UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Nina Radojičić

PRIMENA FAZI LOGIKE ZA REŠAVANJE NP-TEŠKIH PROBLEMA RUTIRANJA VOZILA I LOKACIJE RESURSA METODAMA RAČUNARSKE INTELIGENCIJE

doktorska disertacija

Beograd, 2018.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Nina Radojičić

APPLICATION OF COMPUTATIONAL INTELLIGENCE METHODS FOR SOLVING NP-HARD VEHICLE ROUTING AND RESOURCE LOCATION PROBLEMS - A FUZZY LOGIC APPROACH

Doctoral Dissertation

Belgrade, 2018.

Mentor:
dr Miroslav Marić, vanredni profesor

Matematički fakultet, Univerzitet u Beogradu

Članovi komisije:

dr Gordana Pavlović-Lažetić, redovni profesor
Matematički fakultet, Univerzitet u Beogradu

dr Miroslav Marić, vanredni profesor
Matematički fakultet, Univerzitet u Beogradu

dr Filip Marić, vanredni profesor
Matematički fakultet, Univerzitet u Beogradu

dr Zorica Stanimirović, vanredni profesor
Matematički fakultet, Univerzitet u Beogradu

dr Aleksandar Takači, redovni profesor
Tehnološki fakultet, Univerzitet u Novom Sadu

Datum odbrane: _____

Naslov disertacije: Primena fazi logike za rešavanje NP-teških problema rutiranja vozila i lokacije resursa metodama računarske inteligencije

Rezime:

U okviru ove disertacije, tri NP-teška problema su razmatrana i rešavana različitim metodama računarske inteligencije, sa posebnim akcentom na mogućnosti upotrebe fazi logike za poboljšanje performansi metoda. Pored toga prikazana je i mogućnost korišćenja fazi logike u cilju kreiranja modela koji odgovaraju problemima u praksi.

Prvi problem razmatran u disertaciji je problem rutiranja vozila sa ograničenim rizikom (engl. Risk-constrained Cash-in-Transit Vehicle Routing Problem - RCTVRP). RCTVRP predstavlja specijalan slučaj problema rutiranja vozila (VRP). Kao i kod klasičnog problema VRP, potrebno je da vozila obiđu sve klijente tako da na kraju ukupan pređeni put (ili troškovi putovanja) bude što manji. Kod RCTVRP problema uzima se u obzir i sigurnosni aspekt ruta. Ovaj problem se pojavio u sektoru koji se bavi prenosom novca i robe od velike vrednosti.

Druga dva problema koja su predmet ove disertacije spadaju u grupu lokacijskih problema: problem ravnomernog opterećenja (engl. Load Balancing Problem - LOBA) i problem maksimizacije minimalnog rastojanja (engl. Max-Min Diversity Problem - MMDP). LOBA predstavlja diskretni lokacijski problem kod kojeg je potrebno odabrati snabdevače tako da budu ravnomerno opterećeni od strane pridruženih korisnika. Cilj je odrediti skup od određenog broja snabdevača iz skupa potencijalnih snabdevača koji će biti uspostavljeni tako da je minimizovana razlika između najvećeg i najmanjeg broja korisnika dodeljenih nekom uspostavljenom snabdevaču. Ovaj problem se često javlja u praksi, na primer, prilikom raspoređivanja antena za mobilne telefone, škola, izbornih jedinica ili centara za prikupljanje čvrstog otpada. Problem MMDP je diskretni lokacijski problem kod koga je cilj odabrati podskup od tačno određenog broja elemenata datog skupa tako da je najmanje rastojanje između odabranih elemenata maksimalno. MMDP potiče iz situacija iz prakse, na primer, kada je potrebno rasporediti postrojenja tako da ne postoje dva postrojenja koja su previše blizu. MMDP ima primene i u društvenim i biološkim naukama (na primer u cilju očuvanja ekologije).

U cilju rešavanja RCTVRP, GRASP (Greedy Randomized Adaptive Search Procedure) metod je hibridizovan sa metodom ponovnog povezivanja puta (engl. Path Reliking - PR). Pažljivo određena fazi modifikacija je implementirana u predloženi

GRASP metod u cilju poboljšanja performansi prilikom rešavanja RCTVRP problema. Dodatno, u ovoj disertaciji je predložena nova struktura za PR metod koja se može primeniti za rešavanje drugih problema rutiranja vozila. U cilju smanjenja vremenske složenosti predloženog algoritma, implementirana je nova struktura podataka za RCTVRP. Predloženi fazi GRASP algoritam hibridizovan sa PR metodom daje bolje rezultate u poređenju sa hibridnom verzijom bez fazi modifikacije. Pored toga, eksperimentalni rezultati na javno dostupnim test primerima ukazuju da predloženi metod nadmašuje sve metode koje su do sada primenjivane u dostupnoj literaturi na RCTVRP problem.

Za rešavanje problema LOBA predložena su dva hibridna pristupa: kombinacija redukovane i standardne metode promenljivih okolina (RVNS-VNS), kao i hibridizacija evolutivnog algoritma i VNS metode (EA-VNS). Predložene metode su testirane i poređenje na test primerima do 100 korisnika i potencijalnih snabdevača. U cilju testiranja predloženih metoda na test primerima većih dimenzija, predloženi su test primeri sa preko 400 korisnika i potencijalnih snabdevača. Eksperimentalni rezultati pokazuju da predloženi hibridni metodi brzo dostižu optimalna rešenja koja su poznata za test primere manjih dimenzija, i brzo daju rešenja za test primere velikih dimenzija. Na osnovu kvaliteta dobijenih rešenja kao i vremena dolaska do rešenja, EA-VNS pristup se najbolje pokazao na problemu LOBA.

Pored toga, implementiran je evolutivni algoritam (EA) za rešavanje problema MMDP. Eksperimentalni rezultati pokazuju da EA brzo dostiže sva optimalna rešenja na manjim test primerima. Međutim, za neke test primere većih dimenzija dolazi do preuranjene konvergencije ka lokalnom maksimumu. Iako su uspostavljeni parametri za EA koji imaju odlične rezultate za većinu problema i kvalitetne metaheuristike rade dobro bez obzira na odabir parametara, podešavanje parametara algoritma može uticati na dobijanje boljih rezultata za problem koji se rešava. Jedan od pristupa je da se parametri podešavaju u toku izvršavanja algoritma. Kao deo ove disertacije izložen je jedan pristup podešavanju parametara na primeru problema MMDP. Naime, EA je unapređen dodavanjem fazi pravila formulisanog na osnovu iskustva sa radom sa evolutivnim algoritmima. Implementirano fazi pravilo menja parametar mutacije u toku izvršavanja algoritma u cilju sprečavanja preuranjene konvergencije ka lokalnom maksimumu. Prikazani rezultati ukazuju da je predloženi fazi EA bolji od klasičnog EA za problem MMDP.

Za sva tri problema, u slučajevima test primera manjih dimenzija koje je CPLEX

rešavač mogao da reši, dobijena optimalna rešenja su se poklapala sa rešenjima koje su davale predložene metode.

Pored toga, u okviru ove disertacije prikazana je i mogućnost korišćenja fazi logike u cilju modeliranja problema na primeru jedne varijacije problema RCTVRP. Pored osnovnih limita rizika, cilj je u model dodati verovatnoću da se pljačka desi u okviru svake rute, umesto samo dozvoljavati rešenja čije rute ispunjavaju uslov rizika. Novi fazi model je predstavljen i obezbeđuje rešenja čije rute imaju manji nivo rizika, te se smatraju boljima. Pored toga, date su dve MIP formulacije i uvedeni model je upoređen sa osnovnim modelom na adekvatnom primeru. Pored toga eksperimentalni rezultati ukazuju da novi model daje bezbednije rute.

Ključne reči: računarska inteligencija, metaheuristike, fazi logika, NP-teški problemi, kombinatorna optimizacija

Naučna oblast: Računarstvo

Uža naučna oblast: Računarska inteligencija

Dissertation title: Application of Computational Intelligence Methods for Solving NP-Hard Vehicle Routing and Resource Location Problems - A Fuzzy Logic Approach

Abstract:

In this dissertation, three NP-hard optimization problems are studied and various computational intelligence methods are considered for solving them, with a special emphasis on the possibilities of applying fuzzy logic in order to improve the performances of proposed methods. In addition, it is shown how fuzzy logic can be incorporated into a model to make it more adequate to real world applications. The first problem considered is the Risk-Constrained Cash-in-Transit Vehicle Routing Problem (RCTVRP) that represents a special case of the vehicle routing problem (VRP). Similar to the classical VRP, the aim is to determine the collection routes from one depot to a number of customers in order to minimize the overall travel distance (or cost). Additionally, the safety aspect of the routed risk constraints are introduced in the case of RCTVRP. The RCTVRP concerns the issue of security during the transportation of cash or valuable goods (e.g. in the cash-in-transit industry).

The other two problems studied in this dissertation belong to the class of location problems: the Load Balancing Problem (LOBA) and the Max-Min Diversity Problem (MMDP). The goal of the LOBA problem is to locate a fixed number of facilities, such that the difference between the maximum and minimum number of customers served by each facility is balanced. The LOBA model is useful in cases where customers naturally choose the closest facility. The MMDP consists of selecting a subset of a fixed number of elements from a given set in such a way that the diversity among the selected elements is maximized. This problem also arises in real world situations encompassing a variety of fields, particularly the social and biological sciences.

In order to solve the RCTVRP, a fuzzy GRASP (Greedy Randomized Adaptive Search Procedure) is hybridized with Path Reliking (PR) methodology. Carefully adjusted fuzzy modification incorporated into the proposed GRASP for the RCTVRP improved its performance. Moreover, in this dissertation a new PR structure is implemented and can be used for other vehicle routing problems. To improve the algorithm's time complexity, a new data structure for the RCTVRP is incor-

porated. The proposed fuzzy GRASP with PR hybrid shows better computational performance compared to its non-fuzzy version. Furthermore, computational results on publicly available data sets indicate that the proposed algorithm outperforms all existing methods from the literature for solving the RCTVRP.

For solving the LOBA problem two efficient hybrid metaheuristic methods are proposed: a combination of reduced and standard variable neighborhood search methods (RVNS-VNS) and hybridization of evolutionary algorithm and VNS approach (EA-VNS). The proposed hybrid methods are first benchmarked and compared to all the other methods on existing test instances for the LOBA problem with up to 100 customers and potential suppliers. In order to test the effectiveness of the proposed methods, we modify several large-scale instances from the literature with up to 402 customers and potential suppliers. Exhaustive computational experiments show that the proposed hybrid methods quickly reach all known optimal solutions while providing solutions on large-scale problem instances in short CPU times. Regarding solution quality and running times, we conclude that the proposed EA-VNS approach outperforms other considered methods for solving the LOBA problem.

EA approach is also proposed for solving the MMDP. Computational experiments on a smaller benchmark data set showed that the classic EA quickly reached all optimal solutions obtained previously by an exact solver. However, some of the larger instances of MMDP were challenging for classic EA. Although researchers have established the most commonly used parameter setting for EA that has good performance for most of the problems, it is still challenging to choose the adequate values for the parameters of the algorithm. One approach to overcome this is changing parameter values during the algorithm run. As part of this dissertation this problem was addressed by extending the evolutionary algorithm by adding a fuzzy rule formulated from EA experts' knowledge and experience. The implemented fuzzy rule changes the mutation parameter during the algorithm run. The results on tested instances indicate that the proposed fuzzy approach is more suitable for solving the MMDP than classic EA.

For all three problems addressed whereas the smaller instances that CPLEX was able to solve, obtained optimal solutions were used for comparison with proposed methods and all of the proposed methods obtained these optimal solutions.

Moreover, in this dissertation it has been shown that fuzzy logic is a successful tool in modeling the RCTVRP. In this problem the risk constraints are set by

using a risk threshold T on each route and thus, the routes with risk larger than T are forbidden. However, in this dissertation the aim is to take into account the probability of being robbed along each route instead of just allowing solutions with routes that satisfy the risk constraints. A new fuzzy model for the RCTVRP is developed which takes into account the value of the risk index of each route and the solutions with lower values of risk indexes on their routes are considered superior. In order to achieve that fuzzy numbers are used in the improved model. Moreover, two mixed integer program formulations of new fuzzy model are developed and presented in this dissertation. The introduced fuzzy model is compared with the model from the literature using an adequate example and the advantages of the newly proposed fuzzy RCTVRP is demonstrated. Computational experiments are performed and the comparison of the two models given in the paper show that the newly presented approach leads to safer routes.

Keywords: computational intelligence, metaheuristics, fuzzy logic, NP-hard problems, combinatorial optimization

Research area: Computer Science

Research sub-area: Computational intelligence

Zahvalnica

Posebno bih želela da se zahvalim mentoru prof. dr Miroslavu Mariću na korisnim savetima i nesebičnom angažovanju tokom realizacije ove disertacije. Želim da istaknem zahvalnost prof. dr Gordani Pavlović-Lažetić i prof. dr Filipu Marić na korisnim sugestijama i stručnim savetima koji su doprineli da se kvalitet rada poboljša. Veliku zahvalnost dugujem prof. dr Zorici Stanimirović koja me je zainteresovala za oblast matematičke optimizacije i značajno pomagala i podržavala u realizaciji, kao i konačnom oblikovanju, ovog rada. Takođe, posebnu zahvalnost za korisne savete u vezi fazi logike dugujem prof. dr Aleksandru Takačiju.

Zahvaljujem se svojoj porodici i prijateljima koji su svojom podrškom i razumevanjem ulepšali moje profesionalno usavršavanje.

Sadržaj

1

Uvod

1.1 Matematička optimizacija

Problem *matematičke optimizacije* se može definisati kao pronalaženje $min\{f(x) : x \in X\}$ ili $max\{f(x) : x \in X\}$, gde je su dati prostor rešenja X (dopustiv skup) i realna funkcija $f : X \to \mathbb{R}$ koja se naziva funkcija cilja. Elementi skupa X se nazivaju dopustiva rešenja. Globalni optimum (optimalno rešenje) za problem minimizacije je rešenje $x^* \in X$ tako da $(\forall x \in X)(f(x^*) \leq f(x))$. Slično, u slučaju rešavanja problema maksimizacije, pod optimalnim rešenjem se podrazumeva $x^* \in X$ za koje važi $(\forall x \in X)(f(x^*) \geq f(x))$.

Dopustiv skup X je zadat nizom ograničenja i najčešće je neki podskup od $\mathbb{R}^n$, pa je $x \in X$ zapravo vektor $x = (x_1, x_2, \ldots, x_n)$. Ako je X konačan ili prebrojivo beskonačan, diskretan skup, onda je reč o kombinatornoj optimizaciji. Ukoliko su funkcija cilja f, kao i jednakosti/nejednakosti koje predstavljaju ograničenja linearni, u pitanju je linearno programiranje. Često je postavljeno dodatno ograničenje da neke od koordinata x_i nepoznatog vektora x moraju biti celobrojne i tada je reč o mešovitom celobrojnom linearnom programiranju (engl. *mixed integer linear programming* - MILP, MIP). Specijalno, kod problema celobrojnog linearnog programiranja (engl. *integer linear programming* - ILP, IP) zahteva se celobrojnost svih koordinata vektora x, dok su u slučaju 0-1 linearnog programiranja sve koordinate binarne. Međutim, prilikom kreiranja matematičkih modela za realne probleme, funkcija f može biti i nelinearna. Optimizacioni problem kod koga je funkcija f kvadratna, a ograničenja linearna se naziva problem kvadratnog programiranja (engl.

quadratic programming - QP). Sa druge strane, problemi se nazivaju kvadratno uslovljenima ukoliko su ograničenja kvadratna.

Razvijanje i upotreba optimizacionih metoda ima veliki značaj za uspešno funkcionisanje kompleksnih sistema, kao što su na primer lanci snabdevanja, društvene, telekomunikacione i transportne mreže, itd. Razvoj teorije matematičke optimizacije započeo je i pre postojanja računara, ali njena ekspanzija i rešavanje sve većeg broja problema iz prakse postali su mogući usled napretka računarstva.

O značaju i širokoj primeni optimizacije govorio je, još 1744. godine, jedan od najpoznatijih matematičara svih vremena Ojler (*Leonhard Euler*), koji je izjavio:

„Cum enim mundi universi fabrica sit perfectissima atque a Creatore sapientissimo absoluta, nihil omnino in mundo contingit, in quo non maximi minimive ratio quaepiam eluceat; quamobrem dubium prorsus est nullum, quin omnes mundi effectus ex causis finalibus ope methodi maximorum et minimorum aeque feliciter determinari queant, atque ex ipsis causis efficientibus."

Ovaj citat se ukratko može prevesti kao: Ništa na ovom svetu se ne dešava bez optimizacije i sve sto je racionalno na ovom svetu se može opisati pomoću optimizacije.

Cilj optimizacije nije samo dobro kreiranje i razumevanje modela, već istraživanja u ovoj oblasti treba da daju i algoritme koji efikasno rešavaju probleme koji se razmatraju. Zbog toga je optimizacija neraskidivo vezana za računarske nauke.

Kombinatorna optimizacija

U slučaju problema *kombinatorne optimizacije* potrebno je pronaći najbolji izbor u konačnom (ali obično veoma velikom) ili prebrojivo beskonačnom skupu mogućnosti. Veliki broj raznovrsnih problema iz prakse pripada ovoj klasi problema i stoga je proučavanje rešavanja ovakvih problema od velikog značaja. Međutim, uslov da neke ili sve promenljive nepoznatog vektora u problemima matematičke optimizacije moraju biti celobrojne najčešće dovodi do toga da je pronalaženje optimalnih rešenja teže. Naime, zbog diskretnosti dopustivog skupa X ne mogu se primeniti numeričke metode zasnovane na klasičnoj matematičkoj analizi. Iako u slučaju konačnog skupa X uvek postoje algoritmi potpune pretrage, potpunu pretragu često nije pogodno primeniti ukoliko je kardinalnost skupa X velika.

Primer 1.1. Problem nalaženja najkraćih puteva u grafu (engl. *shortest path problem*): *Pretpostavimo da se određen broj gradova nalazi u nekoj regiji i da želimo da putujemo iz grada s do grada t. Pri tome, poznata su rastojanja između svaka dva grada. Možemo direktno ići iz grada s do grada t ukoliko postoji put koji direktno spaja ta dva grada ili da se krene iz s, prođe kroz jedan ili više gradova i stigne u t. Dakle, putanja od s do t se definiše kao niz dva ili više gradova, tako da je prvi s, a poslednji t. Dužina putanje se definiše kao suma rastojanja između svaka dva uzastopna grada iz putanje. Cilj je među svim putanja od s do t, pronaći onu sa najmanjom dužinom.*

Problem najkraćih puteva se zadaje na usmerenom grafu $G = (V, E)$, gde je V skup čvorova, a E skup grana. Svaki grad odgovara jednom čvoru iz V, a s i t su dva izdvojena čvora. Za svaki par gradova $i, j \in V$ koji su direktno povezani poznato je rastojanje d_{ij} za granu $(i, j) \in E$. Cilj je pronaći niz čvorova $i_1, i_2, \ldots, i_n$, gde je $i_1 = s$, a $i_n = t$, tako da je suma $\sum_{l=1}^{n-1} d_{i_l i_{l+1}}$ minimalna.

Primer 1.2. Problem ranca (engl. *knapsack problem*): *Pretpostavimo da je potrebno u ranac određenog kapaciteta spakovati predmete koji su različitih veličina i vrednosti. Potrebno je odrediti koje predmete uzeti kako bi ranac ima maksimalnu vrednost, a da predmeti mogu da se smeste u ranac.*

Za problem ranca zadaje se prirodan broj K i n predmeta različitih veličina i vrednosti, tako da i-ti predmet ima veličinu $k_i,\ 1 \leq i \leq n$ i vrednost $v_i,\ 1 \leq i \leq n$. Potrebno je pronaći podskup predmeta tako da suma vrednosti odabranih predmeta bude maksimalna, a da suma veličina tih predmeta ne pređe kapacitet ranca K. Dakle, problem se može zapisati na sledeći način:

$$min \sum_{i=1}^{n} v_i x_i,$$

uz uslove:

$$\sum_{i=1}^{n} k_i \leq K,$$

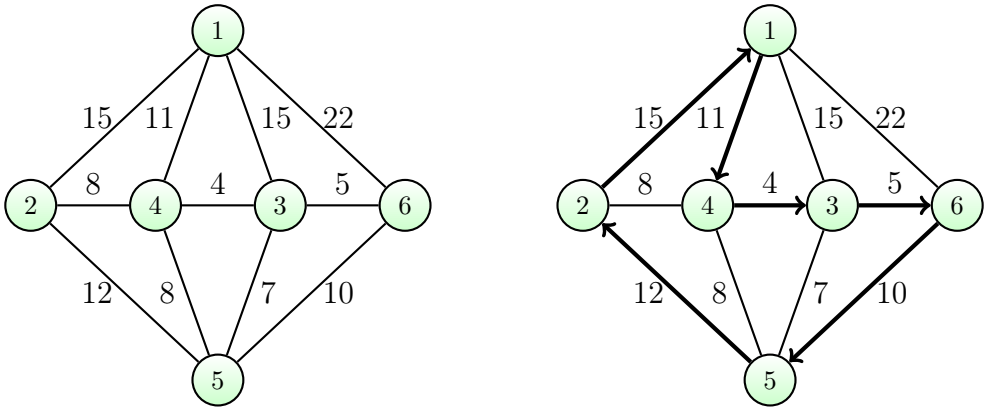
$$x_i \in \{0, 1\}, \forall i \in \{1, ..., n\}.$$

Primer 1.3. Problem trgovačkog putnika (engl. *Traveling Salesman Problem* - TSP): *Trgovački putnik treba da obiđe sve gradove na datoj teritoriji, počevši od*

datog grada i treba da poseti svaki grad tačno jednom. Stoga, ruta može biti predstavljena kao permutacija gradova. Pretpostavka je da su rastojanja između svakog para gradova poznata unapred i cilj je da se odredi permutacija gradova tako da je ukupan pređeni put minimizovan.

Neka je $V = \{1, 2, \ldots, n\}$ skup gradova koje je potrebno obići i zadat je težinski graf $G = (V, E)$ sa nenegativnim težinama d_{ij} dodeljenih svakoj grani $(i, j) \in E$. Tada svaka ruta u kojoj se svaki od n gradova posećuje tačno jednom odgovara Hamiltonovom ciklusu u G. Hamiltonov ciklus (engl. *Hamiltonian cycle*) je ciklus koji sadrži svaki čvor iz grafa tačno jednom. Dakle, dopustivo rešenje za TSP je ruta definisana permutacijom $\pi = (i_1, i_2, \ldots, i_n)$ n gradova, tako da je $i_j \neq i_k$ za svake $j \neq k \in V$. Ova permutacija odgovara Hamiltonovom ciklusu $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n), (i_n, i_1)$, a ukupna dužina rute je $\sum_{k=1}^{n-1} d_{i_k i_{k+1}} + d_{i_n i_1}$.

Razmotrimo jedan test primer (instancu) problema TSP. Na slici 1.1 je dat neusmeren graf sa šest čvorova, a težine grana predstavljaju rastojanja među čvorovima. Jedno dopustivo rešenje prikazane instance je na primer ciklus $(1, 4, 2, 5, 3, 6, 1)$, a ukupna dužina odgovarajuće putanje je 65. Sa desne strane slike 1.1 prikazana je najkraća putanja koja odgovara ciklusu $(1, 4, 3, 6, 5, 2, 1)$, čija ukupna dužina iznosi 57.



Slika 1.1: *Jedna instanca problema TSP (sa leve strane) i optimalno rešenje (sa desne strane)*

Termini test primer i instanca se mogu ravnopravno koristiti. Nije pojednako lako rešiti sve instance jednog problema, a često je težina rešavanja proporcionalna veličini instance koja se rešava. U prethodnom primeru, može se reći da je dimenzija instance jednaka broju čvorova u grafu tj. 6.

Navedeni problem TSP je jedan od najproučavanijih problema kombinatorne optimizacije. Ovaj problem je prvi put matematički formulisao irski matematičar Hamilton (*William Rowan Hamilton*) još u 19. veku. TSP ima široku primenu u mnogim oblastima kao što je raspoređivanje poslova, proizvodnja mikročipova, sekvenciranje DNK, itd. Više o problemu trgovačkog putnika i varijacijama tog problema može se pronaći u literaturi [91].

U okviru disertacije su razmatrane dve važne klase problema kombinatorne optimizacije. Uopštenje problema TSP predstavljaju problemi rutiranja vozila koji su opisani u okviru ove disertacije u poglavlju 2. Pored toga, u poglavlju 3 su predstavljeni problemi lokacije resursa.

1.2 Potreba za heuristikama

Za odabir i razvoj odgovarajućih, efikasnih algoritama za rešavanje pojedinačnih problema, potrebno je razumevanje složenosti tih problema. Primenom egzaktnog algoritma na optimizacioni problem dobija se globalni optimum, ukoliko postoji ili se ispostavi da je skup dopustivih rešenja prazan. Međutim, praktična primena ovakvih algoritama nekada nije moguća zbog limita računarskih resursa (vremenskih ili memorijskih) i složenosti problema matematičke optimizacije koji se rešavaju.

1.3 Klase složenosti algoritama

U ovoj sekciji je izložen kratak pregled teorije složenosti algoritama i objašnjenje klase NP-teških problema. Detaljnije o ovim temama se može pronaći u različitoj literaturi o algoritmima [195] i [26].

Vremenska složenost algoritma

Vremenska složenost algoritma $\mathcal{A}$ je funkcija $t(n)$ kojom se meri koliko je potrebno procesorskog vremena za izvršavanje algoritma, u zavisnosti od dimenzije problema n. Pri tome se podrazumeva da je $t(n)$ je maksimalno vreme koje je potrebno algoritmu za rešavanje bilo kog test primera dimenzije n. Zapravo, jedan od najčešćih razloga da neki algoritam nije praktično primenljiv u praksi je da vreme izvršavanja preterano brzo raste sa porastom veličine test primera koji se rešava.

Prilikom određivanja klase složenosti algoritma, najčešće se ne koristi tačan zapis funkcije $t(n)$, već odgovarajuća funkcija $g(n)$ jednostavnijeg zapisa koja ograničava odozgo funkciju $t(n)$ i simbol veliko O.

Definicija 1.1. *Neka su f i g dve pozitivne funkcije od argumenta n iz skupa $\mathcal{N}$ prirodnih brojeva. Kaže se da je $f(n) = O(g(n))$ ako postoje pozitivne konstante c i N_0 takve da za svako $n > N_0$ važi: $f(n) \leq c \cdot g(n)$.*

Dakle, kažemo da se algoritam izvršava u vremenu $O(g(n))$ ukoliko postoji ceo broj N_0 i realan broj c tako je za svako $n \geq N_0$, kada se algoritam $\mathcal{A}$ primeni na bilo koji test primer veličine n, vreme izvršavanja algoritma manje ili jednako $c \cdot g(n)$.

Algoritam je smatra efikasnim i praktično primenljivim za rešavanje problema $\mathcal{Q}$ ako je njegova vremenska složenost $O(n^k)$, za neki prirodan broj k koji ne zavisi od ulaznih podataka, dok n predstavlja veličinu ulaznih podataka. Za takav algoritam se kaže da se izvršava u polinomijalnom ili polinomskom vremenu (engl. *polynomial-time*), a tada za problem $\mathcal{Q}$ kaže da je *rešiv*. Uprkos tome što se u ovu definiciju uklapa i algoritam složenost na primer $O(n^{100})$, većina rešivih problema ima algoritme čija se vremenska složenost izražava polinomima malog stepena (često kvadratnog ili kubnog).

Definicija 1.2 (Polinomijalni algoritmi)**.** *Algoritam $\mathcal{A}$ koji rešava problem $\mathcal{Q}$ kažemo da je polinomijalan ukoliko postoji polinomska funkcija g tako da $\mathcal{A}$ rešava bilo koju instancu veličine n problema $\mathcal{Q}$ u vremenu $O(g(n))$.*

Postoje problemi za koje se ne zna ni jedan algoritam polinomijalne složenosti. U nastavku su ukratko navedene formalne definicije bazirane na problemima odlučivanja.

Problem odlučivanja

Pod *problemima odlučivanja* (engl. *decision problem*) podrazumevaju se problemi kod kojih se kao rešenje očekuje odgovor „DA" ili „NE". U slučaju problema odlučivanja može se postaviti pitanje „Da li postoji rešenje koje zadovoljava date uslove?". Sa druge strane, optimizacioni problem se odnosi na zadatak da se među svim rešenjima koji zadovoljavaju date uslove pronađe ono rešenje koje minimizuje/maksimizuje funkciju cilja tj. pronalazi dopustivo optimalno rešenje.

Primer 1.4. *Dat je težinski graf $G = (V, E)$ sa nenegativnim težinama grana koje predstavljaju rastojanja. Za TSP, problem odlučivanja bi glasio: „Dat je ceo broj L. Da li postoji Hamiltonov ciklus čija je dužina manja ili jednaka L?", dok je odgovarajući optimizacioni problem pronaći Hamiltonov ciklus najmanje dužine.*

Za problem odlučivanja se kaže da pripada klasi složenosti P ako postoji deterministički algoritam koji ga rešava u polinomijalnom vremenu. Klasa NP sadrži sve problema odlučivanja kod kojih se za jedno potencijalno rešenje može za polinomijalno vreme proveriti da li jeste zaista rešenje tj. da li je odgovor „DA". Dakle, osnovni smisao klase NP je proverljivost za polinomijalno vreme bilo kog zadatog rešenja, a ne zahteva se postojanje algoritma za brzi pronalazak svih rešenja. Klasa NP bi neformalno bila klasa svih problema odlučivanja koji mogu biti rešeni nedeterminističkim algoritmom za polinomijalno vreme.

Slede definicije koje su potrebne u cilju definisanja NP-kompletnih problema, važne potklase problema iz klase NP.

Redukcije polinomijalne vremenske složenosti

Redukcija je svođenje jednog problema $\mathcal{R}$ na drugi problem $\mathcal{Q}$, tako što jedan problem rešavamo korišćenjem crne kutije koja rešava drugi problem. Redukcijom je moguće postići:

- Ukoliko je poznat algoritam za rešavanje problema $\mathcal{Q}$, rešenje problema $\mathcal{R}$ se može dobiti transformacijom u problem $\mathcal{Q}$ i korišćenjem algoritma za rešavanje problema $\mathcal{Q}$.
- Ako se zna da je $\mathcal{R}$ težak problem i zna se donja granica za algoritme koji rešavaju problem $\mathcal{R}$, onda je to istovremeno i donja granica složenosti za problem $\mathcal{Q}$.

Pretpostavimo da su u nastavku $\mathcal{P}_1$ i $\mathcal{P}_2$ dva problema odlučivanja.

Definicija 1.3 (Redukcija u polinomijalanom vremenu)**.** *Kažemo da postoji redukcija u polinomijalanom vremenu problema $\mathcal{P}_1$ na problem $\mathcal{P}_2$ ako i samo ako se problem $\mathcal{P}_1$ može rešiti pomoću algoritma $\mathcal{A}_1$ koji se svodi na polinomijalan broj poziva algoritma $\mathcal{A}_2$ koji rešava problem $\mathcal{P}_2$, uz dodatno još najviše polinomijalano mnogo operacija.*

Definicija 1.4 (Transformacija u polinomijalanom vremenu)**.** *Kažemo da postoji transformacija problema $\mathcal{P}_1$ na problem $\mathcal{P}_2$ u polinomijalanom vremenu, ako se u polinomijalanom vremenu od instance $\mathcal{I}_1$ problema $\mathcal{P}_1$ može konstruisati instanca $\mathcal{I}_2$ problema $\mathcal{P}_2$, tako da je za instancu $\mathcal{I}_1$ odgovor „DA" za problem $\mathcal{P}_1$ ako i samo ako je za instancu $\mathcal{I}_2$ odgovor „DA" za problem $\mathcal{P}_2$.*

Transformacija u polinomijalanom vremenu se može posmatrati kao redukcija u polinomijalanom vremenu koja čini jedan poziv algoritma $\mathcal{A}_2$, nakon što je konstruisana instanca $\mathcal{I}_2$. Koristeći ovu definiciju, može se uvesti važna potklasa klase NP.

NP-teški i NP-kompletni problemi

Definicija 1.5 (NP-kompletni problemi)**.** *Za problem odlučivanja $\mathcal{Q} \in NP$ kaže se da je NP-kompletan ako se svaki drugi problem iz NP može transformisati u njega u polinomijalnom vremenu.*

Dakle, NP-kompletni problemi imaju važno svojstvo da ukoliko postoji polinomijalan algoritam za neki od njih, onda postoji polinomijalan algoritam za svaki od njih. Ukoliko se iz definicije NP-kompletnih problema izostavi uslov da problem $\mathcal{Q}$ mora pripadati klasi NP, dobija se definicija NP-teških problema.

Definicija 1.6 (NP-teški problemi)**.** *Kažemo da je problem $\mathcal{Q}$ NP-težak ako je svaki problem iz klase NP polinomijalno svodljiv na $\mathcal{Q}$.*

Koristeći sada definiciju NP-teških problema, NP-kompletni problemi se alternativno mogu definisati na sledeći način.

Definicija 1.7. *Kažemo da je problem $\mathcal{Q}$ NP-kompletan ako važi:*

- *$\mathcal{Q}$ pripada klasi NP i*
- *$\mathcal{Q}$ je NP-težak.*

Dokaz da je problem NP-kompletan se sastoji od dva glavna koraka koja su navedena u narednoj teoremi 1.1.

Teorema 1.1. *Problem $\mathcal{Q}$ NP-kompletan ako:*

- $\mathcal{Q}$ *pripada klasi NP i*
- *postoji NP-kompletan problem* $\mathcal{R}$ *koji je polinomijalno svodljiv na* $\mathcal{Q}$.

Zapravo, često je dosta lakše koristiti teoremu 1.1 i dokazati polinomijalnu svodljivost, nego direktno pokazati da je problem NP-težak. Međutim, da bi se izveo takav dokaz, neophodno je da je za neki problem $\mathcal{R}$ već dokazano da je NP-kompletan. Klasa NP-kompletnih problema sadrži veliki broj međusobno ekvivalentnih problema.

Prvi problem za koji dokazana NP-kompletnost je problem iskazne zadovoljivosti (engl. *Boolean Satisfiability Problem* - SAT), a za taj dokaz je zaslužan Kuk (*Stephen Cook*) 1971. godine [25]. Kod problema SAT se podrazumeva da je data iskazna formula u KNF i treba utvrditi da li je izraz zadovoljiv. Pri tome KNF je skraćenica za konjunktivnu normalnu formu (engl. *conjunctive normal form*).

Teorema 1.2 (Kukova teorema)**.** *SAT problem je NP-kompletan.*

Kukova teorema je kasnije poslužila da se za mnoge druge probleme dokaže da su NP-kompletni. Na primer, koristeći dokaz polinomijalne svodljivosti problema SAT na problem 3SAT (specijalni slučaj problema SAT gde svaka klauzula sadrži tačno tri literala), dokazano je da i je problem 3SAT NP-kompletan. Primenu teoreme 1.1 u cilju dokazivanja NP-kompletnosti različitih problema može se naći u knjigama [195] i [26], a detaljnija klasifikacija složenosti problema se može pronaći u knjizi [148].

1.4 NP-teški problemi kombinatorne optimizacije

Za problem $\mathcal{Q}$ se kaže da je NP-težak ali nije NP-kompletan ukoliko se svi problemi iz klase NP mogu u polinomijalnom vremenu transformisati u $\mathcal{Q}$, ali pripadnost problema $\mathcal{Q}$ klasi NP nije utvrđena. Termin NP-teški problemi se koristi i za optimizacione probleme koji ne pripadaju klasi NP jer nisu problemi odlučivosti. Među primerima koji su navedeni u sekciji 1.1, takvi su problem ranca i TSP.

Za neke probleme kombinatorne optimizacije pronađen je algoritam koji u polinomijalnom vremenu nalazi optimalno rešenje. Takav je problem nalaženja najkraćih puteva u grafu, kao i problem kada u grafu treba pronaći minimalno povezujuće stablo (engl. *minimum-cost spanning tree* - MCST).

Problem ranca se može rešiti pseudo-polinomijalnim algoritmom (dinamičkim programiranjem) i spada u grupu tzv. slabo NP-kompletnih problema (engl. *weakly NP-complete*)

Sa druge strane, ispostavlja se da za veliki broj problema kombinatorne optimizacije ne postoji algoritam polinomijalne vremenske složenosti koji ga rešava osim ako je P=NP [76]. Veliki broj problema kombinatorne optimizacije spada u grupu NP-teških problema i njihovo rešavanje predstavlja izazov koji privlači mnogobrojne istraživače širom sveta. Različite egzaktne metode su razvijane i korišćene za rešavanje mnogobrojnih optimizacionih problema, ali sa porastom dimenzije primera problema koji se rešava često zahtevaju nedopustivo dugo procesorsko vreme. Zbog toga istraživači posvećuju veliku pažnju konstrukciji efikasnih algoritama koji pronalaze dovoljno dobra rešenja, koja su često i optimalna rešenja.

1.5 Računarska inteligencija

U poslednje dve, tri decenije istraživači su posvetili razvijanju računarskih tehnika za rešavanje problema koje su prethodno bili teški ili nemogući za rešavanje. Ove tehnike, kolektivno nazvane tehnikama računarske inteligencije, inicijalno su uključivale neuronske mreže, fazi sisteme i evolutivne algoritme [47]. Danas se pod računarskom inteligencijom (engl. *Computational Intelligence* - CI) podrazumeva širi spektar metoda koje imaju za cilj da daju rezultate za relativno kratko vreme i mogu se primeniti na veliki broj test primera.

U literaturi [52] se navodi da se računarska inteligencija može posmatrati kao podoblast veštačke inteligencije i da podrazumeva više različitih paradigmi, kao i njihovo kombinovanje. Računarskoj inteligenciji pripadaju i različite metaheurističke metode o kojima je reč u narednoj sekciji.

1.5.1 Metaheuristike

Izračunavanje optimalnih rešenja je nedostižno za mnoge optimizacione probleme značajne za nauku, industriju, ekonomiju, itd. Tako je u praksi često potrebno i dovoljno da se efikasno pronađu dobra rešenja, što se uspešno postiže primenom heuristika i metaheuristika. Određivanje optimalnog rešenja može biti neracionalno u situacijama kada troškovi/vreme pretraživanja premašuju eventualnu dobit i korist

od garancije da je dobijeno rešenje optimalno.

Pod heuristikom se podrazumeva tehnika kojom se traži dobro rešenje zadatka za relativno kratko vreme, bez mogućnosti garantovanja njegove ni optimalnosti, ili čak ni njegove bliskosti optimalnom rešenju. Rešenje dobijeno primenom heuristika mora biti dopustivo i pretpostavlja se da se heurističkom metodom problem rešava brže nego egzaktnim metodama koje bi garantovale optimalno rešenje.

Važnije primene heuristika na probleme kombinatorne optimizacije su započele primenom algoritma A*, A zvezda (engl. *A star*), koji se koristi za pretragu grafova. A* algoritam predstavlja jedan od osnovnih i najpopularnijih algoritama veštačke inteligencije [150] i koristi se za nalaženje najkraće putanje između dva čvora grafa gde se koriste metode heuristike koje procenjuju donju granicu daljine do ciljnog čvora.

Poboljšanja u oblasti heurističkih metoda dovode do rešavanja instanci većih dimenzija različitih problema, smanjivanje vremena potrebnog za pronalaženje rešenja, kao i pronalaženje boljih rešenja. Heuristike se često koriste za rešavanje optimizacionih problema, pogotovu za test primere velikih dimenzija ili za dobijanje početnog (inicijalnog) rešenja neke određene procedure. Heuristike primenjene na probleme velikih dimenzija daju u razumnom vremenu dobra rešenja, koja su često optimalna iako se optimalnost ne može dokazati. Dakle, heuristike predstavljaju aproksimativne metode pažljivo konstruisane za pojedinačne probleme, koje uspešno pronalaze kvalitetna rešenja u razumnom vremenu. Strategije koje su česte komponente različitih heurističkih metoda:

- konstruktivni algoritmi koji grade dopustiva rešenja,
- algoritmi lokalne pretrage koji unapređuju dopustiva rešenja zaustavljajući se na prvom lokalnom optimumu,
- strategije koje omogućavaju izlaske iz lokalnih optimuma.

O tome da ne postoji univerzalni algoritam pretrage koji odgovara svim problemima govori teorema „nema besplatnog ručka" (engl. *No Free Lunch Theorem*)[191]. Stoga se, u oblasti optimizacije, algoritmi pažljivo biraju za specifične probleme.

Klasične heurističke metode se razvijaju u cilju rešavanja jednog problema, ali od 90-ih godina dvadesetog veka preovladao je metaheuristički pristup koji podrazumeva razvijanja pravila i načela koja se mogu primeniti za rešavanje velikog

broja različitih problema. Reč heuristika potiče od grčke reči *heuriskein* koja u prevodu znači *„pronaći”*, dok se prefiks *meta* odnosi na *„metodologiju višeg nivoa”*, dakle *„iznad” heuristika*. Fred Glover je 1986. godine u svom radu [81] uveo termin „metaheuristika” koji se i danas koristi. Metaheuristike se mogu posmatrati kao generalizovane metode računarske inteligencije koje se mogu prilagoditi velikom broju različitih problema matematičke optimizacije. Ova klasa algoritama ima za cilj da se efikasnim pretraživanjem prostora pretrage pronalaze kvalitetna rešenja. Omogućavajući dobijanje dobrih rešenja u relativno kratkom vremenu, meteheuristike su jedne od najuspešnijih tehnika za rešavanje mnogih složenih problema, što objašnjava njihov značaj, razvoj i sve širu primenu. Metaheuristike se sve više koriste za rešavanje i spadaju u najefikasnije strategije za rešavanje različitih složenih problema kombinatorne optimizacije.

Jedna od definicija je ta da metaheuristike predstavljaju iterativni glavni proces koji upravlja i menja operacije manjih heuristika kojima manipuliše kako bi se na efikasan način došlo do kvalitetnog rešenja [188]. Metaheuristike, prema nekim viđenjima (na primer [161]), imaju zajedničke gradivne elemente koji su kombinovani različitim strategijama u cilju prevazilaženja preuranjene konvergencije ka lokalnom optimumu. Uspešna implementacija metaheuristika podrazumeva kombinaciju više uobičajenih komponenti kao što su pohlepni algoritmi, lokalna pretraga, intenzifikacija, diverzifikacija, elitna rešenja, učenje, itd.

Postoje metaheuristike koje su zasnovane na iterativnom popravljanju jednog rešenja, pri čemu se početno rešenje konstruiše na slučajan način ili nekim konstruktivnim algoritmom. U ove metaheuristike spadaju nasumično pohlepna adaptivna pretraga (engl. *Greedy Randomized Adaptive Search Procedure* - GRASP) [60, 160, 64, 65], metoda promenljivih okolina (engl. *Variable Neighborhood Search* - VNS) [144, 145, 94, 97, 46], metoda simuliranog kaljenja (engl. *Simulated Annealing* - SA) [108, 187, 1], tabu pretraga (engl. *Tabu Search* - TS) [82, 83, 80, 87], itd. Pored toga, postoje i populacione metaheuristike, kod kojih se istovremeno popravlja skup jedinki koji se naziva populacija. Jedinke u populaciji odgovaraju rešenju ili delu rešenja. Među populacionim meteheuristikama najpoznatije su evolutivni algoritam (engl. *Evolutionary Algorithm* - EA) [10, 32, 30], metoda roja čestica (engl. *Particle Swarm Optimization* - PSO) [24, 106, 45], metoda optimizacije mravljim kolonijama (engl. *Ant Colony Optimization* - ACO) [41, 40, 39], itd. Metaheuristike se često baziraju na metaforama, a neki od primera su SA, EA i ACO.

Definicija metaheuristika, prema korišćenoj referenci [77], je da su to metode koje kombinuju procedure lokalne pretrage i strategije višeg nivoa koje omogućavaju izlazak iz lokalnih optimuma i robusnu pretragu prostora rešenja. S vremenom su ove metode napredovale, tako da takođe uključuju strategije koje omogućavaju izbegavanje lokalnih optimuma u složenim prostorima pretrage, posebno procedura koje koriste jednu ili više struktura okolina u cilju definisanja dozvoljenih koraka prilikom prelaska sa jednog rešenja na drugo, kreiranje rešenja u konstruktivnom procesu, itd. Stepen do koga se okoline rešenja pretražuju varira u zavisnosti od metode koja se primenjuje. U slučaju nekih populacionih metaheuristika okoline su implicitno definisane menjanjem komponenti jednog rešenja i menjanjem komponentama drugog rešenja. U drugim populacionim metodama, na primer metoda ponovnog povezivanja puta koja će biti detaljno opisana u ovoj disertaciji, strukture okolina su korišćene u svojoj punoj opštosti.

1.5.2 Hibridizacija metoda

U interesu rešavanja problema sa mnogobrojnim primenama, jedan od glavnih ciljeva je razvoj efikasnih metoda optimizacije. Čest pristup rešavanju velikih test primera NP-teških optimizacionih problema predstavljaju metaheuristike, koje se mogu prilagoditi različitim tipovima problema. U cilju iskorišćenja raznovrsnosti metaheurističkih metoda, istraživači sve češće kreiraju različite metode koje kombinuju više različitih algoritamskih ideja iz različitih metaheuristika. Takve metode koje koriste na primer dve različite metaheuristike se nazivaju hibridne. Iskustvo pokazuje da ovakve kombinacije različitih koncepata mogu dovesti do boljih rešenja nego što je to moguće pojedinačnim metaheursitikama, pa ne čudi što se, tokom poslednjih godina, interesovanje za hibridne metaheuristike povećava. Najčešće su se te hibridizacije ostvaruju statički, a određivanje parametara se određuje eksperimentalno.

Kako je hibridizacijom moguće iskoristiti dobre performanse različitih metoda, posebno su se istakle neke kombinacije komplementarnih metoda koje su pokazale dobre performanse kada se primene zajedno.

Različiti hibridni meteheuristički pristupi korišćeni su u brojnoj literaturi [15, 176, 120, 194, 131, 4, 138]. Posebno, hibridi evolutivnih algoritama i lokalne pretrage se nazivaju memetski algoritmi i korišćeni su za rešavanje različitih problema

kombinatorne optimizacije [134, 132]. U nekim slučajevima hibridizacija metaheurističkih metoda se vrši i sa egzaktnim algoritmima, kao na primer u radu [177].

1.6 Fazi mere, fazi skupovi i fazi logika

Matematičar Lofti A. Zadeh je 1965. godine uveo fazi logiku, matematički aparat koji omogućava rad sa raznolikim informacijama koje ne mogu precizno da se izraze uz pomoć klasične logike i teorije klasičnih skupova.

Pre uvođenja osnovnih definicija u nastavku je dat jednostavan primer. Za razliku od binarne logike gde su vrednosti ili 1 ili 0, fazi logika podrazumeva vrednosti iz segmenta $[0, 1]$. Ovo se može posmatrati i kao 0% i 100%.

Primer 1.5. *Može se reći da je petogodišnjak* 100% *mlad, osamnaestogodišnjak* 50% *mlad, dok je čovek sa 80 godina* 0% *mlad. U binarnom svetu, svi ispod 18 su* 100% *mladi, a svi preko 18 su* 0% *mladi.*

1.6.1 Fazi skupovi

Prilikom prenošenja informacija i znanja u svakodnevnom jeziku pojavljuje se velika nepreciznost i neodređenost, raspilinutost (engl. *fuzziness*). Neizvesnost može biti izazvana i nepreciznošću merenja, jezičkim neodređenostima i mnogim drugim faktorima. Zadeh je uveo fazi skupove motivisan činjenicom da se pomoću teorije klasičnih skupova ne mogu predstaviti takvi opisi i podele.

Klasični skupovi

Za dati skup X, svaki njegov podskup A je tačno određen pripadnošću nekog elementa $x \in X$ tom skupu A. Dakle, pripadnost podskupu A je potpuno određena odgovarajućom karakterističnom funkcijom $\chi_A : X \to \{0, 1\}$ zadatom sa:

$$\chi_A(x) = \begin{cases} 1, & \text{ako } x \in A, \\ 0, & \text{ako } x \notin A. \end{cases}$$

Ako su A i B dva podskupa skupa X, kažemo da je B podskup od A ukoliko važi $(\forall x \in X)\ (\chi_B(x) \leq \chi_A(x))$ i pišemo $B \subseteq A$.

Operacije poput unije, preseka i komplementa, mogu se nad klasičnim skupovima predstaviti pomoću karakterističnih funkcija:

- $\chi_{A\cup B}(x) = max\{\chi_A(x), \chi_B(x)\}$,
- $\chi_{A\cap B}(x) = min\{\chi_A(x), \chi_B(x)\}$,
- $\chi_{A^C}(x) = 1 - \chi_A(x)$.

Osnovne osobine operacija sa skupovima su:

- Komutativnost: $A \cup B = B \cup A$ i $A \cap B = B \cap A$.
- Asocijativnost: $(A \cup B) \cup C = A \cup (B \cup C)$ i $(A \cap B) \cap C = A \cap (B \cap C)$.
- Distributivnost: $A\cap(B\cup C) = (A\cap B)\cup(A\cap C)$ i $A\cup(B\cap C) = (A\cup B)\cap(A\cup C)$.
- Idempotentnost: $A \cup A = A$ i $A \cap A = A$.
- Apsorpcija: $A \cap (A \cup B) = A$ i $A \cup (A \cap B) = A$.
- Apsorpcija sa $\emptyset$ i X: $A \cap \emptyset = \emptyset$ i $A \cup X = A$.
- Neutralni element: $A \cup \emptyset = A$ i $A \cap X = A$.
- Involucija: $(A^c)^c = A$.
- Zakon kontradikcije: $A \cap A^c = \emptyset$.
- Zakon isključenja trećeg: $A \cup A^c = X$.
- De Morganovi zakoni: $(A \cup B)^c = A^c \cap B^c$ i $(A \cap B)^c = A^c \cup B^c$.

Fazi skupovi

Teorija fazi skupova predstavlja proširenje teorije klasičnih skupova. Fazi podskup A skupa X je određen funkcijom pripadanja koja može uzimati vrednosti iz celog intervala $[0, 1]$, za razliku od pomenute karakteristične funkcije klasičnih skupova koja uzima samo vrednosti 0 i 1. Funkciju pripadanja fazi podskupu A se često označava sa μ_A:

$$\mu_A : X \to [0, 1].$$

Važno je ovde uočiti razliku u odnosu na klasičan podskup da neki element $x \in X$ pripada datom fazi skupu A sa određenim stepenom između 0 i 1. U nastavku je data formalna definicija.

Definicija 1.8. *Fazi skup A je skup uređenih parova koji se sastoji od elementa $x \in X$ i odgovarajućeg stepena pripadnosti $\mu_A(x)$. Dakle,*

$$A = \{(x, \mu_A(x)) : x \in X\},$$

gde je $\mu_A : X \to [0, 1]$ funkcija pripadanja.

Definicija 1.9. *Neka su A i B neka dva fazi skupa. Dekartov proizvod*

$$A \times B = \{(x, y) : x \in A, y \in B\}$$

može imati podskup R se naziva fazi relacija od A i B i zadat je svojom funkcija pripadanja $\mu_R : A \times B \to [0, 1]$.

Fazi relacija je specijalna vrsta fazi skupova.

Operacije sa fazi skupovima

Kako su operacije sa fazi skupovima uopštenje operacija nad klasičnim skupovima, prirodno što se prethodno napisane formule sa karakterističnim funkcijama zapisuju pomoću funkcija pripadanja:

- $\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$,
- $\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$,
- $\mu_{A^C}(x) = 1 - \mu_A(x)$.

Za fazi podskup B kažemo da je podskup fazi podskupa A ukoliko važi $(\forall x \in X)$ $(\mu_B(x) \leq \mu_A(x))$.

Za fazi podskup A se može definisati njegov α-presek, za $\alpha \in [0, 1]$, kao klasičan skup $[A]_\alpha = \{x : \mu_A \geq \alpha\}$. Na osnovu definicije, odmah važi da je $[A]_\beta \subseteq [A]_\alpha$ za $\alpha \leq \beta$. Takođe, za fazi podskupove A i B važi $[A \cup B]_\alpha = [A]_\alpha \cup [B]_\alpha$ i $[A \cap B]_\alpha = [A]_\alpha \cap [B]_\alpha$.

Za fazi podskup A skupa $\mathbb{R}^n$ kažemo da je *konveksan* ukoliko je svaki njegov α-presek, za $\alpha \in (0,1]$, konveksan skup u klasičnom smislu. Dakle, fazi podskup A od $\mathbb{R}$ je konveksan ako i samo ako je $\mu_A(\alpha x_1 + (1-\alpha)x_2) \geq min\{\mu_A(x_1), \mu_A(x_2)\}$ za sve $x_1, x_2 \in \mathbb{R}$ i $\alpha \in [0,1]$.

1.6.2 Fazi brojevi

Fazi broj A je specijalan fazi podskup proširenog skupa realnih brojeva $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. Fazi broj predstavlja generalizaciju klasičnog realnog broja u smislu da se ne odnosi na jednu vrednost, već na povezan skup mogućih vrednosti, gde svaka moguća vrednosti ima svoju težinu između 0 i 1. Ta težina je zapravo predstavljena funkcijom pripadanja. U literaturi postoje različite definicije fazi brojeva, a ovde je navedena definicija iz knjige [149].

Definicija 1.10. *Fazi podskup A od $\bar{\mathbb{R}}$ je fazi broj ako je*

- *normalizovan ($\mu_A(x) = 1$ za neko $x \in \mathbb{R}$),*
- *konveksan (za sve $x, y, z \in \mathbb{R}$ $x < y < z$ $\mu_A(y) \geq min\{\mu_A(x), \mu_A(z)\}$).*

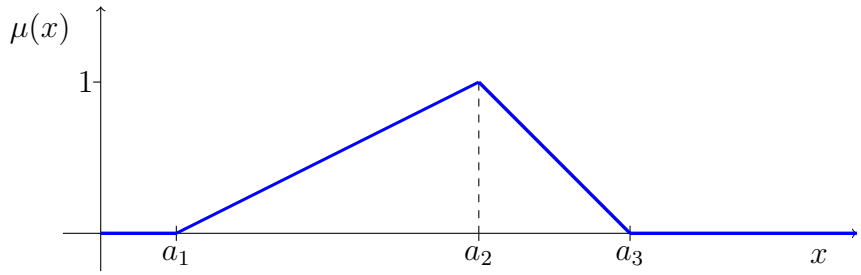
Može se primetiti da prethodna definicija predstavlja uopštenje pojma intervala. Zapravo, za svako $\alpha \in (0,1]$, odgovarajući α-presek fazi A broja je interval. Ukoliko je $\mu_A(x) = 1$ za neko $x \in \bar{\mathbb{R}}$, tada je $\mu_{A|[-\infty,x]}$ monotono neopadajuća funkcija, a $\mu_{A:[x,\infty]}$ monotono nerastuća funkcija.

Primer 1.6. *Trougaoni fazi broj (engl.* Triangular Fuzzy Number - *TFN) je jedan od najčešće korišćenih fazi brojeva. U skraćenoj formi često se označava kao $A = TFN(a_1, a_2, a_3)$, gde su a_1, a_2, a_3 realni brojevi takvi da je $a_1 < a_2 < a_3$, a funkcija pripadanja je definisana kao:*

$$\mu_A(x) = \begin{cases} 1 + \frac{x-a_1}{a_2-a_1}, & ako\ je\ a_1 \leq x \leq a_2, \\ 1 + \frac{a_3-x}{a_3-a_2}, & ako\ je\ a_2 \leq x \leq a_3, \\ 0, & inače. \end{cases}$$

Funkcija pripadanja je prikazana na slici 1.2.

Postoje različite mogućnosti za definisanje funkcije pripadanja, pa i različiti tipovi fazi brojeva. Još neki od poznatih primera su Gausovi i kvadratni, a u okviru



Slika 1.2: *Funkcija pripadanja za trougaoni (linearni) fazi broj*

disertacije su korišćeni i neki drugi fazi brojevi koji su na odgovarajućim mestima opisani.

1.6.3 Fazi mere

S obzirom na to da nije od primarnog značaja za istraživanja u okviru ove disertacije, teorija fazi mere je samo ukratko opisana u nastavku. Fazi mere su značajne kao koncept koji omogućava da se na pogodan način uporede teorija verovatnoća i teorija fazi skupova, ali i u mnogim primenama. Tako su, na primer, fazi mere su proučavane u okviru teorije odlučivanja pod neodređenim uslovima od strane različitih autora [178, 190]. U domaćoj literaturi, fazi mere i njihova primena su detaljno opisane u monografiji čiji je autor profesor Endre Pap [149].

Trougaone norme

U ovom delu disertacije su opisane specijalne operacije na jediničnom intervalu $[0, 1]$ koje se nazivaju trougaone norme. U nastavku su korišćene oznake i definicije vezane za fazi logiku iz monografije autora E. Papa [149]. Trougaone norme (engl. *triangular norms*), koje se kraće označavaju t-norme, predstavljaju klasu realnih binarnih operacija. Osim u teoriji fazi logika i fazi skupova, ove operacije koriste se u teoriji probabilističkih metričkih prostora, teoriji informacija, teoriji igara, nelinearnim jednačinama, itd.

Definicija 1.11. *Trougaona norma (t-norma) je funkcija $T : [0, 1]^2 \to [0, 1]$ takva da ispunjava sledeće uslove:*

(T1) $T(x, y) = T(y, x)$ (komutativnost),

(T2) $T(x, T(y, z)) = T(T(x, y), z)$ *(asocijativnost),*

(T3) $T(x, y) \leq T(x, z)$ *za* $y \leq z$ *(monotonost),*

(T4) $T(x, 1) = x$ *(rubni uslov).*

Na osnovu definicije direktno sledi: $T(0, x) = T(x, 0) = 0$ i $T(1, x) = x$.

Primer 1.7. *Primeri najpoznatijih t-normi:*

- $T_M(x, y) = min\{x, y\}$,
- $T_P(x, y) = xy$,
- $T_L(x, y) = max\{0, x + y - 1\}$,
- $T_W(x, y) = \begin{cases} min\{x, y\}, & ako\ je\ max\{x, y\} = 1, \\ 0, & inače. \end{cases}$

Uz podrazumevanje da $T_1 \leq T_2$ ima značenje da je $T_1(x, y) \leq T_2(x, y)$ za sve $x, y \in [0, 1]$, ove četiri t-norme važi sledeći poredak: $T_W \leq T_L \leq T_P \leq T_M$.

Trougaone konorme

Definicija 1.12. *Trougaona konorma (t-konorma) je funkcija* $S : [0, 1]^2 \to [0, 1]$ *takva da ispunjava sledeće uslove:*

(S1) $S(x, y) = S(y, x)$ *(komutativnost),*

(S2) $S(x, S(y, z)) = S(S(x, y), z)$ *(asocijativnost),*

(S3) $S(x, y) \leq S(x, z)$ *za* $y \leq z$ *(monotonost),*

(S4) $S(x, 0) = x$ *(rubni uslov).*

Primetimo da se t-norme i t-konorme razlikuju samo po rubnim uslovima.

Primer 1.8. *Primeri najpoznatijih t-konormi:*

- $S_M(x, y) = max\{y, x\}$,
- $S_P(x, y) = x + y - xy$,

- $S_L(x, y) = min\{1, x + y\}$,
- $S_W(x, y) = \begin{cases} max\{x, y\}, & ako\ je\ min\{x, y\} = 0, \\ 1, & inače. \end{cases}$

Prvobitno je t-konorma S uvedena kao dualna operacija za datu t-normu T:

$$S(x, y) = 1 - T(1 - x, 1 - y),\ x, y \in [0, 1].$$

Svaka t-konorma S se može ovako predstaviti.

U literaturi [109] se mogu pronaći i mnoge druge trougaone norme i konorme.

Osobine trougaonih normi

Za datu t-normu T i t-konormu S mogu se posmatrati odgovarajuće komutativne totalno uređene polugrupe $([0, 1], T)$ i $([0, 1], S)$. Za $([0, 1], T)$ neutralni element e_T je 1 i anihilator a_T je 0, dok je u slučaju $([0, 1], S)$ neutralni element e_S 1 i anihilator a_S je 1.

1.6.4 Fazi logika

Iako se u širem smislu fazi logika nekad koristi i kao sinonim za teoriju fazi skupova, fazi logika, u užem smislu, predstavlja uopštenje viševrednosne logike. Zadehova stroga negacija c je zadata sa $c(x) = 1 - x$. Za datu t-normu T, njoj odgovarajuća dualna t-konorma S je data kao $S(x, y) = c(T(c(x), c(y)))$. Osnovne logičke operacije se mogu uvesti u $[0, 1]$-verovatnosnoj logici na sledeći način:

- konjunkcija: $x \wedge_T y = T(x, y)$,
- disjunkcija: $x \vee_T y = S(x, y)$.

Neka je x istinitosna vrednost izraza A, a y izraza B. Onda je $x \wedge_T y$ istinitosna vrednost izraza „A i B", a $x \vee_T y$ istinitosna vrednost izraza „A ili B". Pored toga, $c(x)$ predstavlja istinitosnu vrednost izraza „ne A".

Zapravo, ovako definisane operacije odgovaraju klasičnim logičkim operacijama u slučaju klasične logike.

U slučaju Bulove logike važi da je „ne A ili B" ekvivalentno sa „ako A onda B". Analogno se implikacija u $[0,1]$-verovatnosnoj logici može uvesti pomoću funkcije $I_T : [0,1]^2 \to [0,1]$ definisanoj, pomoću T, c i S, kao

$$I_T(x,y) = S(c(x),y) = c(T(x,c(y))).$$

Dakle, u ovom slučaju uvek važi zakon kontrapozicije $I_T(x,y) = I_T(c(y),c(x))$.

Za dve osnovne t-norme T_M i T_L važi:

$$I_{T_M}(x,y) = \begin{cases} y, & \text{za } x+y \geq 1, \\ 1-x, & \text{inače,} \end{cases}$$

$$I_{T_L}(x,y) = \begin{cases} 1, & \text{za } x \geq y, \\ 1-x-y, & \text{inače.} \end{cases}$$

Druga mogućnost za uopštenje klasične implikacije se zasniva na operatoru

$$R_T(x,y) = \sup\{z \in [0,1] : T(x,z) \leq y\}.$$

U tom slučaju, za dve osnovne t-norme T_M i T_L važi:

$$R_{T_M}(x,y) = \begin{cases} 1, & \text{za } x \geq y, \\ y, & \text{inače,} \end{cases}$$

$$R_{T_L}(x,y) = \begin{cases} 1, & \text{za } x \geq y, \\ 1-x-y, & \text{inače.} \end{cases}$$

Iako važi $I_{T_L}(x,y) = R_{T_L}(x,y), \forall x,y \in (0,1)$ u opštem slučaju se ova dva operatora ne poklapaju.

1.6.5 Odnos verovatnoće i fazi logike

Fazi logika i teorija verovatnoće se bave različitim vrstama nesigurnosti i razlikuju im se namene, iako obe mogu predstave određene nivoe subjektivnosti. Teorija fazi skupova koristi koncept pripadnosti fazi skupu, tj. u kojoj meri promenljiva pripada skupu, pri čemu ne mora biti nikakve nesigurnosti u vezi sa stepenom pripadanja. Sa

druge strane, teorija verovatnoće koristi koncept subjektivne verovatnoće, tj. koliko je verovatno da je promenljiva u skupu (ili je u potpunosti u skupu ili u potpunosti nije u skupu, ali je neizvesnost u vezi sa tim da li jeste ili nije). Posledica ove razlike je da teorija fazi skupova oslabljuje aksiome klasične teorije verovatnoća, koji su sami po sebi nastali dodavanjem nesigurnosti (ali ne i stepena) klasičnoj 1/0 Aristotelovoj logici.

Odnos verovatnoće i fazi logike slikovito prikazuje naredni primer iz [149].

Primer 1.9. *Neka je X skup svih tečnosti i neka je A njegov fazi podskup koji predstavlja tečnosti pogodne za piće. Stoga, vrednosti funkcije pripadanja fazi skupu A mogu biti definisane na sledeći način: $\mu_A(\text{čista voda}) = 1$, $\mu_A(\text{voćni sok}) = 0,8$, $\mu_A(\text{alkoholna pića}) = 0,2$, dok je $\mu_A(\text{sona kiselina}) = 0$. Pretpostavimo da žedna osoba treba da izabere između dve boce: na prvoj piše $\mu_A(x) = 0,92$, a na drugoj da je verovatnoća da je pitka tečnost $0,92$. Očigledan izbor osobe koja ne voli da rizikuje je prva boca s obzirom da se u toj boci nalazi neka pitka tečnost (između vode i soka). U drugoj boci može biti pitka voda, ali i otrov.*

1.6.6 Primena fazi logike

Od svojih početaka, teorija fazi skupova razvijana je u različitim pravcima, a danas se termini fazi logike i teorije fazi skupova najčešće koriste kao sinonimi. Matematička teorija fazi logike je veoma razvijena i u stalnom je napretku.

Sve šira primena fazi logike proizilazi iz njene podobnosti za predstavljanje neizvesnih, nepreciznih i nesigurnih informacija. Od velikog je praktičnog značaja postojanje matematičke aparature koja omogućava predstavljanje nepreciznih pojmova i tvrđenja i računanje sa njima. Računarske metode koje se zasnivaju na teoriji fazi skupova imaju različite inženjerske i naučne primene. Primene fazi logike se mogu naći u veštačkoj inteligenciji i računarskim naukama uopšte, medicini, inženjerstvu, teoriji odlučivanja, ekspertskim sistemima, operacionim istraživanjima, robotici, obradi slika, programiranju elektronskih uređaja, itd.

Razlozi za široku primenu fazi logike su mnogobrojni. Konceptualno, fazi logika je jednostavna za razumevanje, fleksibilna i intuitivna. Pored toga, sistemi zasnovani na fazi logika su tolerantni na neprecizne podatke. Dodatno, u okviru fazi logike mogu se modelirati nelinearne funkcije proizvoljne složenosti. Znanja eksperata se mogu pretočiti u sistem zasnovan na fazi logici.

Za razliku od klasične logike (diskretne logike), fazi logika omogućava širi spektar za ocenjivanje situacija iz realnog sveta. To svojstvo omogućava fazi logici da zauzme mesto u aktuelnim oblastima kao što su sistemi kontrole.

Prve primene fazi logike su započete tek tokom 1970-ih godina [166]. Od tada su istraživači iz Japana postali tradicionalno najveći inovatori u različitim oblastima u kojima se fazi logika može primeniti. Danas, gotovo da nema oblasti u kojoj fazi logika nije primenjena. U okviru inženjerskih primena, korišćena je u kamerama, mašinama za pranje i mnogim drugim elektronskim uređajima. Pored toga, fazi logika se koristi u medicinskoj dijagnostici, prepoznavanju rukopisa i gotovo svim sistemima gde postoji neka relacija zavisnosti između ulaznih i izlaznih veličina.

Da bi se razumeli razlozi za intenzivan razvoj fazi skupova, mogu se naglasiti dva svojstva. Fazi skupovi, kao alat za predstavljanje delimičnog pripadanja skupu, su se pokazali veoma korisni i upotrebljivi. Sa druge strane, mnogi matematički alati koji su razvijeni na osnovu fazi logike omogućavaju širok spektar primene u realnim situacijama.

Od 1965. godine do danas, matematički aparat fazi skupova se dosta razvio, počevši od osnova zasnovanih na teoriji skupova do viševrednosne logike kod koje postoji više od dve istinitnosne vrednosti. Napredak u oblasti fazi skupova je obogatio klasičnu dvo-vrednosnu logiku iz nove i dublje perspektive.

Fazi logika predstavlja praktičan način za transformaciju ulaznih podataka u izlazne, što je česta potreba. Na primer, sa informacijom koliko brzo se automobil kreće i koliko snažno motor radi, sistem zasnovan na fazi logici može odrediti adekvatan izbor stepena prenosa.

Sistemi zasnovani na fazi logici odgovaraju i nelinearnim sistemima, kao i sistemima koji imaju više ulaznih podataka i više izlaznih podataka.

Osnovna struktura sistema zasnovanih na fazi logici

Često se fazi pristup koristi da se reši neki problem upravljanja sistemom tako što se neko ekspertsko iskustvo o procesu implementira u sam algoritam. Pri tome se javljaju naredni koraci:

- Fazifikacija (engl. *fuzzification*) - ulazni podaci se modifikuju u adekvatan fazi oblik.

- Zaključivanje na osnovu fazi pravila koja zapravo predstavljaju odnos između dva ili više fazi skupova.
- Defazifikacija (engl. *defuzzification*) - transformiše se iz fazi oblika u odgovarajući izlazni podatak.

1.7 Korišćenje fazi logike u metodama i u matematičkim modelima

Mnogi problemi iz realnog sveta sadrže različite vrste nepouzdanosti, pa njihovi matematički modeli kreirani pomoću klasičnih matematičkih tehnika nisu u potpunosti tačni. Fazi skupovi i različite probabilističke metode se najčešće koriste za modeliranje problema iz realnog sveta. U okviru ove doktorske disertacije biće kreirani novi modeli navedenih problema koji uz pomoć korišćenja fazi teorije bolje opisuju realne situacije. U disertaciji će biti razmatrane neke mogućnosti upotrebe fazi logike u cilju konstrukcije unapređenih metoda za efikasnije rešavanje razmatranih problema, kao u cilju modeliranja problema iz prakse.

U okviru poglavlja 4 i 5 biće predstavljene neke mogućnosti upotrebe fazi logike za poboljšanje performansi metoda za rešavanje razmatranih problema matematičke optimizacije. Metaheurističke metode su kreirane tako da rade korektno bez obzira na odabir parametara koji se koriste. Ipak, dobar odabir parametara može poboljšati rad algoritma. U disertaciji će biti kreiran fazi pristup adaptaciji parametara tokom izvršavanja algoritma u cilju što efikasnijeg pronalaženja rešenja [157].

Fazi logika je našla svoje primene i u modeliranju problema, s obzirom da veliki broj problema iz prakse sadrži različite nepouzdanosti, nepreciznosti i neodređenosti. U poglavlju 6 prikazana je primena fazi logike u cilju preciznijeg predstavljanja jednog od problema koji je predmet ove disertacije.

Preciznost, binarna logika i determinizam su karakteristike većine tradicionalnih alata za formalno modeliranje, rezonovanje i računarstvo uopšte. U tradicionalnoj logici, iskaz može biti tačan ili netačan (a ne nešto između). Kao što je već naglašeno, u klasičnoj teoriji skupova, element ili pripada ili ne pripada skupu. Preciznost podrazumava da parametri u modelu tačno odgovaraju sistemu iz prakse koji se modelira. Stoga, takav model mora biti nedvosmislen, da nema slučajnih događaja

i da su strukture i parametri poznati i fiksirani. Međutim, ovakve pretpostavke se ne mogu nametnuti realnom svetu.

Teorije i alati koji su se dugo koristili za modeliranje nesigurnosti realnog sveta pretežno su teorija verovatnoće i statistika. Kasnije se pojavilo više drugih teorija od kojih se najviše istakla fazi logika.

2

Problemi rutiranja vozila

Jedan od poznatijih problema kombinatorne optimizacije, koji ujedno predstavlja i uopštenje problema trgovačkog putnika je problem rutiranja vozila (engl. *Vehicle Routing Problem* – VRP). Prvi put je formulisan u radu Dantziga i Ramsera 1959. godine [29] kao problem otpremanja kamiona (engl. *Truck Dispatching Problem* - TDP). Pet godina kasnije, ovaj problem je generalizovan [23] i od tada je poznat pod imenom VRP i privlači pažnju sve većeg broja istraživača. Danas postoji čitava klasa problema rutiranja i oni imaju širok spektar primene, posebno u oblasti distribucije i logistike.

Najčešći scenario kod ove klase problema je da je na raspolaganju određeni broj vozila koji put počinju i završavaju u istom skladištu. Pored toga, dat je skup lokacija koje predstavljaju korisnike koje vozila treba da posete, tako da svaka lokacija bude posećena tačno jednom. Najčešći cilj kod ove klase problema je da je potrebno minimizovati ukupan pređeni put i/ili broj vozila.

Problemi rutiranja vozila se najčešće zadaju na kompletnom neusmerenom grafu $G = (V, E)$ sa skupom čvorova $V = \{0, 1, 2, \ldots, n\}$ i skupom grana E. Neka čvor 0 prestavlja skladište u kome se nalazi K vozila sa jednakim kapacitetima Q. U nekim primenama broj vozila K je promenljiva koju treba minimizovati, dok je to nekad fiksiran maksimalan broj vozila koja su na raspolaganju.

VRP je NP-težak problem [118], pa se osim egzaktnih algoritama razvijaju i raznovrsne heuristike za rešavanje ovog problema. Sa razvojem različitih heurističkih i metaheurističkih metoda, rastao je i broj istraživanja i objavljenih radova o problemima iz klase VRP u kojima su predlagane brojne varijante ovog problema i metode

za rešavanje.

Klasa problema rutiranja vozila se može opisati kao skup problema kod kojih je potrebno kreirati kolekciju ruta iz jednog ili više skladišta, tako da se svi korisnici obiđu. U zavisnosti od konkretnih primena problema, mogu postojati i neka dodatna ograničenja. Postoji veliki spektar različitih varijanti problema rutiranja vozila koje su razmatrane u literaturi, a neke od poznatijih podklasa, prema [17], su VRP sa vremenskim ograničenjima, otvoreni VRP, kao i dinamični VRP. Pored toga, u literaturi su često razmatrani i VRP sa ograničenim kapacitetima, kao i VRP sa više skladišta.

VRP sa ograničenim kapacitetima

Problem rutiranja vozila sa ograničenim kapacitetom (engl. *Capacitated Vehicle Routing Problem* - CVRP) podrazumeva da vozila imaju ograničene kapacitete. Osnovna varijanta ovog problema je postoji jedinstveno skladište, da su zadati zahtevi korisnika, kao i da su sva vozila identična i da imaju određen kapacitet. Problem CVRP je kao takav razmatran je u literaturi [186, 74], a javljao se i u varijantama da vozila imaju različite kapacitete i cene [89].

VRP sa vremenskim ograničenjima

Za većinu VRP problema pretpostavlja se da je vreme putovanja između skladišta i korisnika konstantno, ili jednako rastojanju [119]. U praksi, vreme putovanja zavisi od različitih faktora. Tako su se javljale ideje da vremena putovanja u modelu budu predstavljena tako što je radni dan podeljen na periode od kojih svaki ima predodređenu brzinu putovanja [103]. Pri tome, što je veći broj vremenskih intervala, model je realniji jer se brzine putovanja menjaju postepeno [111].

Često proučavan je i problem rutiranja vozila sa vremenskim prozorima (engl. *Vehicle Routing Problem with Time Windows* - VRPTW) [35, 14, 34]. U slučaju VRPTW, usluga nekom korisniku se mora započeti u okviru vremenskog prozora koji je zadao taj korisnik tj. između najranijeg i najkasnijeg vremena kada je dozvoljen početak usluge. U radu [35] predstavljena je LP relaksacija problema VRPTW i predložen algoritam baziran na metodi grananja i sečenja. Bent i sar. [14] su rešavali problem VRPTW tako što su najpre minimizovali broj vozila koristeći algoritam simuliranog kaljenja, a potom primenom metode LNS je urađena minimizacija

ukupnog pređenog puta.

Otvoreni VRP

U slučaju otvorenog VRP (engl. *Open Vehicle Routing Problem* - OVRP) vozila se ne vraćaju u skladište nakon dostave robe korisnicima, ili ukoliko se vrate onda moraju da posete iste korisnike koji su im dodeljeni, radi skupljanja robe, u obrnutom redosledu. OVRP je u literaturi rešavan na različite načine, kao na primer primenom tabu pretrage [18], heurističkom metodom baziranom na minimalnom povezujućem stablu [164], itd.

VRP sa više skladišta

Još jedna varijanta koja se često pominje u literaturi je VRP sa više skladišta (engl. *Multi-depot Vehicle Routing Problem* - MDVRP) i predlagane su različite metode za rešavanje, kao što je na primer tabu pretraga (engl. *Tabu Search* - TS) [55] i hibridni genetski algoritam [101].

Dinamični VRP

U slučaju dinamičnog VRP (engl. *Dynamic Vehicle Routing Problem* - DVRP) ulazni podaci se konstantno ažuriraju, pa se na osnovu novih podataka rute adaptiraju dinamično. Uglavnom se podrazumeva da informacije o zahtevima korisnika stižu dinamično. Za rešavanje problema rutiranja vozila sa dinamičnim zahtevima u literaturi je korišćen adaptivni evolutivni algoritam [12], kao i metode PSO i VNS [107]. VRP sa dinamičnim vremenima putovanja je razmatran u literaturi [121], a više o dinamičnim problemima rutiranja vozila se može pronaći u [152].

U literaturi su se javljale i kombinacije različitih karakteristika. Tako je na primer u [117] razmatran VRP sa ograničenim kapacitetima, stohastičkim zahtevima korisnika i vremenskim prozorima, dok je VRP sa više skladišta i vremenskim prozorima rešavan u [153].

Pored toga, VRP je kombinovan u literaturi sa različitim problemima, kao na primer sa problemom raspoređivanja (engl. *Vehicle Routing and Scheduling*) i kao takav razmatran u brojnim radovima [16, 158, 7, 48]. Kombinovan problem rutiranja vozila i raspodele inventara (engl. *Vehicle Routing and Inventory Allocation Problem*) je rešavan u literaturi na različite načine [58, 22, 124].

Pregled literature o problemima rutiranja vozila se može pronaći u [90, 51], dok se više o novijim istraživanjima i izazovima vezanim za VRP može pronaći u [17]. U nastavku će biti predstavljen problem rutiranja vozila sa ograničenjem rizika koji je razmatran u ovoj disertaciji.

2.1 Problem rutiranja vozila sa ograničenjem rizika

Jedan od zahteva prilikom prenosa novca ili robe od velike vrednosti između banaka, tržnih centara, zlatara i ostalih lokacija koje rade sa velikim količinama novca, je sigurnosni aspekt transporta. Različiti pristupi povećanju bezbednosti su se javljali. Međutim, ulaganje u zaštitnu opremu ipak ne može u potpunosti sprečiti da se pljačkaški napadi na vozila koja prenose robu od velike vrednosti dešavaju (videti [53]). Zbog toga se razmatrala ideja da se posveti pažnja bezbednosti u toku planiranja ruta [171].

Model koji je predstavljen u radu [193] koji podrazumeva fleksibilnost rutiranja i pravljenja rasporeda, uzimajući u obzir mere i vremena i prostora, a sve u cilju smanjivanja rizika od potencijalnih pljačkaških napada. Jedan od pristupa je kreiranje ruta koje su nisu predvidive za napadače, kao što je razmatrano u radu [147], gde je moguće posetiti istog korisnika više puta ali nije dozvoljeno istom granom grafa proći više puta.

U okviru ove disertacije razmatran je problem rutiranja vozila sa ograničenim rizikom (engl. *Risk-constrained Cash-in-Transit Vehicle Routing Problem* - RCTVRP) formulisan u radu [181].

Merenje rizika u problemu RCTVRP

Sigurnosni aspekti prilikom transporta novca i robe od velike vrednosti su razmatrani u literaturi iz različitih perspektiva. Jedna od važnih perspektiva je da se u trenutku kreiranja ruta uzme u obzir sigurnost. U cilju kreiranja matematičkog modela, potrebne su numeričke vrednosti za nivoe rizika na rutama.

U nastavku se koriste sledeće oznake: R_j^r = nivo rizika na ruti r pre nego što vozilo stigne do čvora j i p_{ij} = verovatnoća da se nepoželjan događaj pljačke W desi na grani (i, j).

Kao što je predstavljeno u [181], s obzirom da se događaj W može dogoditi bilo

gde na grani (i, j), ukoliko se vozilo kreće od čvora i ka čvoru j tokom rute r rizik na ruti r pre nego što vozilo stigne u čvor j se može izračunati kao:

$$R_j^r = p_{ij}\nu_{ij}M_i^r,$$

gde ν_{ij} predstavlja verovatnoću da će se pljačka realizovati ako se dogodi pokušaj pljačke i M_i^r je vrednost dobara u vozilu nakon što je napustilo čvor i na ruti r. U nekim primenama, adekvatne vrednosti za p_{ij} i ν_{ij} nisu poznate. Stoga se p_{ij} može zameniti sa rastojanjem c_{ij} između čvorova i i j, dok ν_{ij} se može postaviti na 1, kako je, radi jednostavnosti, pretpostavljeno u nastavku rada. Pored toga, s obzirom na to da je verovatnoća da se pljačka dogodi više od jednom tokom rute mala, može se pretpostaviti da se pljačka ne može desiti više puta tokom iste rute. Detaljnije o merenju rizika tokom rute može se pronaći u radu [181].

Konačno, s obzirom da vozila preuzimaju novac tokom rute i ostavljaju ih tek u skladištu na kraju rute, rizik tokom rute raste. To se može formulisati na sledeći način. Rizik se može izračunati za svaki čvor i gde vozilo dolazi iz čvora j tokom rute r koristeći sledeću rekurzivnu formulu:

$$R_j^r = R_i^r + M_i^r c_{ij}.$$

Specijalno, $R_0^r = 0$, s obzirom na to da je vozilo prazno kada kreće iz skladišta, pa je 0 nivo rizika jednak nuli.

Zbog ograničenja u raspolaganju svim potrebnim realnim podacima, ali i radi jednostavnosti, dve osnovne pretpostavke se podrazumevaju u nastavku:

1. verovatnoća napada (p_{ij}) je proporcionalna dužini puta (c_{ij}) koje vozilo prelazi,
2. verovatnoća (ν_{ij}) je konstantno jednaka jedinici za sve grane, pa se izostavlja.

U okviru ovog problema, ne uzima se u obzir cena vozila korišćenih u rešenju.

Pored toga, osnovna pretpostavka u opisu ovog problema je da su vozila na početku rute prazna i takva napuštaju glavno skladište. Vozila imaju zadatak da preuzmu novac tokom rute sa lokacija gde su korisnici (skup N) i na kraju rute taj prikupljeni novac isporuče u sigurnom sefu koji se nalazi u glavnom skladištu.

Da bi se izračunao rizik svake rute, indeks rizika R_i^r se računa za svaki čvor i u

ruti r, rekurzivno počevši od centralnog skladišta 0 koristeći formulu:

$$R_j^r = R_i^r + D_i^r c_{ij}, \tag{2.1}$$

gde je D_i^r količina novca u vozilu kada napušta čvor i tokom rute r.

Indeks rizika neke rute se akumulira (povećava) tokom putovanja vozila tom rutom. Globalni rizik rute r, označen sa GR^r, predstavlja rizik vozila na ruti r pred povratak u skladište. Ovo je u skladu sa realnim primenama:

- što je veća količina novca koji se prenosi, veći je rizik,
- što je duži put kojim se novac prenosi, veći je rizik.

U RCTVRP, globalni rizik svake rute je limitiran određenom maksimalnom vrednošću, koja se naziva *prag rizika* (engl. *risk threshold*) i označava sa T. Za svaki test primer parametar T je unapred zadat kao konstanta.

2.1.1 Matematička formulacija problema RCTVRP

Problem je definisan na usmerenom grafu $G = (V, A)$. U modelu je, zbog jednostavnosti, centralno skladište predstavljeno sa dva imaginarna čvora: s (engl. *start*) početni čvor od koga sva vozila kreću i e (engl. *end*) krajnji čvor gde se sve rute završavaju. Dakle, skup čvorova $V = \{s, e\} \cup N$ sadrži skladište i skup korisnika $N = \{1, \ldots, n\}$.

Svaki korisnik $i \in N$ ima nenegativanu vrednost parametra „ponuda" (engl. *demand*) d_i koja predstavlja vrednost (na primer količinu novca) koju vozilo prilikom posete treba da se preuzme od korisnika i. Imaginarni čvorovi imaju vrednost parametra ponuda jednaku nuli ($d_s = d_e = 0$). Skup svih grana je definisan kao

$$A = (N \times N) \cup (\{s\} \times N) \cup (N \times \{e\}).$$

Svakoj grani $(i, j) \in A$ je dodeljeno nenegativno rastojanje (ili dužina putovanja) c_{ij}.

Sva vozila polaze prazna iz skladišta ($d_s = 0$) i obave jednu rutu, obilazeći niz korisnika pre nego što se vrate u skladište (e) gde se prikupljeni novac isporučuje. Na samom početku rute, indeks rizika za svako vozilo je jednak nuli. Vozilo kada se

kreće između čvorova i i j povećava svoj indeks rizika za vrednost jednaku proizvodu dužine puta (ili vremena putovanja) c_{ij} i količine novca koju nosi pri izlasku iz čvora i.

U cilju definisanja ovog problema kao problema mešovitog celobrojnog programiranja (MIP), dve familije promenljivih su definisane za svaki čvor. Neka D_i^r označava količinu novca koju prenosi vozilo na ruti r kada napušta čvor i. Neka je R_i^r vrednosti indeksa rizika vozila na ruti r kada stiže u čvor i. Jednostavno se može uočiti da će, za datog korisnika, sve osim jedne od ovih vrednosti biti nule, s obzirom da se svaki korisnik obilazi tačno jednom.

Bulovska promenljiva x_{ij}^r je jednaka 1 ukoliko grana $(i, j) \in A$ pripada putu vozila na ruti r, a 0 inače.

Broj ruta se određuje u okviru optimizacionog problema i ne može biti veći od n što je slučaj u rešenju gde svaka ruta sadrži po tačno jedan čvor. Dakle, bez umanjenja opštosti, indeks rute r se može definisati na skupu korisnika N.

Binarne promenljive x_{ij}^r su definisana kao:

$$x_{ij}^r = \begin{cases} 1, & \text{ako grana}(i,j) \in A \text{ je sadržana u putanji vozila na ruti } r, \\ 0, & \text{inače.} \end{cases}$$

Cilj je odrediti rute tako da se minimizuje vrednost funkcije cilja, tj. ukupna dužina/cena pređenog puta.

U nastavku je naveden matematički model problema RCTVRP iz literature [181].

$$\min \sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r, \tag{2.2}$$

uz uslove:

$$\sum_{j\in N} x_{sj}^r = \sum_{i\in N} x_{ie}^r, \qquad \forall r \in N, \tag{2.3}$$

$$\sum_{j\in N} x_{sj}^1 = 1, \tag{2.4}$$

$$\sum_{i\in N} x_{ie}^r \geq \sum_{j\in N} x_{sj}^{r+1}, \qquad \forall r \in N \setminus \{n\}, \tag{2.5}$$

$$\sum_{r\in N} \sum_{j\in V\setminus\{s\}} x_{ij}^r = 1, \qquad \forall i \in N, \tag{2.6}$$

$$\sum_{h\in V\setminus\{e\}} x_{hj}^r - \sum_{k\in V\setminus\{s\}} x_{jk}^r = 0, \qquad \forall j \in N;\ \forall r \in N, \tag{2.7}$$

$$D_s^r = 0, \qquad \forall r \in N, \tag{2.8}$$

$$D_j^r \geq D_i^r + d_j - (1 - x_{ij}^r) \cdot M_1, \qquad \forall (i,j) \in A;\ \forall r \in N, \tag{2.9}$$

$$0 \leq D_i^r \leq M_1, \qquad \forall i \in V;\ \forall r \in N, \tag{2.10}$$

$$R_s^r = 0, \qquad \forall r \in N, \tag{2.11}$$

$$R_j^r \geq R_i^r + D_i^r \cdot c_{ij} - (1 - x_{ij}^r) \cdot M_2, \qquad \forall (i,j) \in A;\ \forall r \in N, \tag{2.12}$$

$$0 \leq R_i^r \leq T, \qquad \forall i \in V;\ \forall r \in N, \tag{2.13}$$

$$x_{ij}^r \in \{0,1\}, \qquad \forall (i,j) \in A;\ \forall r \in N. \tag{2.14}$$

Cilj je minimizovati ukupni pređeni put na svim rutama (izraz 2.2). Uslovi (2.3) nameću da svaka ruta počinje i završava u centralnom skladištu, dok je uslovom (2.4) omogućeno da prva ruta ($r = 1$) počinje od centralnog skladišta i da ruta numerisana sa $r = 1$ mora postojati. Uslovi (2.5) ne dozvoljavaju postojanje rute $r + 1$ ukoliko ruta r ne postoji, čime je obezbeđeno da su rute redom numerisane. Svaki korisnik mora biti posećen tačno jednom na osnovu uslova (2.6). Uslovi (2.7) definišu da na ruti r, vozilo može napustiti korisnika j samo ako je prethodno posetilo korisnika j. Uslovi (2.8)-(2.10) imaju ulogu da definišu vrednost ponuda koja se akumulira (povećava) kada se vozilo kreće od skladišta preko svakog korisnika i tokom rute r, gde je M_1 dovoljno veliki broj (dovoljno je $M_1 = \sum_{i\in V} d_i$). Uslovi (2.11)-(2.13) omogućavaju da globalni rizik svake rute r ne pređe zadati prag T, gde je M_2 dovoljno veliki broj (dovoljno je $M_2 = M_1 \cdot T \cdot \mathit{najvećeRastojanje}$, gde je $\mathit{najvećeRastojanje} = \max_{i,j\in V} c_{ij}$). S obzirom da se vrednosti D_i^r i R_i^r povećavaju

tokom svake rute r, podrute su automatski onemogućene.

Pored toga, u uslovima (2.12) vrednost c_{ij} se može zameniti verovatnoćom p_{ij} da se pljačka dogodi na putu (i, j), ukoliko su takvi podaci dostupni.

2.1.2 Test primeri za problem RCTVRP

Za tesiranja za problem RCTVRP korišćena su sledeća četiri skupa test primera:

- *„Skup R"* sadrži 180 test primera generisane na slučajan način, kao što je opisano u radu [181]. Devet osnovnih test primera sa različitim brojem čvorova (4, 6, 8, 10, 12, 14, 16, 18 i 20) su generisani tako da su koordinate slučajno odabrane iz intervala $[-20, 20]$. Svaki od osnovnih test primera je kombinovan sa pet različitih nivoa praga rizika i četiri različite vrednosti standardne devijacije korišćenje za dobijanje vrednosti d_i svakog čvora i. Prva vrednost za prag rizika je definisana kao $RL1 = \max_{i \in N}\{d_i c_{i0}\}$, dok su ostale vrednost ($RL1.5$, $RL2$, $RL2.5$ i $RL3$) dobijene pomoću formule $RL\alpha = \alpha \dot{R}L1$, gde je $\alpha \in \{1.5, 2, 2.5, 3\}$. Vrednosti d_i dodeljene svakom čvoru su generisane tako da standardna devijacija σ_d za niz vrednosti d_i je jednaka 1, 4, 16 ili 64.

- *„Skup V"*, takođe korišćen u pomenutom radu [181], sastoji se iz 70 test primera specijalno konstruisanih na osnovu 14 test primera iz poznatog skupa test primera za problem rutiranja vozila *VRPLIB*[1] sa brojem čvorova od 22 pa do 301 (22, 26, 30, 36, 45, 51, 72, 101, 121, 135, 151, 200, 256 i 301). Ove osnovne instance su kombinovane sa 5 različitih nivoa rizika, na isti način kao i za prethodni skup test primera (skup R).

- *„Skup O"* i *„skup S"* čine 64 test primera (svaki po 32) sa brojem čvorova od 10 do 337. Autori rada [182] kreirali su ove instance tako da optimalna rešenja budu poznata unapred radi testiranja. U skupu O čvorovi su raspoređeni tako da su poravnati duž pravih linija i formiraju određeni broj oblika koji odgovaraju rutama optimalnog rešenja tako da se dve zamišljene linije koji počinju iz centra (centralno skladište) spajaju čvorove sve do krajnjeg dodatnog čvora koji se nalazi na vrhu obeliska. Sa druge strane, u skupu S čvorovi su locirani

[1] http://or.dei.unibo.it/library/vrplib-vehicle-routing-problem-library

na određenom broju spiralnih oblika koje kreću od centralnog skladišta. Pragovi rizika za ove instance su definisani tako da optimalno rešenje ima određeni broj čvorova u svakoj ruti. Optimalna rešenja za instance iz skupova O i S imaju onoliko ruta koliko je „latica" (3, 4, 6, 8 i 16). Vrednosti d_i su celobrojne i specijalno dodeljene svakom čvoru tako da odgovaraju rednom broju tog čvora u ruti optimalnog rešenja. Preciznije, $d_a = 1$ za svaki čvor a koji će biti prvi posećen u ruti, drugom se dodeljuje vrednost 2, trećem 3, itd.

3

Problemi lokacije resursa

Jednu od najznačajnijih i često proučavanih klasa problema matematičke optimizacije predstavljaju lokacijski problemi. Ovi problemi se odnose na određivanje lokacije objekata tako da se minimizuje ili maksimizuje zadata funkcija cilja. Matematičke formulacije lokacijskih problema najčešće i nastaju iz potrebe za rešavanjem konkretnih praktičnih problema, pa je tako čest slučaj da u prostoru u kome postavljaju objekti već postoje neki drugi relevantni objekti.

Klasa lokacijskih problema se može podeliti na različite načine. Na primer, najčešća je podela na kontinualne, diskretne i mrežne lokacijske probleme. Kod kontinualnih, lokacije se određuju u kontinualnom prostoru, dok je u slučaju diskretnih lokacijskih problema potrebno rasporediti objekte na izabrane lokacije iz zadatog konačnog skupa. U slučaju mrežnih lokacijskih problema zadata je odgovarajuća matematička struktura (težinski graf ili mreža) i potrebno je odabrati čvorove mreže (diskretni mrežni problemi) ili tačke na granama mreže (kontinualni mrežni problemi). Većina diskretnih lokacijskih problema pripada klasi NP-teških problema, pa se osim egzaktnih algoritama razvijaju i heuristike za rešavanje ovih problema. U okviru ove disertacije, razmatrana su dva diskretna lokacijska problema, pa je u nastavku pažnja usmerena ka diskretnim lokacijskim problemima.

Postoje različite podgrupe lokacijskih problema [44]. Na primer, u situacijama kada je uspostavljena hijerarhija između lokacija reč je o hijerarhijsko-lokacijskim problemima [129, 132]. Objekti kojima je potrebno odrediti položaje su uglavnom centri koji pružaju usluge, snabdevači. Često se u prostoru već nalaze drugi objekti, najčešće klijenti koji predstavljaju korisnike koji će biti dodeljeni uspostavljenim

uslužnim centrima. Problemi kod kojih je pored lociranja objekata (habova), potrebno i izvršiti pridruživanje ne-hab objekata (korisnika) uspostavljenim habovima se nazivaju hab lokacijski problemi i često su proučavani [170, 134, 133, 142].

Praktične primene lokacijskih problema su mnogobrojne. Problemi iz ove klase se javljaju prilikom planiranja infrastrukture za mnogobrojne potrebe, kao što su mreže različitog tipa uključujući i telekomunikacione i računarske. Različiti modeli lokacijskiih problema razvijani su na primer za određivanje lokacije i izgradnju industrijskih postrojenja, bolnica, banaka, prodajnih objekata, itd.

Jedan od poznatijih lokacijskih problema je problem p-medijane (engl. *p-Median Problem* - PMP). U slučaju ovog problema, dat je skup D korisnika od kojih svaki korisnik $i \in D$ ima svoj zahtev d_i, kao i skup F potencijalnih lokacija za objekte koje treba uspostaviti. Poznate su cene transporta c_{ij} od svakog objekta $j \in F$ do svakog korisnika $i \in D$, kao i broj p. Potrebno je odrediti podskup $F' \subseteq F$, takav da je $|F'| = p$, kako bi se na tim lokacijama otvorili objekti i svaki korisnik se pridružuje nekom uspostavljenom objektu, tako da su minizovani troškovi transporta d_i jedinica do svakog od korisnika $i \in D$ od objekata iz F' koji je pridružen tom korisniku i. Kako nisu nametnuta ograničenja kapaciteta, svaki korisnik će biti pridružen najbližem uspostavljenom objektu. Primeri primene za PMP su mnogobrojne [184, 128, 162], a predlagane su različite metode za rešavanje ovog problema [93, 162]. PMP se koristi u situacijama kada je unapred poznat broj objekata koje treba otvoriti. Takva situacija je i u dva lokacijska problema koja su razmatrana u okviru ove disertacije, a koji se ipak razlikuju od problema PMP.

Javljaju se i različita proširenja lokacijskih problema, kao na primer sa kombinacija sa problemom rutiranja (engl. *facility location with vehicle routing*) [146], sa upravljanjem inventarom (engl. *facility location with inventory management*) [167], itd. Jedna od varijanti lokacijskih problema jeste i pouzdani lokacijski problem (engl. *reliable facility location*) [168], gde se mora podrazumevati da uspostavljeni objekti mogu biti nekad i van funkcije i da se u tim situacijama korisnici koji su im bili dodeljeni preusmere na druge objekte koji su u funkciji.

Različiti pristupi rešavanju lokacijskih problema su se javljali u literaturi. S obzirom da su gotovo svi lokacijski problemi NP-teški, upotreba egzaktnih pristupa je ograničena[13]. Pored specijalizovanih heurističkih pristupa, u poslednje vreme se posebna pažnja posvećuje primeni metaheuristika za rešavanje različitih lokacijskih problema. Takođe javljali su se i hibridni pristupi, kao što je na primer hibridna

heuristika koja kombinuje više različitih tradicionalnih metaheuristika i koja je predložena za rešavanje problema PMP [162].

Problemi lokacije resursa sa merom ravnomernosti

Prilikom odabira lokacija potrebno je ispoštovati različite uslove. Modeli problema lokacije resursa koji uključuju neku meru ravnomernosti (ujednačenosti, uravnoteženosti, pravednosti) razmatrani su u literaturi zbog mnogobrojnih primena. Ravnomernost se najčešće određuje prema cenama, vremenu trajanja putovanja ili rastojanjima, a u odnosu na skup korisnika ili snabdevača.

Definisanje mere ravnomernosti kod lokacijskih problema razmatrano je u različitim radovima, kao na primer [136]. U radu [92] dat je uvod u merenje ravnomernosti kod lokacije resursa. Autori su predložili dve mere: varijansa rastojanja između korisnika i snabdevača i Lorencova kriva. Za mrežni lokacijski problem u radu [126] uvedena je mera nazvana „maksimalna apsolutna devijacija" (engl. *maximum absolute deviation*). Algoritmi za rešavanje različitih mrežnih lokacijskih problema sa merama ravnomernosti izloženi su u radu [139].

Aspekt ravnomernosti u lokacijskih problema ima važne primene u javnom sektoru. Na primer, funkcija cilja može da meri neke nejednakosti u disperziji raspodela rastojanja između korisnika i snabdevača i cilj je minimizovati tu nejednakost. Takav princip se može primeniti i na raspoređivanje nepoželjnih ili opasnih objekata (videti [50]).

U okviru ove disertacije razmatrana su dva lokacijska problema. Prvi lokacijski problem se odnosi na ravnopravnost snabdevača i naziva se problem ravnomernog opterećenja, dok je drugi je problem maksimizacije minimalnog rastojanja. Ova dva problema su detaljno opisana u nastavku ovog poglavlja, a više o lokacijskim problemima se može naći u preglednoj literaturi [44, 5, 13].

3.1 Problem ravnomernog opterećenja

Problem ravnomernog opterećenja (engl. *the Load Balance problem* - LOBA) predstavio je A. Marín u radu [135]. Izložene su dve formulacije ovog problema celobrojnog programiranja i razvijeno je nekoliko familija validnih nejednakosti i tehnika preprocesiranja, koje omogućavaju da se smanji dimenzija problema (broj promen-

ljivih i uslova). U istom radu, autor je predložio algoritam grananja i sečenja (engl. *Branch-and-Cut algorithm* - BnC) sa poboljšanjima za rešavanje problema LOBA. Obe formulacije su testirane na instancama sa do 50 potencijalnih snabdevača i 100 korisnika. U slučaju instanci najvećih dimenzija, optimalna rešenja prikazana u razmatranom radu [135] dobijena su okviru jednog sata.

Kasnije su se u literaturi razmatrali slični problemi, pa se tako pojavio skup ograničenja koja se mogu koristi u slučaju problema kod kojih se korisnici pridružuju najbližem alociranom snabdevaču [56]. Još jedan u nizu problema koji se bave ravnomernošću je problem u kome je cilj maksimizacija minimalnog rastojanja između dva uzastopna prolaska kroz objekat [11].

LOBA predstavlja diskretni lokacijski problem kod kojeg je potrebno rasporediti snabdevače tako da budu ravnomerno opterećeni od strane pridruženih korisnika. Kod problema LOBA, dat je prirodan broj p, skup korisnika i skup potencijalnih snabdevača (npr. fabrike, postrojenja). Poznate su i razdaljine (cena transporta) između svakog korisnika i svakog snabdevača. Pretpostavka je da se svaki korisnik pridružuje snabdevaču koji mu je najbliži. Cilj je odrediti skup od tačno p snabdevača koji će biti uspostavljeni tako da je minimizovana razlika između najvećeg i najmanjeg broja korisnika dodeljenih nekom uspostavljenom snabdevaču. Ovaj problem se često javlja u praksi, na primer, prilikom raspoređivanja antena za mobilne telefone, škola, izbornih jedinica ili centara za prikupljanje čvrstog otpada.

Prva formulacija prezentovana u radu [135] nazvana je standardnom, s obzirom na to da je izvedena na osnovu standardne formulacije problema p-medijane dodavanjem dve nove promenljive: maksimalni i minimalni broj korisnika dodeljenih nekom snabdevaču kao i ograničenja da se korisnik pridružuje najbližem snabdevaču. Druga formulacija problema LOBA predložena u pomenutom radu [135] deli osnovnu strukturu sa nekim formulacijama diskretno uređenog problema medijane (engl. *The discrete ordered median problem* - DOMP) [174]. U nastavku je predstavljena standardna formulacija problema LOBA.

3.1.1 Matematička formulacija problema LOBA

Jedna od formulacija koja predložena u radu [135] korišćena je i u ovoj disertaciji. Neka je $A = \{1, ..., n\}$ skup koji se koristi da predstavi korisnike (potrošače) i neka je $B = \{1, ..., m\}$ skup koji predstavlja potencijalne snabdevače (npr. fabrike,

postrojenja). Neka je $C = (c_{ij}), i \in A, j \in B$, matrica troškova (cena transporta, razdaljina ili slično). Nisu nametnuti dodatni uslovi za matricu C tako da elementi matrice mogu biti i negativni brojevi, ne moraju da zadovolje nejednakost trougla ili simetriju.

Rešenje problema LOBA dato je kao skup snabdevača $X(X \subseteq B), |X| = p$. Svaki korisnik $i \in A$ će biti snabdeven od strane uspostavljenog snabdevača $j \in X$ koji mu je najbliži tj. takvog da je $c_{ij} = \min_{k \in X} c_{ik}$. U slučaju nerešenog rezultata korisnik može biti dodeljen bilo kojem od njemu najbližih snabdevača.

Zadatak je minimizovati razliku između najvećeg i najmanjeg broja korisnika dodeljenih bilo kom snabdevaču iz X.

Za matematičku formulaciju problema ravnomernog opterećenja, koju koristimo u radu, potrebno je uvesti promenljive u i l, gde u predstavlja najveći broj korisnika dodeljenih nekom uspostavljenom snabdevaču, a l minimum broja korisnika dodeljenih nekom uspostavljenom snabdevaču. Pored toga, u formulaciji se koriste dva skupa binarnih promenljivih y_j, $j \in B$ za lociranje snabdevača i x_{ij},$i \in A, j \in B$ za pridruživanje korisnika uspostavljenim snabdevačima.

Binarne promenljive koje se koriste su definisane na sledeći način:

$$y_j = \begin{cases} 1, & \text{ako je snabdevač } j \text{ uspostavljen,} \\ 0, & \text{inače,} \end{cases}$$

za sve $j \in B$,

$$x_{ij} = \begin{cases} 1, & \text{ako je korisnik } i \text{ dodeljen snabdevaču } j, \\ 0, & \text{inače,} \end{cases}$$

za sve $i \in A, j \in B$.

Dodatno, neka su definisane konstante $M_i = \max_{j \in B} c_{ij}$, za svako $i \in A$.

Imajući u vidu navedenu notaciju, problem LOBA može biti formulisan na sledeći način [135]:

$$\min(u - l), \tag{3.1}$$

uz uslove:

$$\sum_{j=1}^{m} y_j = p, \tag{3.2}$$

$$x_{ij} \leq y_j, \qquad \forall i \in A; \forall j \in B, \tag{3.3}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad \forall i \in A, \tag{3.4}$$

$$u \geq \sum_{i=1}^{n} x_{ij}, \qquad \forall j \in B, \tag{3.5}$$

$$l \leq \sum_{i=1}^{n} x_{ij} + n(1 - y_j), \qquad \forall j \in B, \tag{3.6}$$

$$\sum_{k \in B} c_{ik} x_{ik} + (M_i - c_{ij}) y_j \leq M_i, \qquad \forall i \in A; \forall j \in B, \tag{3.7}$$

$$y_j \in \{0, 1\}, \qquad \forall j \in B, \tag{3.8}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in A; \forall j \in B. \tag{3.9}$$

Cilj je minimizovati razliku između u i l (3.1). Uslov (3.2) omogućava da je broj uspostavljenih snabdevača tačno p. Uslovi (3.3) i (3.4) obezbeđuju da se korisnici mogu dodeliti tačno jednom i to već uspostavljenom snabdevaču. Ograničenja (3.5) - (3.6) predstavljaju gornju i donju granicu za promenljive u i l, redom. Svaki korisnik je dodeljen svom najbližem snabdevaču, s obzirom na ograničenje (3.7). Na kraju, (3.8) i (3.9) ukazuju na binarnu prirodu promenljivih y_j i x_{ij}.

3.1.2 Test primeri za problem LOBA

Za tesiranja za problem LOBA korišćena su sledeća četiri skupa test primera:

- Test primeri specijalno konstruisani za problem LOBA i predstavljeni u radu [135]. Ovaj skup test primera, nazvan *„Datos"*, je tako generisan da elementi u matrici troškova su postavljeni na $c_{ij} = j$, $i \in A$, $j \in B$. U cilju dobijanja test primera različitih težina za iste A, B i cene alokacije C, autor je koristio parametar pl (engl. *perturbation level*). Pedeset osam test primera označenih sa „m-n-p-pl" su dobijeni kombinovanjem različitih vrednosti za $m(20, 30, 50)$, $n(20, 30, 50, 100)$, $p(3, 4, 6, 10)$ i $pl(10, 30, 50, 100, 200, 400)$.

- Skup test primera *„Pmed"* su dobijeni na osnovu skupa test primera *pmed*10 iz poznate biblioteke test instanci *OR-Library*[1]. Ovaj skup instanci je kao i prethodni korišćen u radu [135]. Od inicijalnog skupa od 100 tačaka, kreirani su podskupovi od $20, 30, ..., 100$ tačaka, koji predstavljaju i korisnike i potencijalne snabdevače ($n = m$). Zaokružena Euklidska rastojanja između tačaka su korišćena za matricu troškova, a vrednosti za $p = 3, 5, 7$.

- *„Galvao"* test primeri G1-G8 su generisani na slučajan način u [75]. Skup korisnika u skupu *Galvao* je jedak skupu potencijalnih snabdevača $A = B$ (i $m = n$). Test primeri G1-G4 sadrže 100 čvorova, dok G5-G8 imaju po 150 čvorova, a vrednosti za p su postavljene na 5, 10 i 15.

- Skup od šest instanci koje su kreirani za problem p-medijane sa kapacitetima [127]. Ove test instance odgovaraju stvarnim podacima iz jednog grada u Brazilu (*São José dos Campos*) i javno su dostupni[2]. U ovim instancama, nazvanim *„SCJ"*, skup korisnika A se poklapa sa skupom potencijalnih snabdevača B, pa važi $m = n$. Veličine (n, p) u $SCJ1 - SCJ4$ su $(100, 10)$, $(200, 15)$, $(300, 25)$, $(300, 30)$, $(402, 30)$ i $(402, 40)$, redom.

3.2 Problem maksimizacije minimalnog rastojanja

Problem maksimizacije minimalnog rastojanja (engl. *Max-Min Diversity Problem* - MMDP) se sastoji od odabira podskupa elemenata datog skupa na način da najmanje rastojanje među odabranim elementima bude maksimizovano. Problem je NP-težak i može se formulisati kao problem celobrojnog linearnog programiranja [159]. Maksimizovanje raznolikosti neke kolekcije tačaka javljala se u više varijanti [159]

Od 1980-tih, nekoliko metoda za rešavanje ovog problema su razvijene i primenjene u različitim oblastima, kao što su na primer socijalne i biološke nauke. U radu [159] predstavljen je heuristički metod za pronalaženje aproksimativnog rešenja, koji se, za ovaj problem, pokazao bolje nego prethodno korišćene metaheuristike (tabu pretraživanje i simulirano kaljenje).

[1] http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[2] http://www.lac.inpe.br/~lorena/instancias.html

Problem MMDP je rešavan i veoma zanimljivim heurističkim pristupom, rešavanjem problema maksimalne klike [33]. Ovaj pristup se, na osnovu eksperimentalnih rezultata, pokazao veoma uspešnim za rešavanje problema MMDP. Postoje različiti algoritmi za rešavanje problema maksimalne klike [192].

3.2.1 Matematička formulacija problema MMDP

U slučaju problema MMDP potrebno je odrediti podskup M od m elemenata, ($|M| = m$) od skupa N od n elemenata tako da je minimalno rastojanje između odabranih elemenata maksimizovano. Definicija rastojanja između elemenata se može prilagoditi različitim primenama [112]. MMDP potiče iz realnih situacija kada je potrebno rasporediti postrojenja, a ima primene i u društvenim i biološkim naukama (na primer u cilju očuvanja ekologije) [86]. U većini ovih primena, podrazumeva se da se svaki element može predstaviti skupom atributa. Neka je s_{ik} stanje ili vrednost k-og atributa elementa i, gde je $k = 1, \ldots, K$. Rastojanje između elemenata i i j se može definisati na sledeći način:

$$d_{ij} = \sqrt{\sum_{k=1}^{K} (s_{ik} - s_{jk})^2}.$$

U ovom slučaju d_{ij} je Euklidsko rastojanje između i i j.

Koristeći definisana rastojanja d_{ij} MMDP se formuliše kao problem binarnog celobrojnog kvadratnog programiranja, gde za $i = 1, \ldots, n$, promenljiva x_i ima vrednost 1 ako je element i odabran, a 0 u suprotnom. U nastavku je dat ovaj jednostavan model za problem MMDP.

$$\max Z_{MM}(x), \text{ gde je } Z_{MM}(x) = \min_{i<j} d_{ij} x_i x_j \tag{3.10}$$

uz uslove:

$$\sum_{i=1}^{n} x_i = m, \tag{3.11}$$

$$x_i \in \{0, 1\}, \qquad i = 1, \ldots, n. \tag{3.12}$$

Uslov (3.11) se odnosi na to da je potrebno da se tačno m elemenata odabere, a

uslov (3.12) nameće da promenljive x_i moraju biti binarne. Za razliku od prethodna dva problema, u slučaju problema MMDP potrebno je maksimizovati funkciju cilja.

Da je MMDP NP-težak problem nezavisno su pokazali Erkut u radu [54] i Ghosh u radu [79]. Kako se navodi u radu [159] i za egzaktne i za heurističke metode teže je rešiti MMDP nego klasičan problem maksimizacije raznolikosti (engl. *Maximum Diversity Problem* - MDP) [3]. U slučaju problema MDP potrebno je odrediti podskup M od m elemenata skupa N tako da je maksimizovan zbir rastojanja d_{ij} za sve $i, j \in M, i \neq j$.

3.2.2 Test primeri za problem MMDP

U ovoj disertaciji eksperimentalna testiranja za problem MMDP urađena su na narednim skupovima test primera, koji su korišćeni i u radu Resendea [159].

- Skup test primera *„Glover"* se sastoji od 75 test primera. Vrednosti elemenata matrica su računaju kao Euklidska rastojanja, a tačke su slučajno generisane sa koordinatama u opsegu od 0 do 100. Broj koordinata za svaku tačku je takođe slučajno generisan između 2 i 21. Glover je 1998. godine razvio generator test problema i konstruisao instance gde je $n = 10$, $n = 15$ i $n = 30$. Vrednosti za m su između $0.2n$ i $0.8n$.

- Skup test primera *„Geo"* je predstavljen u okviru rada [159] i sastoji se od 60 matrica konstruisanih istim generatorom kao i za skup instanci Glover, sa tim što su korišćene vrednosti za n iz skupa $\{100, 250, 500\}$. Za svaku vrednost n korišćene su po dve vrednosti za m: $m = 0.1n$ i $m = 0.3n$ i generisano je po deset test primera. Ove instance su slične geometrijskim instancama predstavljenim u radu [54].

- Skup *„Ran"* sadrži 60 test primera u kojima su elementi matrica slučajno generisani pomoću generatora uvedenog u radu [169]. Generisano je dvadeset instanci gde je n iz skupa $\{100, 250, 500\}$ i za svaku vrednost n posmatrano je $m = 0.1n$ i $m = 0.3n$. Slučajni celi brojevi su generisani između 50 i 100 u svim instancama osim u slučaju instance za koju su $n = 500$ i $m = 150$ i u kojoj su generisani između 1 i 200.

4

Rešavanje problema rutiranja vozila

U ovom poglavlju će biti prikazane dve metode koje su, u okviru ove disertacije, predložene za rešavanje problema RCTVRP, kao i rezultati koji su njihovom primenom dobijeni. Prva predložena metoda je metaheuristika GRASP, a druga metoda ponovnog povezivanja puta. Opisana je implementacija hibridizacije ove dve metode za rešavanje RCTVRP problema.

U skorije vreme za rešavanje NP-teških optimizacionih problema često se predlažu hibridizacije različitih metoda. GRASP sa metodom ponovnog povezivanja puta je uspešno primenjivan na različite optimizacione probleme kao što su raspoređivanje poslova [2], maksimizacija minimalnog rastojanja [159] i mnogi drugi (e.g. [63, 62, 68, 100]). Uspešna primena ovog hibrida je poslužila kao inspiracija da se ovakav pristup testira i na problemu RCTVRP, dok je priroda ovog problema dovela do fazi modifikacije prikazane u sekciji 4.2.

4.1 Metoda GRASP

Nasumično pohlepna adaptivna pretraga (engl. *Greedy Randomized Adaptive Search Procedure* - GRASP) predstavlja metaheurističku metodu koja se često primenjuje na probleme kombinatorne optimizacije. GRASP su uveli Feo i Resende 1989. godine [59] kao probabilističku heuristiku za rešavanje problema pokrivenosti skupa (engl. *set covering problem*). GRASP je, već ubrzo nakon toga, prepoznat kao metaheuristička metoda opšte namene. Ova metoda je primenjivana na različite probleme koji su se javljali u nauci i industriji, uključujući neke tipove probleme pra-

vljenja rasporeda [61], u teoriji grafova [160], neke vrste problema rutiranja vozila [27], različite lokacijske probleme [36], itd.

Osnovna ideja GRASP metode je da se unapred definisan broj iteracija ponavljaju sledeći koraci:

- konstrukcija rešenja korišćenjem pohlepne slučajne metode,
- unapređenje konstruisanog rešenja primenom lokalne pretrage i
- ažuriranje najboljeg pronađenog rešenja.

Konstrukcija rešenja odnosno prva faza GRASP metode se zapravo sastoji iz tri dela:

1. pohlepnog algoritma,
2. adaptivne funkcije,
3. probabilističkog izbora.

Pohlepne heuristike u svakom koraku koriste izbor koji je najbolji u datom trenutku, praveći lokalno optimalan izbor u nadi da će takav postupak voditi ka globalnom optimumu. Pohlepne heuristike u nekim slučajevima (primer: minimalno povezujuće stablo) zaista daju optimalno rešenje, ali često to nije slučaj.

Konstruktivni algoritmi najčešće podrazumevaju da se u svakom koraku dodaje po jedan element trenutnom parcijalno konstruisanom rešenju. U slučaju konstrukcije rešenja pohlepnim algoritmom, odabir sledećeg elementa je često deterministički. Sa druge strane, u slučaju probabilističkog izbora, sledeći element se bira iz liste kandidata (engl. *Restricted Candidate List* - RCL).

Lokalna pretraga (engl. *Local search* - LS) je jedna od najviše korišćenih heurističkih metoda za rešavanje teških optimizacionih problema. Osnovna ideja LS metode je da se počevši od inicijalnog tj. početnog rešenja x, menja x sa boljim rešenjem x' iz njegove okoline $N(x)$ sve do trenutka kada ne postoji bolje rešenje u toj okolini. Okolina $N(x)$ predstavlja skup rešenja koja se mogu dobiti malim izmenama rešenja x. Adekvatno definisanje okolina za dati problem je važno za uspešnu primenu lokalne pretrage.

GRASP predstavlja iterativni postupak ponovnog ponavljanja koraka: faze konstrukcije i faze lokalne pretrage. Prvo se u svakoj iteraciji, konstruiše se početno

rešenje primenom nasumično pohlepne adaptivne heuristike (engl. *greedy randomized adaptive heuristic*). Poželjno je da konstruisano početno rešenje bude dopustivo. Zatim se lokalna pretraga primeni na konstruisano početno rešenje. Ukoliko je rešenje u trenutnoj iteraciji bolje od trenutnog najboljeg rešenja (engl. *Incumbent Solution*), najbolje rešenje se ažurira. Ove dve faze se ponavljaju dok ne bude zadovoljen neki od kriterijuma zaustavljanja, kada algoritam kao rešenje vraća najbolji lokalni optimum. Dakle GRASP se može posmatrati i kao hibrid polupohlepne procedure konstrukcije rešenja i metoda lokalne pretrage. Algoritam 1 prikazuje opisanu osnovnu strukturu GRASP metode.

Algoritam 1 Osnovni GRASP pseudokod

```
1:  Procedura GRASP
2:      NajboljeRešenje ← Beskonačno;
3:      while not KriterijumZaustavljanjaGRASP() do
4:          Rešenje ← KonstrusatiPohlepnoSlučajnoRešenje();
5:          NajboljeRešenje ← LokalnaPretraga(Rešenje);
6:          if Rešenje bolje nego NajboljeRešenje then
7:              NajboljeRešenje ← Rešenje;
8:      return NajboljeRešenje

9:  Procedura KonstrusatiPohlepnoSlučajnoRešenje
10:     Rešenje ← {}
11:     while not ZavrsenaKonstrukcijaRešenja() do
12:         RCL ← KonstrusatiRCL();
13:         s ← SlučajnoOdabran(RCL);
14:         Rešenje ← Rešenje ∪{s}
15:     return Rešenje
```

Više o GRASP metodi može se naći u radovima kao što su [60, 64], a u literaturi [65, 66, 67, 161] su oprisane i neke uspešne primene pri rešavanju NP-teških problema. U praksi se u poslednje vreme pokazalo korisno hibridizovati GRASP metodu sa nekim drugim metodama u cilju pronalaženja boljih rešenja, a u okviru ovog poglavlja biće GRASP je hibridizovan sa metodom ponovnog povezivanja puta.

4.2 Metoda GRASP za rešavanje RCTVRP

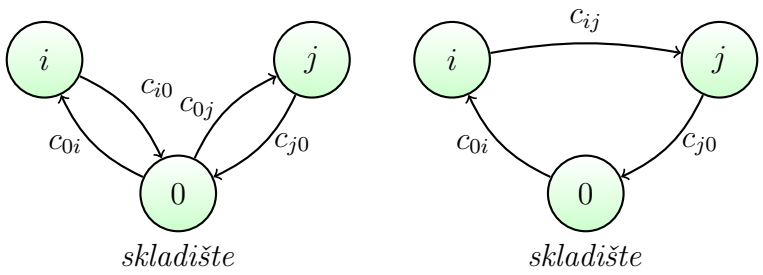
U ovoj podsekciji opisana je GRASP za RCTVRP. U ovoj implementaciji GRASP metode, rešenje problema RCTVRP je predstavljeno kao lista ruta, gde svaka ruta sadrži uređenu listu čvorova. Dve faze GRASP metode su prilagođene problemu RCTVRP na način opisan u nastavku.

Konstruktivna faza za RCTVRP

Osnovna ideja koja stoji iza implementirane konstruktivne faze za RCTVRP zasniva se na konstruktivnoj heuristici za rešavanje problema rutiranja vozila koja se naziva Klark-Vrajtov algoritam ušteda (engl. *Clarke Wright savings algorithm* - CW) [23].

Kod klasičnog CW algoritma kreće se od rešenja u kome svakog korisnika uslužuje po jedno vozilo (tzv. zvezdastog rešenja) i iterativno se spajaju po dve rute. Može se primetiti da u slučaju Euklidskih rastojanja, zvezdasto rešenje ima najveću vrednost ukupnog pređenog puta, pa spajanje ruta dovodi do smanjivanja ukupnog pređenog puta tj. do manjih vrednosti funkcije cilja. Ideja je da se unapred izračunaju uštede koje bi se postigle spajanjem svaka dva para korisnika. Algoritam se zasniva na uštedi troškova koja se dobija spajanjem dve rute u jednu kao što je na primer prikazano na slici 4.1, gde čvor 0 predstavlja skladište. Inicijalno, u levom delu slike, korisnici i i j su u dve odvojene rute, a moguće spajanje je da ova dva korisnika budu u istoj ruti. Na primer spajanje koje je predstavljeno na desnoj strani slike dovodi do uštede $c_{ij} - c_{i0} - c_{0j}$. Pravi se lista ušteda za svaki od parova korisnika, tako što se izračuna: $s_{ij} = c_{i0} + c_{j0} - c_{ij}$, za svako $i, j \in N$, $i \neq j$. Potom se u okviru CW algoritma lista sortira opadajuće na osnovu ušteda, a zatim u svakoj iteraciji heurisitka pohlepno pokušava da spoji par čvorova sa najvećom uštedom. Spajanje je moguće ukoliko su oba čvora povezana sa skladištem i ukoliko su zadovoljena druga ograničenja koja zavise od konkretnog VRP problema koji se rešava (na primer suma zahteva ne sme preći kapacitet vozila).

U cilju implementiranja prve faze GRASP metode za problem RCTVRP, originalna CW metoda je prilagođena pomoću tri izmene, slično kao i u radu [181]. Što se ograničenja tiče, u slučaju RCTVRP problema, ne uzima se u obzir kapacitet vozila, ali postoje ograničenja rizika. Stoga je umesto provere kapaciteta korišćena provera indeksa rizika u cilju kreiranja dopustivih rešenja. Pored toga, da bi se CW heu-



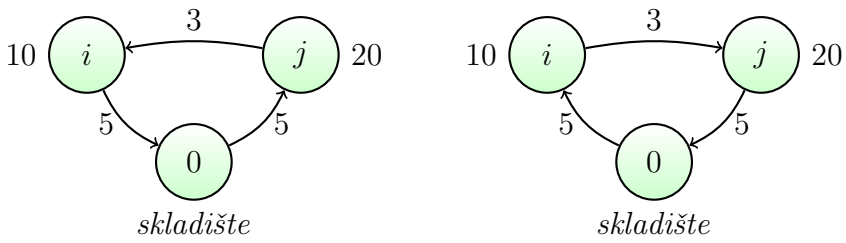
Slika 4.1: *Koncept ušteda - osnovna ideja CW algoritma*

ristika prilagodila tako da bude iskorišćena kao prva faza GRASP metode, umesto odabira najboljeg poboljšanja, spajanje se slučajno bira iz ograničene liste kandidata (engl. *restricted candidate list* - RCL). Pri tome, RCL se konstruiše tako da sadrži moguća poboljšanja/spajanja iz sortirane liste poboljšanja koji su najviše G_α puta lošiji od vrednosti uštede najboljeg moguće poboljšanja. Dodatno, za svaku rutu r, razmatra se i ruta r' dobijena obrtanjem redosleda čvorova u ruti r. Prilagođena CW heuristika je efikasno implementirana korišćenjem strukture podataka objašnjene u delu 4.5.

Fazi modifikacija pri konstrukciji rešenja

Pohlepni i polupohlepni algoritmi se gotovo uvek implementiraju u skladu sa funkcijom cilja. Međutim, u slučaju problema sa jakim ograničenjima koja nisu uključena u funkciju cilja, kao što je to slučaj sa RCTVRP, predlog je naredna fazi modifikacija. Umesto jednostavnog poređenja poboljšanja vrednosti funkcije cilja, u nekim problemima može biti od koristi da se u obzir uzme i koliko dobro su zadovoljena ograničenja. Kako je već navedeno u poglavlju 1.6, fazi logika predstavlja veoma koristan matematički alat za predstavljanja stepena pripadanja objekta nekom skupu [110, 163]. Stoga se navedeni stepen zadovoljenja određenog ograničenja može predstaviti korišćenjem fazi logike.

U cilju objašnjenja osnovne ideje koja stoji iza fazi modifikacije prilikom konstrukcije rešenja, sledi jedan jednostavan motivacioni primer predstavljen na slici 4.2. Neusmerenom grafu na slici težine grana predstavljaju razdaljine između čvorova. Skladište je čvor 0, dva korisnika i i j su predstavljana čvorovima, a količina novca koja treba od njih da se preuzme prikazana je levo od čvora i, a desno od čvora j. Kao što se može videti sve različite rute mogu imati iste vrednosti pređenog puta i



Slika 4.2: *Motivacija za fazi modifikaciju pri konstrukciji rešenja*

različite indekse rizika. Na slici 4.2, obe rute imaju isti ukupan pređeni put 13, dok je indeks rizika rute sa leve strane 210, dok je indeks rizika rute na desnoj strani 180. Stoga se ove dve rute smatraju istog kvaliteta u toku konstrukcije rešenja prethodno opisanim postupkom baziranim na CW heuristici. Međutim, ruta sa manjim indeksom rizika može biti korisna u narednim koracima spajanja ruta u toku konstrukcije rešenja. Ovo zapažanje se može generalizovati.

Prethodno opisani CW algoritam na kome je baziran proces konstrukcije rešenja je uveden za klasičan problem rutiranja vozila, pa se u okviru ove disertacije predlaže da se u slučaju RCTVRP indeksi rizika uzmu u obzir. Može se primetiti da što je manji indeks rizika potencijalne spojene rute, ta ruta je „bolja", tj. verovatnije je da će biti spojena u narednim iteracijama u toku konstrukcije rešenja. Tako da je za računanje kvaliteta parcijalnih rešenja, umesto korišćenja funkcije cilja korišćena sledeća mera:

$$kvalitet\_rešenja = \ pređeni\_put - \sum_{\forall\ ruta\ r} \mu(GR^r),$$

gde je GR^r globalni rizik na ruti r, a μ je funkcija pripadanja definisana na sledeći način:

$$\mu(x) = \begin{cases} 1, & \text{ako } x < t, \\ \frac{T-x}{t-T}, & \text{ako } t \leq x < T, \\ 0, & \text{inače;} \end{cases}$$

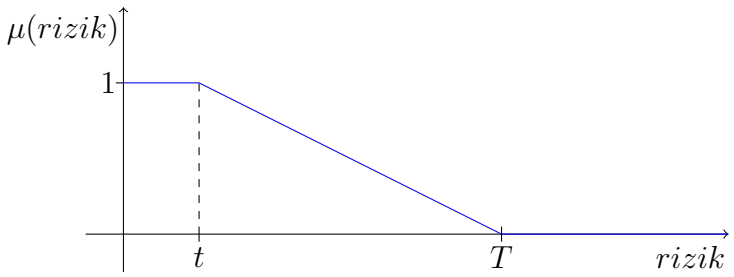
gde je t indeks rizika najbezbednije moguće rute $t = \min_{i \in N} c_{i0} d_i$ i T zadati limit rizika. Prikaz ove funkcije pripadanja je na slici 4.3.

Zapravo, kvalitet jedne rute je definisan pomoću njenog ukupnog pređenog puta

i fazi broja određenog pomoću indeksa rizika rute. Preciznije,

$$kvalitet\_rute = \begin{cases} pređeni\_put - 1, & \text{ako } r < t, \\ pređeni\_put - \rho, & \text{ako } t \leq x < T, \\ pređeni\_put, & \text{inače;} \end{cases}$$

gde je ρ vrednost između 0 i 1 koja opada kako rizik raste. Ovo zapravo znači da kvalitet svake rute može biti ocenjen maksimalnom vrednošću 1, ako je ruta bezbedna.



Slika 4.3: *Funkcija pripadanja za zadati fazi broj*

Primetimo da se korišćena maksimalna vrednost 1 može zameniti sa nekim malim brojem $\varepsilon > 0$ koji nema uticaja na spajanja ukoliko je razlika između pređenih puteva značajna. Sa druge strane, kada razlika između pređenih puteva dva rešenja je veoma slična ili čak ista, biramo spajanje koje će dovesti da manjih vrednosti indeksa rizika. Stoga, ovaj pristup čuva dobre karakteristike klasične faze konstrukcije zasnovane na CW algoritmu.

Faza lokalne pretrage za RCTVRP

Odabir struktura okolina za problem RCTVRP je težak zadatak imajući u vidu da može da uključuje zamenu redosleda čvorova u okviru rute, ali i razmenu čvorova među različitim rutama. Najpre, može se primetiti da LS operatori koji vrše izmene u okviru jedne rute su LS operatori za TSP koji su korišćeni u različitoj literaturi [189, 104, 8]. U ovom radu je, za potrebe rešavanja problema RCTVRP, implementirani operatori: *2-opt* i *3-opt*.

- Operator *2-opt* predstavljen na slici 4.4, je često korišćeni LS operator za TSP

[122], koji je prvi put opisan u radu [28]. Za rutu sa n čvorova, postoji $O(n^2)$ ruta u njenoj *2-opt* okolini.

- Operator *3-opt* je predstavljen na slici 4.5 i opisan u [123]. U *3-opt* okolini rute sa n čvorova se nalazi $O(n^3)$ različitih ruta.

Pretraživanje k-opt se može uopštiti za veće vrednosti k. Međutim, Lin (videti u [123]) je ustanovio eksperimentalno da rute dobijene primenom npr. *4-opt* nisu značajno bolje od onih dobijenih primenom *3-opt* operatora, bez obzira što je vreme izračunavanja za *4-opt* veće.

Pored toga, korišćena su naredna četiri LS operatora koja rade na dve rute:

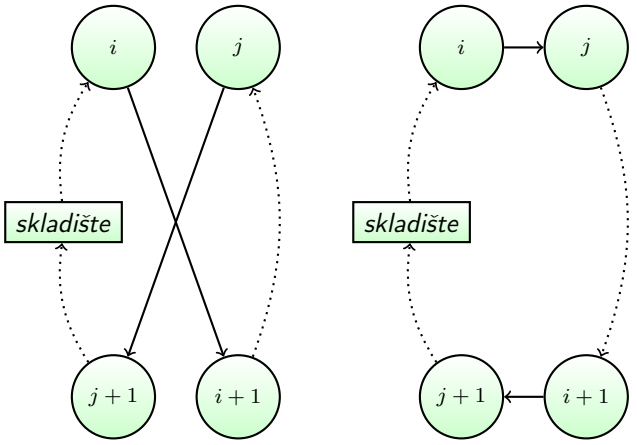
- Operator *2-relocate* opisan u radu [181] (slika 4.6);
- Operator *2-exchange* opisan u radu [181] (slika 4.7);
- Operator *2-opt* za 2 rute, opisan u radu [154], predstavlja generalizaciju prethodno opisanog 2-OPT i razmenjuje krajnje segmente dve rute ne menjajući redosled čvorova (slika 4.8);
- Operator *cross-exchange* je opisan u radu [181] (slika 4.9).

U implementaciji lokalne pretrage za RCTVRP, u svakom koraku je izvršena najbolja zamena, uz uzastopno menjanje struktura okolina. Prvo se iterativno razmatraju *2-opt* zamene i vrši se najbolja zamena. Kada je lokalni minimum dostignut u smislu *2-opt* okoline, razmatraju se *3-opt* okoline. Na kraju, kada dalja poboljšanja ne postoje u okolinama na pojedinačnim rutama, operatori koji rade na dve rute se primenjuju na svake dve rute.

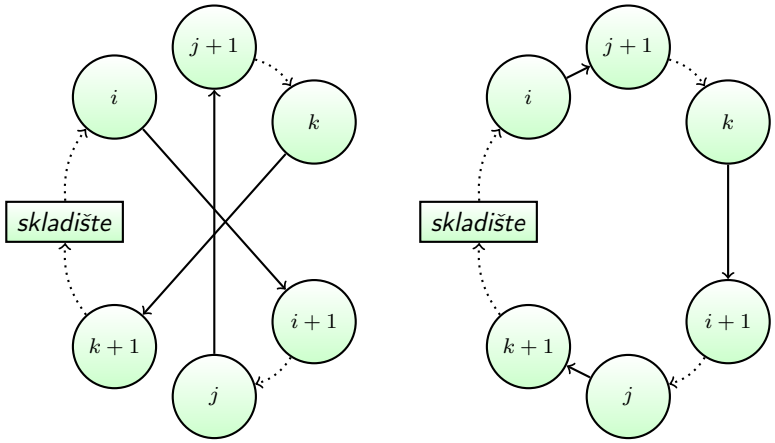
Za potrebe ove disertacije prikazani operatori su implementirani na efikasan način. Svako izračunavanje je realizovano korišćenjem specijalno kreirane strukture koja je opisana u sekciji 4.5.

4.3 Metoda ponovnog povezivanja puta

Metoda ponovnog povezivanja puta (engl. *Path relinking* - PR) uvedena je kao način da se pretraže trajektorije (putanje) između dobrih (elitnih) rešenja prethodno dobijenih pomoću tabu pretrage (engl. *Tabu Search* - TS) [114, 84, 85]. PR metoda



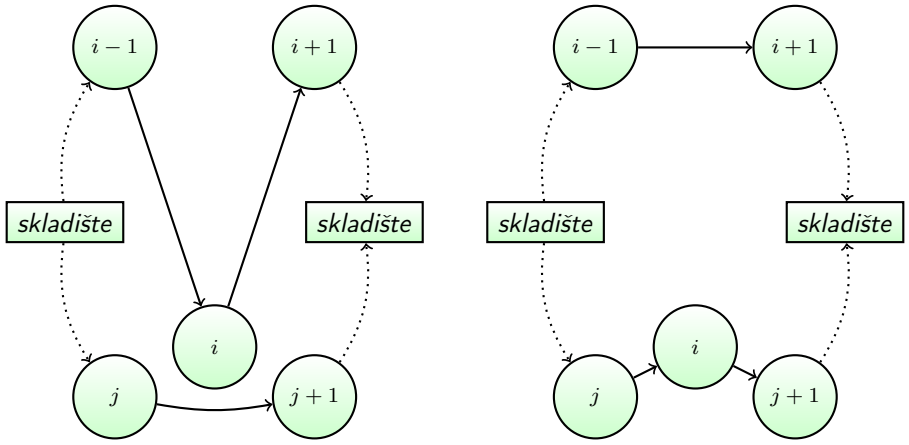
Slika 4.4: *Grafički prikaz 2-opt operatora lokalne pretrage*



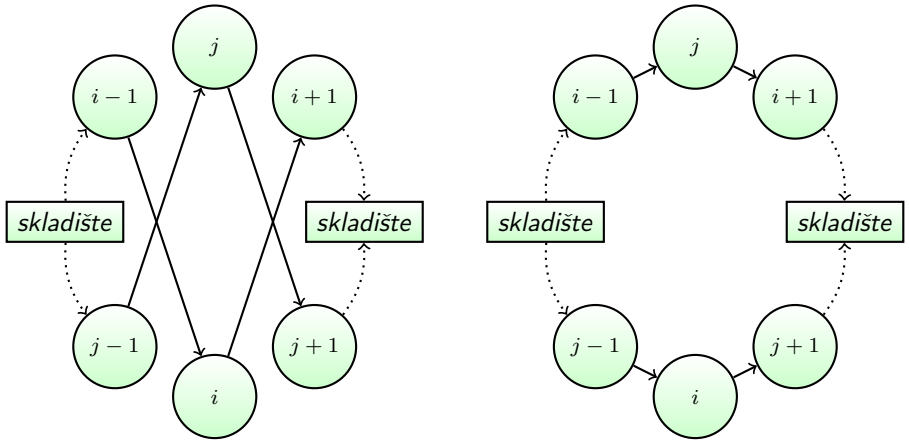
Slika 4.5: *Grafički prikaz 3-opt operatora lokalne pretrage*

danas se koristi i u kombinaciji sa nekim drugim heuristikama, najčešće u cilju poboljšanja performansi metaheuristika koje su bazirane na lokalnoj pretrazi [88]. PR je uspešno hibridizovan sa GRASP metodom, najpre u radu [115], a kasnije je ovaj hibrid korišćen i u drugim radovima [2, 63, 62, 159, 100].

Laguna i Martí [115] prilagodili su PR metodu kontekstu GRASP metode kao način intenzifikacije pretrage u okviru metaheurističke metode. U ovom smislu, povezivanje se sastoji iz pronalaženja puta između rešenja dobijenog metodom GRASP i odabranog elitnog rešenja. Zbog toga, PR ima drugačiju interpretaciju u slučaju kombinovanja sa GRASP metodom jer rešenja dobijena u nekoj iteraciji GRASP metode nisu povezana sa rešenjem iz naredne iteracije nekim nizom koraka, kao što je to slučaj sa TS.



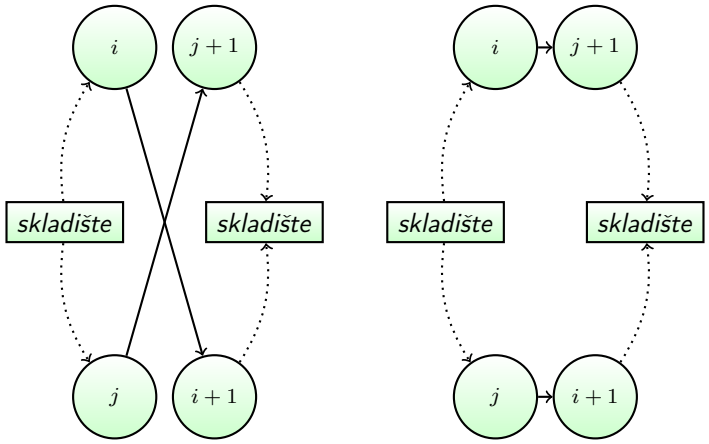
Slika 4.6: *Grafički prikaz 2-relocate operatora lokalne pretrage*



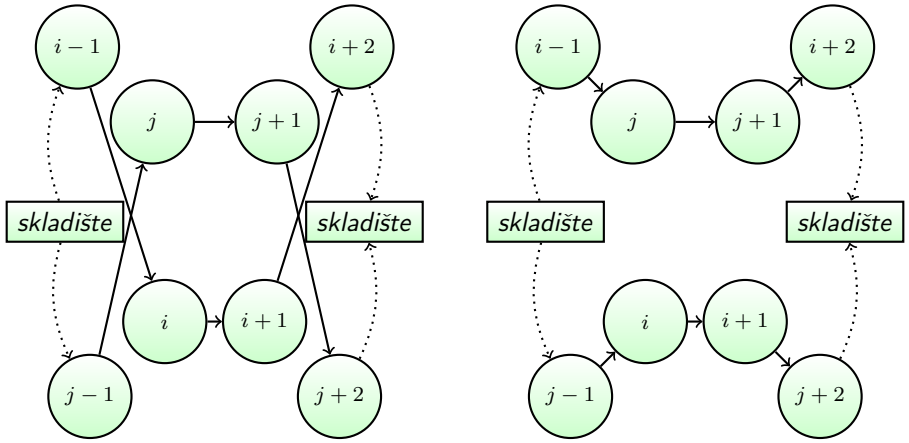
Slika 4.7: *Grafički prikaz 2-exchange operatora lokalne pretrage*

Kao što je u radu [88] istaknuto, PR predstavlja veoma efikasan metod u rešavanju raznolikih problema diskretne optimizacije. Metoda ponovnog povezivanja puta predstavlja način pretraživanja trajektorija između dobrih rešenja. Glavna ideja ove metode je pretpostavka da dobra rešenja imaju neke zajedničke karakteristike, pa kretanjem po trajektoriji između dva dobra rešenja prolazi se kroz rešenja koja su potencijalno bolja. Stoga, PR ima sličnosti sa evolutivnim metodama. Međutim, važna razlika je to što PR koristi sistematična, deterministička pravila za kombinovanje rešenja.

Osnovna ideja PR metode je da dobra rešenja problema imaju neke zajedničke karakteristike. Stoga postoji očekivanje da se može pronaći još bolje rešenje generisanjem puta, odnosno niza međurešenja, između dva dobra rešenja. Počevši od



Slika 4.8: *Grafički prikaz operatora 2-opt na 2 rute*



Slika 4.9: *Grafički prikaz operatora cross-exchange*

jednog elitnog rešenja koje se zove početno rešenje (engl. *initial solution*), PR metoda podrazumeva kretanje po putanji koja vodi ka vodećem rešenju (engl. *guiding solution*). Sva rešenja na putu imaju zajedničke karakteristike ova dva rešenja, a što se više udaljavamo od početnog rešenja, međurešenja na putu postaju sve sličnija vodećem rešenju. To se postiže biranjem koraka koji uvode karakteristike vodećeg rešenja i dodaju ih u međurešenja koja nastaju od početnog rešenja.

Pohlepna varijanta metode ponovnog povezivanja puta

Pohlepna varijanta PR metode (engl. *Greedy path relinking*) podrazumeva da se u svakom koraku zamena bira pohlepno. PR radi na skupu rešenja, zvanom elitni skup (engl. *elite set* - ES), koji se najčešće konstruiše metodom sa kojom se PR hi-

bridizuje. U okviru ove disertacije, metoda GRASP je korišćena za kreiranje elitnog skupa. Ukoliko se kao kriterijum za popunjavanje elitnog skupa jednostavno koristi kvalitet rešenja, elitni skup se može popuniti sa |ES| najboljih rešenja dobijenih metodom GRASP. Međutim, pokazalo se da je primena PR metode nije uspešna ukoliko su rešenja na koja se primenjuje previše slična (videti [162]). Zbog toga se prilikom konstrukcije elitnog skupa treba voditi računa kako o kvalitetu, tako i o raznolikosti rešenja. Na početku, ES je prazan, pa nakon $p = |ES|$ iteracija GRASP metode skup se popuni sa dobijenim rešenjima. ES sadrži sortirana rešenja, od najboljeg (x_1) ka najlošijem (x_p). Zatim, u narednim GRASP iteracijama, proverava se da se neko rešenje x' kvalifikuje da uđe u skup. Ukoliko je X' bolje od x_1 direktno se ubacuje u skup ES. Sa druge strane ukoliko je x' bolje od najlošijeg iz skupa x_p i dovoljno različito od rešenja koja su trenutno u skupu ES ($d(x', ES) \geq D$), takođe se ubacuje u ES. Parametar D koji predstavlja granicu rastojanja se može podesiti eksperimentalno. Na primer, D može biti medijana skupa.

Da bi veličina elitnog skupa ostala nepromenjena, nakon dodavanja nekog novog rešenja u skup potrebno je izbaciti neko rešenje iz skupa. Logična strategija je da se izbaciti najlošije rešenje iz elitnog skupa. Osim toga, u cilju očuvanja kvaliteta i raznolikosti, kao dobra strategija pokazalo se izbacivanje rešenja koje je najsličnije novom dodatom rešenju x' od onih koji su lošiji od x' [159].

Implementacija hibrida GRASP i PR metode može biti statička, u smislu da se najpre koristi GRASP da se konstruiše ES a onda se primenjuje PR da generiše rešenja između svaka dva rešenja iz ES. Na primer, za proizvoljna dva rešenja $x_i, x_j \in ES$ metoda ponovnog povezivanja puta se primenjuje u oba smera, tj. $PR(x_i, x_j)$ i $PR(x_j, x_i)$. Zatim se na najbolje rešenje sa obe putanje primeni lokalna pretraga. Najbolje rešenje se uvek čuva u elitnom skupu (x_1) i ažurira se kada se nađe novo bolje rešenje. U ovom slučaju, algoritam se završava kada se PR primeni na sve sve parove.

Sa druge strane, hibridizacija metode GRASP sa metodom ponovnog povezivanja puta može sadržati dinamičko ažuriranje elitnog skupa, kao što je predloženo u radu [115]. U tom slučaju, svako rešenje x' koje se dobije pomoću GRASP metode se direktno šalje u PR algoritam, koji se primenjuje na x' i neko izabrano x_j iz ES. Odabir se radi na osnovu određene verovatnoće i vrednosti rešenja. Kao i u statičkoj varijanti, lokalna pretraga se primenjuje na najbolje rešenje dobijeno na putu, ali sada izlaz dobijen lokalnom pretragom se odmah testira za ulazak u elitni skup.

Pohlepno-slučajna varijanta metode ponovnog povezivanja puta

Pohlepno-slučajna varijanta PR metode (engl. *Greedy randomized path relinking* - GRPR) podrazumeva da kretanja između početnog i vodećeg rešenja prate pohlepno-slučajnu strategiju. Ta strategija se odnosi na to da se u svakom koraku, umesto da se primenjuje lokalna promena koja daje najbolji rezultat, promena koja će se realizovati se slučajno bira iz liste dobrih kandidata. Stoga, ovaj postupak ima sličnosti sa konstruktivnom fazom GRASP metode.

Okrnjena varijanta metode ponovnog povezivanja puta

U slučaju ove varijante, kriterijum zaustavljanja je drugačiji. Umesto kretanja po putanji sve dok se ne dođe do vodećeg rešenja, samo k koraka je dozvoljeno i onda se vraća najbolje rešenje na tom kraćem putu.

Evolutivna varijanta metode ponovnog povezivanja puta

Evolutivna varijanta PR metode (engl. *Evolutionary path relinking* - EvPR) je uveden kao završna faza prilikom hibridizacije metoda GRASP i PR [162, 6]. Kao i u dinamičkoj varijanti GRASP metode sa GRP, i u slučaju GRASP metode sa EvPR u svakoj iteraciji se primenjuje GRASP i PR u cilju popunjavanja elitnog skupa. Nakon određenog broja iteracija GRASP sa GRP se završava. Sa druge strane, u varijanti GRASP metode sa EvPR nakon toga se dešava završna faza koja podrazumeva primenu PR metode na svaka dva para rešenja iz ES. Rešenja koja se dobijaju ovom primenom PR metode su kandidati za elitni skup ES i PR se primenjuje opet na njih sve dok ima elemenata koji ulaze u ES.

4.4 Metoda ponovnog povezivanja puta za rešavanje problema RCTVRP

Metoda PR je prvi put primenjena na problem rutiranja vozila u radu [99] gde je hibridizovana sa TS metodom. U tom radu razmotrili su različite strategije za popunjavanje elitnog skupa i biranje početnog i vodećeg rešenja.

Za rešavanje problema RCTVRP metodom PR, odabrana je strategija za popunjavanje elitnog skupa koja kreira skup koga odlikuju i kvalitet i raznolikost i koja

se najbolje pokazala za rešavanje problema VRP metodom PR [99], a u okviru ove disertacije prilagođena problemu RCTVRP. Označimo sa $D_{s_1}^{s_2}$ nivo različitosti dva rešenja s_1 i s_2 problema RCTVRP, tj. broj grana koje su sadržane u jednom od rešenja, ali ne u oba. Medijana svih rešenja s elitnog skupa $\mathcal{R}$ relativno u odnosu na najbolje rešenje b se može izračunati kao:

$$Median = \frac{\sum_{s \in \mathcal{R} \setminus \{b\}} D_b^s}{|\mathcal{R}| - 1},$$

gde je $|\mathcal{R}|$ broj rešenja u elitnom skupu.

Rešenje s se dodaje u elitni skup $\mathcal{R}$ ukoliko je zadovoljen bar jedan od sledećih uslova:

- $|\mathcal{R}| < ES_{size}$, gde je ES_{size} predefinisani parametar;
- rešenje s je bolje od najboljeg rešenja iz elitnog skupa $\mathcal{R}$;
- rešenje s je bolje nego najlošije rešenje iz elitnog skupa $\mathcal{R}$ i $D_b^s > Median$.

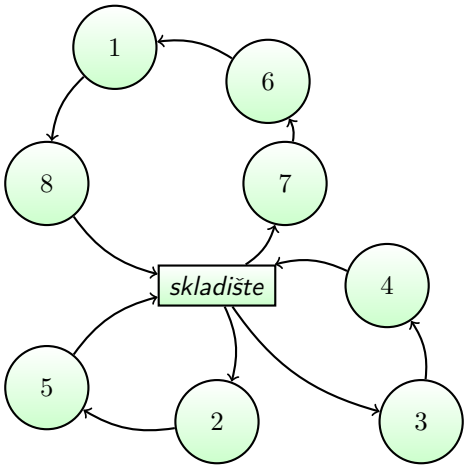
U svakom od prethodnih slučajeva, najlošije rešenje iz elitnog skupa $\mathcal{R}$ se zameni sa s, pa broj elemenata u $\mathcal{R}$ ostaje konstantan od trenutka kada se napuni.

Što se tiče odabira inicijalnog i vodećeg rešenja, s ozbirom da je PR hibridizovan sa GRASP metodom, jedno rešenje je rešenje iz trenutne GRASP iteracije. Drugo rešenje se bira iz elitnog skupa $\mathcal{R}$ na slučajan način, ali u cilju davanja prednosti boljim rešenjima, korišćena je apsolutna vrednost normalne raspodele davajući prednost boljim rešenjima iz elitnog skupa $\mathcal{R}$.

Odabir strukture okolina za kretanje po putanji je deo PR metode koji najviše zavisi od problema koji se rešava. PR je prvi put primenjen na VRP u radu [99], gde je kretanje po putanjama realizovano tako da identični delovi dva rešenja ostanu uključeni u svakom među rešenju na putanji. Međutim, kako je naglašeno u radu [99], nije uvek očigledno odlučiti koji su identični delovi dva rešenja u slučaju VRP. Stoga, oni su predložili uparivanje ruta rešavanjem problema raspoređivanja (engl. *assignment problem*).

U okviru ove disertacije, predlaže se nova struktura za PR metodu u slučaju rešavanja problema RCTVRP, a može se koristiti i za ostale probleme iz VRP klase. Svako rešenje problema RCTVRP se predstavi kao uređena lista grana uključenih u rešenje. Na primer, rešenje s predstavljeno na slici 4.10 se može jednoznačno

zapisati kao $Lista_s = \{(0,2),\ (0,3),\ (0,7),\ (1,8),\ (2,5),\ (3,4),\ (4,0),\ (5,0),\ (6,1),\ (7,6),\ (8,0)\}$.



Slika 4.10: *Primer jednog rešenja problema RCTVRP sa tri rute*

Spajanje putanja u okviru PR metode počinje sa dve liste grana, označene sa $Lista_{\text{vodećeRešenje}}$ i $Lista_{\text{početnoRešenje}}$. Grane koje se nalaze u oba skupa (tj. koji pripada preseku $Lista_{\text{vodećeRešenje}} \cap Lista_{\text{početnoRešenje}}$) ostaju uključene i u svim rešenjima na putanji. Da bi se obezbedio napredak ka vodećem rešenju, mogući PR koraci imaju sledeći oblik: dve grane (a,b) i (c,d) trenutnog rešenja se mogu zameniti sa granama (a,d) i (c,b), gde je $(a,d) \in Lista_{\text{vodećeRešenje}} \setminus Lista_{\text{trenutnoRešenje}}$, dok su $(a,b),(c,d) \in Lista_{\text{trenutnoRešenje}} \setminus Lista_{\text{vodećeRešenje}}$. U svakom koraku se bira najbolja zamena, u smislu funkcije cilja, i ona se izvršava. Dozvoljeno je kretanje i kroz prostor nedopustivih rešenja, ukoliko u nekom koraku nema zamena koje vode u dopustiva rešenja. Ovo je omogućeno dodavanjem velikog broja na vrednost funkcije cilja u slučaju nedopustivih rešenja.

Primetimo da primena prethodno opisanih PR zamena ne može promeniti broj grana u listi. Međutim, inicijalno i vodeće rešenje mogu imati različit broj ruta, tj. različit broj grana u odgovarajućim listama. Stoga, pre početka iterativne primene zamena, broj grana u listama se izjednačava dodavanjem odgovarajućeg broja fiktivnih grana (engl. *dummy edges*). Fiktivna grana $(0,0)$ odgovara praznoj ruti i stoga podrazumevamo da ne menja rešenje kome je dodata, ali dozvoljava spajanja (ukoliko je $(a,d) = (0,0)$) i deljenja (ako je $(a,b) = (0,0)$) ruta tokom PR zamena. Opisani PR za RCTVRP je prikazan algoritmom 2.

Algoritam 2 PR za RCTVRP

Procedura PR($Lista_{\text{početnoRešenje}}$, $Lista_{\text{vodećeRešenje}}$)
  NajboljeRešenje ← Rešenje($Lista_{\text{početnoRešenje}}$);
  while $|Lista_{\text{početnoRešenje}}| > |Lista_{\text{vodećeRešenje}}|$ **do**
    $Lista_{\text{vodećeRešenje}} \leftarrow Lista_{\text{vodećeRešenje}} \cup \{(0,0)\}$;
  while $|Lista_{\text{početnoRešenje}}| < |Lista_{\text{vodećeRešenje}}|$ **do**
    $Lista_{\text{početnoRešenje}} \leftarrow Lista_{\text{početnoRešenje}} \cup \{(0,0)\}$;
  $Lista_{\text{trenutnoRešenje}} \leftarrow Lista_{\text{početnoRešenje}}$;
  while $Lista_{\text{trenutnoRešenje}} \neq Lista_{\text{vodećeRešenje}}$ **do**
    $Lista_{\text{trenutnoRešenje}} \leftarrow$ NajboljaZamena($Lista_{\text{trenutnoRešenje}}$,$Lista_{\text{vodećeRešenje}}$);
    NovoRešenje ← LokalnaPretraga(Rešenje($Lista_{\text{trenutnoRešenje}}$));
    if NovoRešenje bolje nego NajboljeRešenje **then**
      NajboljeRešenje ← NovoRešenje;
  return NajboljeRešenje

4.5 Struktura podataka za spajanje ruta za RCTVRP

Vremenska složenost direktnog računanja globalnog indeksa rizika i ukupnog pređenog puta je $O(n)$, gde je n ukupan broj čvorova. U predloženom algoritmu za rešavanje problema RCTVRP računanje vrednosti funkcije cilja je potrebno u fazama konstrukcije, lokalne pretrage i PR metode. Većina rešenja koja je u tim situacijama potrebno vrednovati su zapravo bliska prethodnim rešenjima (u njihovim su okolinama), pa se ova izračunavanja mogu uraditi efikasnije.

Za svaku rutu r umesto evidentiranja samo liste čvorova, čuvaju se i naredne vrednosti:

- $indeksRizika_r$ = globalni indeks rizika rute r;
- $novac_r$ = ukupna vrednost novca koja se akumulira u vozilu tokom rute r;
- $rastojanje_r$ = ukupan pređeni put rute r;
- $indeksRizika_{\bar{r}}$ = globalni indeks rizika rute $\bar{r}$ dobijene obrtanjem redosleda čvorova u ruti r.

Može se primetiti da se prve tri vrednosti poklapaju za rute r i $\bar{r}$. Ukoliko se vrši

spajanje ruta r_1 i r_2 u novu rutu r_3, onda se u konstantnom vremenu može izračunati:

$$\begin{aligned} indeksRizika_{r_3} =& indeksRizika_{r_1} + indeksRizika_{r_2} \\ &+ novac_{r_1} \cdot (rastojanje_{r_2} + \text{ušteda}); \\ novac_{r_3} =& novac_{r_1} + novac_{r_2}; \\ rastojanje_{r_3} =& rastojanje_{r_1} + rastojanje_{r_2} + \text{ušteda}; \\ indeksRizika_{\bar{r}_3} =& indeksRizika_{\bar{r}_1} + indeksRizika_{\bar{r}_2} \\ &+ novac_{r_2} \cdot (rastojanje_{r_1} + \text{ušteda}). \end{aligned} \tag{4.1}$$

koristeći prethodno izračunate vrednosti ušteda na osnovu CW heuristike. Preciznije, ušteda $= c_{ij} - c_{i0} - c_{0j}$, gde je i poslednji čvor rute r_1, dok je j prvi čvor u ruti r_2. Prethodne formule 4.1 se mogu posmatrati kao jedna operacija spajanja dve rute, označiti sa $\mathcal{S}pajanje$, i sva izračunavanja se izvršavaju u $O(1)$.

Da se prethodno opisana operacija $\mathcal{S}pajanje$ mogla univerzalno koristiti sve vrednosti u strukturi $\{indeksRizika_r, novac_r, rastojanje_r, indeksRizika_{\bar{r}}\}$ treba da budu izračunate za svaku rutu i njenu svaku podrutu. Pod podrutom (engl. *subroute*) neke rute r u kojoj se obilaze čvorovi $k_1, k_2, ..., k_m$ se podrazumeva ruta u kojoj se obilaze čvorovi $k_i, k_{i+1}, ..., k_{j-1}, k_j$, za neke i i j takve da je $1 \leq i \leq j \leq m$. Na primer, ako pretpostavimo da je jedna od ruta trenutnog rešenja $0 \to 1 \to 2 \to 3 \to 0$ (vozilo kreće od skladišta, posećuje čvorove 1, 2 i 3, i vraća se u skladište), potrebno je izračunati vrednosti definisane strukture podataka za svaku podrutu: $(0 \to 1 \to 0)$, $(0 \to 1 \to 2 \to 0)$, $(0 \to 1 \to 2 \to 3 \to 0)$, $(0 \to 2 \to 0)$, $(0 \to 2 \to 3 \to 0)$, $(0 \to 3 \to 0)$. Vremenska i prostorna složenost računanja ove četiri vrednosti iz strukture za sve podrute rute sa n čvorova su $O(n^2)$. Primetimo da je time za svaku podrutu r poznati i svi potrebni podaci za rute koja se od r može dobiti promenom smera.

Koristeći ovaj pristup, spajanje bilo koje dve podrute u rešenju i računanje funkcije cilja se može uraditi u $O(1)$. Na primer, za 2-opt LS operator (Figure 4.4), svi parametri strukture za novodobijenu rutu se mogu dobiti primenom:

$$\mathcal{S}pajanje(\mathcal{S}pajanje(podruta(\text{skladište}, i), podruta(j, i{+}1)), podruta(j{+}1, \text{skladište})).$$

Stoga, umesto računanja vrednosti funkcije cilja za sva rešenja u 2-opt okolini rute sa n čvorova u $O(n^3)$, isto se može uraditi u $O(n^2{+}n^2) = O(n^2)$. Naravno, ovde

je napravljen kompromis, s obzirom da se koristi dodatni memorijski prostor $O(n^2)$. Sličan postupak se može primeniti na sve operatore u fazama konstrukcije, LS i PR, što je i urađeno u okviru implementacije GRASP i PR metoda u za rešavanje problema RCTVRP.

4.6 Eksperimentalni rezultati za RCTVRP

Testiranje predloženih algoritama je izvršeno na PC računaru sa Intel Core i7-860 procesorom sa radnim taktom 2.8 GHz pod Windows 7 Professional operativnim sistemom. Algoritmi su implementirani na C# programskom jeziku.

Eksperimentalni testovi implementirane hibridne meteheuristike GRASP i PR (i fazi i obična verzija) su izvršavani dok neki od narednih kriterijuma zaustavljanja nije dostignut:

- 100000 uzastopnih iteracija bez poboljšanja najbolje pronađenog rešenja;
- vremensko ograničenje od t_{sec} sekundi.

Drugi kriterijum zaustavljanja je dodat kako bi bilo moguće adekvatno poređenje sa rezultatima iz [182] na dva skupa podataka O i S. Imajući u vidu da su autori za potrebe rada [182] 20 puta pokretali algoritam sa po 30 sekundi maksimalnog vremena izvršavanja. S obzirom na karakteristiku GRASP metaheuristike da je svaka iteracija ponovno pokretanje algoritma, svi testovi za potrebe ove disertacije su izvedeni tako što se izvršava jedno pokretanje algoritma za svaki test primer. Radi adekvatnog poređenja sa rezultatima iz [182], pronađena je evaluacija korišćenih procesora (*CPU comparison*[1]). Procesor korišćen u radu [182] ima efikasnost 6629, dok procesor korišćen za potrebe ovog rada ima 5063. Stoga je bilo dozvoljeno postaviti t_{sec} na 785 sekundi radi poređenja sa njihovih ukupnih 600 sekundi. Ipak, odabrano je $t_{sec} = 600$ sekundi.

Podešavanje parametara

Parametar G_α u fazi konstrukcije metode GRASP se automatski podešava na osnovu reaktivne GRASP strategije (engl. *Reactive GRASP strategy*), koja kada je

[1] http://www.cpubenchmark.net/

prvi put korišćena [155] i tada je ovakva metoda nazvana reaktivni GRASP (engl. *Reactive GRASP*). Kasnije uspešno primenjene u različitim radovima [151, 57] i danas se može nazvati i uobičajenim načinom podešavanja parametra G_α, pa se u nazivu metode izostavlja reč reaktivni.

U cilju određivanja veličine elitnog skupa (ES_{size}) test je urađen na sledeći način. Odabrano je 60 test primera tako što je na slučajan način odabrano po 15 test primera iz svakog od skupova skupova: skup R, skup V, skup O i skup S. U literaturi se za vrednost parametra ES_{size} najčešće koriste brojevi između 10 i 100. Međutim, kako su vrednosti bliske 10 češće zastupljene u literaturi (videti [137, 2, 151, 57]) za testiranje su odabrane naredne vrednosti: 10, 20, 30, 50, 75 i 100. Test je izveden na klasičnoj verziji algoritma (bez fazi modifikacije).

Rezultati testiranja za parametar ES_{size} su predstavljeni u tabeli 4.1. Vrednosti za ES_{size} su predstavljene u prvoj koloni, prosečna vrednost funkcije cilja u drugoj, prosečan broj iteracija potreban algoritmu da dođe do najboljih rešenja je dat u trećoj koloni, dok četvrta kolona sadrži prosečno vreme u sekundama. Poslednja kolona se odnosi na broj najboljih rešenja dostignutih sa određenom vrednošću ES_{size} parametra.

Tabela 4.1: ES_{size} *analiza parametra*

ES_{size}	prosek(sol)	prosek(iter)	prosek(t) [s]	# najboljih
10	1909.82	668.62	169.49	48
20	1907.59	672.23	170.50	43
30	1909.67	739.57	167.15	45
50	1909.40	597.82	156.26	51
75	1909.54	569.28	145.30	48
100	1909.92	597.22	153.60	46

Može se primetiti na osnovu podataka u tabeli 4.1 da nema direktne veze između dobijenih rezultata i vrednosti ES_{size} parametra. Dobijeni rezultati su dobri za sve vrednosti parametara, pa se na osnovu toga ne može odrediti najbolja od testiranih vrednost za ES_{size}. Dakle, predloženi algoritam je veoma stabilan bez obzira na vrednost ES_{size} parametra. Ipak, detaljnijom analizom može se uočiti da su za parametar $ES_{size} = 50$ postignuti malo bolji rezultati u odnosu na ostale razmatrane vrednosti parametra kada je u pitanju broj dostignutih najboljih rešenja. Stoga, za dalja testiranja je izabrano $ES_{size} = 50$.

Poređenje dobijenih rešenja

U cilju predstavljanja efikasnosti fazi verzije predložene hibridne metode u odnosu na verziju istog hibrida bez fazi modifikacije, komparativni eksperimenti su izvedeni na 60 test primera predstavljenih u sekciji 4.6. Poređenje rezultata je dato u tabeli 4.2 koja ima sličan redosled kolona kao i prethodna tabela 4.1. Uzimajući u obzir predstavljene rezultate, može se primetiti da fazi verzija algoritma bolja od hibridne verzije bez fazi modifikacije. Fazi verzija je dostigla najbolja rešenja na više test primera, koristeći u proseku manje vreme dolaska do rešenja. Takođe, u odnosu na broj iteracija potrebnih za dolazak do rešenja interesantno je primetiti da je prosečan broj iteracija manji kada se koristi fazi modifikacija, pa se može zaključiti da je konvergencija ka dobrim rešenjima brža u slučaju fazi verzije.

Tabela 4.2: *Ukupni i prosečni rezultati za poređenje fazi i ne-fazi verzije predloženog algoritma*

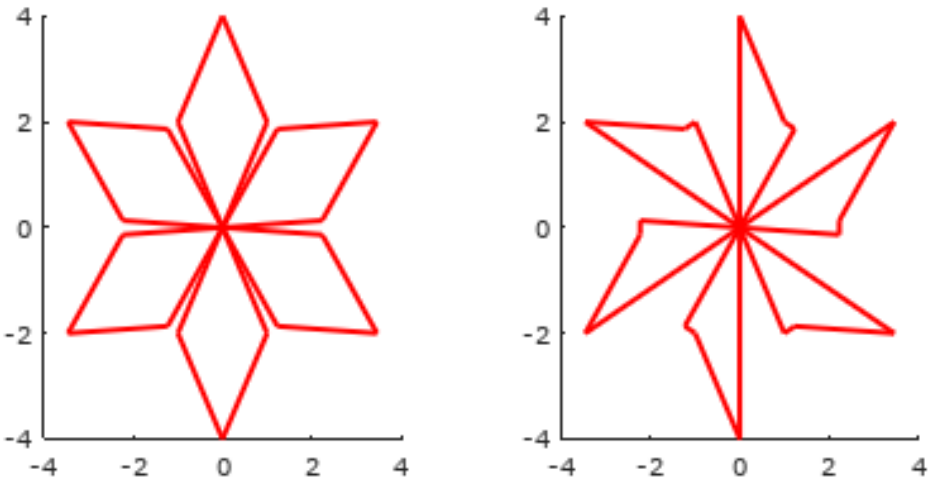
TIP	prosek(sol)	prosek(iter)	prosek(t) [s]	# najboljih
GRASP-PR	1909.40	597.82	156.26	45
F-GRASP-PR	1905.20	442.22	145.00	49

Test primeri iz sva četiri skupa su testirana predloženim algoritmom, koristeći $ES_{size} = 50$ i vremenski limit od 600 sekundi.

Za neke test primere iz skupova O i S dobijeni rezultati su bolji nego očekivane optimalne vrednosti koje su izračunate kao što je opisano u radu [182]. Ovo se dešava iako su test primeri iz ova dva skupa posebno kreirani tako da optimalna rešenja znaju unapred. Ipak, u nekim instancama je „jeftinije" u smislu pređenog puta da se reorganizuju rute jer su neki čvorovi koji bi trebalo da budu u različitim rutama previše blizu jedan drugom, a prag rizika ne zabranjuje te reorganizovane rute. Jedan primer prikazan je na slici 4.11, gde je za test primer veličine 19 iz skupa O, predviđeno optimalno rešenje sa leve strane slike i ima vrednost funkcije cilja 53.66, dok je vrednost funkcije cilja dobijena metodom predloženom u ovoj disertaciji 52.44 a grafički prikaz tog rešenja je u desnom delu slike 4.11. Skladište je čvor na poziciji $(0, 0)$. Pronađeno je da je to slučaj sa 5 od 34 test primera iz skupa O, i 17 od 34 test primera iz skupa S. U slučajevima da je metodom F-GRASP-PR dobijeno rešenje bolje od očekivanog optimuma, to rešenje je računato u tabeli 4.3 radi poređenja sa rezultatima iz rada [182].

Tabela 4.3: *Poređenje broja dostignutih najboljih poznatih rešenja na test primerima iz skupa O i S*

Metoda	skup O		skup S	
najbolja iz [182]	17/32	53%	13/32	41%
F-GRASP-PR	29/34	85%	30/34	88%



Slika 4.11: *Rešenja dobijena za test primer veličine 19 iz skupa O*

Autori rada [182] su predložili hibridnu metaheurističku metodu označenu sa ACO-LNS i poredili je sa prethodno predloženim metodama za rešavanje problema RCTVRP, nazvanih m-NNg, p-NNg, m-TNNg, p-TNNg, m-CWg, p-CWg i p-TLK (videti rad [181]). Od svih predloženih metoda u radovima [181, 182], ACO-LNS je dostigla najveći broj optimalnih rešenja iz skupa O (17/32), dok se metoda m-CWg najbolje pokazala na test primerima iz skupa S sa postignutih 13 od 32. Dakle, na skupu test primera O metod F-GRASP-PR je dostigao 85% optimalnih rešenja, dok je metoda koja se najbolje pokazala na ovom skupu podataka od svih metoda iz [182] dostigla 53% optimalnih rešenja. Razlika je još uočljivija na skupu S gde je u ovoj disertaciji predloženi metod F-GRASP-PR dostigao 88% optimalnih rešenja, dok je najbolja metoda na ovom skupu podataka iz [182] stigla samo 41%. Primetimo da je je za metod F-GRASP-PR moglo da se koristi 785 sekundi za svako pokretanje algoritma radi poređenja sa rezultatima iz [182]. Ipak, vreme je postavljeno na 600 sekundi kao što je objašnjeno na početku ove sekcije.

Poređenje sa F-GRASP-PR metodom predloženom u ovoj disertaciji je predsta-

vljeno u tabeli 4.3, gde je pokazano da metoda F-GRASP-PR, u smislu kvaliteta rešenja, nadmašuje sve pomenute metode. Može se primetiti da skupovi test primera S i O imaju po 34 test primera, ali je u radu [182] prikazano broj postignutih optimalnih rešenja od 32.

Poređenje na ostalim skupovima test primera nije bilo moguće s obzirom da u radovima [182] i [181] nisu prikazali najbolja dobijena rešenja.

U tabelama 4.4-4.7, prikazani su eksperimentalni rezultati dobijeni na test primerima iz svih razmatranih skupova podataka.

Prva kolona za tabele 4.4 i 4.5 sadrži ime test primera, dok prve dve kolone tabele 4.6 sadrži nivo rizika i veličinu test primera. U obe tabele ostale duple kolone sadrže dobijena rešenja i odgovarajuće vreme u sekundama dobijeno primenom predloženih GRASP-PR i F-GRASP-PR metoda. Na test primerima manjih dimenzija za koje je rezultat dobijen korišćenjem rešavača CPLEX 12.6 rezultati se poklapaju (videti rezultate iz sekcije 6.4). Za 6 test primera iz skupa R ne postoji dopustivo rešenje (vrednosti -1 prikazane u tabeli 4.5), s obzirom da već kod zvezastog rešenja neka od ruta ima nivo rizika koji je veći od dozvoljenog praga T.

Skupovi test primera O i S su posebno interesantni za testiranje zbog specijalnog načina kako su konstruisane da bi optimalne vrednosti bile poznate unapred, što se koristi za finalnu proveru eksperimentalnih rezultata. Rezultati na ova dva skupa test primera su dati u tabeli 4.7. Prva kolona sadrži ime (veličinu) test primera. Potom su, prvo za skup O, a potom i skup S date naredne vrednosti za sve test primere:

- očekivana optimalna vrednost funkcije cilja za svaki test primer, izračunata na osnovu objašnjenja datom u radu [182];
- vrednost funkcije cilja rešenja dobijenog metodom GRASP-PR;
- vreme u sekundama do rešenja dobijenog metodom GRASP-PR;
- vrednost funkcije cilja rešenja dobijenog metodom F-GRASP-PR;
- vreme u sekundama do rešenja dobijenog metodom F-GRASP-PR.

Tabela 4.4: *Eksperimentalni rezultati na manjim test primerima iz skupa R*

Naziv	GRASP-PR		F-GRASP-PR		Naziv	GRASP-PR		F-GRASP-PR	
instance	obj	t[s]	obj	t[s]	instance	obj	t[s]	obj	t[s]
4_1_1.0	343.05	0.03	343.05	0.04	8_3_1.0	658.32	0.04	658.32	0.04
4_1_1.5	343.05	0.03	343.05	0.04	8_3_1.5	487.53	0.04	487.53	0.04
4_1_2.0	343.05	0.04	343.05	0.04	8_3_2.0	431.01	0.04	431.01	0.04
4_1_2.5	343.05	0.03	343.05	0.04	8_3_2.5	416.90	0.06	416.90	0.06
4_1_3.0	286.13	0.04	286.13	0.04	8_3_3.0	384.23	0.04	384.23	0.04
4_3_1.0	343.05	0.03	343.05	0.04	8_5_1.0	574.23	0.04	574.23	0.04
4_3_1.5	343.05	0.03	343.05	0.04	8_5_1.5	487.53	0.04	487.53	0.04
4_3_2.0	343.05	0.04	343.05	0.04	8_5_2.0	431.01	0.04	431.01	0.04
4_3_2.5	343.05	0.03	343.05	0.04	8_5_2.5	416.90	0.06	416.90	0.06
4_3_3.0	286.13	0.04	286.13	0.04	8_5_3.0	379.64	0.06	379.64	0.06
4_5_1.0	343.05	0.04	343.05	0.04	8_7_1.0	485.04	0.04	485.04	0.04
4_5_1.5	343.05	0.04	343.05	0.04	8_7_1.5	416.90	0.06	416.90	0.06
4_5_2.0	343.05	0.04	343.05	0.04	8_7_2.0	384.23	0.04	384.23	0.04
4_5_2.5	343.05	0.04	343.05	0.04	8_7_2.5	327.71	0.06	327.71	0.06
4_5_3.0	286.13	0.04	286.13	0.04	8_7_3.0	327.71	0.06	327.71	0.06
4_7_1.0	343.05	0.04	343.05	0.04	10_1_1.0	608.90	0.04	608.90	0.04
4_7_1.5	332.93	0.04	332.93	0.04	10_1_1.5	540.57	0.06	540.57	0.06
4_7_2.0	286.13	0.04	286.13	0.04	10_1_2.0	466.03	0.06	466.03	0.06
4_7_2.5	286.13	0.04	286.13	0.04	10_1_2.5	422.80	0.04	422.80	0.04
4_7_3.0	286.13	0.04	286.13	0.04	10_1_3.0	417.33	0.06	417.33	0.07
6_1_1.0	594.64	0.04	594.64	0.04	10_3_1.0	608.90	0.04	608.90	0.04
6_1_1.5	594.34	0.04	594.34	0.04	10_3_1.5	534.25	0.04	534.25	0.04
6_1_2.0	438.51	0.04	438.51	0.04	10_3_2.0	466.03	0.06	466.03	0.06
6_1_2.5	387.99	0.04	387.99	0.04	10_3_2.5	422.80	0.06	422.80	0.06
6_1_3.0	387.68	0.04	387.68	0.04	10_3_3.0	417.33	0.06	417.33	0.06
6_3_1.0	645.17	0.04	645.17	0.03	10_5_1.0	608.90	0.04	608.90	0.04
6_3_1.5	594.34	0.04	594.34	0.04	10_5_1.5	481.99	0.04	481.99	0.04
6_3_2.0	445.44	0.04	445.44	0.04	10_5_2.0	428.77	0.04	428.77	0.04
6_3_2.5	387.99	0.04	387.99	0.04	10_5_2.5	421.57	0.06	421.57	0.06
6_3_3.0	387.68	0.04	387.68	0.04	10_5_3.0	412.43	0.07	412.43	0.08
6_5_1.0	594.64	0.04	594.64	0.04	10_7_1.0	576.47	0.04	576.47	0.04
6_5_1.5	594.34	0.04	594.34	0.04	10_7_1.5	468.50	0.06	468.50	0.06
6_5_2.0	387.99	0.04	387.99	0.04	10_7_2.0	424.08	0.06	424.08	0.06
6_5_2.5	387.99	0.04	387.99	0.04	10_7_2.5	406.83	0.04	406.83	0.04
6_5_3.0	387.68	0.04	387.68	0.04	10_7_3.0	406.83	0.06	406.83	0.06
6_7_1.0	534.96	0.04	534.96	0.04	12_1_1.0	1347.19	0.04	1347.19	0.04
6_7_1.5	445.44	0.04	445.44	0.04	12_1_1.5	1123.46	0.04	1123.46	0.04
6_7_2.0	362.88	0.04	362.88	0.04	12_1_2.0	840.46	0.06	840.46	0.06
6_7_2.5	360.68	0.04	360.68	0.04	12_1_2.5	714.43	0.04	714.43	0.05
6_7_3.0	358.63	0.05	358.63	0.06	12_1_3.0	594.74	0.08	594.74	0.08
8_1_1.0	541.56	0.04	541.56	0.04	12_3_1.0	1345.07	0.04	1345.07	0.04
8_1_1.5	487.53	0.04	487.53	0.04	12_3_1.5	1129.26	0.04	1129.26	0.04
8_1_2.0	431.01	0.04	431.01	0.04	12_3_2.0	839.27	0.07	839.27	0.07
8_1_2.5	416.90	0.04	416.90	0.04	12_3_2.5	714.43	0.06	714.43	0.06
8_1_3.0	379.64	0.04	379.64	0.04	12_3_3.0	602.45	0.07	602.45	0.08
Prosek	401.00	0.04	401.00	0.04	Prosek	564.47	0.05	564.47	0.05

Tabela 4.5: *Eksperimentalni rezultati na većim test primerima iz skupa R*

Naziv	GRASP-PR		F-GRASP-PR		Naziv	GRASP-PR		F-GRASP-PR	
instance	obj	t[s]	obj	t[s]	instance	obj	t[s]	obj	t[s]
12_5_1.0	-1.00		-1.00		16_7_1.0	884.70	0.06	884.70	0.07
12_5_1.5	995.93	0.06	995.93	0.06	16_7_1.5	655.12	0.09	655.12	0.09
12_5_2.0	724.19	0.04	724.19	0.04	16_7_2.0	584.10	0.07	584.10	0.07
12_5_2.5	696.98	0.08	696.98	0.08	16_7_2.5	520.38	0.09	520.38	0.09
12_5_3.0	595.28	0.04	595.28	0.04	16_7_3.0	476.75	0.21	476.75	0.21
12_7_1.0	-1.00		-1.00		18_1_1.0	-1.00		-1.00	
12_7_1.5	911.37	0.07	911.37	0.07	18_1_1.5	1058.47	0.06	1058.47	0.07
12_7_2.0	717.18	0.04	717.18	0.04	18_1_2.0	819.26	0.14	819.26	0.14
12_7_2.5	656.27	0.10	656.27	0.10	18_1_2.5	734.73	0.10	734.73	0.10
12_7_3.0	563.44	0.09	563.44	0.09	18_1_3.0	680.00	0.05	680.00	0.05
14_1_1.0	1026.92	0.04	1026.92	0.04	18_3_1.0	-1.00		-1.00	
14_1_1.5	836.74	0.10	836.74	0.10	18_3_1.5	1026.37	0.20	1026.37	0.19
14_1_2.0	756.68	0.06	756.68	0.06	18_3_2.0	819.26	0.19	819.26	0.20
14_1_2.5	549.40	0.09	549.40	0.09	18_3_2.5	730.55	0.09	730.55	0.09
14_1_3.0	543.23	0.12	543.23	0.13	18_3_3.0	680.00	0.09	680.00	0.09
14_3_1.0	1026.92	0.04	1026.92	0.04	18_5_1.0	-1.00		-1.00	
14_3_1.5	836.74	0.08	836.74	0.08	18_5_1.5	1059.65	0.05	1059.65	0.05
14_3_2.0	756.68	0.04	756.68	0.04	18_5_2.0	816.79	0.30	816.79	0.30
14_3_2.5	549.40	0.11	549.40	0.11	18_5_2.5	752.46	0.20	752.46	0.20
14_3_3.0	543.23	0.06	543.23	0.06	18_5_3.0	619.77	0.20	619.77	0.19
14_5_1.0	951.66	0.04	951.66	0.04	18_7_1.0	-1.00		-1.00	
14_5_1.5	758.81	0.07	758.81	0.07	18_7_1.5	923.73	0.14	923.73	0.13
14_5_2.0	549.40	0.06	549.40	0.07	18_7_2.0	782.82	0.13	782.82	0.13
14_5_2.5	541.50	0.12	541.50	0.12	18_7_2.5	703.04	0.55	703.04	0.57
14_5_3.0	482.56	0.09	482.56	0.10	18_7_3.0	590.82	0.43	590.82	0.45
14_7_1.0	812.17	0.08	812.17	0.08	20_1_1.0	1254.04	0.04	1254.04	0.04
14_7_1.5	623.20	0.11	623.20	0.11	20_1_1.5	1070.76	0.04	1070.76	0.04
14_7_2.0	512.57	2.10	512.57	2.08	20_1_2.0	787.19	0.08	787.19	0.09
14_7_2.5	488.47	0.13	488.47	0.13	20_1_2.5	741.46	0.20	741.46	0.20
14_7_3.0	427.96	0.19	427.96	0.19	20_1_3.0	693.60	0.06	693.60	0.06
16_1_1.0	1067.62	0.04	1067.62	0.04	20_3_1.0	1255.40	0.05	1255.40	0.05
16_1_1.5	790.19	0.13	790.19	0.14	20_3_1.5	1070.76	0.05	1070.76	0.05
16_1_2.0	651.20	0.08	651.20	0.08	20_3_2.0	785.37	0.23	785.37	0.22
16_1_2.5	565.46	0.35	565.46	0.35	20_3_2.5	725.39	0.19	725.39	0.19
16_1_3.0	550.35	0.12	550.35	0.12	20_3_3.0	618.39	0.10	618.39	0.11
16_3_1.0	1029.39	0.04	1029.39	0.04	20_5_1.0	1252.49	0.04	1252.49	0.04
16_3_1.5	790.19	0.10	790.19	0.10	20_5_1.5	1070.76	0.09	1070.76	0.09
16_3_2.0	650.56	0.09	650.56	0.10	20_5_2.0	809.34	0.09	809.34	0.09
16_3_2.5	565.46	0.34	565.46	0.33	20_5_2.5	709.53	0.77	709.53	0.77
16_3_3.0	550.35	0.11	550.35	0.11	20_5_3.0	655.54	0.25	655.54	0.25
16_5_1.0	1014.01	0.04	1014.01	0.04	20_7_1.0	1091.65	0.04	1091.65	0.04
16_5_1.5	853.95	0.06	853.95	0.07	20_7_1.5	871.25	0.10	871.25	0.10
16_5_2.0	661.21	0.10	661.21	0.10	20_7_2.0	722.98	0.23	722.98	0.23
16_5_2.5	593.59	0.11	593.59	0.12	20_7_2.5	668.37	0.11	668.37	0.11
16_5_3.0	559.76	0.26	559.76	0.26	20_7_3.0	583.89	0.07	583.89	0.07
Prosek	705.31	0.14	705.31	0.15	Prosek	813.58	0.15	813.58	0.15

Tabela 4.6: *Eksperimentalni rezultati na test primerima iz skupa V*

		GRASP-PR		F-GRASP-PR	
rl	n	obj	t[s]	obj	t[s]
1	22	782.50	0.07	782.50	0.07
1	26	872.44	0.12	872.44	0.12
1	30	656.03	1.08	656.03	1.10
1	36	953.57	0.27	953.57	0.26
1	45	891.79	356.23	891.79	351.58
1	51	1442.00	174.64	1442.00	174.79
1	72	348.81	372.66	348.81	361.87
1	101	2195.91	362.11	2195.91	357.46
1	121	5967.56	595.59	5967.02	595.15
1	135	2116.38	267.73	2116.38	266.17
1	151	3058.27	345.01	3058.27	345.38
1	200	3991.58	347.42	3991.58	349.85
1	256	83956.13	0.18	83956.13	0.17
1	301	25084.58	337.52	25084.58	341.12
1.5	22	578.19	1.10	578.19	1.04
1.5	26	680.85	1.55	680.85	1.61
1.5	30	514.34	0.39	514.34	0.39
1.5	36	749.84	2.95	749.84	2.82
1.5	45	776.36	10.66	776.36	10.23
1.5	51	1084.04	441.07	1084.04	452.39
1.5	72	291.06	56.19	291.06	57.06
1.5	101	1664.93	449.22	1664.93	447.39
1.5	121	4051.77	492.23	4051.77	475.48
1.5	135	1597.77	594.40	1597.77	587.90
1.5	151	2256.89	559.34	2256.89	560.52
1.5	200	2905.64	158.00	2905.64	158.95
1.5	256	58430.71	278.43	58430.71	275.43
1.5	301	18608.89	565.03	18608.89	553.88
2	22	496.68	3.34	496.68	3.39
2	26	566.46	0.56	566.46	0.54
2	30	488.97	0.96	488.97	0.95
2	36	660.22	89.76	660.22	89.34
2	45	735.57	6.46	735.57	6.34
2	51	904.63	309.63	904.63	299.66
2	72	262.02	63.33	262.02	63.59
2	101	1403.88	50.75	1403.88	51.34
2	121	3129.77	297.83	3129.77	299.91
2	135	1403.02	226.98	1403.02	220.14
2	151	1903.87	138.41	1903.87	140.11
2	200	2382.18	284.04	2382.18	279.97
2	256	40034.44	354.36	40034.44	353.36
2	301	15503.93	356.78	15503.93	356.17
2.5	22	442.45	0.64	442.45	0.61
2.5	26	535.14	465.43	535.14	450.84
2.5	30	477.66	0.37	477.66	0.36
2.5	36	584.86	46.77	584.86	47.50
2.5	45	727.17	11.54	727.17	10.90
2.5	51	809.32	408.82	809.32	401.21
2.5	72	247.82	234.08	247.82	232.06
2.5	101	1254.02	591.21	1253.97	600.03
2.5	121	2631.68	488.97	2631.68	489.07
2.5	135	1221.79	500.43	1221.79	488.30
2.5	151	1633.43	150.78	1633.43	151.51
2.5	200	2057.07	61.38	2057.07	61.68
2.5	256	31838.86	395.19	31838.86	392.07
2.5	301	13606.93	355.19	13606.93	359.59
3	22	401.68	1.58	401.68	1.66
3	26	493.80	22.67	493.80	23.30
3	30	477.66	0.18	477.66	0.16
3	36	547.62	23.59	547.62	23.62
3	45	705.95	20.73	705.95	19.87
3	51	751.34	332.57	751.34	331.49
3	72	239.85	4.58	239.85	4.62
3	101	1142.54	61.81	1142.54	62.35
3	121	2271.90	587.23	2271.90	567.79
3	135	1150.89	392.93	1150.89	379.63
3	151	1485.82	126.92	1485.82	122.26
3	200	1862.87	509.46	1862.87	506.89
3	256	26140.51	424.53	26140.51	422.12
3	301	12216.36	256.33	12216.36	251.73
Prosek		5776.307	220.4327	5776.298	218.5459

Tabela 4.7: *Eksperimentalni rezultati na test primerima iz skupova O i S*

	SET O					SET S				
	očekivan	GRASP-PR		F-GRASP-PR		očekivan	GRASP-PR		F-GRASP-PR	
n	obj	obj	t[s]	obj	t[s]	obj	obj	t[s]	obj	t[s]
10	26.83	26.83	0.08	26.83	0.07	19.98	19.98	0.04	19.98	0.04
13	35.78	35.78	0.07	35.78	0.07	26.65	26.65	0.06	26.65	0.07
16	38.83	38.83	0.07	38.83	0.07	35.41	35.41	0.06	35.41	0.07
19	53.66	52.44	0.17	52.44	0.16	39.97	38.18	0.25	38.18	0.25
21	51.78	51.78	0.06	51.78	0.06	47.22	47.22	0.08	47.22	0.09
25	71.55	64.88	1.92	64.88	1.88	53.29	49.72	47.14	49.72	47.10
28	62.83	62.83	0.05	62.83	0.05	61.80	61.80	0.13	61.80	0.13
31	77.66	77.66	1.44	77.66	1.43	70.82	70.02	1.43	70.02	1.43
37	83.78	83.78	0.07	83.78	0.07	82.39	82.39	3.72	82.39	3.66
40	86.83	86.83	0.64	86.83	0.65	98.59	98.59	0.17	98.59	0.17
41	103.56	101.98	18.35	101.98	18.30	94.44	88.00	169.16	88.00	167.57
49	143.11	101.29	255.79	101.29	255.71	106.73	81.74	346.24	81.74	347.27
52	110.83	110.83	0.15	110.83	0.15	118.71	118.71	1.14	118.71	1.19
53	115.78	115.78	0.88	115.78	0.86	123.11	123.11	0.66	123.11	0.66
55	125.66	125.66	8.05	125.66	7.88	134.02	132.93	186.12	132.93	179.44
64	134.83	134.83	1.44	134.83	1.44	150.60	150.60	1.89	150.60	1.84
69	147.78	147.78	2.67	147.78	2.60	166.12	166.12	1.32	166.12	1.33
73	167.55	167.55	27.88	167.55	28.14	164.79	162.33	280.68	162.33	274.93
79	173.66	173.66	40.69	173.66	41.87	197.19	197.16	3.92	197.16	3.88
81	207.11	176.33	266.79	176.33	260.79	170.08	135.45	437.32	135.45	445.36
85	179.78	179.78	2.37	179.78	2.30	201.18	201.18	1.75	201.18	1.77
94	194.83	194.83	6.03	194.83	5.95	214.27	212.48	7.18	212.48	7.05
103	221.66	221.66	182.86	221.66	178.84	247.72	246.76	317.31	246.76	310.95
105	231.55	231.55	129.35	231.55	126.98	246.22	245.09	101.40	245.09	100.92
125	259.78	259.78	3.93	259.78	3.91	285.69	283.31	14.38	283.31	14.55
127	269.67	269.67	178.15	269.67	177.23	307.72	307.26	82.31	307.26	83.93
137	295.55	295.55	358.18	295.55	348.82	335.94	336.21	570.62	336.21	583.91
145	335.11	339.97	450.48	339.97	438.47	310.78	282.60	276.87	282.60	283.11
169	359.55	359.55	388.99	359.55	373.60	410.30	415.88	492.20	415.88	509.00
187	389.67	389.67	316.28	389.67	314.80	428.54	424.89	52.61	424.89	53.46
209	463.11	511.23	572.91	511.23	568.41	494.82	477.12	151.05	477.12	150.10
249	519.55	529.78	406.99	529.78	412.11	599.98	599.98	249.02	599.98	249.13
273	591.10	654.66	526.58	654.66	527.41	664.48	675.02	48.66	675.02	48.55
337	719.10	720.36	296.27	719.56	56.53	833.86	869.53	198.32	836.14	140.97
Prosek	207.34	208.69	130.78	208.66	122.28	221.87	219.51	118.98	218.53	118.06

5

Rešavanje problema lokacije resursa

U ovom poglavlju su prikazane metode računarske inteligencije predložene za rešavanje dva lokacijska problema koja su predmet ove disertacije LOBA i MMDP. Za rešavanje problema LOBA predložene su dve hibridne metode, pa su najpre opisani evolutivni algoritam, kao i različite varijante metode promenljivih okolina. Sa druge strane, za rešavanje problema MMDP prikazana je jedna mogućnost upotrebe fazi pravila za podešavanje parametara evolutivnog algoritma u toku izvršavanja algoritma.

5.1 Evolutivni algoritmi

Evolutivni algoritmi (engl. *Evolutionary Algorithms* - EA) predstavljaju metaheurističke metode optimizacije koje imitiraju proces prirodne evolucije date populacije jedinki. Nastali su kao jedno od ostvarenja ideje da se iskoristi princip prirodne evolucije kod izrade algoritma. Teorija evolutivnih algoritama svoje utemeljenje pronalazi kao apstrakcija biološke evolucije. U cilju detaljnijeg opisa osnova ovih metoda, u daljem tekstu je naveden deo biološko računarske terminologije vezane za temu biološke evolucije. Živi organizmi se sastoje od ćelija, a svaka ćelija sadrži skup hromozoma, koji su izgrađeni od lanaca DNK (engl. DNA - *Deoxyribonucleic acid*) koji kodiraju informacije o odgovarajućem organizmu. Geni su zapravo funkcionalni blokovi DNK i svaki gen predstavlja odgovarajuće svojstvo organizma. U toku reprodukcije odabrani roditelji ukrštaju gene i time se formiraju novi hromozomi i na taj način se genetski materijal prenosi iz generacije u generaciju. Takođe, mutacija

dovodi do malih promena DNK elemenata. Kvalitet (prilagođenost) novonastalog organizma se meri njegovim uspehom u životu. I evolutivnim procesima živi organizmi postaju sve prilagođeniji uslovima u prirodi, tj. životnoj okolini. Upravo su na ovim principima evolucije izgrađeni i evolutivni algoritmi.

Američki naučnik Džon Holand (*John Henry Holland*) je u knjizi [102] predložio algoritam kao računarski proces koji imitira evoluciju u prirodi tako što ga primenjuje na apstraktne jedinke. Polazni cilj razvoja tog algoritma nije bila praktična primena zarad rešavanja nekog specifičnog problema, već formalna studija o evoluciji i adaptaciji u prirodi i načinima na koje je taj proces moguće ubaciti u računarstvo.

Analogija evolucije kao prirodnog procesa i evolutivnih algoritama kao metode optimizacije, ogleda se u procesu selekcije, ukrštanja i mutacije koji se zajedno nazivaju genetskim operatorima. Slično kao što su u evolucijskom procesu okolina i uslovi u prirodi ključ selekcije jedinki nad nekom vrstom živih bića, tako je i funkcija prilagođenosti ključ selekcije nad populacijom rešenja u slučaju EA. Naime, u prirodi jedinka koja je najbolje prilagođena uslovima i okolini u kojoj živi ima najveću verovatnoću preživljavanja i razmnožavanja, a time i prenošenja svog genetskog materijala na svoje potomke. Taj princip da bolje prilagođene jedinke prolaze u narednu generaciju i prenose svoja osobine na svoje potomke primenjuje se i kod evolutivnih algoritama i utiče na to da se očekuje da je svaka nova generacija jedinki bolje prilagođena nego prethodna. Za evolutivne algoritme jedinke su potencijalna rešenja problema koji se rešava, a svojstva jedinki su zapisana u hromozomima pomoću genetskog koda. Selekcijom se biraju dobre jedinke koje se prenose u sledeću populaciju, a ukrštanjem genetskog materijala stvaraju se nove jedinke. Takav ciklus selekcije, ukrštanja i mutacije genetskim materijalom jedinki ponavlja se sve dok nije zadovoljen uslov zaustavljanja evolucijskog procesa.

Evolutivni algoritmi su se, tokom poslednje četiri decenije, pokazali kao veoma uspešan opšti alat za rešavanje čitavog niza problema iz prakse. Veliki broj istraživača se bavio prilagođavanju evolutivnih algoritama različitim problema. Više o evolutivnim algoritmima može se naći u literaturi [31, 143, 165], a neke uspešne primene pri rešavanju NP-teških problema opisane su radovima [173, 130, 175, 116].

Prostor kodiranih rešenja kod EA

Definisanje prostora kodiranih rešenja X' zavisi u opštem slučaju od konkretnog oblika problema koji se rešava. Način kodiranja dopustivih rešenja, kao i azbuka simbola nad kojom se ono vrši trebalo bi da omoguće da se, sa što manjim brojem simbola i što prirodnije, izraze glavne karakteristike ovih rešenja. U osnovnoj varijanti EA koristi se tzv. binarno kodiranje u kome se svakoj tački iz X dodeljuje kôd unapred zadate dužine nad binarnom azbukom $\{0, 1\}$. Pri tome vrednosti simbola u kôdu, kao i njihove pozicije, zavise od tačke čiji je to kôd. Ovakav način kodiranja se prirodno nameće kada se problem koji se rešava formuliše kao zadatak matematičkog programiranja sa binarnim promenljivim i takvi problemi se najčešće veoma uspešno rešavaju evolutivnim algoritmima, što je slučaj i kod problema LOBA. Za neke probleme dobro su se pokazala i kodiranja nad azbukama veće kardinalnosti, o čemu se može naći i u radu [69]. Odabranom načinu kodiranja treba prilagoditi i genetske operatore. O korišćenju tradicionalnih genetskih operatora i za neka ne-binarna kodiranja može se videti u radu [20].

Definisanje početne populacije

Postoje različite strategije za dobijanje početne populacije. Na primer, početna populacija se može u celosti kreirati na slučajan način, ali i pomoću neke specijalne heuristike za rešavanje razmatranog optimizacionog problema. Da bi se obezbedila mogućnost EA da ispita različite delove prostora rešenja X', početna populacija treba da sadrži kodove što raznovrsnijih rešenja a to se najčešće postiže tako što se populacija generiše na slučajan način. Nekada je pogodno koristiti i heuristike za generisanje dela (ili cele) početne populacije, ali onda ona mora biti dobro izabrana i da je vreme izvršavanja relativno kratko. Za potrebe implementacije problema LOBA početna populacija je slučajno generisana, sa verovatnoćom p/m da vrednost bita bude 1 (inače 0). Broj članova populacije bi trebalo da bude dovoljno veliki da omogući kreiranje ovakve jedne raznovrsne populacije, ali ne previše veliki, jer bi u tom slučaju njeno generisanje zahtevalo mnogo vremena. U implementaciji EA za problem LOBA se dobro pokazala veličina populacije od 150 jedinki.

Jedan od bitnih aspekata prilikom implementacije EA je i način na koji se vrši zamena generacija za koju se najčešće koristi neki od sledećih pristupa [129].

- *Generacijski* pristup podrazumeva da treba u svakoj generaciji menjati sve

jedinke u populaciji.

- *Stacionarni* pristup se odnosi na taktiku da u svakoj generaciji treba generisati samo deo populacije, dok se preostale jedinke prenose iz prethodne populacije.
- *Elitistički* pristup zahteva da se u svaku generaciju propustiti određen broj elitnih jedinki na koje se ne primenjuju genetski operatori.

Funkcija prilagođenosti

Različiti načini računanja funkcije prilagođenosti se mogu pronaći u literaturi [172, 129]. Na osnovu specifičnosti samog problema koji se rešava određuje se odgovarajući način računanja funkcije prilagođenosti. U nekim situacijama se za funkciju prilagođenosti može uzeti funkcija cilja odgovarajućeg problema. Kako se u slučaju EA ne zahteva neprekidnost i glatkost funkcije prilagođenosti, on se može primeniti i u slučajevima kada nisu ispunjeni uslovi za primenu nekih klasičnih metoda.

Operator selekcije

Ovaj operator bira jedinke iz trenutne populacije koje će ostaviti potomke za sledeću generaciju. Najčešće se definiše tako da upravo one jedinke koje imaju bolju prilagođenost, imaju i veće šanse. Pristup koji je najjednostavniji i za shvatanje i za implementaciju je da se selekcijom biraju jedinke sa najvećom vrednošću funkcije prilagođenosti, nije uvek najbolji izbor jer često može da uzrokuje završavanje algoritma u lokalnom optimumu. Kako i loše prilagođene jedinke mogu imati neke dobre gene, ne treba u potpunosti eliminisati već im omogućiti šansu da rekombinacijom njihovih dobrih gena sa drugim jedinkama proizvedu dobro prilagođene jedinke.

Metod koji se češće koristi je selekcija koja je zasnovana na slučajnosti, ali tako da jedinke sa većom vrednošću funkcije prilagođenosti imaju veće šanse da budu izabrane. Ovakav vid selekcije se implementira korišćenjem ruleta, na kom svaka jedinka dobija svoj odsečak veličine koja je propocionalna vrednosti funkcije prilagođenosti za tu jedinku. Nedostatak proste rulet selekcije je mogućnost preuranjene konvergencije usled postepenog preovlađivanja visoko prilagođenih jedinki u populaciji koje ne odgovaraju globalnom optimumu. Ipak, često se koristi i kombinacija prethodno navedena dva metoda, prema kojem određeni broj najboljih jedinki

garantovano prelazi u narednu generaciju zarad ubrzavanja algoritma, dok se ostale jedinke biraju ruletnom selekcijom.

Jedna od popularnijih tehnika je i turnirska selekcija, kod koje se populacija na slučajan način deli u grupe od po n jedinki, koje se zatim „nadmeću" radi preživljavanja i prelaska u narednu generaciju [70]. Broj jedinki koje učestvuju na turniru je veličina turnira (u oznaci n), koja predstavlja parametar turnirske selekcije (koji se najčešće unapred zadaje). Ponekad je problem odabrati odgovarajući broj n, te postoji i metod koji omogućava da u proseku taj broj ne bude ceo. Taj metod se naziva fino gradirana turnirska selekcija (FGTS), a uvedena je i razmatrana u radovima [70, 72, 73, 71]. FGTS je primenjivana prilikom rešavanja različitih lokacijskih problema, kao na primer u radovima [172, 129].

Ukrštanje

Operator ukrštanja se u najopštijem slučaju, po ugledu na istoimeni genetski proces, definiše kao postupak u kome se slučajno uzajamno razmenjuju delovi kodova dva rešenja (roditelja) i tako dobijaju kodovi 2 nova rešenja (potomka). Razmena genetskog materijala jedinki-roditelja može se sprovesti pomoću jednopozicionog, dvopozicionog, višepozicionog ili uniformnog ukrštanja, ali postoje i drugi, složeniji, vidovi ovog genetskog operatora. Najjednostavniji je jednopozicioni operator ukrštanja kod koga se na slučajan način se bira broj $k \in \{0, 1, \ldots, N-1\}$, koji predstavlja tzv. tačku ukrštanja tj. poziciju u kodovima roditelja u odnosu na koju će razmeniti sadržaj. Zatim, simboli u ova dva koda, od pozicije $k+1$ do poslednje pozicije N, uzajamno razmenjuju mesta. Operator ukrštanja može biti definisan i u odnosu na 2 tačke ukrštanja. Kod dvopozicionog ukrštanja slučajno se biraju dve tačke ukrštanja $k_1, k_2 \in \{0, 1, \ldots, N-1\}$ i uzajamno se razmenjuju delovi kodova roditelja od pozicije $k_1 + 1$ do pozicije k_2.

Koji operator ukrštanja koristimo zavisi od samog problema koji rešavamo. U situacijama kada su geni međusobno zavisni, treba se opredeliti za jednopoziciono ukrštanje koje će delimično sačuvati strukturu genetskog koda. Sa druge strane, ukoliko su geni nezavisni među sobom najbolje je koristiti uniformno ukrštanje.

Operator mutacije

Operator mutacije je unarni operator koji se koristi da bi se povremeno unela raznovrsnost među jedinkama jedne populacije (posebno u slučaju kada su ove jedinke veoma slične), kao i da bi se sprečio neopravdan gubitak pojedinih simbola na nekim od pozicija kodova populacije. Tokom rada evolutivnog algoritma može se očekivati da će, usled uzajamnog ukrštanja kodova jedne generacije, doći do „konvergencije" u okviru trenutne populacije, tj. da će iz iteracije u iteraciju doći do sve veće i veće sličnosti između njenih kodova. To znači da vremenom dolazi do grupisanja populacije u jednoj uskoj oblasti prostora X'. Takva populacija može, usled forsiranja jedinki koje su bolje prilagođene, sadržati kodove visoko kvalitetnih rešenja i lokalizovati oblast koja verovatno sadrži kôd nekog lokalnog optimuma problema. Korišćenjem samo selekcije i ukrštanja, moglo bi doći do prerane konvergencije, tj. do preranog izjednačavanja kodova jedne populacije, čime bi se pretraživanje moglo „zaglaviti" u nekoj oblasti od X' koja ne sadrži kôd globalnog optimuma. Da bi se donekle sprečila ovakva prerana konvergencija koristi se mutacija (sa malom verovatnoćom p_{mut}) koja omogućava povratak raznolikosti u populaciju, a time i dalje rasejavanje po prostoru X'. Operator mutacije u opštem slučaju vrši, kao i odgovarajući genetski proces, promenu sadržaja koda (hromozoma) nekog rešenja (jedinke) slučajnom zamenom pojedinih simbola ovog koda sa nekim drugim simbolima iz azbuke simbola. Ovakav operator se primenjuje na svaki potomak a unapred zadatom verovatnoćom mutacije. Mutacija, u slučaju binarno kodiranih rešenja, podrazumeva promenu jednog bita iz 1 u 0, i obrnuto.

Pored toga, primenom operatora selekcije i ukrštanja, može se desiti da na određenoj poziciji sve (ili skoro sve) jedinke u populaciji imaju istu fiksiranu vrednost bita. Takve pozicije se nazivaju „zaleđeni" bitovi i u tom slučaju prostor pretrage je smanjen. Zbog toga se javila strategija da se u slučaju „zaleđenih" bitova mutacija vrši sa verovatnoćom nekoliko puta većom nego što je to slučaj za ostale bitove [173]. Time se postiže da se pretražuju i delovi prostora pretrage koji su bili izgubljeni usled zaleđenog bita.

Kriterijum zaustavljanja

Iterativno se smenjuju generacije jedinki u EA sve dok se ne ispuni kriterijum zaustavljanja. Mogu se koristiti različiti kriterijumi da bi se prekinulo izvršavanje

koraka EA, a najčešće se koriste:

- dostignut maksimalni, unapred zadati, broj generacija,
- sličnost jedinki u populaciji,
- ponavljanje najbolje jedinke maksimalni, unapred zadati, broj puta,
- ograničeno vreme izvršavanja algoritma i
- prekid od strane korisnika.

Nekada se može kao kriterijum zaustavljanja koristiti dostizanje optimalnog rešenja (ako je ono unapred poznato) i dokazana optimalnost najbolje jedinke (ukoliko je to moguće).

5.2 Evolutivni algoritam za problem LOBA

S obzirom na formulaciju problema LOBA, u implementaciji EA za ovaj problem, prirodno je bilo koristiti binarno kodiranje. Kodiranje jedinki je, za problem LOBA, je binarno, gde se svakom rešenju dodeljuje kôd unapred zadate dužine m, gde je m broj potencijalnih snabdevača ($m = |J|$). Jedinica na i-tom mestu podrazumeva da je i-ti snabdevač odabran da bude uspostavljen, a nula da nije uspostavljen.

U implementaciji evolutivnog algoritma za problem LOBA se dobro pokazala veličina populacije od 150 jedinki, a početna populacija je slučajno generisana, sa verovatnoćom p/m da vrednost bita bude 1 (inače 0). Na ovaj način, omogućava se raznolikost genetskog materijala početne populacije. Međutim kako neke od na taj način generisanih jedinki ne predstavljaju dopustiva rešenja (tj. imaju sumu bitova različitu od p), pa se odmah izvrši odgovarajući broj slučajno odabranih zamena bitova (1 u 0 ili 0 u 1) u cilju postizanja tačno m bitova postavljenih na 1 u genetskom kodu. Genetski operatori koji se nakon toga primenjuju sprečavaju nastajanje nedopustivih jedinki, pa se time obezbeđuje da se samo jedinke koje odgovaraju dopustivim rešenjima mogu pojaviti u narednim generacijama.

Funkcija prilagođenosti, u slučaju problema LOBA, je formulisana tako da one jedinke koje imaju najbolju vrednost funkcije cilja (za problem LOBA to je najmanja vrednost, poželjno 0) slika u 0, a one koji imaju najlošiju vrednost funkcije cilja slika

u 1. Skaliranje u jedinični interval podrazumeva da funkcija prilagođenosti uzima vrednosti iz intervala $[0, 1]$.

Nije poželjno u populaciji imati dve identične jedinke. To se reguliše, tako što se ukoliko se neka jedinka već pojavila, svakoj takvoj se vrednost funkcije prilagođenosti postavlja na nulu i na taj način se smanjuje verovatnoća prelaska takve jedinke u narednu populaciju. Na isti način se postupa i ukoliko se u populaciji javlja više od 10 jedinki sa istom vrednošću funkcije cilja.

Implementacija EA za problem LOBA koristi koncept stacionarnog EA sa elitističkom strategijom. Trećina populacije se zamenjuje u svakoj generaciji, i oni nastaju primenom operatora selekcije, ukrštanja i mutacije na sve jedinke iz prethodne populacije. Dve trećine tj. 100 najbolje prilagođenih jedinki se direktno propusta u sledeću generaciju, čime se skraćuje vreme izvršavanja algoritma i obezbeđuje čuvanje dobrih rešenja. Prednost elitističkog pristupa, pored uštede procesorskog vremena, je i to što se kvalitet rešenja može bolje predvideti.

Genetski operatori i vrednosti parametara su odabrani sa ciljem da se uspostavi balans između efikasnosti i kvaliteta rešenja. U EA implementaciji za problem LOBA su korišćeni:

- turnirska selekcija sa parametrom 5;
- za dve odabrane jedinke sa verovatnoćom 0.85 primenjuje se operator dvopozicionog ukrštanja sa slučajnim odabirom dve tačke ukrštanja;
- mutacija sa zaleđenim bitovima, tako da je verovatnoća mutacije $p_{mut} = 0.4/m$ (gde je m dužina binarnog koda svake jedinke). Posebno, u slučaju zaleđenih bitova, verovatnoća mutacije se množi dodatno sa odabranim faktorom 3.5.

Kriterijum zaustavljanja koji se koristi je dostignuti maksimalni broj generacija: 10000. U cilju implementacije EA za probleme u ovoj disertaciji korišćenja biblioteka funkcija otvorenog koda GAFramework[1].

5.3 Metoda promenljivih okolina (VNS)

Metoda promenljivih okolina (engl. *Variable neighborhood search* - VNS) je metaheuristika za rešavanje problema matematičke optimizacije čija je osnovna ideja

[1] https://code.google.com/archive/p/gaframework/source

sistematska promena okolina u fazama lokalne pretrage, kao i fazama izlaska iz lokalnih optimuma u cilju nalaženja globalnog optimuma [145, 95, 94, 96]. Lokalno pretraživanje je jedna od najviše korišćenih heuristika podrazumeva da se krenuvši od početnog rešenja, nizom uzastopnih lokalnih promena kojima se svaki put poboljšava vrednost funkcije cilja, dostiže lokalni optimum.

Osnovna šemu VNS-a je prvi put predstavljena u radu [144], gde je navedeno da se VNS može implementirati pomoću bilo kog algoritama lokalnog pretraživanja kao potprograma, a prikazana je efikasnost i ilustrovana su poboljšanja na primeru problema trgovačkog putnika. Od tada pa do danas metoda promenljivih okolina ima brojne primene u rešavanju različitih problema [93, 9, 156, 38, 37, 180].

Neka je N_k $(k = 1, \ldots, k_{max})$ konačan skup struktura okolina, koje se kraće mogu zvati okolinama. Sa $N_k(x)$ se označava skup rešenja koji se nalaze u k-oj okolini rešenja x. Kako se navodi u radu [94], većina heuristika, koje koriste lokalnu pretragu, koristi jednu okolinu, tj. $k_{max} = 1$. Okoline se određuju su u skladu sa problemom koji se rešava i mogu biti izvedene na osnovu različitih metrika.

Važna komponenta VNS metode je takozvana faza razmrdavanja (engl. *shaking*) koja podrazumeva izlazak u k-tu okolinu od polazne tačke x, tj. generiše se tačka x' iz k-te okoline rešenja x $(x' \in N_k(x))$.

Pored faze razmrdavanja, važna komponenta je i promena okolina u kojoj se trenutno rešenje x poredi sa rešenjem x' iz k-te okoline rešenja x. Ukoliko je x' bolje od rešenja x, trenutno rešenje se ažurira tj. nastavlja se od tačke x' i vrednost k se postavlja na 1 tj. pretražuje se okolina N_1. Ukoliko poboljšanje nije postignuto, tj. x' nije bolje od rešenja x, samo se k uvećava za 1 tj. posmatra se naredna okolina.

Detaljnije o VNS metodi se može naći u radovima Mladenovića i Hansena [145, 95, 96]. U nastavku će biti prikazane osnovna i redukovana metoda promenljivih okolina, a o različitim varijantama VNS metode može se pročitati u [97].

Osnovna metoda promenljivih okolina

Osnovna metoda promenljivih okolina (engl. *Basic Variable Neighborhood Search* - BVNS), koja se uglavnom naziva samo metodom promenljivih okolina je podrazumeva kombinaciju faze razmrdavanja, lokalne pretrage i promene okolina. Ovi koraci se ponavljaju sve dok neki unapred zadat kriterijum zaustavljanja ne postane ispunjen. Pri tome se kod lokalne pretrage mogu primeniti naredne dve strategije u

pronalaženju boljeg rešenja x' u okolini rešenja x nakon čega se pretraga nastavlja od rešenja x'. Najčešće se od mogućih rešenja u okolini bira najbolje za potencijalno x' i to je strategija najboljeg poboljšanja. Međutim u situacijama kada je ta strategija vremenski previše zahtevna primenjuje se strategija prvog poboljšanja koja podrazumeva da se kada se u okolini pronađe prvo rešenje x' koje je bolje od x, trenutno rešenje se ažurira na x' i pretraga se nastavlja od njega.

Može se primetiti da su metode GRASP i VNS komplementarne u smislu da je slučajnost uključena u fazu konstrukcije kod metode GRASP, dok se kod VNS u fazi razmrdavanja na slučajan način bira rešenje nad kojim će se primeniti lokalna pretraga.

Redukovana metoda promenljivih okolina

Redukovana metoda promenljivih okolina (engl. *Reduced Variable Neighborhood Search* - RVNS) ima za cilj poboljšanje efikasnosti i primenjuje se najčešće za rešavanje test primera velikih dimenzija. Lokalna pretraga je često vremenski najzahtevniji deo osnovnog VNS algoritma. U metodi RVNS ponavljaju se koraci razmrdavanja i zamene okolina sve dok neki odabrani kriterijum zaustavljanja ne postane ispunjen.

5.4 RVNS-VNS metod za rešavanje problema LOBA

Za rešavanje problema LOBA implementirana je osnovna metoda promenljivih okolina bazirana na strategiji prvog poboljšanja.

U predloženom VNS konceptu, svako rešenje problema LOBA je predstavljeno kao permutacija $(i_1, ..., i_m)$ elemenata skupa $\{1, 2, .., m\}$, gde je $m = |J|$. Pri tome, prvih p brojeva u permutaciji su redni brojevi snabdevača koje treba uspostaviti.

Način kako se definišu strukture okolina je veoma značajan za uspešnu primenu VNS-a. Okoline datog rešenja S je potrebno definisati uzimajući u obzir prirodu problema i načina na koji su rešenja kodirana. U slučaju problema LOBA, u okviru ove disertacije, odabrano je da strukture okolina budu definisane na sledeći način. Ukoliko nasumično jednog uspostavljenog snabdevača u rešenju S zamenimo sa nasumično odabranim neuspostavljenim snabdevačem, dobija se rešenje S' iz okoline N_1 inicijalnog rešenja S. Ukoliko se dva slučajno odabrana uspostavljena snab-

devača zamene se dva koji nisu uspostavljena, dobija se rešenje S'' iz okoline N_2 rešenja S, itd. Zapravo, ako se dva rešenja S_1, S_2 razlikuju u tačno k uspostavljenih snabdevača ($k = 1, 2, ..$), onda važi $S_1 \in N_k(S_2)$, kao i obrnuto $S_1 \in N_k(S_1)$.

Zbog svoje efikasnosti za rešavanje problema LOBA implementirana je i redukovana metoda promenljivih okolina (RVNS) koja je kombinovana sa klasičnom VNS metodom na sledeći način. Inicijalno rešenje za metod RVNS se bira na slučajan način i RVNS se izvršava dok se ne dostigne maksimalan broj iteracija *max-iter-RVNS*. Najbolje rešenje dobijeno primenom metode RVNS se koristi kao inicijalno rešenje za VNS metod. Pre glavne VNS petlje, na najbolje rešenje I dobijeno u RVNS fazi primenjuje se procedura $ProceduraPopravljanja$ bazirana na mnogostrukoj primeni lokalne pretrage. Na taj način dobija se dodatno poboljšanje trenutno najboljeg rešenja koje omogućava da se VNS metoda kreće po obećavajućim okolinama. Koraci u VNS metodi se izvršavaju *max-iter-VNS* puta.

Broj iteracija za za RVNS i VNS u hibridnoj RVNS-VNS metodi su postavljeni na $min\{n, n/3\}$ za RVNS i 10 000 za VNS metod. Dodatan kriterijum zaustavljanja je ukupno vreme izvršavanja $t_{max} = 100$ sekundi (tj. $t_{RVNS}+t_{VNS} < t_{max}$). Pseudokod za metodu RVNS-VNS prikazan je algoritmom 3.

5.5 Hibridni EA-VNS metod za rešavanje problema LOBA

Za rešavanje problema LOBA implementiran je i hibrid evolutivnog algoritma i VNS metode, koji će biti označavan sa EA-VNS. U prvoj fazi se primeni EA za dobijanje kvalitetnih inicijalnih rešenja za VNS metodu koja se koristi u drugoj fazi. EA se izvršava određeni broj generacija koji predstavlja kompromis između korišćenja vremena i kvaliteta rešenja. Najbolja jedinka dobijena primenom EA metode se prekodira u VNS reprezentaciju rešenja i koristi kao početno rešenje u VNS fazi.

Za EA-VNS metod koristi se više kriterijuma zaustavljanja. EA se zaustavlja kada se dostigne maksimalan broj generacija (10 000), dok je maksimalan broj iteracija za VNS metod ograničen sa 100 000. Pored toga, ograničenje je postavljeno i na ukupno vreme izvršavanja EA-VNS metoda na maksimalnih $t_{max} = 100$ sekundi (i.e. $t_{EA} + t_{VNS} < t_{max}$).

Algoritam 3 RVNS-VNS metod za problem LOBA

RVNS algoritam:
Kreiranje inicijalnog rešenja S na slučajan način;
$p \leftarrow$ je broj uspostavljenih snabdevača u S;
$M \leftarrow min(p, n-p)$;
$iter \leftarrow 1$;
while $iter \leq max\_iter\_RVNS$ **do**
    $k \leftarrow 1$;
    while $k \leq M$ **i** $iter \leq max\_iter\_RVNS$ **i** $t_{RVNS} < t_{max}$ **do**
        $S' \leftarrow FazaRazmrdavanja(S, k), S' \in N_k(S)$;
        if $funkcijaCilja(S') \leq funkcijaCilja(S)$ **then**
            $S \leftarrow S'$; //Pretraga se nastavlja od okoline N_1 rešenja S';
            $k \leftarrow 1$;
        else
            $k \leftarrow k + 1$;
        $iter \leftarrow iter + 1$;
Neka je I najbolje rešenje dobijeno pomoću RVNS;
VNS algoritam:
$S \leftarrow ProceduraPopravljanja(I)$;
$p \leftarrow$ je broj uspostavljenih snabdevača u S;
$M \leftarrow min(p, n-p)$;
$iter \leftarrow 1$;
while $iter \leq max\_iter\_VNS$ **i** $t_{RVNS} + t_{VNS} < t_{max}$ **do**
    $k \leftarrow 1$;
    while $k \leq M$ **i** $iter \leq max\_iter\_VNS$ **do**
        $S' \leftarrow FazaRazmrdavanja(S, k), S' \in N_k(S)$;
        $S'' \leftarrow LokalnaPretraga(S')$;
        if $funkcijaCilja(S'') \leq funkcijaCilja(S')$ **then**
            $S \leftarrow S''$; //Pretraga se nastavlja od okoline N_1 rešenja S'';
            $k \leftarrow 1$;
        else
            $k \leftarrow k + 1$;
        $iter \leftarrow iter + 1$;

Osnovna šema za RVNS-VNS prikazan je algoritmom 4.

5.6 Eksperimentalni rezultati za problem LOBA

U ovoj sekciji su prikazani eksperimentalni rezultati za predložene metode za rešavanje problema LOBA, uz odgovarajuća poređenja rezultata. Testiranja metoda EA, VNS, RVNS-VNS i EA-VNS izvršena su na računaru sa Intel Core i7-860 procesorom sa 2.8 GHz, 8GB RAM memorije i na Windows 7 Professional operativnom sistemu. Rešavač CPLEX 12.1[2] je korišćen za dobijanje optimalnih rešenja (ukoliko je moguće) za razmatrane test primere i izvršavan je na istoj platformi. Sve implementacije su urađene u programskom jeziku $C\sharp$, na .NET platformi.

Tabele 5.1-5.2 sadrže eksperimentalne rezultate za skup primera „Datos", kao i poređenje sa egzaktnim BnC metodom i heuristikom HEUR iz radu [135]. Napomena je da su BnC i HEUR testirani pod Linux operativnim sistemom na računaru Pentium IV sa 2.5 GHz i 3 GB RAM memorija.

Tabele 5.1 i 5.2 imaju isti raspored kolona. Prva kolona sadrži ime test primera i parametre m, n, p i p_l. Ostali parovi kolona sadrže optimalna/najbolja rešenja i odgovarajuća vremena izvršavanja u sekundama za:

- rešavač CPLEX 12.1
- egzaktni metod BnC iz [135];
- heuristički metod Heur iz [135];
- evolutivni algoritam (EA);
- osnovna metoda promenljivih okolina (VNS);
- hibridizacija redukovane i osnovne metode promenljivih okolina (RVNS-VNS);
- hibridizacija evolutivnog algoritma i osnovne metode promenljivih okolina (EA-VNS).

U slučajevima da neka od metoda nije dala rešenja na odgovarajućem mestu u tabeli je znak „-".

[2] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Algoritam 4 EA-VNS hibridni metod

```
1:
2: EA metod:
3: Inicijalizacija:
4: Generisanje početne populacije P;
5: Odabir kriterijuma zaustavljanja: max_iter_EA, t_max;
6: iter ← 1;
7: while iter ≤ max_iter_EA i t_EA < t_max do
8:     Računanje funkcijaCilja(X) za svaku jedinku X ∈ P;
9:     Selekcija;
10:    Ukrštanje;
11:    Mutacija;
12:    iter ← iter + 1;
13:
14: Neka je I najbolje rešenje dobijeno pomoću EA;
15: Kodiranje rešenja I u skladu sa VNS predstavljenjem rešenja;
16:
17: VNS algoritam:
18: S ← ProceduraPopravljanja(I);
19: p ← je broj uspostavljenih snabdevača u S;
20: M ← min(p, n − p);
21: iter ← 1;
22: while iter ≤ max_iter_VNS i t_EA + t_VNS < t_max do
23:     k ← 1;
24:     while k ≤ M i iter ≤ max_iter_VNS i t_EA + t_VNS < t_max do
25:         S' ← FazaRazmrdavanja(S, k), S' ∈ N_k(S);
26:         S'' ← LokalnaPretraga(S');
27:         if funkcijaCilja(S'') ≤ funkcijaCilja(S') then
28:             S ← S''; //Pretraga se nastavlja od okoline N_1 rešenja S'';
29:             k ← 1;
30:         else
31:             k ← k + 1;
32:         iter ← iter + 1;
```

Tabela 5.1: *Poređenje na test primerima iz skupa „Datos", $m \leq 30$*

Instanca	CPLEX 12.1		BnC		Heur		EA		VNS		RVNS-VNS		EA-VNS	
m n p p_l	Opt	t[s]	Opt	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]
20 20 3 1	1	3.688	-	-	-	-	opt	0.000	opt	0.000	opt	0.004	opt	0.011
20 20 3 10	15	3.547	15	27	opt	0	opt	0.040	opt	0.008	opt	0.013	opt	0.027
20 20 3 50	3	2.274	3	1	opt	1	opt	0.010	opt	0.007	opt	0.010	opt	0.024
20 20 6 1	1	3.318	-	-	-	-	opt	0.000	opt	0.000	opt	0.009	opt	0.009
20 20 6 10	12	1.631	12	19	opt	0	opt	0.010	opt	0.010	opt	0.010	opt	0.031
20 20 6 50	6	3.169	6	17	opt	0	opt	0.010	opt	0.000	opt	0.009	opt	0.024
30 30 3 1	0	4.947	-	-	-	-	opt	0.000	opt	0.000	opt	0.004	opt	0.009
30 30 3 10	25	38.376	25	1080	opt	3	opt	0.010	opt	0.000	opt	0.004	opt	0.022
30 30 3 30	22	34.019	22	11	opt	3	opt	0.160	opt	0.010	opt	0.005	opt	0.034
30 30 3 100	9	35.340	9	1035	opt	3	opt	0.040	opt	0.000	opt	0.005	opt	0.034
30 30 4 1	1	30.935	-	-	-	-	opt	0.000	opt	0.000	opt	0.004	opt	0.010
30 30 4 10	24	32.643	24	7	opt	3	opt	0.010	opt	0.007	opt	0.009	opt	0.023
30 30 4 30	17	43.394	17	13	opt	3	opt	0.020	opt	0.000	opt	0.004	opt	0.024
30 30 4 100	9	74.433	9	320	10	3	opt	0.540	opt	0.000	opt	0.005	opt	0.028
30 50 3 10	45	46.645	45	24	opt	4	opt	0.010	opt	0.008	opt	0.010	opt	0.030
30 50 3 50	34	56.436	34	17	opt	6	opt	0.010	opt	0.009	opt	0.016	opt	0.021
30 50 3 100	27	79.010	27	19	opt	6	opt	0.010	opt	0.000	opt	0.005	opt	0.021
30 50 3 200	13	102.220	13	16	opt	5	opt	0.020	opt	0.000	opt	0.004	opt	0.020
30 50 6 10	45	19.897	45	25	opt	4	opt	0.770	46	0.010	opt	0.007	opt	10.366
30 50 6 50	34	9.366	34	23	opt	5	opt	0.030	opt	0.013	opt	0.019	opt	0.029
30 50 6 100	19	52.992	19	23	opt	4	opt	0.100	opt	0.013	opt	0.018	opt	0.040
30 50 6 200	10	17.154	10	23	opt	5	opt	1.110	opt	0.023	opt	0.031	opt	0.035
30 50 10 10	45	6.728	45	86	46	3	opt	0.090	opt	0.206	opt	0.071	opt	0.020
30 50 10 50	27	8.033	27	85	opt	4	opt	0.040	opt	0.000	opt	0.008	opt	0.038
30 50 10 100	18	40.577	18	68	20	3	opt	0.790	opt	0.154	opt	0.107	opt	0.309
30 50 10 200	5	44.412	5	57	opt	3	opt	0.990	opt	0.020	opt	0.027	opt	0.058
30 100 3 10	94	48.340	94	92	opt	9	opt	0.060	opt	0.009	opt	0.011	opt	0.023
30 100 3 50	85	400.760	85	90	opt	9	opt	0.080	opt	0.000	opt	0.012	opt	0.018
30 100 3 100	75	412.056	75	89	opt	14	opt	0.010	opt	0.000	opt	0.004	opt	0.028
30 100 3 200	63	315.694	63	88	opt	9	opt	0.430	opt	0.008	opt	0.013	opt	0.024
30 100 6 10	92	48.749	92	85	opt	10	opt	1.550	opt	0.039	opt	0.047	opt	0.049
30 100 6 50	84	874.667	84	96	opt	8	opt	0.040	opt	0.015	opt	0.019	opt	0.024
30 100 6 100	64	401.656	64	160	opt	8	opt	0.650	opt	0.040	opt	0.037	opt	0.097
30 100 6 200	48	411.906	48	148	opt	8	opt	0.300	opt	0.019	opt	0.028	opt	0.053
30 100 10 10	96	32.902	96	651	opt	5	opt	0.010	opt	0.000	opt	0.022	opt	0.021
30 100 10 50	77	139.707	77	510	opt	5	opt	0.090	opt	0.040	opt	0.067	opt	0.052
30 100 10 100	63	129.800	63	666	opt	8	opt	0.250	opt	0.000	opt	0.045	opt	0.062
30 100 10 200	41	75.513	41	492	opt	8	opt	0.060	opt	0.000	opt	0.038	opt	0.127
Prosek		107.55		181.26		5.06		0.22		0.02		0.02		0.31

Tabela 5.2: *Poređenje na test primerima iz skupa „Datos”,* $50 \leq m \leq 100$

Instanca	CPLEX 12.1		BnC		Heur		EA		VNS		RVNS-VNS		EA-VNS	
m n p p_l	Opt	t[s]	Opt	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]
50 50 3 10	46	148.63	46	232	opt	34	opt	0.01	opt	0.00	opt	0.01	opt	0.02
50 50 3 50	39	423.94	39	123	opt	45	opt	0.61	opt	0.01	opt	0.01	opt	0.11
50 50 3 100	32	463.13	32	158	opt	35	opt	0.02	opt	0.00	opt	0.00	opt	0.02
50 50 3 400	11	370.74	11	202	opt	43	opt	0.68	opt	0.01	opt	0.01	opt	0.04
50 50 6 10	43	233.78	43	421	opt	38	opt	2.17	opt	0.02	opt	0.02	opt	17.22
50 50 6 50	33	3264.01	33	264	opt	37	opt	0.70	opt	0.00	opt	0.02	opt	0.08
50 50 6 100	24	7926.63	24	301	opt	36	opt	0.57	opt	0.04	opt	0.11	opt	0.05
50 50 6 400	3	1376.23	3	194	opt	37	opt	0.46	opt	0.03	opt	0.07	opt	0.10
50 50 10 10	43	223.64	43	412	45	25	opt	1.16	opt	0.19	opt	0.49	opt	15.11
50 50 10 50	30	238.75	30	383	opt	30	opt	0.28	opt	0.03	opt	0.03	opt	0.05
50 50 10 100	20	442.00	20	406	23	33	opt	1.47	opt	0.26	opt	2.15	opt	5.40
50 50 10 400	2	1802.68	2	539	opt	28	opt	1.52	opt	0.11	opt	0.07	opt	0.04
50 100 3 10	96	1416.08	96	1085	opt	62	opt	0.01	opt	0.01	opt	0.02	opt	0.01
50 100 3 50	87	1065.99	87	1107	opt	61	opt	0.07	opt	0.00	opt	0.01	opt	0.07
50 100 3 100	79	1126.60	79	1613	opt	72	opt	0.02	opt	0.00	opt	0.00	opt	0.08
50 100 3 400	48	1749.61	48	636	opt	89	opt	0.02	opt	0.00	opt	0.01	opt	0.02
50 100 6 10	95	1665.54	-	-	95	95	opt	0.02	opt	0.00	opt	0.01	opt	0.02
50 100 6 50	83	13273.56	-	-	83	51	opt	1.18	opt	0.03	opt	1.85	opt	0.12
50 100 6 100	70	110415.51	-	-	76	73	opt	0.17	opt	0.04	opt	0.09	opt	0.04
50 100 6 400	35	7592.41	-	-	35	-	opt	0.31	opt	0.07	opt	0.05	opt	0.07
50 100 10 10	96	1056.70	96	504	opt	-	opt	0.02	opt	0.02	opt	0.03	opt	0.02
50 100 10 50	83	31282.51	83	500	opt	-	opt	0.20	opt	0.00	opt	0.02	opt	0.08
50 100 10 100	69	11380.78	69	518	opt	-	opt	0.58	opt	3.16	opt	40.26	opt	0.17
50 100 10 400	18	18080.49	-	-	opt	-	opt	0.77	opt	0.08	opt	0.12	opt	0.08
100 100 3 1	-	-	-	-	-	-	1	0.00	1	0.00	1	0.01	1	0.01
100 100 3 10	96	19296.37	-	-	-	-	opt	0.16	opt	0.02	opt	0.02	opt	0.07
100 100 3 50	-	-	-	-	-	-	91	0.04	91	0.00	91	0.00	91	0.02
100 100 3 100	-	-	-	-	-	-	86	1.18	86	0.00	86	0.01	86	0.07
100 100 3 1000	-	-	-	-	-	-	41	0.67	41	0.02	41	0.03	41	0.02
100 100 3 2000	-	-	-	-	-	-	11	0.22	11	0.00	11	0.02	11	0.05
100 100 3 2500	-	-	-	-	-	-	3	0.02	3	0.00	3	0.03	3	0.12
100 100 3 3000	-	-	-	-	-	-	1	0.49	1	0.00	1	0.01	1	0.15
100 100 3 3500	-	-	-	-	-	-	1	0.12	1	0.00	1	0.01	1	0.05
100 100 3 5000	-	-	-	-	-	-	1	0.01	1	0.00	1	0.01	1	0.01
Prosek		9452.65		505.16		48.63		0.47		0.12		1.34		1.17

Na osnovu rezultata iz tabele 5.1 može se primetiti da na svim manjim i srednjim test primerima skupa „Datos" sa do 30 korisnika i 100 potencijalnih snabdevača, sva četiri predložena metoda brzo dostižu optimalna rešenja prethodno dobijena pomoću CPLEX 12.1 rešavača. Jedini izuzetak među predloženim metodama je metoda VNS i to samo na test primeru 30-50-6-10, dok metode BnC i HEUR iz [135] nisu dale rešenje za test primere 20-20-3-1, 20-20-6-1, 30-30-3-1 i 30-30-4-1. Za veće test primere iz skupa „Datos" rezultati su predstavljeni u tabeli 5.2 i sva četiri predložena metoda su dostigla optimalna rešenja za veoma kratko vreme. Metod BnC nije rešio 5 takvih „Datos" test primera sa 50 korisnika i 100 potencijalnih snabdevača, dok heuristika Heur se pokazala najlošije s obzirom da nije dostigla optimalna rešenja na 6 test primera. Pored toga, test primeri iz skupa „Datos" sa 100 korisnika i 100 potencijalnih snabdevača su bili nedostižni za obe metode BnC i Heur iz [135]. Takođe, CPLEX 12.1 rešavač nije mogao da reši ove test primere zbog memorijskih ili vremenskih ograničenja, osim za test primer 100-100-3-10. Na osnovu rezultata iz tabele 5.2, može se videti da su sva četiri predložena metoda rešavala test primere iz skupa „Datos" sa 100 korisnika i 100 potencijalnih snabdevača u kratkom vremenu izvršavanja dostižući ista najbolja rešenja. Na kraju tabela mogu se videti prosečna vremena za svaku od metoda. Ta vremena ukazuju da su se na skupu podataka „Datos" najefikasnije pokazale metode VNS iRVNS-VNS.

Rezultati i poređenja na test primerima iz skupa „Pmed" su predstavljeni u tabeli 5.3 na isti način kao i podaci u prethodne dve tabele (tabele 5.1 i 5.2). Može se videti iz tabele 5.3 da su obe predložene hibridne metode RVNS-VNS i EA-VNS veoma brzo dostigle optimalna rešenja koja su dobijena primenom rešavača CPLEX ili BnC metode. EA nije dostigao rešenja u slučaju jednog test primera 90-90-7, a VNS metod u slučaju test primera 100-100-5. Kada je vreme izvršavanja u pitanju, EA-VNS metod se bolje pokazao u odnosu na ostale predložene metode. Vreme izvršavanja za metod EA-VNS je bilo do 0.2 sekundi, dok su vremena izvršavanja za RVNS-VNS, VNS, EA do 1.570, 3.4 i 0.7 sekundi redom.

BnC metod se pokazao uspešnim u nalaženju optimalnih rešenja za sve testirane instance iz skupa „Pmed", a sva optimalna rešenja je dostigla i heuristika Heur. Međutim, obe ove metode BnC i HEUR iz [135] su zahtevale znatno duže vreme izvršavanja u odnosu na metode predložene u ovom radu. Na skupu test primera „Pmed", maksimalno vreme izvršavanja algoritma BnC je 1490 sekundi, a heuristike Heur 303 sekundi. Na kraju tabele mogu se videti prosečna vremena za svaku od

Tabela 5.3: *Poređenje na test primerima iz skupa „Pmed”*

Instanca	CPLEX 12.1		BnC		Heur		EA		VNS		RVNS-VNS		EA-VNS	
m n p	Opt	t[s]	Opt	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]
20 20 3	1	3.80	1	0	opt	0	opt	0.00	opt	0.00	opt	0.00	opt	0.00
20 20 5	0	1.02	0	0	opt	0	opt	0.00	opt	0.00	opt	0.00	opt	0.00
20 20 7	1	1.64	1	0	opt	0	opt	0.01	opt	0.00	opt	0.00	opt	0.00
30 30 3	0	4.44	0	0	opt	0	opt	0.00	opt	0.00	opt	0.00	opt	0.00
30 30 5	0	17.03	0	0	opt	0	opt	0.18	opt	0.00	opt	0.00	opt	0.00
30 30 7	-	-	1	1	opt	1	opt	0.40	opt	0.00	opt	0.00	opt	0.00
40 40 3	1	146.84	1	0	opt	0	opt	0.00	opt	0.00	opt	0.00	opt	0.00
40 40 5	0	43.68	0	4	opt	4	opt	0.26	opt	0.00	opt	0.00	opt	0.00
40 40 7	1	305.95	1	7	opt	7	opt	0.18	opt	0.00	opt	0.00	opt	0.00
50 50 3	1	1080.87	1	0	opt	0	opt	0.00	opt	0.00	opt	0.00	opt	0.00
50 50 5	0	185.70	0	56	opt	40	opt	0.84	opt	0.00	opt	0.00	opt	0.00
50 50 7	1	4345.88	1	48	opt	40	opt	0.34	opt	0.20	opt	0.10	opt	0.10
60 60 3	0	68.46	0	0	opt	0	opt	0.07	opt	0.00	opt	0.00	opt	0.00
60 60 5	0	580.92	0	13	opt	13	opt	0.31	opt	0.00	opt	0.00	opt	0.00
60 60 7	1	15740.64	1	13	opt	13	opt	0.59	opt	0.00	opt	0.00	opt	0.00
70 70 3	1	5016.17	1	10	opt	10	opt	0.01	opt	0.00	opt	0.00	opt	0.00
70 70 5	0	1126.66	0	103	opt	103	opt	0.42	opt	0.10	opt	0.40	opt	0.00
70 70 7	0	10678.25	0	1005	opt	123	opt	1.57	opt	3.40	opt	0.70	opt	0.20
80 80 3	1	116936.73	1	5	opt	5	opt	0.00	opt	0.00	opt	0.00	opt	0.01
80 80 5	0	7568.90	0	62	opt	62	opt	0.63	opt	0.10	opt	0.00	opt	0.01
80 80 7	-	-	1	254	opt	233	opt	0.50	opt	0.10	opt	0.10	opt	0.01
90 90 3	0	7497.11	0	1	opt	1	opt	0.04	opt	0.00	opt	0.00	opt	0.01
90 90 5	0	67416.41	0	1475	opt	303	opt	0.33	opt	0.00	opt	0.20	opt	0.01
90 90 7	-	-	1	39	opt	39	2	1.05	opt	0.00	opt	0.10	opt	0.01
100 100 3	-	-	1	9	opt	9	opt	0.02	opt	0.00	opt	0.00	opt	0.01
100 100 5	-	-	0	921	opt	111	opt	0.52	2	0.00	opt	0.00	opt	0.01
100 100 7	-	-	1	1490	opt	141	opt	1.27	opt	0.40	opt	0.20	opt	0.01
Prosek		11369.86		204.30		46.59		0.35		0.16		0.07		0.01

metoda. Ta vremena ukazuju da se na skupu test primera „Pmed" najefikasnije pokazala metoda EA-VNS.

Tabela 5.4 sadrži rezultate za test primere iz skupa „Galvao", a tabela 5.5 za skup „SJC". Ova dva skupa test primera („Galvao" i „SCJ") nisu prethodno razmatrani u literaturi za problem LOBA, pa su prikazani samo rezultati dobijeni pomoću četiri predložene metode. Pored toga, pokušano je i dobijanje optimalnih rešenja uz pomoć rešavača CPLEX 12.1, ali rešenja nisu dobijena ni nakon nekoliko dana izvršavanja često zbog memorijskih ograničenja. Sa druge strane, četiri razmatrana heuristička pristupa su dala rešenja za test primere iz oba skupa „Galvao" i „SCJ" u relativno kratkom vremenu izvršavanja (često ispod jednog minuta), ali su dobijena rešenja različitog kvaliteta. Stoga, tabele 5.4 i 5.5 imaju jednu dodatnu kolonu „najbolji metod" koja sadrži podatke koji metod je dao najbolje rešenje na svakoj od test primera. U situacijama kada su sve četiri metode dale isto rešenje u koloni je ispisano „ista rešenja".

Na osnovu tabele 5.4 može se uočiti da je za test primere iz skupa „Galvao", koji imaju sa 100 i 150 korisnika i isti broj potencijalnih snabdevača, metod EA-VNS dostigao najbolja rešenja na 8, RVNS-VNS na 7, a VNS na 4 test primera. Na ostalim test primerima sve četiri metode su dala ista rešenja. Prilikom poređenja vremena izvršavanja dva heuristička pristupa koja su se najbolje pokazala EA-VNS i RVNS-VNS na „Galvao" test primerima, može se zaključiti da je RVNS-VNS brži s obzirom da mu je maksimalno vreme izvršavanja 33.974 sekundi u poređenju sa 69.816 sekundi kod EA-VNS. Međutim, treba napomenuti da su ovakva duža vremena izvršavanja EA-VNS i RVNS-VNS dosegnuta za teže instance i da je na njima RVNS-VNS imao kraće vreme u poređenju sa EA-VNS zbog različitih kriterijuma zaustavljanja. U mnogim drugim slučajevima, EA-VNS je bio efikasniji nego RVNS-VNS u smislu vremena izvršavanja.

Skup „SCJ" sa test primerima do 402 čvora je još teži za rešavanje. EA-VNS se najbolje pokazao na 10, RVNS-VNS na 7, VNS na 3, a EA na 2 test primera. Na ostalim test primerima iz skupa „SCJ", sva četiri pristupa su dala ista rešenja. Kada se u obzir uzmu i dobijena rešenja i vreme izvršavanja, može se zaključiti da je hibridni pristup EA-VNS bolji od ostala tri na ovom skupu test primera. Maksimalno vreme izvršavanja za metod EA-VNS je bilo 50.817 sekundi i to za najveći test primer 402-402-40.

Tabela 5.4: *Rezultati za test primere iz skupa „Galvao”*

Instanca		EA		VNS		RVNS-VNS		EA-VNS		
I_{num}	m n p	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	najbolji metod
1	100 100 3	1	0.01	1	0	1	0.005	1	0.02	ista rešenja
1	100 100 5	0	2.74	0	0.125	0	0.163	0	0.142	ista rešenja
1	100 100 7	1	1.27	1	0.106	1	0.163	1	0.131	ista rešenja
1	100 100 10	2	5	0	3.138	0	33.396	0	16.109	VNS, RVNS-VNS, EA-VNS
2	100 100 3	1	0.01	1	0	1	0.005	1	0.021	ista rešenja
2	100 100 5	0	1.57	0	0.098	0	0.213	0	0.115	ista rešenja
2	100 100 7	1	1.86	1	0.106	1	0.162	1	0.128	ista rešenja
2	100 100 10	2	1.14	0	3.208	0	33.666	0	1.347	VNS, RVNS-VNS, EA-VNS
3	100 100 3	1	0.01	1	0	1	0.006	1	0.02	ista rešenja
3	100 100 5	0	1.68	0	0.124	0	0.161	0	0.114	ista rešenja
3	100 100 7	1	2.87	1	0.106	1	0.159	1	0.125	ista rešenja
3	100 100 10	2	0.71	0	3.202	2	33.974	0	7.837	VNS, EA-VNS
4	100 100 3	1	0.01	1	0	1	0.006	1	0.021	ista rešenja
4	100 100 5	0	3.23	0	0.1	0	0.16	0	0.113	ista rešenja
4	100 100 7	1	2.9	1	0.102	1	0.157	1	0.13	ista rešenja
4	100 100 10	2	1.18	0	3.247	0	33.427	0	1.338	VNS, RVNS-VNS, EA-VNS
5	150 150 5	0	7.43	0	0.825	0	0.383	0	0.367	ista rešenja
5	150 150 7	1	4.79	1	0.208	1	0.483	1	0.166	ista rešenja
5	150 150 10	2	4.42	2	0	0	2.465	0	69.652	RVNS-VNS, EA-VNS
5	150 150 15	2	6.05	2	1.139	2	1.101	2	0.504	ista rešenja
6	150 150 5	0	4.18	0	0.126	0	0.382	0	0.362	ista rešenja
6	150 150 7	1	2.56	1	0.209	1	0.477	1	0.167	ista rešenja
6	150 150 10	2	3.04	2	0	0	33.396	0	69.674	RVNS-VNS, EA-VNS
6	150 150 15	2	4.33	2	1.151	2	1.101	2	0.521	ista rešenja
7	150 150 5	0	2.82	0	0.807	0	0.38	0	0.363	ista rešenja
7	150 150 7	1	6.3	1	0.214	1	0.481	1	0.169	ista rešenja
7	150 150 10	2	6.08	2	0	0	2.475	0	69.816	RVNS-VNS, EA-VNS
7	150 150 15	2	6.28	2	1.148	2	1.1	2	0.522	ista rešenja
8	150 150 5	0	3.09	0	0.118	0	0.38	0	0.591	ista rešenja
8	150 150 7	1	5.31	1	0.202	1	0.48	1	0.164	ista rešenja
8	150 150 10	2	2.89	2	0	0	2.462	0	69.673	RVNS-VNS, EA-VNS
8	150 150 15	2	6.92	2	1.137	2	1.101	2	0.526	ista rešenja
Prosek		1.125	3.209	0.875	0.655	0.688	5.765	0.625	9.717	

Tabela 5.5: *Rezultati za test primere iz skupa „SCJ”*

Instanca		EA		VNS		RVNS-VNS		EA-VNS		
I_{num}	m n p	Naj	t[s]	Naj	t[s]	Naj	t[s]	Naj	t[s]	najbolja metoda
1	100 100 3	1	0.01	1	0.00	1	0.00	1	0.03	ista rešenja
1	100 100 5	0	0.52	0	0.00	0	0.10	0	0.08	ista rešenja
1	100 100 7	1	2.89	1	0.05	1	0.20	1	0.03	ista rešenja
1	100 100 10	2	4.34	2	0.06	2	0.10	2	0.06	ista rešenja
Prosek		0.5	0.27	0.5	0.00	0.5	0.05	0.5	0.05	
2	200 200 5	0	4.66	0	8.51	0	0.10	0	0.33	ista rešenja
2	200 200 10	2	6.58	2	0.60	0	2.20	2	0.27	RVNS-VNS, EA-VNS
2	200 200 15	3	14.37	1	48.15	2	3.80	1	9.61	VNS, EA-VNS
2	200 200 20	4	7.20	2	9.12	2	2.00	2	2.52	VNS, RVNS-VNS, EA-VNS
Prosek		2.25	8.20	1.25	16.60	1	2.03	1.25	3.18	
3	300 300 10	2	22.83	2	2.95	2	0.80	2	0.99	ista rešenja
3	300 300 15	4	19.56	2	51.26	2	6.50	2	10.34	VNS, RVNS-VNS, EA-VNS
3	300 300 20	5	14.60	4	9.86	2	17.60	2	23.97	RVNS-VNS, EA-VNS
3	300 300 30	4	23.07	5	15.04	4	16.10	4	12.21	EA, RVNS-VNS, EA-VNS
Prosek		3.75	20.02	3.25	19.78	2.5	10.25	2.5	11.88	
4	402 402 10	2	30.49	2	1.98	2	6.00	1	9.57	EA-VNS
4	402 402 20	4	33.86	4	23.29	4	50.90	3	18.20	EA-VNS
4	402 402 30	6	28.22	6	57.54	5	50.90	5	14.07	RVNS-VNS, EA-VNS
4	402 402 40	6	44.07	8	52.81	6	43.70	6	50.82	EA, RVNS-VNS, EA-VNS
Prosek		4.5	34.16	5	33.90	4.25	37.88	3.75	23.16	

5.7 Evolutivni algoritam za problem MMDP

Na osnovu matematičkih formulacija problema MMDP i LOBA navedenih u poglavlju 3 može se javiti ideja da će evolutivni algoritam koji se pokazao kao veoma uspešan na primeru problema LOBA imati dobre performanse i za problem MMDP. U nastavku su navedene pojedinosti implementacije evolutivnog algoritma za MMDP. Naime s obzirom na prirodu problema MMDP, i u ovoj implementaciji evolutivnog algoritma je korišćeno binarno kodiranje. Svako rešenje problema MMDP je predstavljeno kao niz bitova dužine n, gde je $n = |N|$. Vrednost svakog bita u genetskoj reprezentaciji predstavlja da li je odgovarajući element iz skupa N odabran ili ne.

Vrednost funkcije prilagođenosti svake jedinke je proporcionalna sa $Z_{MM}(x)$, sa izuzetkom da se penalizuje ta vrednost u slučaju rešenja koja nisu dopustiva. Veličina populacije je fiksirana na 150. Genetski operatori (selekcija, mutacija i ukrštanje) se primenjuju sve dok se se ne ispuni kriterijum zaustavljanja, što je broj generacija $N_g = 200000$. Pored toga, algoritam se zaustavlja ukoliko je ponavljanje

najbolje jedinke u $N_c = 10000$ uzastopnih generacija ili ako se najbolja vrednost funkcije prilagođenosti ponovi u $N_f = 100000$ uzastopnih generacija.

Da bi se obezbedila raznolikost i bolji kvalitet genetskog materijala, početna populacija se slučajno generiše tako da se vrednost svakog bita postavi na 1 sa verovatnoćom m/n. Kao i u rešavanju problema LOBA, u slučaju da generisane jedinke ne predstavljaju dopustiva rešenja (tj. imaju sumu bitova različitu od m), pa se odmah izvrši odgovarajući broj slučajno odabranih zamena bitova (1 u 0 ili 0 u 1) u cilju postizanja tačno m bitova postavljenih na 1 u genetskom kodu.

U ovoj EA implementaciji za problem MMDP iterativno se ponavljaju operatori selekcije, ukrštanja i mutacije. Kao operator selekcije, korišćena je fino gradirana turnirska selekcija (FGTS) sa parametrom 6.6, što znači da je u 60% slučajeva veličina turnira 6, a u ostalim 7. Potom je primenjen standardni jednopozicioni operator ukrštanja, sa verovatnoćom ukrštanja postavljenom na 0.85. Parametar mutacije je inicijalno postavljen na $p_{mut} = 0.4/n$.

Podešavanje parametara

Kao što je navedeno u uvodu, ne može se očekivati da jedan algoritam dobro radi za sve probleme matematičke optimizacije. Pored toga, parametri koji se koriste u algoritmima se podešavaju u cilju postizanja boljih performansi. Odabir parametara ne samo da zavisi od problema koji se rešava, već se nekada dešava da za različite test primere istog problema različiti parametri pokažu kao efikasniji. Čest pristup je da se parametri postave na osnovu nekog, najčešće slučajno odabranog, podskupa skupa test primera i da ostaju fiksirani tokom izvršavanja algoritma. Metaheurističke metode su kreirane tako da dobro rade za širok spektar problema i njihovih test primera, bez obzira na odabir parametara. Ipak, dobar odabir parametara može uticati na poboljšanje performansi algoritma.

Za evolutivne algoritme uopšte, podešavanje parametara je komplikovan zadatak zbog nelinearne zavisnosti među parametrima [125]. Performanse evolutivnog algoritma mogu se razlikovati u zavisnosti od odabranih parametara. Tokom godina istraživači su utvrdili vrednosti parametara koji funkcionišu dobro za veliki broj različitih problema [130, 176].

Detaljnije o problemu upravljanja parametrima (engl. *parameter control problem*) uz pregled literature u kojima su navedeni pristupi za njegovo rešavanje,

može se naći u radu [105]. Jedan mogućih načina da se pristupi problemu upravljanja parametara jeste podešavanje parametara u toku izvršavanja algoritma. U okviru istraživanja koja su deo ove disertacije, korišćena su fazi pravila za promenu vrednosti parametara u toku izvršavanja algoritma.

U nastavku su prikazani nedostaci eksperimentalnog podešavanja parametara kod evolutivnih algoritama, navedeni u radu [49]:

- Parametri nisu nezavisni i testiranje svih različitih kombinacija na sistematičan način je praktično nemoguće.
- Proces podešavanja parametara je vremenski zahtevan, čak i kada se parametri optimizuju pojedinačno, bez obzira na njihovu interakciju.
- Eksperimentalno odabrane vrednosti parametara nisu garantovano optimalne za dati problem.

Iako je priroda problema MMDP i LOBA na prvi pogled slična, implementacija evolutivnog algoritma se znatno uspešnije pokazala u rešavanju problema LOBA. Iako su ova dva problema što se tiče kodiranja rešenja i odabira genetskih operatora kompatibilne, prostor pretrage se znatno razlikuje. U slučaju problema LOBA, predloženi EA se sistematičnije kreće po prostoru pretrage primenom genetskih operatora. Međutim, u slučaju problema MMDP, zbog drugačijeg rasporeda dobrih rešenja nisu u prostoru pretrage u odnosu na primenjene genetske operatore. Stoga na nekim test primerima problema MMDP dolazi do preuranjene konvergencije kada se primeni predloženi EA.

Među različitim testiranim vrednostima parametara EA, nisu pronađene vrednosti parametara koje imaju dobre performanse na svim testiranim instancama problema MMDP. Stoga se razumnim čini menjanje nekih parametra u toku izvršavanja algoritma. Na primer, ukoliko u okviru izvršavanja EA ne menja najbolja jedinka u okviru određenog broja uzastopnih generacija, može biti korisno povećati verovatnoću mutacije i time omogućiti veću raznovrsnost populacije.

U okviru ove sekcije teze prikazano je menjanje parametara p_{mut} u toku izvršavanja EA. Iako se veličina populacije može menjati, ostavljena je fiksirana na 150 u cilju jasnijeg identifikovanja uticaja menjanja vrednosti jednog parametara, dok su ostali fiksirani.

5.7.1 Fazi EA za rešavanje problema MMDP

Fazi pravila su u literaturi korišćena u cilju integracije ekspertskog znanja u različitim oblastima [140]. Pod ekspertskim znanjem se podrazumevaju činjenice nastale meritornim iskustvom u nekoj oblasti. U okviru istraživanja koja su deo ove disertacije, ideja je da se u evolutivni algoritam integriše ekspertsko znanje u vidu fazi pravila koja prilagođavanjem parametara imaju za cilj poboljšanje performansi evolutivnog algoritma.

Razmatramo narednu činjenicu sa kojom bi se složio značajan broj istraživača u oblasti evolutivnih algoritama. Ukoliko je verovatnoća mutacije mala i raznolikost u populaciji mala, onda verovatnoću mutacije treba malo povećati.

Pre implementacije fazi pravila, potrebno je odrediti ulazne i izlazne parametre i odgovarajuće funkcije pripadanja. U ovom istraživanju, odabrana izlazna vrednost je parametar verovatnoća mutacije, dok su ulazni podaci takođe vrednost parametra mutacije p_{mut} i odgovarajuća mera raznolikosti vrednosti funkcije cilja u populaciji.

Merenje raznolikosti populacije

U radu [105], naglašene su dve mere raznolikosti populacije: jedna se tiče genetskog materijala u populaciji, dok se druga bavi vrednostima funkcije prilagođenosti.

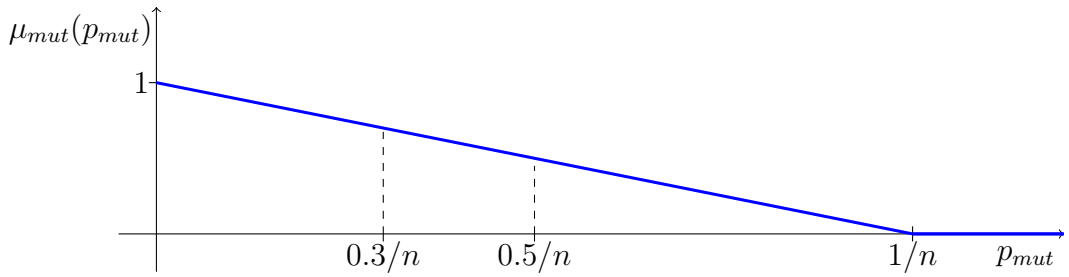
Različite mere mogu se koristiti da se otkrije konvergencija u populaciji. Razmotrimo dve mere predložene u radu [141], a kasnije razmatrane i u radu [98]:

$$PDM_1 = \frac{f_{best}}{\bar{f}} \text{ i } PDM_2 = \frac{\bar{f}}{f_{worst}},$$

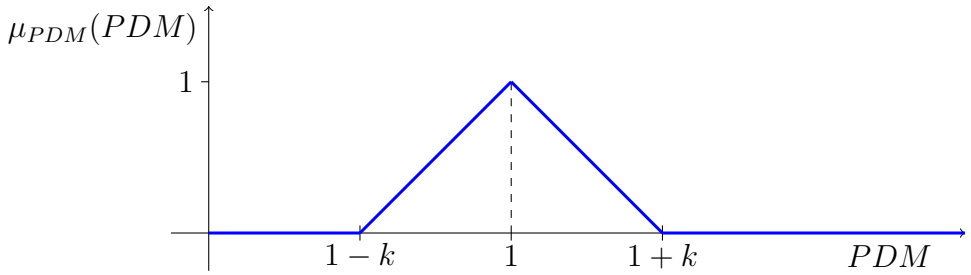
gde je f_{best} vrednost funkcije prilagođenosti najbolje jedinke u populaciji, f_{worst} najlošije, a $\bar{f}$ prosečna vrednost funkcije prilagođenost u okviru trenutne populacije. Pri tome, funkcija prilagođenosti je skalirana na $[0, 1]$ tako da je teorijski najbolja jedinka ima vrednost funkcije prilagođenosti nula, a najlošija jedan. Dakle, vrednosti $PDM_1, PDM_2 \in [0, 1]$, a njihova blizina jedinici ukazuje na konvergenciju u populaciji. Sa druge strane, u slučaju da je raznolikost u populaciji velika ove dve vrednosti su bliske nuli.

Fazi pravilo za parametar mutacije

U ovoj implementaciji, parametar mutacije se menja na osnovu fazi pravila nakon svakih pet generacija evolutivnog algoritma. Zapravo, ukoliko je razlika između dve mere PDM_1 i PDM_2 opala u poređenju sa prethodnom generacijom, verovatnoću mutacije treba „malo" povećati. Fazi pravilo se dakle uvodi na osnovu ekspertskog znanja odnosno intuitivne činjenice da će populacija koja konvergira imati koristi od malog povećanja verovatnoće mutacije. Ulazni podaci za ovo fazi pravilo su: p_{mut}, PDM_1 i PDM_2, a kao izlaz se očekuje vrednost p'_{mut} koja se eventualno razlikuje od prethodne p_{mut}. Na slici 5.1 je prikazana funkcija pripadanja skupu velika verovatnoća mutacije koja je odabrana u ovoj implementaciji. Sa druge strane funkcija pripadanja trougaonog tipa je odabrana za predstavljanje termina „malo" povećati, kao i razlike između PDM_1 i PDM_2 slika 5.2. Pri tome je parametar k dobijen na osnovu prve generacije kao $k = |f_{best} - f_{worst}|$.



Slika 5.1: *Funkcija pripadanja skupu mala verovatnoća mutacije*



Slika 5.2: *Funkcija pripadanja skupu „blizu jedinice" za PDM_1 i PDM_2*

U slučaju fazi pravila i antecedens i konsekvens implikacije su modelirane kao fazi skupovi, što je slučaj i kod razmatranog fazi pravila formulisanog kao „ako je p_{mut} mala i ako su PDM_1 i PDM_2 bliski jedinici onda p_{mut} malo povećati". Ova fazi

relacija $\mathcal{R}$ se može definisati kao fazi skup $\mathcal{R} = \{(PDM_1, PDM_2, p_{mut}), \mu_{\mathcal{R}}(x, y)\}$, gde je $\mu_{\mathcal{R}}(x, y) = \mu_{PDM}(PDM_1) \cdot \mu_{PDM}(PDM_2) \cdot \mu_{mut}(p_{mut})$. Defazifikacija podrazumeva da se dobijena vrednost x iskoristi za dobijanje novog p'_{mut} na osnovu inverza funkcije μ_{mut} tj. $\mu^{-1}_{mut}(x)$.

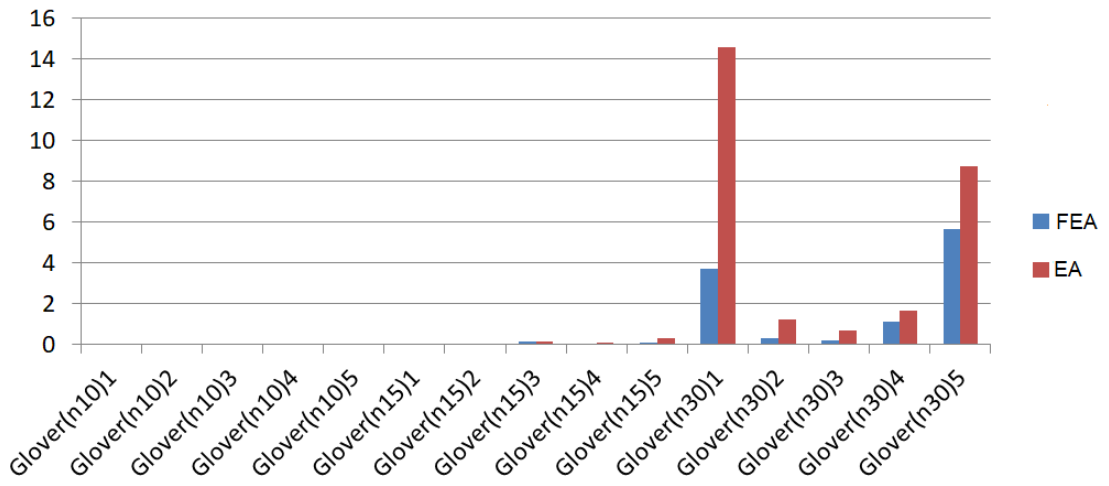
5.8 Eksperimentalni rezultati za MMDP

Prethodno opisani metodi su implementirani u C# programskom jeziku, na .NET platformi i testirani na računaru sa i7-4700MQ 2.39 GHz procesorom i 8GB RAM memorije, na Windows 10 operativnom sistemu. U cilju evaluacije efektivnosti predloženog pristupa, optimalna rešenja za test primere manjih dimenzija su dobijena i pomoću rešavača CPLEX 12.1. Implementacija za CPLEX je urađena u C# programskom jeziku na osnovu modela prikazanog u sekciji 3.2.1.

CPLEX 12.1 je uspešno dao optimalna rešenja za test primere iz skupa „Glover" korišćenjem matematičke formulacije problema MMDP iz sekcije 3.2.1. Za sve test primere iz skupa Glover, obe EA metode su dostigle optimalna rešenja prethodno dobijena pomoću CPLEX rešavača. U poređenju sa predloženim evolutivnim algoritmom, evolutivni algoritam sa dodatnim fazi pravilom (FEA) zahtevao je kraće vreme izvršavanja u proseku kada se uporede svi test primeri iz skupa „Glover". Na slici 5.3 grafički su predstavljena vremena izvršavanja za EA i FEA za neke od test primera iz skupa „Glover" sa $n \in \{10, 15, 20\}$ i $m = 6$. Vremena izvršavanja su računata kao prosečna vremena za 20 pokretanja programa algoritma.

S obzirom na to da je nedeterministički pristup čest kod metaheurističkih metoda, više pokretanja jedne metode za rešavanje istog test primera može dovesti do različitih rezultata. Naime, metaheuristike koriste pseudo-slučajne brojeve i izvršavanje zavisi od početnog semena (engl. *seed*).

U tabelama 5.6 i 5.7 su prikazani kompletni rezultati za test primere iz skupa „Glover". Za sve test primere iz skupa „Glover" oba predložena pristupa su dostigla optimalne rezultate, pre toga dobijene primenom rešavača CPLEX. Može se videti na osnovu većine test primera iz skupa „Glover" da su vremena izvršavanja za FEA bolja od predložene EA metode. U cilju ocenjivanja performansi evolutivnog algoritma prilikom različitih pokretanja za isti test primer, prosečna dobijena vrednost i standardna devijacija dobijenih rešenja su računata. Na osnovu tih vrednosti se se



Slika 5.3: *Poređenje vremena izvršavanja u sekundama za test primere iz skupa „Glover”*

može videti koliko je algoritam stabilan, odnosno da su rezultati dobijeni za različita pokretanja bliski jedan drugom.

Dakle, kao što je napomenuto predloženo fazi pravilo se primenjuje jednom u svakih pet generacija evolutivnog algoritma i neki od poziva fazi pravila menjaju parametar mutacije. U tabeli 5.8 je sumirano ovo poređenje, predstavljanjem prosečnih vrednosti broja generacija N_{gen}, broja poziva fazi pravila N_f i broja promena parametra mutacije N_{mut}. Rezultati su grupisani prema vrednosti veličine m i prosečne vrednosti su prikazane. Pored toga prikazan je i procenat promene parametra mutacije u odnosu na ukupan broj generacija. I ovom slučaju algoritam je pokretan 20 puta za svaki test primer. Iako je kriterijum zaustavljanja zavisio od konkretnog pokretanja algoritma za svaki test primer, na osnovu prosečnog broja generacija za 20 pokretanja algoritma može se zaključiti koji je najčešći kriterijum zaustavljanja za svaki od test primera. Za instance manjih dimenzija ($n = 25$) može se zaključiti da se algoritam zaustavlja ukoliko je jedan od naredna dva kriterijuma zadovoljen: maksimalan broj ponavljanja funkcije cilja ili je dostignut maksimalan broj ponavlja najbolja jedinka u populaciji. Za veće test primere najčešći kriterijum ponavljanje najbolje jedinke maksimalan broj puta. Pored toga, može se primetiti da su slični kriterijumi zaustavljanja dostignuti za FEA i EA.

Za dva veća skupa test primera „Geo” i „Ran”, CPLEX rešavač nije mogao da dođe do rezultata, ali je poređenje moguće sa najboljim poznatim rezultatima iz

Tabela 5.6: *Poređenje vremena izvršavanja za prvu polovinu test primera iz skupa „Glover”*

	EA				**FEA**			
Instanca	Naj	agap	sigma	t[s]	Naj	agap	sigma	t[s]
m=2								
Glover(n10)1	243.97	0	0	0.01	243.97	0	0	0.01
Glover(n10)2	210.65	0	0	0.00	210.65	0	0	0.00
Glover(n10)3	146.62	0	0	0.01	146.62	0	0	0.01
Glover(n10)4	186.18	0	0	0.00	186.18	0	0	0.00
Glover(n10)5	209.62	0	0	0.00	209.62	0	0	0.00
m=3								
Glover(n10)1	184.36	0	0	0.01	184.36	0	0	0.01
Glover(n10)2	183.79	0	0	0.00	183.79	0	0	0.00
Glover(n10)3	114.43	0	0	0.02	114.43	0	0	0.02
Glover(n10)4	167.25	0	0	0.00	167.25	0	0	0.00
Glover(n10)5	189.86	0	0	0.00	189.86	0	0	0.00
Glover(n15)1	100.41	0	0	0.03	100.41	0	0	0.03
Glover(n15)2	218.19	0	0	0.04	218.19	0	0	0.05
Glover(n15)3	198.45	0	0	0.03	198.45	0	0	0.04
Glover(n15)4	167.32	0	0	0.01	167.32	0	0	0.03
Glover(n15)5	177.22	0	0	0.00	177.22	0	0	0.01
m=4								
Glover(n10)1	175.63	0	0	0.01	175.63	0	0	0.01
Glover(n10)2	158.07	0	0	0.02	158.07	0	0	0.02
Glover(n10)3	107.22	0	0	0.02	107.22	0	0	0.01
Glover(n10)4	137.02	0	0	0.00	137.02	0	0	0.00
Glover(n10)5	179.71	0	0	0.00	179.71	0	0	0.00
Glover(n15)1	83.80	0	0	0.22	83.80	0	0	0.17
Glover(n15)2	206.57	0	0	0.02	206.57	0	0	0.02
Glover(n15)3	181.24	0	0	0.02	181.24	0	0	0.02
Glover(n15)4	139.05	0	0	3.89	139.05	0	0	2.93
Glover(n15)5	175.20	0	0	0.01	175.20	0	0	0.01
m=6								
Glover(n10)1	163.00	0	0	0.01	163.00	0	0	0.01
Glover(n10)2	136.58	0	0	0.01	136.58	0	0	0.01
Glover(n10)3	71.92	0	0	0.00	71.92	0	0	0.00
Glover(n10)4	120.47	0	0	0.00	120.47	0	0	0.01
Glover(n10)5	159.06	0	0	0.06	159.06	0	0	0.06
Glover(n15)1	64.57	0	0	0.03	64.57	0	0	0.02
Glover(n15)2	177.91	0	0	0.03	177.91	0	0	0.04
Glover(n15)3	162.42	0	0	0.15	162.42	0	0	0.14
Glover(n15)4	126.89	0	0	0.10	126.89	0	0	0.04
Glover(n15)5	142.20	0	0	0.33	142.20	0	0	0.10
Glover(n30)1	185.56	0	0	14.58	185.56	0	0	3.69
Glover(n30)2	180.30	0	0	1.24	180.30	0	0	0.29
Glover(n30)3	164.01	0	0	0.71	164.01	0	0	0.22
Glover(n30)4	105.17	0	0	1.66	105.17	0	0	1.09
Glover(n30)5	104.24	0	0	8.76	104.24	0	0	5.67
Prosek	157.65	0.00	0.00	0.80	157.65	0.00	0.00	0.37

Tabela 5.7: *Poređenje vremena izvršavanja za drugu polovinu test primera iz skupa „Glover”*

	EA				**FEA**			
Instanca	Naj	agap	sigma	t[s]	Naj	agap	sigma	t[s]
m=8								
Glover(n10)1	131.15	0	0	0.01	131.15	0	0	0.01
Glover(n10)2	118.75	0	0	0.00	118.75	0	0	0.00
Glover(n10)3	63.31	0	0	0.00	63.31	0	0	0.00
Glover(n10)4	92.22	0	0	0.00	92.22	0	0	0.00
Glover(n10)5	140.08	0	0	0.00	140.08	0	0	0.00
m=9								
Glover(n15)1	40.40	0	0	0.03	40.40	0	0	0.03
Glover(n15)2	154.42	0	0	0.18	154.42	0	0	0.26
Glover(n15)3	152.13	0	0	0.14	152.13	0	0	0.16
Glover(n15)4	105.53	0	0	0.01	105.53	0	0	0.01
Glover(n15)5	122.41	0	0	0.01	122.41	0	0	0.01
Glover(n30)1	165.74	0	0	30.67	165.74	0	0	29.03
Glover(n30)2	161.12	0	0	7.78	161.12	0	0	5.41
Glover(n30)3	141.89	0	0	0.80	141.89	0	0	0.55
Glover(n30)4	85.64	0	0	5.93	85.64	0	0	4.11
Glover(n30)5	90.27	0.29	0.70	25.41	90.27	0.29	0.70	12.18
m=12								
Glover(n15)1	29.38	0	0	0.00	29.38	0	0	0.01
Glover(n15)2	138.78	0	0	0.01	138.78	0	0	0.01
Glover(n15)3	126.50	0	0	0.01	126.50	0	0	0.00
Glover(n15)4	97.71	0	0	0.00	97.71	0	0	0.00
Glover(n15)5	107.66	0	0	0.00	107.66	0	0	0.00
Glover(n30)1	158.48	0	0	5.64	158.48	0	0	3.81
Glover(n30)2	148.61	0.04	0.18	7.21	148.61	0.04	0.18	7.81
Glover(n30)3	129.17	0	0	0.93	129.17	0	0	0.81
Glover(n30)4	69.53	0.04	0.19	20.12	69.53	0.04	0.19	20.96
Glover(n30)5	73.58	0.24	0.56	51.60	73.58	0.24	0.56	35.39
m=18								
Glover(n30)1	143.30	0	0	11.58	143.30	0	0	12.42
Glover(n30)2	136.89	0	0	16.25	136.89	0	0	19.78
Glover(n30)3	108.14	0	0	14.83	108.14	0	0	15.58
Glover(n30)4	56.88	0	0	0.97	56.88	0	0	0.53
Glover(n30)5	61.31	0	0	2.56	61.31	0	0	1.91
m=24								
Glover(n30)1	119.96	0	0	0.06	119.96	0	0	0.06
Glover(n30)2	124.11	0	0	1.56	124.11	0	0	1.67
Glover(n30)3	96.18	0	0	0.20	96.18	0	0	0.18
Glover(n30)4	46.74	0	0	0.09	46.74	0	0	0.07
Glover(n30)5	0.06	0	0	0.07	0.06	0	0	0.06
Prosek	106.80	0.02	0.046	5.84	106.80	0.02	0.04	4.93

Tabela 5.8: *Analiza primene fazi pravila za test primere iz skupa „Glover”*

Različite grupe instanci iz skupa Glover		Prosek za 20 izvršavanja algoritma			[%]
n	m	N_{gen}	N_f	N_{mut}	
10	2	25301.09	5060.22	3661.2	14.47
10, 15	3	56357.89	11271.58	4108.6	7.29
10, 15	4	70040.08	14008.02	8588.4	12.26
10, 15, 30	6	82253.69	16450.74	8122.27	9.87
10	8	16300.18	3260.04	1836.4	11.27
15, 30	9	98085.21	19617.04	9969.1	10.16
15, 30	12	77347.35	15469.47	8256.9	10.68
30	18	103999.2	20799.84	10097.8	9.71
30	24	100160.5	20032.09	9696.8	9.68

radova [33] i [159]. Rezultati iz rada [33] i dalje predstavljaju najbolje dostignute rezultate za test primere većih dimenzija. Štaviše za neke test primere je i dokazana optimalnost, ali treba uzeti u obzir da je maksimalno vreme izvršavanja za potrebe istraživanja u tom radu postavljeno na 5 sati.

Iako rezultati predloženog FEA ne popravljaju rezultate iz literature, predstavljaju unapređenje za EA za problem MMDP. Međutim može se zaključiti da je metod GRASP uspešniji nego EA za rešavanje većih test primera problema MMDP.

Dakle, iako su istraživači uspostavili najčešće korišćene parametre za evolutivni algoritam koji imaju dobre performanse za veliki broj različitih problema, odabir parametara je i dalje izazov za primene ovakvih algoritama. Podešavanje parametara u toku izvršavanja programa, što se može postići primenom fazi pravila, je jedan od načina u cilju prevazilaženja ovog izazova. U ovoj disertaciji je urađen eksperiment na problemu MMDP implementacijom dodatnog fazi pravila na osnovu iskustva sa evolutivnim algoritmima i na primeru MMDP problema se ovakav pristup pokazao bolji od klasičnog evolutivnog algoritma.

6

Fazi brojevi u modeliranju problema

Preciznost, binarna logika i determinizam su karakteristike većine tradicionalnih alata za formalno modeliranje, rezonovanje i računarstvo uopšte. U tradicionalnoj logici, iskaz može biti tačan ili netačan (a ne nešto između). U klasičnoj teoriji skupova, element ili pripada ili ne pripada skupu. Preciznost podrazumava da parametri u modelu tačno odgovaraju sistemu iz prakse koji se modelira. Stoga, takav model mora biti nedvosmislen, da nema slučajnih događaja i da su strukture i parametri poznati i fiksirani. Međutim, ovakve pretpostavke se ne mogu nametnuti realnom svetu.

Teorije i alati koji su se dugo koristili za modeliranje nesigurnosti realnog sveta pretežno su teorija verovatnoće i statistika. Kasnije se pojavilo više drugih teorija od kojih se najviše istakla fazi logika. Fazi logika je predstavljena u uvodnom poglavlju, a u nastavku je najpre data motivacija da se ona uključuje u modeliranje problema.

6.1 Motivacija za fazi modele

Veliki broj problema iz prakse sadrži određeni nivo neodređenosti. Neretko objašnjenja tih problema prirodnim jezikom sadrže nepreciznosti koja su razumljiva čak i u stručnim krugovima. Uprkos sveobuhvatnosti objašnjenja problema na prirodnom jeziku, ovakvi problemi predstavljaju izazov kada je potrebno kreirati odgovarajući matematički model.

Sledeća dva primera su u potpunosti jasna u okviru prirodnog jezika:

- vreme putovanja od A do B je oko 5 sati;

- dnevni profit je između 10 i 20 hiljada novčanih jedinica.

Prilikom kreiranja modela ukoliko je potrebno ove vrednosti predstaviti kao realne brojeve, prethodni podaci bi verovatno bili zapisani kao vrednosti 5 sati i 15000 novčanih jedinica. Međutim, optimalno rešenje dobijeno na osnovu modela generisanog sa ovim striktnim vrednostima, može se znatno razlikovati od optimalnog rešenja gde su tačne vrednost 5 sati i 3 minuta i 12359 novčanih jedinica.

Utemeljivač fazi logike, Lofti A. Zadeh, je upravo imao ideju o korišćenju termina prirodnih jezika u okviru oblasti fazi logike. Od tada mnogi praktični problemi su formulisani korišćenjem fazi koncepata. Fazi logika je primenjivana u mnogim oblastima, od modeliranja arhitektonskih zahteva i dizajniranja sistema za automatsko upravljanje, pa do modeliranja različitih optimizacionih problema što je razmatrano u radovima [42, 179, 43].

Da bi se kreirao matematički model nekog problema, potrebno je zapisati različita ograničenja. Primer jednog ograničenja u slučaju VRP je da svaki korisnik treba bude posećen tačno jednom. Međutim, nije moguće sva ograničenja zapisati pomoću binarnih vrednosti. Podela korisnika na posećene i neposećene je jasna, ali je pitanje kako klasifikovati rutu kao bezbednu ili ne. Skup bezbednih ruta nije tako jednostavan za definisanje, jer ne postoji gruba granica kada rute prestaju da budu bezbedne. Preciznije merenje rizika neće rešiti ovaj problem u potpunosti. Međutim, ukoliko se doda nivo rizika svakoj ruti, onda se rute mogu biti rangirane na osnovu nivoa do kog ispunjavaju sigurnost. Fazi skupovi predstavljaju pogodan matematički alat za tu svrhu.

6.2 Pregled literature koja je povezana sa fazi VRP modelima

U cilju kreiranja matematičkih modela različitih varijanti problema rutiranja vozila, fazi logika je korišćena u literaturi kao adekvatan način rada da se reši izazov konstituisanja realnijih modela za praktične probleme. Mnogi radovi razmatraju probleme rutiranja vozila sa fazi zahtevima gde zahtevi korisnika nisu precizni. Istraživači u radu [185] promenili su klasičan VRP koristeći trougaoni fazi broj za predstavljanje zahteva u svakom čvoru (korisniku), jer su pretpostavili da su količine koje treba preuzeti od svakog korisnika samo aproksimativno poznate.

Na sličan način, fazi zahtevi su uključeni u modifikaciju problema rutiranja vozila gde vozila ne moraju da se vrate u skladište nakon što usluže poslednjeg korisnika u svojoj ruti, to je tzv. problem otvorenog rutiranja vozila (engl. *the open vehicle routing problem*), koji je razmatran u radu [21]. Osim toga, u radu [113], razmatrani su fazi zahtevi za jednu varijantu problema rutiranja vozila.

Problemi rutiranja vozila sa fazi vremenskim prozorima su razmatrani u literaturi [183]. S obzirom da nije uvek neophodno striktno poštovati vremenske prozore, Tang i koautori su u radu [183] primenili fazi funkcije pripadanja u cilju karakterisanja nivoa servisa koji se odnosi na nepoštovanje pravila vremenskih prozora u problemima rutiranja vozila i predstavili formulaciju problema kao višekriterijumski model sa dva cilja: minimizaciju ukupnog pređenog puta i maksimizaciju nivoa servisa korisnika od stane snabdevača.

Sa druge strane, u radu [78] razmatra se višekriterijumski dinamički problem rutiranja vozila sa fazi vremenskim prozorima (engl. *Dynamic Vehicle Routing Problem with Fuzzy Time Windows*), gde je skup vremenskih zahteva pristiže u realnom vremenu u slučajno određenim trenucima.

S obzirom da su vozilima dodeljene rute na osnovu vremenskih prozora koja određuje korisnik i koja su veoma bitna za korisnikov nivo satisfakcije, autori u radu [78] predstavljaju ovu informaciju u preferenciji korisnika kao konveksan fazi broj u odnosu na zadovoljstvo vremenom usluge.

U skorije vreme, Brito i sar. [19] su izučavali takozvani otvoreno-zatvoreni problem rutiranja vozila (engl. *Close-open Vehicle Routing Problem*), koji je varijanta problema rutiranja vozila gde nije neophodno da se sva vozila vrate u skladište nakon obavljanja svojih usluga. U tom radu razmatrano je da su kapaciteti i vremenski prozori fleksibilni i modelirani su kao fazi ograničenja, s obzirom da su zahtevi korisnika i vreme putovanja u realnim situacijama neprecizne.

U dostupnoj literaturi, do sada fazi teorija nije korišćena za modeliranje problema RCTVRP.

6.3 Fazi model za RCTVRP

U okviru ove disertacije razvijen je novi i unapređen model za RCTVRP, nazvan FRCTVRP, koji omogućava preciznije razlikovanje rešenja dobrog kvaliteta. Kao

što je objašnjeno za RCTVRP izračunati rizik svake rute manji od praga rizika se ne uzima dalje u razmatranje. Jedina provera u RCTVRP modelu jeste da je izračunati rizik manji od praga rizika T. Međutim, iako nije eksplicitno uključeno u model predložen u [181], ono što se na osnovu opisa problema podrazumeva jeste da što je manji indeks rizika, ruta je sigurnija. Stoga se skup bezbednih ruta može na odgovarajući način predstaviti pomoći fazi skupova, umesto običnih skupova.

U okviru ove disertacije, ideja je da se ova činjenica uključi tako što se izmeni funkcija cilja. Naime, s obzirom na to da je RCTVRP problem minimizacije, umesto da se samo minimizuje ukupan pređeni put, nova funkcija cilja će uključivati i rastojanja i meru koliko su rute bezbedne.

U cilju postizanja da izračunata vrednost ima adekvatan uticaj u novom predloženom modelu uvedena je sledeća funkcija cilja:

$$\min \sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r F(R_j^r). \tag{6.1}$$

Pri tome, uvedena funkcija F je definisana kao:

$$F(r) = 1 + \mu(r),$$

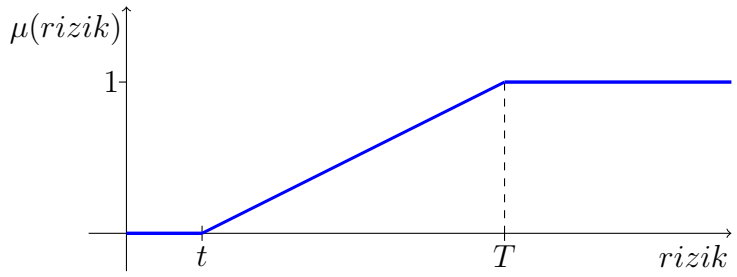
gde je funkcija μ predstavlja funkciju pripadanja:

$$\mu(x) = \begin{cases} 0, & \text{ako je } x < t, \\ \frac{x-t}{T-t}, & \text{ako je } t \leq x < T, \\ 1, & \text{inače.} \end{cases}$$

Fazi broj čija je ovo funkcija pripadanja se u literaturi naziva levo rame (engl. *left shoulder fuzzy number*) i prikazan je na slici 6.1.

Parametar t ($t < T$) je postavljen na $t = \max_{i \in N} c_{ie} d_i$, s obzirom da deluje pravično da se sve rute sa po jednim korisnikom budu označene kao rute sa najmanjom vrednošću funkcije F. Rute čiji nivo rizika je veći od praga rizika T su i dalje nedozvoljene. Funkcija f preslikava nivoe rizika u brojeve iz intervala $[0, 1]$.

Ovakva funkcija cilja je odabrana da bi se predstavilo svojstvo „što manji rizik to bolje", što je i bila namera da se uključi u funkciju cilja. Zapravo, funkcija cilja je promenjena tako da dužine grana u okviru „rizičnih" ruta budu uvećanje množenjem



Slika 6.1: *Funkcija pripadanja za odgovarajući fazi broj*

sa faktorom između 1 i 2. Na taj način, minimizovanjem funkcije cilja među svim rutama koje imaju istu dužinu prednost imaju bezbednije rute. Može se primetiti da se i neke druge rastuće funkcije mogu testirati, čime se predloženi model prilagođava različitim primenama. Na primer, moguće je predloženi model promeniti tako da se razmatra razlika u indeksima rizika samo u slučajevima da su rute iste ukupne dužine.

Međutim, u ovoj disertaciji je predstavljena varijanta u kojoj je funkcija cilja takva da predstavlja kompromis između ukupnog pređenog puta i indeksa rizika na rutama, u smislu da je dopušteno da ruta bude malo duža ukoliko je dosta bezbednija. Pored toga, ovakav izbor rastuće funkcije je pogodan i za formulaciju MILP modela predstavljenih u ovom radu.

U cilju formulisanja modela FRCTVRP kao problema celobrojnog mešovitog linearnog programiranja (MILP), nove promenljive FR_j^r sa vrednostima iz intervala $[1, 2]$ su uključene u model.

Model za FRCTVRP se može prikazati na sledeći način:

$$\min \sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r FR_j^r, \tag{6.2}$$

uz uslove:

$$\sum_{j\in N} x_{sj}^r = \sum_{i\in N} x_{ie}^r, \qquad \forall r \in N, \tag{6.3}$$

$$\sum_{j\in N} x_{sj}^1 = 1, \tag{6.4}$$

$$\sum_{i\in N} x_{ie}^r \geq \sum_{j\in N} x_{sj}^{r+1}, \qquad \forall r \in N \setminus \{n\}, \tag{6.5}$$

$$\sum_{r\in N} \sum_{j\in V\setminus\{s\}} x_{ij}^r = 1, \qquad \forall i \in N, \tag{6.6}$$

$$\sum_{h\in V\setminus\{e\}} x_{hj}^r - \sum_{k\in V\setminus\{s\}} x_{jk}^r = 0, \qquad \forall j \in N, \forall r \in N, \tag{6.7}$$

$$D_s^r = 0, \qquad \forall r \in N, \tag{6.8}$$

$$D_j^r \geq D_i^r + d_j - (1 - x_{ij}^r) \cdot M_1, \qquad \forall (i,j) \in A, \forall r \in N, \tag{6.9}$$

$$0 \leq D_i^r \leq M_1, \qquad \forall i \in V, \forall r \in N, \tag{6.10}$$

$$R_s^r = 0, \qquad \forall r \in N, \tag{6.11}$$

$$R_j^r \geq R_i^r + D_i^r \cdot c_{ij} - (1 - x_{ij}^r) \cdot M_2, \qquad \forall (i,j) \in A, \forall r \in N, \tag{6.12}$$

$$0 \leq R_i^r \leq T, \qquad \forall i \in V, \forall r \in N, \tag{6.13}$$

$$FR_i^r \geq 1 + (R_i^r - t)/(T - t), \qquad \forall i \in V \setminus \{s\}, \forall r \in N, \tag{6.14}$$

$$1 \leq FR_i^r \leq 2, \qquad \forall i \in V \setminus \{s\}, \forall r \in N, \tag{6.15}$$

$$x_{ij}^r \in \{0, 1\}, \qquad \forall (i,j) \in A, \forall r \in N. \tag{6.16}$$

Uslovi (6.14) i (6.15) u navedenom modelu su tako definisani da FR_j^r predstavljaju vrednosti $F(R_j^r)$, gde je F prethodno definisana funkcija $F(t) = 1 + f(t)$.

Može se primetiti da je broj promenljivih i ograničenja porastao u odnosu na početni model iz literature. Stoga, u nastavku je predstavljena unapređena verzija modela za FRCTVRP, gde su promenljive R_j^r izbačene, a uslovi (6.11)-(6.15) su zamenjeni uslovima (6.26)-(6.28) dobijenim korišćenjem formule $FR_i^r = 1 + \frac{R_i^r - t}{T - t}$ (tj. $R_i^r = t + (FR_i^r - 1) \cdot (T - t)$). Očigledno je da ova formula važi samo kada je R_i^r iz intervala $[t, T]$. Međutim, to je ispunjeno za sve validne rute.

Konačno, unapređena formulacija problema FRCTVRP je:

$$\min \sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r FR_j^r, \tag{6.17}$$

uz uslove:

$$\sum_{j \in N} x_{sj}^r = \sum_{i \in N} x_{ie}^r, \qquad \forall r \in N, \tag{6.18}$$

$$\sum_{j \in N} x_{sj}^1 = 1, \tag{6.19}$$

$$\sum_{i \in N} x_{ie}^r \geq \sum_{j \in N} x_{sj}^{r+1}, \qquad \forall r \in N \setminus \{n\}, \tag{6.20}$$

$$\sum_{r \in N} \sum_{j \in V \setminus \{s\}} x_{ij}^r = 1, \qquad \forall i \in N, \tag{6.21}$$

$$\sum_{h \in V \setminus \{e\}} x_{hj}^r - \sum_{k \in V \setminus \{s\}} x_{jk}^r = 0, \qquad \forall j \in N, \forall r \in N, \tag{6.22}$$

$$D_s^r = 0, \qquad \forall r \in N, \tag{6.23}$$

$$D_j^r \geq D_i^r + d_j - (1 - x_{ij}^r) \cdot M_1, \qquad \forall (i,j) \in A, \forall r \in N, \tag{6.24}$$

$$0 \leq D_i^r \leq M_1, \qquad \forall i \in V, \forall r \in N, \tag{6.25}$$

$$FR_s^r = \frac{T - 2t}{T - t}, \qquad \forall r \in N, \tag{6.26}$$

$$FR_j^r \geq FR_i^r + (D_i^r \cdot c_{ij} - (1 - x_{ij}^r) \cdot M_2)/(T - t), \qquad \forall (i,j) \in A, \forall r \in N, \tag{6.27}$$

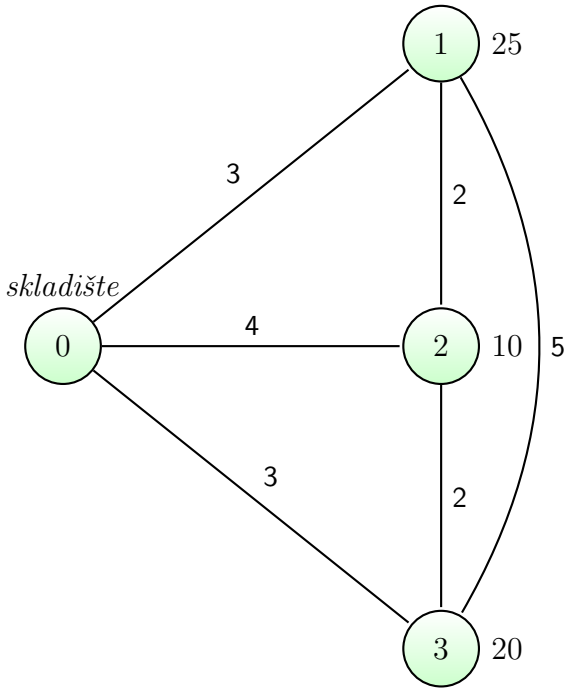
$$1 \leq FR_i^r \leq 2, \qquad \forall i \in V \setminus \{s\}, \forall r \in N, \tag{6.28}$$

$$x_{ij}^r \in \{0, 1\}, \qquad \forall (i,j) \in A, \forall r \in N. \tag{6.29}$$

6.3.1 Motivacioni primer

Razlika između modela iz literature RCTVRP i u disertaciji razvijenog novog fazi modela FRCTVRP se može predstaviti na sledećem jednostavnom primeru. Pretpostavimo da je dat graf sa četiri čvora kao što je prikazano na slici 6.2 i da je limit rizika $T = 160$. Čvor 0 predstavlja skladište. Tri čvora označena brojevima 1, 2 i 3 predstavljaju korisnike. Količine novca koje treba preuzeti od njih zapisani su sa desne strane svakog čvora koji označava korisnika. U nastavku će biti razmotren ovaj primer za oba modela RCTVRP i FRCTVRP.

Najpre razmatramo ovaj primer i klasičan RCTVRP model iz literature [181].



Slika 6.2: *Primer na neusmerenom težinskom grafu gde težine grada predstavljaju rastojanja između čvorova.*

Može se lako izračunati da nije moguće posetiti sva tri korisnika u okviru jedne rute, zbog ograničenja koja se odnose na rizik. Rešenje u kome tri vozila obilaze svaki u svojoj ruti po jednog korisnika je dopustivo, ali u tom slučaju funkcija cilja dostiže maksimalnu vrednost. Zato su predstavljena detaljnije naredna tri dopustiva rešenja (sva tri sa po dve rute):

Rešenje 1:

Ruta ($r = 1$): $0 \to 2 \to 1 \to 0$

$D_0^1 = 0,\ D_2^1 = d_2 = 10,\ D_1^1 = D_2^1 + d_1 = 35,$

$R_0^1 = 0,\ R_2^1 = R_0^1 + D_0^1 \cdot c_{02} = 0,\ R_1^1 = R_2^1 + D_1^1 \cdot c_{21} = 20,$

$GR^1 = R_1^1 + D_1^1 \cdot c_{10} = 125 < T = 160.$

Ruta ($r = 2$): $0 \to 3 \to 0$

$D_0^2 = 0,\ D_3^2 = d_3 = 20,$

$R_0^2 = 0,\ R_3^2 = R_0^2 + D_0^2 \cdot c_{03} = 0,$

$GR^2 = R_3^2 + D_3^2 \cdot c_{30} = 60 < T = 160.$

Rešenje 2:

Ruta $(r = 1)$: $0 \to 1 \to 0$

$D_0^1 = 0,\ D_1^1 = d_1 = 25,$

$R_0^1 = 0,\ R_1^1 = R_0^1 + D_0^1 \cdot c_{01} = 0,$

$GR^1 = R_1^1 + D_1^1 \cdot c_{10} = 75 < T = 160.$

Ruta $(r = 2)$: $0 \to 2 \to 3 \to 0$

$D_0^2 = 0,\ D_2^2 = d_2 = 10,\ D_3^2 = D_2^2 + d_3 = 30,$

$R_0^2 = 0,\ R_2^2 = R_0^2 + D_0^2 \cdot c_{02} = 0,\ R_3^2 = R_2^2 + D_2^2 \cdot c_{23} = 20,$

$GR^2 = R_3^2 + D_3^1 \cdot c_{30} = 110 < T = 160.$

Rešenje 3:

Ruta $(r = 1)$: $0 \to 1 \to 0$

$D_0^1 = 0,\ D_1^1 = d_1 = 25,$

$R_0^1 = 0,\ R_1^1 = R_0^1 + D_0^1 \cdot c_{01} = 0,$

$GR^1 = R_1^1 + D_1^1 \cdot c_{10} = 75 < T = 160.$

Ruta $(r = 2)$: $0 \to 3 \to 2 \to 0$

$D_0^2 = 0,\ D_3^2 = d_3 = 20,\ D_2^2 = D_3^2 + d_2 = 30,$

$R_0^2 = 0,\ R_3^2 = R_0^2 + D_0^2 \cdot c_{03} = 0,\ R_2^2 = R_3^2 + D_3^2 \cdot c_{32} = 40,$

$GR^2 = R_2^2 + D_2^1 \cdot c_{20} = 160 \leq T = 160.$

Primetimo da sva tri prethodna rešenja su dopustiva i imaju istu vrednost funkcije cilja:

$$\sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r = 15.$$

Dakle, na osnovu modela iz rada [181], sva tri izdvojena rešenja su istog kvaliteta. Međutim, sa bezbednosne tačke gledišta rešenje 3 je najlošije od ova tri rešenja, s obzirom na to da je globalni rizik na dve rute $GR^1 = 75$ i $GR^2 = 160$. Bilo bi

poželjno da u dobrom modelu rešenja kod kojih je isti ukupni put, ali rizici isti, dobiju različite vrednosti funkcije cilja.

U slučaju predloženog FRCTVRP modela situacija je drugačija. Naravno, sva ograničenja su ostala ista, pa je dovoljno razmatrati opet samo dopustiva rešenja. U slučaju „zvezdastog" rešenja, vrednost funkcije cilja je ista kao u prethodnom primeru, zbog definicije funkcije F i parametra t. U nastavku su izračunate ostale vrednosti.

Rešenje 1: Rešenje koje sastoji iz dve rute $0 \to 2 \to 1 \to 0$ i $0 \to 3 \to 0$ ima sledeću vrednost funkcije cilja:

$$\sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r F(R_j^r) = 16.765.$$

Rešenje 2: Za dve rute $0 \to 1 \to 0$ i $0 \to 2 \to 3 \to 0$, dobija se:

$$\sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r F(R_j^r) = 16.235.$$

Rešenje 3: Računanje funkcije cilja u slučaju dve rute $0 \to 1 \to 0$ i $0 \to 3 \to 2 \to 0$:

$$\sum_{r \in N} \sum_{(i,j) \in A} c_{ij} x_{ij}^r F(R_j^r) = 19.$$

Ovaj primer demonstrira da novi FRCTVRP model dovodi do toga da se preferiraju manje rizične rute među dozvoljenim rutama. Kada se koristi RCTVRP model, predstavljena tri rešenja imaju istu vrednost funkcije cilja. Sa druge strane, kada se koristi novi FRCTVRP model, ista rešenja imaju tri različite vrednosti funkcije cilja. Drugo rešenje je najbolje, s obzirom na to da je malo bolje od prvog. Rešenje broj 3 ima najveću vrednost funkcije cilja, što je očekivano s obzirom na rizičnije rute korišćene u rešenju.

Može se primetiti da novi, u disertaciji razvijeni, model FRCTVRP uključuje verovatnoću nepoželjnog događaja (pljačka) na svakoj od ruta, ali čuva dobre karakteristike prethodnog modela zadržavajući ograničenje rizika na svakoj ruti u rešenju.

6.4 Eksperimentalni rezultati za FRCTVRP

Primer iz prethodnog poglavlja pokazuje da model FRCTVRP podrazumeva da su među rešenjima sa istim ukupnim pređenim putem bolja ona rešenja koja imaju rute koje su manje rizične. U slučaju većih test primera razmatranog rešenja, broj potencijalnih rešenja je veći i poređenje dva modela je zanimljivije. Stoga su eksperimenti izvršeni tako što su prethodna dva modela implementirana i rešavana korišćenjem rešavača CPLEX 12.6.

Eksperimenti su izvršeni na test primerima sa 4, 6, i 8 korisnika iz skupa R (skup instanci specijalno kreiran za razmatrani problem u [181]). Rešavač CPLEX je uspeo da pronađe optimalna rešenja na razmatranim test primerima za oba modela. Rezultati se mogu videti u tabeli 6.1, čije kolone redom označavaju:

- Ime test primera;
- Dobijene optimalne vrednosti funkcije cilja za RCTVRP model;
- Rute optimalnog rešenja za RCTVRP model (uređene liste čvorova koje treba posetiti u okviru svake rute su razdvojene znakom „|”);
- Vreme u sekundama potrebno za dolazak do rešenja za RCTVRP model;
- Dobijene optimalne vrednosti funkcije cilja za FRCTVRP model;
- Rute optimalnog rešenja za FRCTVRP model (uređene liste čvorova koje treba posetiti u okviru svake rute su razdvojene znakom „|”);
- Vreme u sekundama potrebno za dolazak do rešenja za FRCTVRP model.

Na primer, u petom redu se može videti da za test primer „4_1_3.0” optimalno rešenje u slučaju modela RCTVRP ima optimalnu vrednost funkcije cilja 286.1275 i da postoje dve rute u tom rešenju: prva ruta sadrži korisnike 3 i 1, dok druga ruta sadrži samo jednog korisnika: 2. Takođe, može se videti u istom redu da je CPLEX rešavaču bilo potrebno 0.2 sekundi da dođe do rešenja, dok je za isti test primer bilo potrebno 0.09 sekundi u slučaju modela FRCTVRP. U istom redu, može se videti da optimalno rešenje ima vrednost funkcije cilja jednako 335.2504 za model FRCTVRP. Međutim, ova dva rešenja su zapravo ista. Važno je napomenuti da funkcija cilja za dva modela nije ista, tako da rešenja dobijena za ova dva modela

ne treba porediti na osnovu dobijenih vrednosti funkcije cilja. Stoga poređenja su urađena na osnovu čvorova u rutama, tj. ukoliko su liste čvorova iste onda se rešenja mogu smatrati istima. Može se primetiti i da ukoliko su rute iste, ukupan pređeni put je isti.

Na osnovu rezultata u tabeli 6.1 može se videti da su na nekim test primerima rute optimalnih rešenja za dva modela iste i u tim slučajevima su označene rute za model FRCTVRP. Najpre, može se primetiti da su za test primere kod kojih je indeks praga rizika $RL1$ (test primeri kod kojih se naziv završava sa „1.0") isti rezultati. Međutim, primetno je da što je vrednost praga rizika veća, razlika između dva modela je značajnija. Ovo je očekivano, s obzirom da veće vrednosti praga rizika dovode do većeg broja ruta koje zadovoljavaju uslov praga rizika.

Prikazani eksperimentalni rezultati pokazuju da dva modela dovode do različitih optimalnih rešenja na mnogim testiram primerima. U nekim primerima broj ruta u ta dva rešenja se razlikuje, kao što je slučaj kod test primera „4_7_1.5". Može se primetiti, da broj ruta u optimalnom rešenju za model FRCTVRP može biti veći ili jednak broju ruta u slučaju klasičnog RCTVRP. Često u slučaju različitih rešenja, broj ruta ostaje isti, ali čvorovi nisu grupisani na isti način. Na primer, u slučaju test primera „4_7_2.5" dva modela imaju različita optimalna rešenja. Iako oba rešenja imaju isti ukupni pređeni put (286.1275), razlika u indeksima rizika na rutama je različita. Za RCTVRP indeksi rizika su: 28548.9 i 6785.72 dok za novi model FRCTVRP optimalno rešenje ima naredne vrednosti za indekse rizika na rutama: 23119.3 i 6785.72 jer je razlika u redosledu obilaska čvorova u okviru prve rute.

Detaljnije poređenje test primera za koje su dobijene drugačije optimalne rute u dva modela je dato u tabeli 6.2. Prva kolona sadrži ime test primera, dok je konstanta limit rizika T data u drugoj koloni. Potom su, prvo za RCTVRP, a potom i za FRCTVRP, date sledeće vrednosti: ukupan pređeni put i globalni indeksi rizika za sve rute u optimalnom rešenju.

Prednost novog modela razvijenog u disertaciji je da omogućava da se napravi razlika i uoče manje rizične rute među svim mogućim rutama. Cilj je i dalje pronaći rešenja sa što manjim pređenim putem, s tim što ovaj model predstavlja kompromis između pređenog puta i rizika kao što je objašnjeno u sekciji 6.3. Stoga, model FRCTVRP omogućava pronalaženje ruta koje su manje rizične i pogodnije za prenos robe od velike vrednosti. Uzimajući u obzir sedam test primera koji imaju

Tabela 6.1: *Poređenje modela RCTVRP i FRCTVRP na test primerima iz skupa R*

	RCTVRP			**FRCTVRP**		
Instanca	F_{cilja}	**Čvorovi u rutama**	**T[s]**	F_{cilja}	**Čvorovi u rutama**	**T[s]**
4_1_1.0	343.0474	**1|2|3|**	0.06	343.0474	**1|2|3|**	0.04
4_1_1.5	343.0474	**1|2|3|**	0.06	343.0474	**1|2|3|**	0.06
4_1_2.0	343.0474	**1|2|3|**	0.29	343.0474	**1|2|3|**	0.18
4_1_2.5	343.0474	**1|2|3|**	0.07	343.0474	**1|2|3|**	0.06
4_1_3.0	286.1275	2|3→ 1|	0.20	331.0777	**2|3→ 1|**	0.09
4_3_1.0	343.0474	**1|2|3|**	0.05	343.0474	**1|2|3|**	0.05
4_3_1.5	343.0474	**1|2|3|**	0.05	343.0474	**1|2|3|**	0.04
4_3_2.0	343.0474	**1|2|3|**	0.23	343.0474	**1|2|3|**	0.17
4_3_2.5	343.0474	**1|2|3|**	0.07	343.0474	**1|2|3|**	0.06
4_3_3.0	286.1275	**2|3→ 1|**	0.30	333.2687	**2|3→ 1|**	0.05
4_5_1.0	343.0474	**1|2|3|**	0.05	343.0474	**1|2|3|**	0.04
4_5_1.5	343.0474	**1|2|3|**	0.19	343.0474	**1|2|3|**	0.16
4_5_2.0	343.0474	**1|2|3|**	0.07	343.0474	**1|2|3|**	0.07
4_5_2.5	343.0474	**1|2|3|**	0.11	343.0474	**1|2|3|**	0.06
4_5_3.0	286.1275	**2|3→ 1|**	0.18	327.1712	**2|3→ 1|**	0.22
4_7_1.0	343.0474	**1|2|3|**	0.07	343.0474	**1|2|3|**	0.07
4_7_1.5	332.933	1→ 2|3|	0.12	343.0474	1|2|3|	0.12
4_7_2.0	286.1275	**1→ 3|2|**	0.23	331.5104	**1→ 3|2|**	0.16
4_7_2.5	286.1275	2|3→ 1|	0.06	316.4339	1→ 3|2|	0.05
4_7_3.0	286.1275	2|3→ 1|	0.06	308.8765	1→ 3|2|	0.20
6_1_1.0	594.6428	**1|3|4→ 2|5|**	0.74	594.6428	**1|3|4→ 2|5|**	0.29
6_1_1.5	594.3352	1|3→ 2|4|5|	0.32	594.6428	1|3|4→ 2|5|	0.42
6_1_2.0	438.5142	1|2|3→5→ 4|	0.30	494.5922	1|3→ 4→ 2|5|	0.49
6_1_2.5	387.9904	**3→ 1|5→ 4→ 2|**	0.29	459.2829	**3→ 1|5→ 4→ 2|**	0.51
6_1_3.0	387.6828	1→ 3→ 2|5→ 4|	0.22	441.4709	3→ 1|5→ 4→ 2|	0.50
6_3_1.0	645.1667	**1|2|3|4|5|**	0.33	645.1667	**1|2|3|4|5|**	0.39
6_3_1.5	594.3352	1|3→ 2|4|5|	0.39	596.3519	1|3|5|4→ 2|	0.28
6_3_2.0	445.4363	2|3→ 5|4→ 1|	0.36	515.8627	1→ 2|3→ 4|5|	0.46
6_3_2.5	387.9904	**3→ 1|5→ 4→ 2|**	0.17	462.7753	**3→ 1|5→ 4→ 2|**	0.44
6_3_3.0	387.6828	1→ 3→ 2|5→ 4|	0.31	444.0907	3→ 1|5→ 4→ 2|	0.45
6_5_1.0	594.6428	**1|3|4→ 2|5|**	0.20	594.6428	**1|3|4→ 2|5|**	0.43
6_5_1.5	594.3352	1|3→ 2|4|5|	0.28	594.6428	1|3|4→ 2|5|	0.47
6_5_2.0	387.9904	**3→ 1|5→ 4→ 2|**	0.18	486.8541	**3→ 1|5→ 4→ 2|**	0.51
6_5_2.5	387.9904	**3→ 1|5→ 4→ 2|**	0.23	453.9269	**3→ 1|5→ 4→ 2|**	0.51
6_5_3.0	387.6828	1→ 3→ 2|4→ 5|	0.18	437.4531	3→ 1|5→ 4→ 2|	0.52
6_7_1.0	534.9608	**3|4→ 2|5→ 1|**	0.27	534.9608	**3|4→ 2|5→ 1|**	0.50
6_7_1.5	445.4363	1→ 4|2|5→ 3|	0.20	505.5844	1→ 2|3|5→ 4|	0.53
6_7_2.0	362.8849	**1→ 2|5→ 3→ 4|**	0.31	417.5379	**1→ 2|5→ 3→ 4|**	0.47
6_7_2.5	360.6816	**1|5→ 3→ 4→ 2|**	0.36	389.7075	**1|5→ 3→ 4→ 2|**	0.33
6_7_3.0	358.6308	1→ 3→ 4|5→ 2|	0.29	382.4555	1|5→ 3→ 4→ 2|	0.28
8_1_1.0	541.5618	3→ 2|4|5|6→ 1|7|	4.18	584.5117	1→ 6|2→ 5|3|4|7|	7.15
8_1_1.5	487.5307	**1→ 6→ 4|3→ 2|5|7|**	1.05	536.7901	**1→ 6→ 4|3→ 2|5|7|**	3.22
8_1_2.0	431.0115	**1→ 6→ 4|3→ 2→ 5|7|**	0.93	472.3581	**1→ 6→ 4|3→ 2→ 5|7|**	1.93
8_1_2.5	416.9024	2→ 5|4→ 6→ 1|7→ 3|	1.35	458.5821	1→ 6→ 4|3→ 2→ 5|7|	2.84
8_1_3.0	379.6434	5→ 1→ 6→ 4|7→ 3→ 2|	2.58	427.3815	1→ 6→ 4|5|7→ 3→ 2|	2.21
8_3_1.0	658.3218	**1|2|3|5|6→ 4|7|**	2.72	658.3218	**1|2|3|5|6→ 4|7|**	2.87
8_3_1.5	487.5307	**1→ 6→ 4|3→ 2|5|7|**	1.04	540.3774	**1→ 6→ 4|3→ 2|5|7|**	3.28
8_3_2.0	431.0115	**1→ 6→ 4|3→ 2→ 5|7|**	0.79	474.4815	**1→ 6→ 4|3→ 2→ 5|7|**	2.83
8_3_2.5	416.9024	2→ 5|4→ 6→ 1|7→ 3|	3.52	459.9981	1→ 6→ 4|3→ 2→ 5|7|	2.93
8_3_3.0	384.2311	4→ 6→ 1|5|7→ 3→ 2|	0.75	428.2584	1→ 6→ 4| 5|7→ 3→ 2|	2.63
8_5_1.0	574.233	**1→ 6|2→ 5|3|4|7|**	1.12	574.233	**1→ 6|2→ 5|3|4|7|**	2.47
8_5_1.5	487.5307	**1→ 6→ 4|3→ 2|5|7|**	0.89	531.5638	**1→ 6→ 4|3→ 2|5|7|**	1.03
8_5_2.0	431.0115	**1→ 6→ 4|3→ 2→ 5|7|**	1.21	469.6237	**1→ 6→ 4|3→ 2→ 5|7|**	2.68
8_5_2.5	416.9024	1→ 6→ 4|5→ 2|7→ 3|	1.23	456.7588	1→ 6→ 4|3→ 2→ 5|7|	2.78
8_5_3.0	379.6434	5→ 1→ 6→ 4|7→ 3→ 2|	1.04	426.5219	1→ 6→ 4|5|7→ 3→ 2|	2.71
8_7_1.0	485.0425	1→ 6|3→ 2→ 5|4|7|	1.07	485.0425	3→ 2→ 5|4|6→ 1|7|	1.90
8_7_1.5	416.9024	1→ 6→ 4|2→ 5|7→ 3|	0.82	442.2885	1→ 6→ 4|3→ 2→ 5|7|	2.24
8_7_2.0	384.2311	4→ 6→ 1|5|7→ 3→ 2|	0.64	433.0913	1→ 6→ 4|5|7→ 3→ 2|	2.50
8_7_2.5	327.7119	**1→ 6→ 4|7→ 3→ 2→ 5|**	0.36	368.4427	**1→ 6→ 4|7→ 3→ 2→ 5|**	1.23
8_7_3.0	327.7119	4→ 6→ 1|7→ 3→ 2→ 5|	0.47	358.2634	1→ 6→ 4|7→ 3→ 2→ 5|	1.87

Tabela 6.2: *Poređenje indeksa rizika dobijenih za modele RCTVRP i FRCTVRP na test primerima iz skupa R*

		RCTVRP					
Instanca	**Limit Rizika**	**Pređeni put**	**Indeksi rizika**				
4_7_1.5	19995	332.93	13262.2	19561.7			
4_7_2.5	33325	286.13	28548.9	6785.72			
4_7_3.0	39990	286.13	28548.9	6785.72			
6_1_1.5	18720	594.34	8513.76	8483.65	15870.8	9148.45	
6_1_2.0	24960	438.51	3355.24	24817.4	22184.1		
6_1_3.0	37440	387.68	35123.7	22184.1			
6_3_1.5	17856	594.34	8812.47	15318.8	8186.31	8938.95	
6_3_2.0	23808	445.44	23601	3380.85	23445		
6_3_3.0	35712	387.68	34130.8	21986			
6_5_1.5	19584	594.34	8417.89	17268.1	7727.87	9707.14	
6_5_3.0	39168	387.68	20853.7	37696.7			
6_7_1.5	22752	445.44	19944.8	5148.11	22241.7		
6_7_3.0	45504	358.63	7523.1	43318			
8_1_1.0	14738	541.56	11782	4321.4	14565.3	3837.09	14615.7
8_1_2.5	29475	416.9	11455.1	24957.5	24333.2		
8_1_3.0	35370	379.64	35145.8	30337.1			
8_3_2.5	29250	416.9	12136.8	24811.1	24081.1		
8_3_3.0	35100	384.23	4454.37	24811.1	30532.2		
8_5_2.5	29925	416.9	26164.9	14035.2	15774		
8_5_3.0	35910	379.64	34331.5	31331.9			
8_7_1.0	17550	485.04	16907.5	5813.78	17538.1	13897	
8_7_1.5	26325	416.9	14290.7	20946.9	25535.6		
8_7_2.0	35100	384.23	34437.9	3590.09	33059.9		
8_7_3.0	52650	327.71	33059.9	42252.6			
		FRCTVRP					
Instanca	**Limit Rizika**	**Pređeni put**	**Indeksi rizika**				
4_7_1.5	19995	343.05	6785.72	4408.22	13262.2		
4_7_2.5	33325	286.13	23119.3	6785.72			
4_7_3.0	39990	286.13	23119.3	6785.72			
6_1_1.5	18720	594.64	12464.4	9148.45	8513.76	11929.3	
6_1_2.0	24960	463.23	9148.45	8513.76	24500.1		
6_1_3.0	37440	387.99	24817.4	25720.9			
6_3_1.5	17856	594.64	8186.31	12287.3	11889.1	8938.95	
6_3_2.0	23808	465.43	8186.31	20716.1	12691.6		
6_3_3.0	35712	387.99	23884.8	25548.4			
6_5_1.5	19584	594.64	13039.7	12682.5	7727.87	9707.14	
6_5_3.0	39168	387.99	25201	26099.3			
6_7_1.5	22752	445.44	21671.5	5148.11	22241.7		
6_7_3.0	45504	360.68	9707.14	33090.5			
8_1_1.0	14738	574.23	11782	3837.09	8694.49	11455.1	12498.1
8_1_2.5	29475	431.01	21537.6	11782	17407.1		
8_1_3.0	35370	384.23	17407.1	4321.4	30337.1		
8_3_2.5	29250	431.01	17337.9	11692.1	22065.4		
8_3_3.0	35100	384.23	4454.37	17337.9	30532.2		
8_5_2.5	29925	431.01	11961.9	22541.6	15774		
8_5_3.0	35910	384.23	4520.85	31331.9	15774		
8_7_1.0	17550	485.04	17314.1	5813.78	17538.1	16907.5	
8_7_1.5	26325	431.01	17538.1	16907.5	20946.9		
8_7_2.0	35100	384.23	34437.9	3590.09	20946.9		
8_7_3.0	52650	327.71	20946.9	42252.6			

isti ukupan pređeni put u oba modela a različita rešenja, zbir svih indeksa rizika je 378358.29 u slučaju modela RCTVRP, dok je 340943.69 u slučaju FRCTVRP. Stoga, na osnovu samo ovih test primera može se videti da se rizik može smanjiti korišćenjem FRCTVRP umesto RCTVRP, nekad i bez povećavanja ukupnog puta.

Može se primetiti da je razlika među dva razmatrana modela značajnija kod test primera većih dimenzija. Dodatni eksperimentalni testovi su izvršeni na test primerima veličine 22 iz skupova test primera V, kao i manjih test primera veličina 10, 13, 16, 19 i 21 iz dva skupa test primera S i O, opisanih u sekciji 2.1.2. Ovi rezultati su prikazani u tabeli 6.3. Prva kolona sadrži ime test primera, nakon čega je naveden nivo rizika (rl) u slučaju test primera iz skupa V. Treća kolona sadrži veličinu svakog test primera, dok je ostatak tabele u istom formatu kao i u slučaju tabele 6.1. Podaci iz tabele 6.3 idu u prilog činjenici da je od značaja korišćenje modela FRCTVRP prilikom rada sa test primerima velikih dimenzija, jer za samo dva test primera veličine 10 rešenja su ista u oba modela.

Može se videti da se prednost u vremenu rešavanja uz pomoć CPLEX rešavača ne može jasno dodeliti ni jednom od dva poređena modela. Zapravo, to je i očekivano jer je novi razvijani model kreiran tako da ne bude teži od početnog.

Tabela 6.3: *Eksperimentalni rezultati poređenja modela RCTVRP i FRCTVRP na test primerima iz skupova V, O i S*

			RCTVRP		
Skup	rl	veličina	F_{cilja}	**Čvorovi u rutama**	**t[s]**
V	1	22	874.70	1→6|2→7|3→8→12|4|5|9|10|11→13|15→18|16|17→21|19|20→14|	4128.71
V	1.5	22	722.83	4→3|8→2→1→6|9→7|10→5|11|12|13|15|17|18|19→16|21→20→14|	449.17
V	2	22	654.53	2→1→6|4→3|8|9→7→5|10|12|13→11|14|15|16|17→20→18|21→19|	3429.91
V	2.5	22	655.24	1→2→6|4→3→11→10|5→7→16|8→14→17→20→18|9|12|13|15|21→19|	3086.47
V	3	22	559.77	1→2→5→7→10|6→3→4|9→8|11→13|12|14→16→17→18|15|20→21→19|	3135.43
O		10	26.83	**2→1→3|5→4→6|8→7→9**|	15.22
O		13	35.78	2→1→3|5→4→6|8→7→9|11→10→12|	10534.86
O		16	47.55	2→3→1→4→5|7→8→6→9|10|12|13→11→14→15|	3277.63
O		19	57.97	1→16|3→17→18→14|4→6→2|7→9→5|10→13|11→15|12→8|	3062.91
O		21	70.71	1→4→3→2|7→6→8→9→10|12|14→11→13→20|15|17|19→16→18→5|	2844.82
S		10	19.98	**3→2→1|6→5→4|9→8→7**|	34.54
S		13	26.65	3→2→1|6→5→4|9→8→7|12→11→10|	12829.08
S		16	39.84	5→4→3→2→1|9→10→8→7→6|11|12→13→14→15|	2545.69
S		19	44.63	2→3|7|8→9→4|6→5→1|11→12→10|15→14|16|17→18→13|	2751.47
S		21	73.25	2→1|3→4→5|10→9→6→8→7|13|14→15|16|17|18→19→20→11→12|	2923.67
			FRCTVRP		
Skup	rl	veličina	F_{cilja}	**Čvorovi u rutama**	**t[s]**
V	1	22	940.62	1|3|4|5|8→2→6→10|9→7|11|12|13|14|16|17|18→15|19|20|21|	744.64
V	1.5	22	761.37	1→2→6|3→4→8|5|7→9→10|11|12|13|14|18→15|19|20→16|21→17|	1960.54
V	2	22	701.71	3→4|5→7→9|8→6→1→2→10|11→13|12|15|16|17|18→20→14|19|21|	2042.16
V	2.5	22	710.80	1→2→6|4→3→8|5→7|11|12|13|14|15→10→9|16|19→17|21→20→18|	3574.96
V	3	22	602.92	2→1→6|10→7→5→9|11→3→4→8|12|13|15|16|17→20→18|21→19→14|	7434.60
O		10	31.50	**2→1→3|5→4→6|8→7→9**|	43.91
O		13	44.77	2→1→3|5→4→6|8→7→9|10→11|12|	4920.04
O		16	51.62	1→4→3→2|5|7→8→6→9→10|12→13→11→14→15|	4729.92
O		19	68.23	1|3→17|4|5→9|6→2|7→8→12|10→11→15|13|16→18→14|	8780.10
O		21	98.77	2|3→1→4|5|6→9→8→7|10|12|13→11→14|15|17|18→16|19|20|	7836.38
S		10	22.34	**3→2→1|9→8→7|6→5→4**|	53.91
S		13	31.87	3→2→1|6→5→4|7|9→8|12→11→10|	5358.31
S		16	61.10	1|2|5→4→3|6|10→9→8→7|11|13→12|15→14|	5167.44
S		19	55.10	1|3→2|6→5→4|7|9→8|10|12→11|13|15→14|16|18→17|	5890.35
S		21	78.31	2→1|5→4→3|7→6|8→10→9|12|13|15→14|11|20→19→18→17→16|	9723.64

7

Zaključak

U okviru ove disertacije su rešavana tri NP-teška problema različitim metodama računarske inteligencije. Poseban značaj prilikom rešavanja teških optimizacionih problema sve više imaju metaheuristike koje predstavljaju procedure visokog nivoa koje efikasno pronalaze kvalitetna rešenja. Prilikom rešavanja problema iz prakse, od velike važnosti je da se, u razumnom vremenu, dobiju optimalna ili bar rešenja dobrog kvaliteta. U okviru istraživanja za potrebe ove disertacije predložene su različite metaheurističke metode, koje su pažljivo implementirane i unapređene u skladu sa problemima koji su rešavani. Inicijalne metoda za rešavanje svakog od tri problema su najčešće odabrane na osnovu intuicije da će ta metoda da se pokaže dobro na tom tipu problema. Može se očekivati ukoliko se neki metod dobro pokazao na nekom sličnom problemu, da može poslužiti za rešavanje i razmatranog problema. Međutim, to nije uvek slučaj i o tome je diskutovano u poglavlju 5. U svakom slučaju, predložene metode su razvijene u skladu sa svakim od problema. Istraživanja su bila usmerena i u pravcu daljih unapređenja metoda rešavanja razmatranih problema, kao i u cilju preciznije formulacije problema iz prakse. U disertaciji se na više mesta ispostavilo da dobra rešenja u ovom domenu dolaze iz sveta fazi logike.

Prvi problem koji je predmet istraživanja u okviru ove disertacije pripada klasi problema rutiranja vozila. Naziva se problem rutiranja vozila sa ograničenim rizikom (engl. *Risk-constrained Cash-in-Transit Vehicle Routing Problem* - RCTVRP) i javio u sektoru za rad sa velikim količinama robe od velike vrednosti. Opisano je rešavanje problema RCTVRP pomoću hibridizacije metode GRASP sa metodom

ponovnog povezivanja puta (PR). Tom prilikom je dizajnirana specijalna struktura podataka koja omogućava efikasnije izvršavanja algoritama. U disertaciji je predložena PR metoda pogodna za rešavanje RCTVRP, ali i drugih problema rutiranja vozila. Pored toga, etapi konstrukcije rešenja u okviru metode GRASP dodata je fazi modifikacija u skladu sa problemom RCTVRP koja je dovela do poboljšanja performansi predloženog hibrida. Predloženi metod se najbolje pokazao za rešavanje problema RCTVRP u poređenju sa svim rezultatima dostupnim u literaturi.

U disertaciji su razmatrana i dva problema iz klase problema lokacije resursa: problem ravnomernog opterećenja (engl. *Load Balancing Problem* - LOBA) i problem maksimizacije minimalnog rastojanja (engl. *Max-Min Diversity Problem* - MMDP). Za problem LOBA predložene su dve hibridne metode: EA-VNS i RVNS-VNS. Predložene hibridne metode su se uspešno pokazale na svim test primerima i predstavljaju unapređenje rezultata iz literature. Za rešavanje problema MMDP predloženi evolutivni algoritam je unapređen fazi pravilom koje omogućava podašavanje parametra u toku izvršavanja algoritma. Ova modifikacija je unapredila performanse predloženog evolutivnog algoritma za problem MMDP.

Pored toga, u disertaciji je prikazana mogućnost korišćenja fazi logike u cilju modeliranja problema iz prakse. U tom cilju razvijen je novi fazi model za verziju problema RCTVRP, koji predstavlja unapređenje već poznatog klasičnog modela RCTVRP u smislu da uključuje adekvatno poređenje među dopustivim rutama korišćenjem fazi brojeva. Pored toga, dve matematičke formulacije fazi varijante problema su predstavljene u disertaciji. Poređenje FRCTVRP modela sa klasičnim RCTVRP modelom na javno dostupnim test primerima ukazuje da fazi pristup koji je predložen u ovoj disertaciji kao optimalna rešenja daje bezbednije rute.

Naučni doprinos ove disertacije predstavljaju sledeći najvažniji rezultati:

- Za etapu konstrukcije rešenja prilikom rešavanja problema RCTVRP metodom GRASP uvedena je modifikacija zasnovana na fazi logici, koja poboljšava performanse algoritma.

- U radu je kreirana nova struktura podataka i predložene su nove formule za ažuriranje podataka o rutama koje omogućavaju smanjenje vremenske složenosti lokalne pretrage za rešavanje RCTVRP problema.

- Predstavljen je novi pristup određivanja trajektorija u metodi ponovnog po-

vezivanja puta za rešavanje problema rutiranja vozila.

- Primenom predložene i implementirane hibridne metode dobijena su rešenja na instancama problema velikih dimenzija za problem LOBA koje do sada nisu rešavane.
- Dobijena su rešenja koja su bolja od do sada poznatih na instancama problema velikih dimenzija za problem RCTVRP.
- Predložena su fazi pravila za menjanje vrednosti parametra evolutivnog algoritma u toku izvršavanja i eksperimentima je pokazano da takav pristup unapređuje primenu tog algoritma na problem MMDP.
- Kreiran je nov model za varijaciju problema RCTVRP primenom fazi logike takav da optimalna rešenja imaju bezbednije rute, što su potvrdila i odgovarajuća eksperimentalna poređenja modela.

Dakle, može se zaključiti da istraživanja realizovana u ovoj disertaciji daju važan naučni doprinos rešavanju problema rutiranja vozila i lokacije resursa metodama računarske inteligencije, matematičkom modeliranju ovih problema, kao i primenama fazi logike. Rezultati ove disertacije otvaraju nove mogućnosti primene fazi logike u cilju povećavanja efikasnosti i kvaliteta primenjenih metoda i prilikom modeliranja realnih problema.

Literatura

[1] Emile Aarts, Jan Korst, and Wil Michiels. Simulated annealing. In *Search methodologies*, pages 187–210. Springer, 2005.

[2] Renata M Aiex, Silvio Binato, and Mauricio GC Resende. Parallel grasp with path-relinking for job shop scheduling. *Parallel Computing*, 29(4):393–430, 2003.

[3] Bahram Alidaee and Haibo Wang. A note on heuristic approach based on ubqp formulation of the maximum diversity problem. *Journal of the Operational Research Society*, 68(1):102–110, 2017.

[4] Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768, 2015.

[5] Sibel Alumur and Bahar Y Kara. Network hub location problems: The state of the art. *European journal of operational research*, 190(1):1–21, 2008.

[6] Diogo V Andrade and Mauricio GC Resende. Grasp with evolutionary path-relinking. In *Proc. of Seventh Metaheuristics International Conference (MIC 2007)*, 2007.

[7] Konstantinos N Androutsopoulos and Konstantinos 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.

[8] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[9] Charles Audet, Vincent Béchard, and Sébastien Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[10] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[11] Maria Barbati, Giuseppe Bruno, and Alfredo Marín. Balancing the arrival times of users in a two-stage location problem. *Annals of Operations Research*, 246(1-2):273–288, 2016.

[12] Mohamed Barkaoui and Michel Gendreau. An adaptive evolutionary approach for real-time vehicle routing and dispatching. *Computers & Operations Research*, 40(7):1766–1776, 2013.

[13] Sumanta Basu, Megha Sharma, and Partha Sarathi Ghosh. Metaheuristic applications on discrete facility location problems: a survey. *Opsearch*, 52(3):530–561, 2015.

[14] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004.

[15] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[16] Lawrence Bodin and Bruce Golden. Classification in vehicle routing and scheduling. *Networks*, 11(2):97–108, 1981.

[17] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

[18] José Brandão. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, 2004.

[19] Julio Brito, F Javier Martínez, JA Moreno, and José L Verdegay. An ACO hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints. *Applied Soft Computing*, 32:154–163, 2015.

[20] Leo Budin, Marin Golub, and Andrea Budin. Traditional techniques of genetic algorithms applied to floating-point chromosome representations. *sign*, 1(11):52, 2010.

[21] Erbao Cao and Mingyong Lai. The open vehicle routing problem with fuzzy demands. *Expert Systems with Applications*, 37(3):2405 – 2411, 2010.

[22] T William Chien, Anantaram Balakrishnan, and Richard T Wong. An integrated inventory allocation and vehicle routing problem. *Transportation science*, 23(2):67–76, 1989.

[23] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[24] Maurice Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

[25] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[26] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[27] Teodor Gabriel Crainic, Simona Mancini, Guido Perboli, and Roberto Tadei. Grasp with path relinking for the two-echelon vehicle routing problem. In *Advances in Metaheuristics*, pages 113–125. Springer, 2013.

[28] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

[29] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[30] Dipankar Dasgupta and Zbigniew Michalewicz. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.

[31] Lawrence Davis. Handbook of genetic algorithms. 1991.

[32] Lawrence D Davis, Kenneth De Jong, Michael D Vose, and L Darrell Whitley. *Evolutionary algorithms*, volume 111. Springer Science & Business Media, 2012.

[33] Federico Della Croce, Andrea Grosso, and Marco Locatelli. A heuristic approach for the max–min diversity problem based on max-clique. *Computers & Operations Research*, 36(8):2429–2433, 2009.

[34] Guy Desaulniers, Oli BG Madsen, and Stefan Ropke. The vehicle routing problem with time windows. *Vehicle routing: Problems, methods, and applications*, 18:119–159, 2014.

[35] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[36] Juan A Díaz, Dolores E Luna, José-Fernando Camacho-Vallejo, and Martha-Selene Casas-Ramírez. Grasp and hybrid grasp-tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems with Applications*, 82:67–76, 2017.

[37] Aleksandar Djenić, Miroslav Marić, Zorica Stanimirović, and Predrag Stanojević. A variable neighbourhood search method for solving the long-term care facility location problem. *IMA Journal of Management Mathematics*, 28(2):321–338, 2017.

[38] Aleksandar Djenić, Nina Radojičić, Miroslav Marić, and Marko Mladenović. Parallel VNS for bus terminal location problem. *Applied Soft Computing*, 42:448–458, 2016.

[39] Marco Dorigo and Mauro Birattari. Ant colony optimization. In *Encyclopedia of machine learning*, pages 36–39. Springer, 2011.

[40] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[41] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new metaheuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 1470–1477. IEEE, 1999.

[42] Darko Drakulić, Aleksandar Takači, and Miroslav Marić. New model of maximal covering location problem with fuzzy conditions. *Computing & Informatics*, 35(3), 2016.

[43] Darko Drakulić, Aleksandar Takači, and Miroslav Marić. Fuzzy covering location problems with different aggregation operators. *Filomat*, 31(2):513–522, 2017.

[44] Zvi Drezner. *Facility location: a survey of applications and methods*. Springer Verlag, 1995.

[45] Ke-Lin Du and MNS Swamy. Particle swarm optimization. In *Search and Optimization by Metaheuristics*, pages 153–173. Springer, 2016.

[46] Abraham Duarte, Nenad Mladenović, Jesús Sánchez-Oro, and Raca Todosijević. *Variable Neighborhood Descent*, pages 1–27. Springer International Publishing, Cham, 2016.

[47] Russell C Eberhart and Yuhui Shi. *Computational intelligence: concepts to implementations*. Elsevier, 2011.

[48] Richard Eglese and Sofoclis Zambirinis. Disruption management in vehicle routing and scheduling for road freight transport: a review. *TOP*, Feb 2018.

[49] Aguston Eiben, Zbigniew Michalewicz, Marc Schoenauer, and Jim Smith. Parameter control in evolutionary algorithms. *Parameter setting in evolutionary algorithms*, pages 19–46, 2007.

[50] Horst A Eiselt and Gilbert Laporte. Objectives in location problems. Technical report, 1995.

[51] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[52] Andries P. Engelbrecht. *Computational Intelligence. An Introduction.* Wiley, 2 edition, 2007.

[53] A Erasmus. The cit industry in south africa–its challenges and solutions. In *ESTA business conference, exhibition and general assembly. Bratislava, Slovakia*, 2012.

[54] Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[55] John Willmer Escobar, Rodrigo Linfati, Paolo Toth, and Maria G Baldoquin. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20(5):483–509, 2014.

[56] Inmaculada Espejo, Alfredo Marín, and Antonio M Rodríguez-Chía. Closest assignment constraints in discrete location problems. *European Journal of Operational Research*, 219(1):49–58, 2012.

[57] H Faria, Silvio Binato, Mauricio GC Resende, and Djalma M Falcão. Power transmission network design by greedy randomized adaptive path relinking. *IEEE Transactions on Power Systems*, 20(1):43–49, 2005.

[58] Awi Federgruen and Paul Zipkin. A combined vehicle routing and inventory allocation problem. *Operations research*, 32(5):1019–1037, 1984.

[59] Thomas A Feo and Mauricio GC Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71, 1989.

[60] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[61] Thomas A Feo, Kishore Sarathy, and John McGahan. A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers & Operations Research*, 23(9):881–895, 1996.

[62] Paola Festa, Panos M Pardalos, Leonidas S Pitsoulis, and Mauricio GC Resende. Grasp with path relinking for the weighted maxsat problem. *Journal of Experimental Algorithmics (JEA)*, 11:2–4, 2007.

[63] Paola Festa, Panos M Pardalos, Mauricio GC Resende, and Celso C Ribeiro. Randomized heuristics for the max-cut problem. *Optimization methods and software*, 17(6):1033–1058, 2002.

[64] Paola Festa and Mauricio GC Resende. Grasp: An annotated bibliography. In *Essays and surveys in metaheuristics*, pages 325–367. Springer, 2002.

[65] Paola Festa and Mauricio GC Resende. An annotated bibliography of grasp. *Operations Research Letters*, 8:67–71, 2004.

[66] Paola Festa and Mauricio GC Resende. An annotated bibliography of grasp–part i: Algorithms. *International Transactions in Operational Research*, 16(1):1–24, 2009.

[67] Paola Festa and Mauricio GC Resende. An annotated bibliography of grasp–part ii: Applications. *International Transactions in Operational Research*, 16(2):131–172, 2009.

[68] Paola Festa and Mauricio GC Resende. Hybridizations of grasp with path-relinking. *Hybrid metaheuristics*, 434:135–155, 2013.

[69] Paul Field. Nonbinary transforms for genetic algorithm problems. In *AISB Workshop on Evolutionary Computing*, pages 38–50. Springer, 1994.

[70] Vladimir Filipović. Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama. *Magistarski rad, Univerzitet u Beogradu, Matematički fakultet*, 1998.

[71] Vladimir Filipović. Fine-grained tournament selection operator in genetic algorithms. *Computing and Informatics*, 22(2):143–161, 2012.

[72] Vladimir Filipović, Jozef Kratica, Dušan Tošić, and Ivana Ljubić. Fine grained tournament selection for the simple plant location problem. In *Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications-WSC5*, pages 152–158, 2000.

[73] Vladimir Filipović, Dušan Tošić, and Jozef Kratica. Experimental results in applying of fine grained tournament selection. In *Proceedings of the 10th Congress of Yugoslav Mathematicians*, pages 331–336, 2001.

[74] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[75] Roberto Diéguez Galvão and Charles ReVelle. A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88(1):114–123, 1996.

[76] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[77] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[78] Seyed Farid Ghannadpour, Simak Noori, Reza Tavakkoli-Moghaddam, and Keivan Ghoseiri. A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application. *Applied Soft Computing*, 14, Part C:504 – 527, 2014.

[79] Jay B Ghosh. Computational aspects of the maximum diversity problem. *Operations research letters*, 19(4):175–181, 1996.

[80] F Glover and M Laguna. Tabu search, modern heuristic techniques for combinatorial problems, edited by cr reeves, 1993.

[81] Fred Glover. Future paths for integer programming and links to ai. *Computers & Operations Research*, 13(5):553–549, 1986.

[82] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[83] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[84] Fred Glover. Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in computer science and operations research*, pages 1–75. Springer, 1997.

[85] Fred Glover. A template for scatter search and path relinking. In *Artificial evolution*, pages 1–51. Springer, 1998.

[86] Fred Glover, Ching-Chung Kuo, and Krishna S Dhir. Heuristic algorithms for the maximum diversity problem. *Journal of information and Optimization Sciences*, 19(1):109–132, 1998.

[87] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 3261–3362. Springer, 2013.

[88] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29(3):653–684, 2000.

[89] Bruce Golden, Arjang Assad, Larry Levy, and Filip Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66, 1984.

[90] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[91] Gregory Gutin and Abraham P Punnen. *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media, 2006.

[92] J Halpern and O Maimon. Accord and conflict among several objectives in locational decisions on tree networks. *Locational Analysis of Public Facilities*, pages 391–304, 1983.

[93] Pierre Hansen and Nenad Mladenović. Variable neighborhood search for the p-median. *Location Science*, 5(4):207–226, 1997.

[94] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

[95] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

[96] Pierre Hansen and Nenad Mladenović. Developments of variable neighborhood search. In *Essays and surveys in metaheuristics*, pages 415–439. Springer, 2002.

[97] Pierre Hansen and Nenad Mladenović. Variable neighborhood search. In *Search methodologies*, pages 313–337. Springer, 2014.

[98] Francisco Herrera and Manuel Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing*, 8:95–125, 1996.

[99] Sin C. Ho and Michel Gendreau. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1):55–72, 2006.

[100] Sin C. Ho and Wai Yuen Szeto. Grasp with path relinking for the selective pickup and delivery problem. *Expert Systems with Applications*, 51:14–25, 2016.

[101] William Ho, George TS Ho, Ping Ji, and Henry CW Lau. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 21(4):548–557, 2008.

[102] John 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.

[103] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379 – 396, 2003.

[104] David Johnson. Local optimization and the traveling salesman problem. *Automata, languages and programming*, pages 446–461, 1990.

[105] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015.

[106] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[107] Mostepha R Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali 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.

[108] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[109] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular norms*, volume 8. Springer Science & Business Media, 2013.

[110] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey, 1995.

[111] A.L. Kok, E.W. Hans, and J.M.J. Schutten. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5):910 – 918, 2012.

[112] Ching-Chung Kuo, Fred Glover, and Krishna S Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

[113] R.J. Kuo, Ferani E. Zulvia, and Kadarsah Suryadi. Hybrid particle swarm optimization with genetic algorithm for solving capacitated vehicle routing problem with fuzzy demand – a case study on garbage collection system. *Applied Mathematics and Computation*, 219(5):2574 – 2588, 2012.

[114] Manuel Laguna and Fred Glover. Bandwidth packing: a tabu search approach. *Management science*, 39(4):492–500, 1993.

[115] Manuel Laguna and Rafael Marti. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11(1):44–52, 1999.

[116] Bojana Lazović, Miroslav Marić, Vladimir Filipović, and Aleksandar Savić. An integer linear programming formulation and genetic algorithm for the maximum set splitting problem. *Publications de l'Institut Mathematique*, 92(106):25–34, 2012.

[117] Hongtao Lei, Gilbert Laporte, and Bo Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775–1783, 2011.

[118] Jan Karel Lenstra and AHG Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[119] Xiangyong Li, Stephen C.H. Leung, and Peng Tian. A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications*, 39(1):365 – 374, 2012.

[120] T Warren Liao, PC Chang, RJ Kuo, and C-J Liao. A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. *Applied Soft Computing*, 21:180–193, 2014.

[121] Tsai-Yun Liao and Ta-Yin Hu. An object-oriented evaluation framework for dynamic vehicle routing problems under real-time information. *Expert Systems with Applications*, 38(10):12548–12558, 2011.

[122] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21(2):498, 1973.

[123] Shen Lin. Computer solutions of the traveling salesman problem. *The Bell system technical journal*, 44(10):2245–2269, 1965.

[124] SC Liu and CC Lin. A heuristic method for the combined location routing and inventory problem. *The International Journal of Advanced Manufacturing Technology*, 26(4):372–381, 2005.

[125] FJ Lobo, Cláudio F Lima, and Zbigniew Michalewicz. *Parameter setting in evolutionary algorithms*, volume 54. Springer Science & Business Media, 2007.

[126] M Cruz López-de-los Mozos and Juan A Mesa. The maximum absolute deviation measure in location problems on networks. *European Journal of Operational Research*, 135(1):184–194, 2001.

[127] Luiz AN Lorena and Edson LF Senne. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876, 2004.

[128] Luiz Antonio Nogueira Lorena and Edson Luiz França Senne. Local search heuristics for capacitated p-median problems. *Networks and Spatial Economics*, 3(4):407–419, 2003.

[129] Miroslav Marić. Rešavanje nekih np-teških hijerarhijsko-lokacijskih problema primenom genetskih algoritama. *Doktorska disertacija*, 2009.

[130] Miroslav Marić. An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem. *Computing and Informatics*, 29(2):183–201, 2012.

[131] Miroslav Marić, Zorica Stanimirović, and Srdjan Božović. Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, 227(1):3–23, 2015.

[132] Miroslav Marić, Zorica Stanimirović, Aleksandar Djenić, and Predrag Stanojević. Memetic algorithm for solving the multilevel uncapacitated facility location problem. *Informatica*, 25(3):439–466, 2014.

[133] Miroslav Marić, Zorica Stanimirović, Nikola Milenković, and Aleksandar Đenić. Metaheuristic approaches to solving large-scale bilevel uncapacitated facility location problem with clients' preferences. *Yugoslav Journal of Operations Research*, 25(3):361–378, 2015.

[134] Miroslav Marić, Zorica Stanimirović, and Predrag Stanojević. An efficient memetic algorithm for the uncapacitated single allocation hub location problem. *Soft Computing*, 17(3):445–466, 2013.

[135] Alfredo Marín. The discrete facility location problem with balanced allocation of customers. *European Journal of Operational Research*, 210(1):27–38, 2011.

[136] Michael T Marsh and David A Schilling. Equity measurement in facility location analysis: A review and framework. *European Journal of Operational Research*, 74(1):1–17, 1994.

[137] Geraldo R. Mateus, Mauricio G. C. Resende, and Ricardo M. A. Silva. Grasp with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics*, 17(5):527–565, Oct 2011.

[138] Jorge E Mendoza, Louis-Martin Rousseau, and Juan G Villegas. A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. *Journal of Heuristics*, 22(4):539–566, 2016.

[139] Juan A Mesa, Justo Puerto, and Arie Tamir. Improved algorithms for several network location problems with equality measures. *Discrete applied mathematics*, 130(3):437–448, 2003.

[140] József Mezei, Matteo Brunelli, and Christer Carlsson. A fuzzy approach to using expert knowledge for tuning paper machines. *Journal of the Operational Research Society*, 68(6):605–616, Jun 2017.

[141] A Michael and Hideyuki Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 76–83. Morgan Kaufmann, 1993.

[142] Stefan Mišković and Zorica Stanimirović. A hybrid metaheuristic method for the deterministic and robust uncapacitated multiple allocation p-hub centre problem. *European Journal of Industrial Engineering*, 11(5):631–662, 2017.

[143] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[144] Nenad Mladenović. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. In *Papers presented at Optimization Days*, volume 12, 1995.

[145] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[146] Gábor Nagy and Saïd Salhi. Location-routing: Issues, models and methods. *European journal of operational research*, 177(2):649–672, 2007.

[147] Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler Calvo. A hybrid tabu search for the m-peripatetic vehicle routing problem. In *Matheuristics*, pages 253–266. Springer, 2009.

[148] Zoran Ognjanović and Nenad Krdžavac. Uvod u teorijsko računarstvo. *FON, Beograd*, 2004.

[149] Endre Pap. *Fazi mere i njihova primena.* Prirodno-matematički fakultet, 1999.

[150] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[151] Luciana S Pessoa, Mauricio GC Resende, and Celso C Ribeiro. A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Computers & Operations Research*, 40(12):3132–3146, 2013.

[152] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[153] Michael Polacek, Richard F Hartl, Karl Doerner, and Marc Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10(6):613–627, 2004.

[154] Jean-Yves Potvin, Guy Lapalme, and Jean-Marc Rousseau. A generalized k-opt exchange procedure for the mtsp. *INFOR: Information Systems and Operational Research*, 27(4):474–481, 1989.

[155] Marcelo Prais and Celso C. Ribeiro. Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000.

[156] Nina Radojičić. Rešavanje nekih NP-teških problema diskretne optimizacije. *master rad, Matematički fakultet, Beograd*, 2011.

[157] Nina Radojičić. An approach to solving the min-max diversity problem using genetic algorithm with fuzzy decisions. *Transactions on Internet Research (TIR), Special issue - Special Issue on Solving Some Computationally Hard Problems II*, 13(1):21–27, 2017.

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

[159] Mauricio GC Resende, Rafael Martí, Micael Gallego, and Abraham Duarte. Grasp and path relinking for the max–min diversity problem. *Computers & Operations Research*, 37(3):498–508, 2010.

[160] Mauricio GC Resende and Celso C Ribeiro. A grasp for graph planarization. *Networks*, 29(3):173–189, 1997.

[161] Mauricio GC Resende and Celso C Ribeiro. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer, 2016.

[162] Mauricio GC Resende and Renato F Werneck. A hybrid heuristic for the p-median problem. *Journal of heuristics*, 10(1):59–88, 2004.

[163] Da Ruan. *Fuzzy set theory and advanced mathematical applications*, volume 4. Springer Science & Business Media, 2012.

[164] Dimitrios Sariklis and Susan Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51(5):564–573, 2000.

[165] Kumara Sastry, David E Goldberg, and Graham Kendall. Genetic algorithms. In *Search methodologies*, pages 93–117. Springer, 2014.

[166] Rudolf Seising. *The Fuzzification of Systems: The Genesis of Fuzzy Set Theory and its Initial Applications-Developments up to the 1970s*. Springer Publishing Company, Incorporated, 2007.

[167] Zuo-Jun Max Shen, Collette Coullard, and Mark S Daskin. A joint location-inventory model. *Transportation science*, 37(1):40–55, 2003.

[168] Zuo-Jun Max Shen, Roger Lezhou Zhan, and Jiawei Zhang. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS Journal on Computing*, 23(3):470–482, 2011.

[169] Geiza C Silva, Luiz S Ochi, and Simone L Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. *Lecture notes in computer science*, 3059:498–512, 2004.

[170] Marcos Roberto Silva and Claudio B Cunha. New simple and efficient heuristics for the uncapacitated single allocation hub location problem. *Computers & Operations Research*, 36(12):3152–3165, 2009.

[171] Lance Smith, Erin Louis, et al. Cash in transit armed robbery in australia. *Trends and issues in crime and criminal justice*, (397):1, 2010.

[172] Zorica Stanimirović. Genetski algoritam za rešavanje nekih np-teških hab lokacijskih problema. *Doktorska Disertacija*, 2007.

[173] Zorica Stanimirović. An efficient genetic algorithm for the uncapacitated multiple allocation p-hub median problem. *Control & Cybernetics*, 37(3), 2008.

[174] Zorica Stanimirović, Jozef Kratica, and Djordje Dugošija. Genetic algorithms for solving the discrete ordered median problem. *European Journal of Operational Research*, 182(3):983–1001, 2007.

[175] Zorica Stanimirovic, Miroslav Maric, Srdjan Bozovic, and Predrag Stanojevic. An efficient evolutionary algorithm for locating long-term care facilities. *Information Technology and Control*, 41(1):77–89, 2012.

[176] Zorica Stanimirovic, Miroslav Maric, Nina Radojicic, and Srdjan Bozovic. Two efficient hybrid metaheuristic methods for solving the load balance problem. *Applied and Computational Mathematics*, 13(3):332–349, 2014.

[177] Predrag Stanojević, Miroslav Marić, and Zorica Stanimirović. A hybridization of an evolutionary algorithm and a parallel branch and bound for solving the capacitated single allocation hub location problem. *Applied Soft Computing*, 33:24–36, 2015.

[178] Michio Sugeno. Fuzzy measures and fuzzy integrals—a survey. In *Readings in Fuzzy Sets for Intelligent Systems*, pages 251–257. Elsevier, 1993.

[179] Aleksandar Takači, Ivana Štajner-Papuga, Darko Drakulić, and Miroslav Marić. An extension of maximal covering location problem based on the choquet integral. *Acta Polytechnica Hungarica*, 13(4), 2016.

[180] Đurđica Takači, Miroslav Marić, Gordana Stankov, and Aleksandar Djenić. Efficiency of using VNS algorithm for forming heterogeneous groups for CSCL learning. *Computers & Education*, 109:98–108, 2017.

[181] Luca Talarico, Kenneth Sörensen, and Johan Springael. Metaheuristics for the risk-constrained cash-in-transit vehicle routing problem. *European Journal of Operational Research*, 244(2):457–470, 2015.

[182] Luca Talarico, Johan Springael, Kenneth Sörensen, and Fabio Talarico. A large neighbourhood metaheuristic for the risk-constrained cash-in-transit vehicle routing problem. *Computers & Operations Research*, 78:547 – 556, 2017.

[183] Jiafu Tang, Zhendong Pan, Richard Y.K. Fung, and Henry Lau. Vehicle routing problem with fuzzy time windows. *Fuzzy Sets and Systems*, 160(5):683 – 695, 2009.

[184] Barbaros C Tansel, Richard L Francis, and Timothy J Lowe. State of the art—location on networks: a survey. part i: the p-center and p-median problems. *Management science*, 29(4):482–497, 1983.

[185] Dušan Teodorović and Goran Pavković. The fuzzy set theory approach to the vehicle routing problem when demand at nodes is uncertain. *Fuzzy Sets and Systems*, 82(3):307 – 317, 1996.

[186] Paolo Toth and Daniele Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512, 2002.

[187] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[188] Stefan Voß, Silvano Martello, Ibrahim H Osman, and Catherine Roucairol. *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Springer Science & Business Media, 2012.

[189] Christos Voudouris and Edward Tsang. Guided local search and its application to the traveling salesman problem. *European journal of operational research*, 113(2):469–499, 1999.

[190] Zhenyuan Wang and George J Klir. *Fuzzy measure theory*. Springer Science & Business Media, 2013.

[191] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[192] Qinghua Wu and Jin-Kao Hao. A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709, 2015.

[193] Shangyao Yan, Sin-Siang Wang, and Ming-Wei Wu. A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem. *Computers & Industrial Engineering*, 63(2):464 – 473, 2012.

[194] Emmanouil E Zachariadis, Christos D Tarantilis, and Chris T Kiranoudis. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with applications*, 36(2):1070–1081, 2009.

[195] Miodrag Živković. Algoritmi. *Matematički fakultet, Beograd*, 2000.

Biografija autora

Nina Radojičić rođena je 25. oktobra 1987. godine u Čačku. Osnovnu školu „Vuk Karadžić" i Gimnaziju u Čačku završila je kao nosilac diplome „Vuk Karadžić". Tokom osnovne škole i Gimnazije učestvovala je i nagrađivana na takmičenjima iz matematike, hemije, srpskog jezika, kao i na sportskim takmičenjima.

Matematički fakultet u Beogradu upisala je 2006. godine, smer Računarstvo i informatika. Diplomirala je u junskom ispitnom roku 2010. godine sa prosečnom ocenom 10,00 kao student generacije. Master studije, na studijskom programu Matematika, modul Računarstvo i informatika, završila je 2011. godine sa prosečnom ocenom 10,00. Master rad pod nazivom „Rešavanje nekih NP-teških problema diskretne optimizacije" odbranila je pod mentorstvom dr Miroslava Marića. Doktorske akademske studije studijskog programa Informatika upisala je 2011. godine. Sve ispite predviđene planom studija položila je sa ocenom 10.

Dobitnik je stipendije fonda za mlade talente Republike Srbije za studente završnih godina studija 2009. godine, kao i 2010. godine kao student master studija. Pored toga, 2009. godine joj je uručena pohvalnica za uspeh tokom studija „Udruženja univerzitetskih profesora i naučnika Srbije". Tokom 2010/11. godine bila je stipendista Republičke fondacije za razvoj naučnog i umetničkog podmlatka, a po dve godine za redom dobijala je stipendije Republike Srbije i grada Čačka. Nakon završetka osnovnih studija, tokom avgusta i septembara 2010. godine je u okviru prakse radila na projektu o mrežnim simulacijama na Univerzitetu Fernuni, Hagen (Nemačka).

Od oktobra 2010. godine radila je kao saradnik u nastavi na Matematičkom fakultetu u Beogradu za naučnu oblast Računarstvo i informatika, a od školske 2012/2013. godine zaposlena je kao asistent na istoj Katedri. Držala je vežbe iz predmeta: Programiranje 1, Programiranje 2, Relacione baze podataka, Programiranje baza podataka, Baze podataka 2, Bioinformatika (Master studije), Kriptogra-

fija (Master studije), Konstrukcija i analiza algoritama (Studijski program Informatika), Konstrukcija i analiza algoritama (Studijski program Matematika, modul Računarstvo i informatika) i Metodika nastave računarstva C sa praktikumom (Master studije). Koautor je zbirke zadataka iz Programiranja 2, kao i više naučnih radova.

Tokom doktorskih studija stručno usavršavanje je obavljala na fakultetima i istraživačkim institucijama: CIRM Marseille (Francuska) maj 2012. godine; ULPGC Las Palmas (Španija) jul 2013. godine; KTH Stockholm (Švedska) jun 2014. godine; JKU Linz (Austrija) avgust 2014. godine; Zuze Institute Berlin (Nemačka) oktobar 2014. godine i septembar i oktobar 2015. godine; TU Berlin (Nemačka), jun 2017. god.

Od 2011. godine do danas angažovana je na projektu Ministarstva prosvete, nauke i tehnološkog razvoja pod nazivom „Digitalizacija kulturnog i naučnog nasleđa sa primenama u srednjoškolskoj i univerzitetskoj nastavi matematike, informatike, astronomije, istorije i srpskog jezika", broj 044006. Od novembra 2011. godine do januara 2013. godine bila je zaposlena sa delom radnog vremena kao istraživač na Matematičkom institutu SANU.

Изјава о ауторству

Име и презиме аутора Нина Радојичић

Број индекса 2038/2011

Изјављујем

да је докторска дисертација под насловом

Примена фази логике за решавање NP-тешких проблема рутирања возила и локације ресурса методама рачунарске интелигенције

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

Потпис аутора

У Београду, _____

_____

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Нина Радојичић

Број индекса 2038/2011

Студијски програм Информатика

Наслов рада Примена фази логике за решавање NP-тешких проблема рутирања возила и локације ресурса методама рачунарске интелигенције

Ментор др Мирослав Марић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива 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 аутора

У Београду, _____

_____

Изјава о коришћењу

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

Примена фази логике за решавање NP-тешких проблема рутирања возила и локације ресурса методама рачунарске интелигенције

која је моје ауторско дело.

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

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

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, _____

_____

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

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

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

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

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

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