,271	5,065
$T^r_{20,30,15}$	**39,576**	16679,69	12975,456	10,373	44,870	127,055	108,786	21,827	51,027	20000,010	0,000	0,000	51,027	20000,010	0,000	28,934
$T^r_{20,40,20}$	52,544	21667,833	12681,746	2,967	**51,930**	1555,367	1146,830	4,779	52,741	20174,265	12457,650	3,812	52,734	4752,372	3838,095	3,930
$T^r_{25,50,15}$	46,121	21559,712	14706,783	7,476	**44,981**	232,707	188,274	10,343	46,450	19920,055	15125,637	8,064	47,981	304,672	250,062	10,716
$T^r_{30,30,25}$	58,299	26590,842	7372,224	2,115	58,720	6710,664	3418,966	3,380	**57,683**	20151,399	12933,592	1,856	59,196	606,294	169,691	5,150
$T^r_{30,60,15}$	**39,331**	21893,615	12253,396	1,964	39,893	224,105	168,123	3,944	39,489	20139,6	16680,52	2,101	39,343	1421,185	1002,106	3,726
Просек	**51,773**	18573,512	10878,137	**4,048**	52,051	**1023,156**	**602,389**	7,500	53,384	15970,170	10104,149	6,764	53,770	2899,066	675,607	8,543

На основу резултата приказаних у горњем делу Табеле 7.16, може се закључити да је BVNS метода незнатно боља од остале три методе у погледу просечне вредности функције циља (26,057). Такође, ова метода се показала и најстабилнијом, с обзиром да просечно процентуално одступање добијних решења у односу на најбоља позната решења има најнижу вредност (2,629 %). Што се тиче просечног времена извршавања, најбржа метода је BVNSi$_F$, којој је у просеку било неопходно 89,951s да достигне своје најбоље решење. Са друге стране BVNSi$_F$ метода даје решења лошијег квалитета у односу на преостале три методе, с обзиром да је просечна вредност функције циља при примени методе BVNSi$_F$ (26,211) већа од одговарјућих вредности добијених тестирањем осталих метода. Ако предложене методе поредимо у пару, BVNS са SVNS и BVNSi$_B$ са BVNSi$_F$ (имајући у виду да користе исте структуре околина, а разликују се према стратегији у фази локалне претраге), можемо закључити да су на посматраном скупу реалних инстанци средњих димензија, методе које користе стратегију првог побољшања (BVNS и BVNSi$_B$) око 1,5 пута брже од метода које користе стратегију најбољег побољшања (SVNS и BVNSi$_F$), али са друге стране дају решења лошијег квалитета. При анализирању резултата на свакој од инстанци појединачно, може се приметити да су, упркос слабијој стабилности методе BVNSi$_B$ и лошијем квалитету решења код методе BVNSi$_F$, обе варијанте дизајниране BVNSi методе успеле да пронађу два нова најбоља позната решења, у случају инстанци $T_{5,20,20}$ и $T_{7,5,6}$.

Експериметални резултати предложених метахеуристичких метода на скупу од 10 генерисаних инстанци већих димензија (означених словом r), које прате структуру реалних инстанци, представљени су у доњем делу Табеле 7.16. На основу приказаних резултата може се закључити да метода BVNS постиже најбољи просечни квалитет решења с обзиром да је њеном применом добијена најнижа просечна вредност функције циља (51,773) на скупу генерисаних инстанци. Такође, BVNS метода се показала и најстабилнијом са најнижом вредношћу просечног процентуалног одступања добијених решења у односу на најбоља позната (4,048 %). У смислу просечног времена извршавања, најбоље перформансе је показала SVNS метода која је пронашла решења високог квалитета 18 пута брже у односу на BVNS методу, што је заправо резултат стратегије најбољег побољшања која се користи у фази локалне претраге код SVNS. Лошији квалитет решења BVNSi методе, који се примећује и при тестирању инстанци средњих димензија, долази до изражаја на скупу генерисаних тест примера.

Штавише, при тестирању инстанце $T^r_{20,30,15}$, ни једна од BVNSi$_B$ и BVNSi$_F$ метода није успела да поправи иницијално решење за 20000s извршавања. Из тог разлога је и просечно време извршавања на генерисаним инстанцама као и просечно процентуално одступање добијених решења у односу на најбоља позната решења показало лошије перформансе ове методе у поређењу са предложеним BVNS и SVNS метахеуристикама. Међутим и поред слабијих перформанси у односу на BVNS методу у смислу квалитета добијених решења, предложена SVNS метода је успела да пронађе нова најбоље позната решења у случају три генерисане инстанце већих димензија ($T^r_{15,30,25}$, $T^r_{20,40,20}$ и $T^r_{25,50,15}$), док је BVNSi$_B$ методом пронађено најбоље познато решење за инстанцу $T^r_{30,30,25}$.

Имплементације и експериментални резултати тестирања дизајнираних BVNS и SVNS метода изложени су у раду [7]. Предложене BVNS и SVNS методе су показале значајно боље перформансе у односу на варијанту методе променљивих околина која је претходно дизајнирана и изложена у раду [6], па су њен опис и резултати изостављени из ове докторске дисертације.

Глава 8

Закључак

Предмет ове докторске дисертације је нов проблем распоређивања возила при оптимизацији транспорта пољопривредних сировина. Рад је мотивисан реалним проблемом организације транспорта шећерне репе у једној фабрици шећера у Србији, али се развијени математички модели и методе решавања уз адекватне модификације могу применити на сродне проблеме који се јављају при оптимизацији транспорта сировина. Разматрани проблем распоређивања возила VSP је детаљно описан и развијени су одговарајући математички модели који укључују карактеристике и ограничења проблема. Поред полазног проблема, проучаван је и његов потпроблем VSP-P који је добијен уклањањем два сложена ограничења која се односе на сусрете возила на локацијама и у кругу фабрике. Полазни проблем VSP је доста сложенији јер та два услова захтевају посебну пажњу приликом моделирања и дизајнирања метода за решавање.

Проблем распоређивања возила VSP-P формулисан је у облику MIQCP модела који је потом трансформисан у еквивалентан MILP модел. Тестирања добијене MILP формулације егзактним CPLEX решавачем су показала да се оптимална решења добијају само на реалним инстанцама мањих димензија. У циљу решавања инстанци проблема већих димензија, дизајниране су две метахеуристичке методе: Општа метода променљивих околина (BVNS) и Похлепна стохастичко-адаптивна процедура претраге (GRASP). Серије прелиминарних тестова су извршене на подскупу тест инстанци у циљу одређивања адекватних вредности параметара у предложеним методама. Поређењем и анализирањем добијених резултата на скупу свих реалних и генерисаних тест примера закључено је да су обе предложене методе успешне при решавању проблема

VSP-P. Притом се BVNS метода показала бољом у погледу квалитета добијених решења и стабилности, док је GRASP надмашила BVNS са аспекта времена извршавања, али са нешто лошијим квалитетом добијених решења.

За проблем распоређивања возила VSP, најпре је предложена MIQCP формулација из које је одговарајућим трансформацијама добијен еквивалентан MILP модел. С обзиром на значајно повећање броја променљивих и ограничења при линеаризацији MIQCP модела, обе формулације су тестиране Lingo решавачем. Добијена су оптимална или допустива решења само на тест примерима малих димензија, при чему се MILP формулација показала успешнијом од MIQCP. За решавање реалних инстанци већих димензија, дизајниране су три варијанте методе променљивих околина: Општа метода променљивих околина (BVNS), Адаптивна метода променљивих околина (SVNS) и Побољшана општа метода променљивих околина (BVNSi). У BVNS је коришћена стратегија најбољег, а у SVNS стратегија првог побољшања у фази локалне претраге, док је BVNSi метода дизајнирана у две варијанте, једна са стратегијом првог (BVNSi$_F$), а друга са стратегијом најбољег побољшања (BVNSi$_B$) у фази локалне претраге. Извршени су прелиминарни експерименти на репрезентативном скупу инстанци како би се изабрале адекватне вредности параметара предложених метода. Затим су дизајниране методе тестиране на свим малим и средњим реалним инстацама и на генерисаним инстанцама већих димензија. При тестирању инстанци малих димензија, нема великих разлика у перформансама дизајнираних метода, али са порастом димензије тест примера примећене су веће разлике у добијеним резултатима. Генерално, методе које користе стратегију најбољег побољшања у фази локалне претраге (BVNS и BVNSi$_B$) су успешније када је у питању квалитет решења, а методе са стратегијом првог побољшања (SVNS и BVNSi$_F$) имају значајно краће време извршавања и притом дају решења високог квалитета. Општи закључак је да се приступ заснован на методи променљивих околина показао ефикасним за решавање проблема VSP.

Научни допринос ове дисертације састоји се из више аспеката. Најзначајнији резултати овог истраживања су:

- формулисање новог проблема распоређивања возила VSP и његовог потпроблема VSP-P,
- формулисање адекватних математичких модела мешовитог целобројног програмирања са квадратним ограничењима (MIQCP) за VSP и VSP-P,

- линеаризација MIQCP модела, односно њихова трансформација у моделе мешовитог целобројног линеарног програмирања (MILP),
- анализа сложености разматраног проблема. Прецизније, доказано је да је посматрани VSP НП-тежак проблем оптимизације, као и његов потпроблем VSP-P,
- дизајнирање и имплементација неколико метахеуристичких метода за решавање VSP и VSP-P. Елементи предложених метода су прилагођени разматраним проблемима, али се, уз адекватне модификације, могу користити за дизајнирање метахеуристичких метода за решавање сличних проблема,
- развијени софтвери се могу употребити у пракси у реалним транспортним системима.

Резултати приказани у овој докторској дисертацији представљају научни допринос областима комбинаторне оптимизације, математичког моделирања, метахеуристичких метода и операционих истраживања.

На основу добијених резултата може се закључити да предложени математички модели и методе имају велики потенцијал за решавање проблема распоређивања возила. Из тог разлога, један од праваца будућег истраживања је прилагођавање предложених математичких модела и имплементација метахеуристичких метода за решавање сличних проблема. Генерално, то није једноставан задатак јер чак и врло мале разлике у карактеристикама проблема могу довести до крупних промена у математичком моделу као и у имплементацији метахеуристичких метода.

Други правац будућег рада би се састојао у комбиновању предложених имплементација са егзактним методама у циљу добијања оптималних решења или поправљања горње границе на инстанцама великих димензија. Такође, хибридизација са другим метахеуристичким методама може водити ка бољим перформансама у погледу квалитета добијених решења, времена извршавања и стабилности.

У будућим истраживањима планирано је и проширење предложених математичких модела додавањем нових ограничења која се природно јављају у пракси, као што је организација транспорта са више депоа и коришћење возила која нису хомогена, одосно немају исти капацитет и техничке карактеристике.

За те сложеније проблеме изазов је развити метахеуристичке методе решавања полазећи од постојећих имплементација које су предложене у овој докторској дисертацији.

Библиографија

[1] H. Abedinnia, C. H. Glock, and M.D. Schneider. Machine scheduling in production: a content analysis. *Applied Mathematical Modelling*, 50:279–299, 2017.

[2] D.W.I. Agustina, C.K.M. Lee, and R. Piplani. Vehicle scheduling and routing at a cross docking center for food supply chains. *International Journal of Production Economics*, 152:29–41, 2014.

[3] A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345 – 378, 2015.

[4] K.N. Androutsopoulos and K.G. Zografos. An integrated modelling approach for the bicriterion vehicle routing and scheduling problem with environmental considerations. *Transportation Research Part C: Emerging Technologies*, 82:180–209, 2017.

[5] A. Anokić. Variable neighborhood search for optimizing the transportation of agricultural raw materials. *The IPSI BgD Transactions In Internet Research*, 13(1):28–37, 2017.

[6] A. Anokić, Z. Stamirović, T. Davidović, and Dj. Stakić. Variable Neighborhood Search for Vehicle Scheduling Problem Considering the Transport of Agricultural Raw Materials. *Electronic Notes in Discrete Mathematics*, 58:137–142, 2017.

[7] A. Anokić, Z. Stamirović, T. Davidović, and Dj. Stakić. Variable neighborhood search based approaches to a vehicle scheduling problem in agriculture. *International Transactions in Operational Research*, accepted for publication, 2017.

[8] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2011.

[9] J.B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society*, 49(7):700–708, 1998.

[10] P. Augerat, J.M. Belenguer, E. Benavent, A. Corbéran, and D. Naddef. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, 106(2-3):546–557, 1998.

[11] F. Baita, R. Pesenti, W. Ukovich, and D. Favaretto. A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Computers & Operations Research*, 27(13):1249–1269, 2000.

[12] M. Battarra, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11):3041–3050, 2009.

[13] L. Bodin and B. Golden. Classification in vehicle routing and scheduling. *Networks*, 11(2):97–108, 1981.

[14] A. Bortfeldt, T. Hahn, D. Männel, and L. Mönch. Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. *European Journal of Operational Research*, 243(1):82–96, 2015.

[15] G.A. Bula, C. Prodhon, F. A. Gonzalez, H.M. Afsar, and N. Velasco. Variable neighborhood search to solve the vehicle routing problem for hazardous materials transportation. *Journal of Hazardous Materials*, 324(part B):472–480, 2016.

[16] S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

[17] C. Carreto and B. Baker. A GRASP interactive approach to the vehicle routing problem with backhauls. In C.C. Ribeiro and P. Hasen, editors, *Essays and Surveys in Metaheuristics*, pages 185–199. Springer, US, 2002.

[18] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the Multi Trip Vehicle Routing Problem. *European Journal of Operational Research*, 236(3):833–848, 2014.

[19] I. Charon and O. Hudry. The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14(3):133–137, 1993.

[20] I. Charon and O. Hudry. The noising methods: A generalization of some metaheuristics. *European Journal of Operational Research*, 135(1):86–101, 2001.

[21] M. Cheikh, M. Ratli, O. Mkaouar, and B. Jarboui. A variable neighborhood search algorithm for the vehicle routing problem with multiple trips. *Electronic Notes in Discrete Mathematics*, 47:277–284, 2015.

[22] E.K.P. Chong and Žak S.H. *An Introduction to Optimization*. John Wiley & Sons, 2013.

[23] Jr E.G. Coffman. Computer and Job-shop scheduling theory, John Wiley&Sons. *Inc. New York*, 1976.

[24] J.F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle Routing. *Handbooks in operations research and management science*, 14:367–428, 2007.

[25] R. W. Cottle. George B. Dantzig: Operations Research Icon. *Operations Research*, 53(6):892–898, 2005.

[26] G. B. Dantzig. Maximization of a Linear Function of Variables Subject to Linear Inequalities, In: T. C. Koopmans, Ed., Activity Analysis of Production and Allocation. 1951.

[27] G.B. Dantzig and D.R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, 1(3):217–222, 1954.

[28] T. Davidović, D. Teodorović, and M. Šelmić. Bee colony optimization Part I: The algorithm overview. *Yugoslav Journal of Operations Research*, 25(1), 2015.

[29] J. de Armas and B. Melián-Batista. Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. *Computers and Industrial Engineering*, 85:120–131, 2015.

[30] C. Duhamel, P. Lacomme, C. Prins, and C. Prodhon. A GRASP× ELS approach for the capacitated location-routing problem. *Computers and Operations Research*, 37(11):1912–1923, 2010.

[31] C. Duhamel, P. Lacomme, A. Quilliot, and H. Toussaint. A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. *Computers & Operations Research*, 38(3):617–640, 2011.

[32] S. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally diffcult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

[33] S. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[34] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP-Part I: Algorithms. *International Transactions in Operational Research*, 16(1):1–24, 2009.

[35] I. Fister, X. Yang, and J. Brest. A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, 13:34–46, 2013.

[36] B. Fleischmann. The vehicle routing problem with multiple use of vehicles. *Fachbereich Wirtschaftswissenschaften, Universität Hamburg*, Working paper, 1990.

[37] K. Fleszar, I.H. Osman, and K. S. Hindi. A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809, 2009.

[38] M. Fliedner, D. Briskorn, and N. Boysen. Vehicle scheduling under the warehouse-on-wheels policy. *Discrete Applied Mathematics*, 205:52–61, 2016.

[39] S. French. *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Ellis Horwood Chichester, 1982.

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

[41] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[42] F. Glover. Tabu search and adaptive memory programming-advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*, pages 1–75. Springer, 1997.

[43] F.W. Glover and G.A. Kochenberger. *Handbook of Metaheuristics*. Springer Science & Business Media, 2006.

[44] M. Grötschel and M.W. Lawler Padberg. *Polyhedral theory. In E.L. and Lenstra, J.K. and Rinnooy Kan, A.H.G and Schmoys, D.B. (eds.) The traveling salesman problem*. New York: Wiley, 1985.

[45] A. Haghani, M. Banihashemi, and K.H. Chiang. A comparative analysis of bus transit vehicle scheduling models. *Transportation Research Part B: Methodological*, 37(4):301–322, 2003.

[46] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[47] P. Hansen and N. Mladenović. Variable neighborhood search. In E. K. Burke and R. D. Graham, editors, *Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 313–337. Springer-Verlag, New York, 2014.

[48] P. Hansen, N. Mladenović, and J. A.M. Pérez. Variable neighborhood search. *European Journal of Operational Research*, 191(3):593–595, 2008.

[49] P. Hansen, N. Mladenović, and J. A.M. Pérez. Variable neighbourhood search: methods and applications. *Ann. Oper. Res.*, 175(1):367–407, 2010.

[50] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

[51] A. Higgins. Scheduling of road vehicles in sugarcane transport: A case study at an Australian sugar mill. *European Journal of Operational Research*, 170(3):987–1000, 2006.

[52] S.C. Ho and W.Y. Szeto. GRASP with path relinking for the selective pickup and delivery problem. *Expert Systems with Applications*, 51:14–25, 2016.

[53] J.H. Holland. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, 1975.

[54] Lindo Systems Inc. `http://www.lindo.com`, 2016.

[55] J. Kennedy and R. Eberhard. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ*, pages 1942–1948, 1995.

[56] M.R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E.G. Talbi. A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests. *Applied Soft Computing*, 12(4):1426–1439, 2012.

[57] S. Kirkpatrick, C.D. Gelatt, and M.P Vecchi. Optimization by simulated annealing. *Science, New Series*, 220(4598):671–680, 1983.

[58] G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23, 1995.

[59] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757, 2007.

[60] R. Lahyani, L.C. Coelho, M. Khemakhem, G. Laporte, and F. Semet. A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega*, 51:1–10, 2015.

[61] J.K. Lenstra and A.H.G. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[62] S.C. Liu and A.Z. Chen. Variable neighborhood search for the inventory routing and scheduling problem in a supply chain. *Expert Systems with Applications*, 39(4):4149–4159, 2012.

[63] P. Lučić and D. Teodorović. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *The Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis,University of Azores,Sao Miguel, Azores Islands, Portugal*, pages 441–445, 2001.

[64] R. Macedo, C. Alves, S. Hanafi, B. Jarboui, N. Mladenović, B. Ramos, and J.M.V. de Carvalho. Skewed general variable neighborhood search for the location routing scheduling problem. *Computers & Operations Research*, 61:143–152, 2015.

[65] S. Madankumar and C. Rajendran. Mathematical models for green vehicle routing problems with pickup and delivery: A case of semiconductor supply chain. *Computers & Operations Research*, 89:183–192, 2016.

[66] J. Mańdziuk and M. Świechowski. UCT in Capacitated Vehicle Routing Problem with traffic jams. *Information Sciences*, 406-407:42–56, 2017.

[67] Y. Marinakis. Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807–6815, 2012.

[68] L. Martínez and C. A. Amaya. A vehicle routing problem with multi-trips and time windows for circular items. *Journal of the Operational Research Society*, 64(11):1630–1643, 2013.

[69] E.L. Milan, S.M. Fernandez, and L.M.P. Aragones. Sugar cane transportation in Cuba, a case study. *European Journal of Operational Research*, 174(1):374–386, 2006.

[70] ILOG CPLEX Optimization Studio 12.6.3: MIQCP mixed integer programs with quadratic terms in the constraints. IBM Knowledge Center. `http://www.ibm.com`, 2016.

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

[72] S. Moons, K. Ramaekers, and A.A.Y. Caris. Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Computers and Industrial Engineering*, 104:224–245, 2017.

[73] A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47, 2007.

[74] C. Park and J. Seo. A GRASP approach to transporter scheduling and routing at a shipyard. *Computers & Industrial Engineering*, 63(2):390–399, 2012.

[75] M. Pinedo. *Scheduling Theory, Algorithms and Systems*. 4th edition, Springer, 2012.

[76] M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000.

[77] C. Prins. A GRASP× evolutionary local search hybrid for the vehicle routing problem. In B.P. Francisco and Jorge T., editors, *Bio-inspired algorithms for the vehicle routing problem*, pages 35–53. Springer, 2009.

[78] S. Raff. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211, 1983.

[79] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau and Potvin J.Y., editors, *Handbook of metaheuristics*, pages 283–319. Springer, 2010.

[80] M.G.C. Resende and C.C. Ribeiro. GRASP: Greedy randomized adaptive search procedures. In E.K. Burke and G. Kendall, editors, *Search Methodologies - Introductory tutorials in optimization and decision support systems*, pages 287–312. Springer, 2014.

[81] R.A. Sarker and C.S. Newton. *Optimization Modelling: A Practical Approach*. Taylor & Francis Group LLC, 2008.

[82] K. Sethanan and R. Pitakaso. Differential evolution algorithms for scheduling raw milk transportation. *Computers and Electronics in Agriculture*, 121:245–259, 2016.

[83] Zorica Stanimirović. *Genetski algoritam za rešavanje nekih NP-teških hab lokacijskih problema*. PhD thesis, Matematički fakultet, Univerzitet u Beogradu, 2007.

[84] C. Sur and A. Shukla. New bio-inspired meta-heuristics-green herons optimization algorithm-for optimization of travelling salesman problem and road network. In *Proceedings of International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 168–179. Springer, 2013.

[85] E.G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[86] D. Teodorović and M. Dell'Orco. Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Proceedings of the 10th EWGT Meeting, Poznan, 13-16, September 2005*, 2005.

[87] S. Thuankaewsing, S. Khamjan, K. Piewthongngam, and S. Pathumnakul. Harvest scheduling algorithm to equalize supplier benefits: A case study from the Thai sugar cane industry. *Computers and Electronics in Agriculture*, 110:42–55, 2015.

[88] G.G. Wang, S. Deb, and L.S. Coelho. Elephant herding optimization. In *Proceedings of Computational and Business Intelligence (ISCBI), 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pages 1–5. IEEE, 2015.

[89] H. Wang and J. Shen. Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints. *Applied Mathematics and Computation*, 190(2):1237–1249, 2007.

[90] N. Wassan, N. Wassan, G. Nagy, and S. Salhi. The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and a Two-Level Variable Neighbourhood Search. *Computers & Operations Research*, 78:454–467, 2017.

[91] Y. Xiao and A. Konak. A simulating annealing algorithm to solve the green vehicle routing & scheduling problem with hierarchical objectives and weighted tardiness. *Applied Soft Computing*, 34:372–388, 2015.

[92] D. Xu, K. Li, X. Zou, and L. Liu. An unpaired pickup and delivery vehicle routing problem with multi-visit. *Transportation Research Part E: Logistics and Transportation Review*, 103:218–247, 2017.

[93] X.S. Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2(2):78–84, 2010.

[94] X.S. Yang. A new metaheuristic bat-inspired algorithm. In *Proceedings of Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[95] X.S. Yang and S. Deb. Cuckoo search via lévy flights. In *Proceedings of Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress*

on Nature & Biologically Inspired Computing (NaBIC), pages 210–214. IEEE, 2009.

[96] M.X. Zhang, B. Zhang, and Y.J. Zheng. Bio-inspired meta-heuristics for emergency transportation problems. *Algorithms*, 7(1):15–31, 2014.

Биографија аутора

Ана Анокић (Адамовић) је рођена 27.05.1978. године у Београду, где је завршила основну школу као ђак генерације и Математичку гимназију са одличним успехом.

Математички факултет у Београду, смер Нумеричка математика и оптимизација, уписала је 1997. године. Дипломирала је 29.08.2003. године, са посечном оценом 9,19. Исте године је уписала магистарске студије на Математичком факултету у Београду, смер Нумеричка математика и оптимизација, а потом и докторске студије по статуту из 2006. године. Од 2011. је студент докторских студија студијског програма Математика по акредитацији из 2009. године.

Запослена је на Пољопривредном факултету Универзитета у Београду где је два пута бирана у звање асистента приправника у периоду од 2004. до 2008. године и два пута у звање асистента у периоду од 2008. до 2015. године, на Катедри за статистику Института за агроекономију. Тренутно се налази на радном месту сарадника без сарадничког звања у истој институцији. Током овог периода, од 2004. године до данас, држала је рачунске вежбе из предмета: Операциона истраживања и Статистика, на основним студијама и Математичко-статистичке методе 1, на мастер студијама.

Прилог 1.

Изјава о ауторству

Потписани-а Ана Анокић

број индекса 2045/2011

Изјављујем

да је докторска дисертација под насловом

Математички модели и методе решавања новог проблема распоређивања возила при оптимизацији транспорта пољопривредних сировина

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

Потпис докторанда

У Београду, 12.10.2017.

Ана Анокић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора АНА АНОКИЋ

Број индекса 2045/2011

Студијски програм МАТЕМАТИКА

Наслов рада МАТЕМАТИЧКИ МОДЕЛИ И МЕТОДЕ РЕШАВАЊА НОВОГ ПРОБЛЕМА РАСПОРЕЂИВАЊА ВОЗИЛА ПРИ ОПТИМИЗАЦИЈИ ТРАНСПОРТА ПОЉОПРИВРЕДНИХ СИРОВИНА

Ментор проф. др. Зорица Станимировић

Потписани/а Ана Анокић

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

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

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

Потпис докторанда

У Београду, 12.10.2017.

Ана Анокић

Прилог 3.

Изјава о коришћењу

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

МАТЕМАТИЧКИ МОДЕЛИ И МЕТОДЕ РЕШАВАЊА НОВОГ ПРОБЛЕМА РАСПОРЕЂИВАЊА ВОЗИЛА ПРИ ОПТИМИЗАЦИЈИ ТРАНСПОРТА ПОЉОПРИВРЕДНИХ СИРОВИНА

која је моје ауторско дело.

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

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

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

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

Потпис докторанда

У Београду, 12.10.2017.

Ана Анђић