Rješavanje problema usmjeravanja vozila hibridnim staničnim evolucijskim algoritmom

Galić, Ante

Sveučilište u Zagrebu

FAKULTET PROMETNIH ZNANOSTI

mr. sc. Ante Galić

RJEŠAVANJE PROBLEMA USMJERAVANJA VOZILA HIBRIDNIM STANIČNIM EVOLUCIJSKIM ALGORITMOM

DOKTORSKI RAD

Zagreb, 2018.

Sveučilište u Zagrebu

FAKULTET PROMETNIH ZNANOSTI

mr. sc. Ante Galić

RJEŠAVANJE PROBLEMA USMJERAVANJA VOZILA HIBRIDNIM STANIČNIM EVOLUCIJSKIM ALGORITMOM

DOKTORSKI RAD

Mentor:
prof. dr. sc. Tonči Carić

Zagreb, 2018.

University of Zagreb

FACULTY OF TRANSPORT AND TRAFFIC SCIENCES

Ante Galić, M.Sc.

SOLVING THE VEHICLE ROUTING PROBLEM USING HYBRID CELLULAR EVOLUTIONARY ALGORITHM

DOCTORAL THESIS

Supervisor:
Tonči Carić, PhD, Full Professor

Zagreb, 2018.

Povjerenstvo za ocjenu doktorskog rada:

1. prof. dr. sc. Hrvoje Gold, predsjednik
   Sveučilište u Zagrebu, Fakultet prometnih znanosti
2. prof. dr. sc. Tonči Carić, mentor, član
   Sveučilište u Zagrebu, Fakultet prometnih znanosti
3. prof. dr. sc. Domagoj Jakobović, član
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
4. prof. dr. sc. Sadko Mandžuka, zamjena
   Sveučilište u Zagrebu, Fakultet prometnih znanosti

Povjerenstvo za obranu doktorskog rada:

1. prof. dr. sc. Hrvoje Gold, predsjednik
   Sveučilište u Zagrebu, Fakultet prometnih znanosti
2. prof. dr. sc. Tonči Carić, mentor, član
   Sveučilište u Zagrebu, Fakultet prometnih znanosti
3. prof. dr. sc. Domagoj Jakobović, član
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
4. prof. dr. sc. Sadko Mandžuka, zamjena
   Sveučilište u Zagrebu, Fakultet prometnih znanosti

Datum obrane doktorskog rada: 25. siječnja, 2018. g.

O MENTORU

Prof. dr. sc. Tonči Carić rođen je 26. srpnja 1969. u Splitu. Diplomirao je 1993. godine i magistrirao 2000. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Doktorirao je na Fakultetu prometnih znanosti Sveučilišta u Zagrebu 2004. godine. Trenutno je zaposlen na Fakultetu prometnih znanosti u Zagrebu, na Zavodu za inteligentne transporte sustave, kao redovni profesor. Uključen je u znanstveno-istraživački rad na projektima financiranim sredstvima Europske unije, nacionalnim istraživačkim projektima te stručnim projektima. Njegovi znanstveni interesi su u području razvoja inteligentnih transportnih sustava, optimizacija prometnih procesa i baza podataka.

Zahvala

Hvala mentoru prof. dr. sc. Tončiju Cariću na stručnom vodstvu, usmjeravanju i poticanju te na nesebičnoj pomoći pri izradi ove disertacije. Hvala kolegi dr. sc. Juraju Fosinu na brojnim diskusijama vezanim uz obrađenu temu, vrijednim savjetima i sugestijama, prof. dr. sc. Domagoju Jakoboviću na ispravcima i komentarima, prof. dr. sc. Hrvoju Goldu na detaljnom iščitavanju, ispravcima i komentarima te na širenju pozitivnog ozračja i poticanja na znanstveno istraživački rad tijekom čitavog trajanja doktorskog studija. Hvala sinu Andri i supruzi Maji na razumijevanju i strpljenju, roditeljima Marku i Ružici, braći Ivanu, Nikoli i sestri Ani na ohrabrivanju i podršci.

Ante Galić,

Zagreb, 25. siječnja, 2018.

Sažetak

Problem usmjeravanja vozila (VRP) kompleksan je kombinatorički problem s kojim se svakodnevno susreću tvrtke koje obavljaju dostavu robe. Njegovim učinkovitim rješavanjem moguće je značajno smanjiti troškove dostave. Metaheurističkim metodama moguće je relativno brzo pronaći visoko kvalitetna rješenja. Stanični evolucijski algoritam metaheuristički je algoritam kod kojeg su jedinke iz populacije raspoređene unutar toroidalne mreže i mogu biti u interakciji samo sa obližnjim jedinkama. Podešavanjem selekcijskog pritiska moguće je postići odgovarajući omjer diverzifikacije i intenzifikacije koji je ključan za uspješnost algoritma. Hibridizacija postupkom pretraživanja velikog susjedstva ubrzava pronalazak visoko kvalitetnih rješenja. Razvijeni algoritam testiran je na nekoliko skupova ispitnih zadataka te na problemima dostave hrvatskih tvrtki. Rezultati ostvareni na ispitnim zadacima pokazuju da učinkovitost algoritma ne odstupa mnogo od najboljih poznatih algoritama za ovu vrstu problema, dok rezultati ostvareni na problemima hrvatskih tvrtki pokazuju da je primjenom algoritma moguće postići značajne uštede.

Ključne riječi: problem trgovačkog putnika, TSP, problem usmjeravanja vozila, VRP, problem usmjeravanja vozila s vremenskim prozorima, VRPTW, metaheurističke metode, genetski algoritam, evolucijski algoritam, stanični evolucijski algoritam, cEA, stanični genetski algoritam, cGA, kombinatorička optimizacija.

Abstract

Vehicle Routing Problem (VRP) is a complex combinatorial problem encountered daily by companies that are dealing with goods delivery. With its efficient solving it is possible to significantly reduce the cost of delivery. Metaheuristic methods are capable of finding high-quality solutions in reasonable amount of time. The cellular evolutionary algorithm is a metaheuristic algorithm in which the individuals from the population are distributed within the toroidal grid and can interact only with nearby entities. By adjusting the selection pressure, it is possible to achieve the appropriate ratio of diversification and intensification that is crucial to the success of the algorithm. Hybridization by a large neighborhood search accelerates the finding of high quality solutions. The developed algorithm has been tested on several sets of benchmarks and on the delivery problems of Croatian companies. The results obtained on the benchmarks show that the efficiency of the algorithm does not differ much from the best-known algorithms for this type of problem, while the results achieved on the problems of Croatian companies show that it is possible to achieve significant savings by algorithm application.

Keywords: traveling salesman problem, TSP, vehicle routing problem, VRP, vehicle routing problem with time windows, VRPTW, metaheuristics, genetic algorithm, evolutionary algorithm, cellular evolutionary algorithm, cEA, cellular genetic algorithm, cGA, combinatorial optimization.

Sadržaj

Poglavlje 1

Uvod

Problem usmjeravanja vozila (eng. *Vehicle Routing Problem*, kratica VRP) kompleksan je kombinatorički problem s kojim se svakodnevno susreću tvrtke koje obavljaju distribuciju i/ili prikupljanje neke vrste tereta. Njegovim učinkovitim rješavanjem moguće je značajno smanjiti ukupne transportne troškove, povećati konkurentnost tvrtke distributera, te povećati kvalitetu pružene usluge. Planiranje distribucije uključuje dodjeljivanje svake pojedine lokacije iskrcaja nekom od raspoloživih vozila, određivanje optimalnog redoslijeda obilaska lokacija (rute) za svako vozilo, te najpovoljnijeg (najjeftinijeg, najbržeg ili najkraćeg) puta između svakog para lokacija u ruti vozila. Pri planiranju je nužno uvažiti ograničenja kapaciteta vozila, vremenska ograničenja (radno vrijeme vozača, dogovorene termine iskrcaja) ali i sva druga ograničenja čiji broj i kompleksnost varira od tvrtke do tvrtke i od problema do problema. U idealnom slučaju, rješavanjem ovog problema nastoji se pronaći optimalno rješenje kojim će se minimizirati sredstva utrošena za obavljanje distribucije (broj korištenih vozila i vozača, utrošeno gorivo, vrijeme i dr.). Zbog nesagledivog broja mogućih rješenja kod problema s relativno velikim brojem lokacija (više od 100), najčešće je nemoguće znati je li pronađeno rješenje optimalno pa stoga rješavanje problema podrazumijeva ispitivanje što većeg broja mogućih rješenja u što kraćem vremenu. U proteklih pedeset godina razvijene su brojne metode i algoritmi rješavanja. S obzirom na kvalitetu dobivenih rješenja i vrijeme računanja najučinkovitijima su se pokazali metaheuristički algoritmi inspirirani procesima koji se odvijaju u prirodi. Među njih spadaju i evolucijski algoritmi koji simulacijom prirodnog odabira i borbe za opstanak pretražuju prostor rješenja. Praktična primjena algoritama rješavanja VRP problema postala je aktualna naglim razvojem informacijskih tehnologija koji se dogodio u proteklih dvadesetak godina. Povećanje procesne moći računala, dostupnost vektorskih zemljovida za cestovnu navigaciju, javno dostupan satelitski sustav za globalno pozicioniranje stvorili su preduvjete za uspješnu primjenu znanja vezanih uz ovaj problem u svakodnenom planiranju distribucije u gospodarstvu.

1.1 Predmet istraživanja

Izrada optimalnog plana dostave ili prikupljanja tereta skupinom vozila ograničenih kapaciteta predstavlja optimizacijski problem koji se u literaturi naziva problem usmjeravanja vozila. Pripada kategoriji NP-teških problema iz područja kombinatorne optimizacije. Rješavanje problema podrazumijeva pronalazak skupa ruta koje predstavljaju redoslijede obilazaka lokacija iskrcaja (ukrcaja) pri čemu svaku lokaciju obilazi samo jedno vozilo i gdje sva vozila kreću iz skladišta i vraćaju se u skladište nakon obavljenog posla. U praksi se ovaj problem javlja prilikom planiranja dostave robe široke potrošnje, hrane, pića, tiskovina, poštanskih pošiljaka, prikupljanja komunalnog otpada i sl. Uobičajena proširenja osnovnog modela problema uključuju više raspoloživih skladišta, vremenska ograničenja te zahtjeve za prikupljanjem tereta. U praksi se javljaju i dodatna ograničenja koja otežavaju rješavanje problema poput zadanih prioriteta kod posluživanja lokacija, različitih radnih vremena vozača, različitih karakteristika i kapaciteta vozila, različitih tipova tereta, različitih lokacija za povratak i dr. Zbog velikog broja mogućih ograničenja koja otežavaju modeliranje problema, u posljednje vrijeme se ovakvi problemi nazivaju obogaćenim problemima usmjeravanja vozila (eng. *Rich Vehicle Routing Problem*).

Metaheurističke metode rješavanja ne garantiraju pronalazak optimalnih rješenja, ali su u stanju pronaći zadovoljavajuće kvalitetna rješenja za kratko vrijeme. Evolucijski algoritmi metaheurističke su metode koje simuliraju principe biološke evolucije tretirajući rješenja problema kao jedinke koje se bore za opstanak i koje skupa tvore populaciju. Bolje jedinke imaju veću vjerojatnost preživljavanja i prenošenja genetskog materijala na potomstvo. Ocjena kvalitete svake pojedine jedinke ovisi o funkciji cilja koja se minimizira ili maksimizira. Kod staničnog evolucijskog algoritma svaka jedinka zauzima određenu poziciju u toroidalnoj rešetci i može biti u interakciji samo s neposrednim susjedima. Time se postiže efekt izoliranih otoka i pojačava diverzifikacija pretrage prostora rješenja (istraživanje). Hibridizacijom algoritma operatorima lokalnog pretraživanja pojačava se intenzifikacija pretrage odnosno eksploatacija uskog područja u blizini trenutnog rješenja.

Za uspješnu primjenu hibridnog staničnog evolucijskog algoritma u procesu planiranja distribucije u nekoj tvrtki nužno je prikupiti i pripremiti sve potrebne podatke, te modelirati potpuni usmjereni graf kod kojeg težine bridova predstavljaju duljine najkraćih ili najbržih puteva između lokacija iskrcaja (ukrcaja) i skladišta. Ukoliko su zadana i vremenska ograničenja svaka cestovna udaljenost mora imati i odgovarajuću aproksimaciju trajanja putovanja.

Iz navedene problematike istraživanja definira se predmet istraživanja: hibridni stanični evolucijski algoritam za rješavanje različitih varijanti problema usmjeravanja vozila, analiza njegove učinkovitosti i validacija na standardnim ispitnim zadacima i prikupljenim problemima hrvatskih tvrtki.

1.2 Pregled dosadašnjih istraživanja

Problem usmjeravanja vozila (VRP) definiran je 1959. godine iz potrebe za racionalizacijom opskrbe benzinskih postaja gorivom, a prvotno je nazvan problemom otpremanja kamiona [34]. Pripada kategoriji NP-teških problema i generalizacija je problema trgovačkog putnika (eng. *Traveling Salesman Problem*, kratica TSP)[76],[23]. Tijekom proteklih nekoliko desetljeća definiran je niz proširenja osnovnog modela problema [107]. Jedno od najvažnijih proširenja je problem usmjeravanja vozila s vremenskim prozorima (eng. *Vehicle Routing Problem with Time Windows*, kratica VRPTW) kod kojeg je osim ograničenja kapaciteta vozila nužno zadovoljiti i vremenska ograničenja [102],[53]. Trajanje posluživanja unaprijed je poznato, a svaka lokacija se mora poslužiti unutar zadanog vremenskog intervala ili prozora. Skladište ima ograničeno radno vrijeme koje predstavlja vremenski prozor unutar kojeg se sav transport mora obaviti, a vozila vratiti natrag. Drugo važno proširenje osnovnog problema je višeskladišni problem usmjeravanja vozila (eng. *Multi-Depot Vehicle Routing Problem*, kratica MDVRP) kod kojeg je lokacije moguće poslužiti vozilima koja pripadaju različitim skladištima [46],[47]. Kod problema usmjeravanja vozila s povratnim prikupljanjem (eng. *Vehicle Routing Problem with Backhauls*, kratica VRPB) nakon obavljene dostave potrebno je prikupiti teret (npr. praznu ambalažu) [97]. Kod problema prikupljanja i dostave (eng. *Pickup and Delivery Problem*, kratica PDP) teret se preuzima na jednoj lokaciji, a dostavlja na drugu [96]. Kod problema otvorenog usmjeravanja vozila (eng. *Open Vehicle Routing Problem*, kratica OVRP) vozila se ne vraćaju u skladište nakon obavljene dostave [77].

S porastom broja ograničenja i varijabli (lokacije, vozila), složenost problema se naglo povećava što ograničava primjenu egzaktnih metoda rješavanja. Heurističke metode koje su zasnovane na iskustvu vezanom uz specifičan problem, u stanju su za kratko vrijeme pronaći rješenja zadovoljavajuće kvalitete i na problemima velikih dimenzija, no pretraživanje prostora rješenja često završava u lošem lokalnom optimumu [19]. Metaheurističke metode nastoje prevladati ovaj problem usmjeravajući pretragu prema područjima koja sadrže kvalitetna rješenja povremeno prihvaćajući i lošija, a često i nekompletna ili nevažeća rješenja [20]. Prirodom inspirirane metaheurističke metode poput evolucijskih algoritama, optimizacije mravljom kolonijom i simuliranog kaljenja pokazale su se prema dosadašnjim istraživanjima vrlo učinkovitima u rješavanju VRP problema [53],[93],[55],[112].

Evolucijski algoritmi među koje spadaju genetski algoritmi, memetički algoritmi, evolucijske strategije i genetičko programiranje pretragu prostora rješenja obavljaju stohastički, koristeći populaciju rješenja na kojoj simuliraju principe evolucije. Učinkovitost evolucijskog algoritma ovisi o postupku kodiranja rješenja problema u kromosome, funkciji za evaluaciju dobrote jedinki, implementaciji i učestalosti primjene operatora rekombinacije (križanja) i mutacije, postupku selekcije jedinki i dr.

Različiti autori koriste različite postupke kodiranja jedinki. Kodiranje permutacijom indeksa lokacija (vektor koji predstavlja redoslijed obilazaka) omogućuje primjenu operatora križanja tipičnih za TSP probleme, no ovakav pristup iziskuje posebne postupke dekodiranja s obzirom da u permutaciji nije zapisana informacija o pripadnosti lokacija vozilima [106],[94]. U radu [71], autori u redoslijed obilazaka dodaju indekse skladišta kao oznake završetka pojedinih ruta. Sličan efekt uporabom proširenog skupa indeksa postižu autori u radu [1].

Funkcija za evaluaciju dobrote jedinki ovisi o modelu VRP problema i načinu kodiranja. Primjerice, u slučaju VRPTW problema primarni cilj optimizacije je minimizirati broj vozila, a sekundarni minimizirati ukupnu udaljenost. Iz tog razloga u radu [13] paralelno ko-evoluiraju dvije populacije gdje svaka koristi zasebnu funkciju dobrote. U radu [15], autori u funkciju dobrote uključuju i broj neraspoređenih lokacija koje su posljedica specifičnog načina kodiranja i dekodiranja kromosoma. Leksikografska evaluacija dobrote koristi se u radu [67], dok se u radu [53] sličan efekt postiže pomoću odgovarajućih koeficijenata za određivanje prioriteta svake komponente koja se evaluira.

Implementacija operatora rekombinacije (križanja) i mutacije također ovisi o načinu kodiranja kromosoma. U radu [95], napravljena je usporedba osam operatora križanja na problemu usmjeravanja vozila. U radu [13], autori operatore križanja i mutacije primjenjuju izravno na rješenjima izbjegavajući njihovo eksplicitno kodiranje u kromosome.

Mutacije najčešće predstavljaju jednostavne operacije prebacivanja lokacije s jedne pozicije na drugu unutar kromosoma, međusobne zamjene dvaju lokacija i inverzije slučajno odabranog segmenta kromosoma [1],[95].

Za odabir jedinki koje će se reproducirati ili mutirati najčešće se koriste standardne metode turnirske i rulet selekcije. S ciljem sprječavanja iščezavanja najboljeg pronađenog rješenja iz populacije, većina autora koristi pravilo elitizma prema kojem se najbolja jedinka bezuvjetno prenosi u svaku slijedeću generaciju.

S ciljem poboljšanja i ubrzanja pretrage prostora rješenja često se u evolucijske algoritme ugrađuju metode lokalnog pretraživanja, mehanizmi dekompozicije problema kojima se ograničava veličina susjedstva kao i niz drugih elemenata koji takvim algoritmima daju atribut hibridni. Od recentnih istraživanja ističe se rad [108], u kojem je opisan hibridni genetski algoritam koji se na velikom broju testnih VRPTW zadataka pokazao učinkovitijim od drugih najboljih poznatih algoritama u trenutku objavljivanja. U radu [65], autori koriste drugačiji pristup u kojem genetičkim programiranjem evoluiraju konstruktivne heuristike da bi potom njima generirali rješenja problema.

Osnovni problem koji se može uočiti kod evolucijskih algoritama je prerana konvergencija koja se događa uslijed zasićenja populacije jednakim ili vrlo sličnim jedinkama. Stanični genetski algoritam (eng. *Cellular Genetic Algorithm*, kratica cGA) za rješavanje problema usmjeravanja vozila koji će se obrađivati u ovom istraživanju, predložen je u radu [1]. Temelji se na načelu izolacije putem udaljenosti prema kojem se križati mogu samo neposredni susjedi. Pokazao se vrlo

učinkovitim u rješavanju klasičnog VRP problema [3],[4]. Evidentan je nedostatak istraživanja o njegovoj primjeni na drugim varijantama VRP problema kao i na problemima iz gospodarstva i industrije što je jedan od ciljeva ovog istraživanja navedenih u odlomku 1.3.

Razvoj računalnih i informacijsko-komunikacijskih tehnologija tijekom posljednja dva desetljeća omogućio je primjenu algoritama i metoda rješavanja problema usmjeravanja vozila za planiranje distribucije u gospodarstvu i industriji [21],[54],[110]. Primjena računalnih optimizacijskih metoda u logističkim tvrtkama često rezultira značajnim uštedama u transportnim troškovima koje se kreću od 5% do 20% [107].

1.3 Cilj i hipoteze istraživanja

Cilj predloženog istraživanja je razviti učinkovit hibridni stanični evolucijski algoritam za rješavanje problema usmjeravanja vozila iz stvarnog svijeta. Iz navedenog proizlaze slijedeće hipoteze:

- Hibridni stanični evolucijski algoritam u stanju je za kratko vrijeme pronaći kvalitetna rješenja postavljenog problema zbog učinkovitog balansiranja procesa eksploatacije i istraživanja prostora rješenja.

- Uspostava metodologije prikupljanja podataka iz stvarnog svijeta i njihove pripreme za definiranje problema usmjeravanja vozila omogućiti će primjenu razvijenog algoritma u logističkim tvrtkama.

- Primjenom predložene metodologije i algoritma za rješavanje problema usmjeravanja vozila značajno bi se smanjili ukupni troškovi distribucije u logističkim tvrtkama.

1.4 Obrazloženje strukture doktorske disertacije

U prvom dijelu, UVODU, definiran je problem i predmet istraživanja, određena svrha i ciljevi istraživanja, dan osvrt na dosadašnja istraživanja, te je obrazložena struktura doktorske disertacije.

U drugom dijelu rada pod naslovom PROBLEM USMJERAVANJA VOZILA, opisani su najvažniji modeli problema, provedena njihova klasifikacija i objašnjena matematička složenost.

U trećem dijelu rada pod naslovom POSTUPCI RJEŠAVANJA, opisani su konstruktivni heuristički algoritmi rješavanja, principi lokalnog i stohastičkog pretraživanja prostora rješenja.

U četvrtom dijelu rada pod naslovom HIBRIDNI STANIČNI EVOLUCIJSKI ALGORITAM, razrađeni su elementi predloženog hibridnog staničnog evolucij-

skog algoritma za rješavanje problema usmjeravanja vozila te je provedena analiza njegove učinkovitosti na testnim zadacima.

U petom dijelu rada pod naslovom RJEŠAVANJE PROBLEMA IZ STVARNOG SVIJETA, opisani su postupci pripreme podataka i prilagodbe algoritma koje je potrebno provesti za uspješno rješavanje problema iz stvarnog svijeta. Provedena je validacija predloženog algoritma na problemima definiranim na osnovu podataka prikupljenih od hrvatskih tvrtki.

U posljednjem dijelu rada, ZAKLJUČKU, sustavno su formulirani i prikazani rezultati znanstvenog istraživanja kojima su dokazivane postavljene hipoteze.

Poglavlje 2

Problem usmjeravanja vozila

Problemi rutiranja i raspoređivanja (eng. *routing and scheduling*) predstavljaju kompleksne kombinatoričke probleme koji se svakodnevno javljaju u gospodarstvu i industriji prilikom planiranja transporta ljudi i dobara. Problem usmjeravanja vozila (VRP) jedan je od najvažnijih problema u distribucijskoj logistici i osnova je velikom broju problema koji se međusobno razlikuju po vrsti i broju ograničenja. Osnovne komponente svakog VRP problema su skladišta, vozila i korisnici (lokacije koje treba poslužiti). Skupinom vozila koja su locirana u skladištu potrebno je obići sve korisnike uz uvjet da svakog korisnika posjeti samo jedno vozilo i da se sva vozila vrate u skladište. Pri tome je cilj odrediti rute uz koje će ukupni troškovi biti minimalni. Na troškove u prvom redu utječe broj angažiranih vozila pa je stoga minimizacija broja vozila primarni cilj, dok je minimizacija ukupnog pređenog puta ili utrošenog vremena najčešće sekundarni cilj optimizacije.

Standardna ograničenja po kojima se varijante VRP problema razlikuju jesu broj skladišta (jedno ili više njih), maksimalno dozvoljeno trajanje ili duljina rute vozila, različiti kapaciteti vozila, zahtjevi korisnika za dostavom ili prikupljanjem određene količine tereta, te vremenski prozori unutar kojih je potrebno započeti ukrcaj ili iskrcaj. U stvarnom svijetu potrebno je uvažiti i veliki broj dodatnih ograničenja koja su često posljedica specifičnosti lokacija ukrcaja i/ili iskrcaja, specifičnih poslovnih procesa tvrtke koja obavlja distribuciju (prikupljanje) ili zakonskih odredbi (npr. obavezno uzimanje pauze za vozače). Takvi problemi često se nazivaju obogaćenim problemima usmjeravanja vozila (RVRP).

Problem se može modelirati uz pomoć potpunog grafa u kojem vrhovi predstavljaju korisnike, a lukovi troškove putovanja između njih. Zbog jednosmjernih ulica i pravila prometovanja (npr. zabrana skretanja), kod stvarnih problema graf je usmjeren budući da trošak putovanja između dva korisnika ne mora biti jednak u oba smjera. U nastavku su opisani neki od važnijih modela problema.

2.1 Problem trgovačkog putnika

Problem trgovačkog putnika (TSP) poznati je i dobro istraženi kombinatorički problem koji se može promatrati kao poseban slučaj problema usmjeravanja vozila u kojem postoji samo jedno vozilo neograničenog kapaciteta. Pripada kategoriji NP-teških problema [76]. Prema definiciji, trgovački putnik mora pronaći najkraći put kojim će obići skup gradova uz uvjet da kroz svaki prođe samo jednom i da se na kraju putovanja vrati u grad iz kojeg je putovanje započeo. Ako je zadana matrica troškova $D = d(i, j)$, gdje je d_{ij} trošak putovanja između grada i i grada j, $(i, j = 1, 2, ..., n)$, potrebno je pronaći permutaciju $P = (i_1, i_2, i_3, ...i_n)$ cijelih brojeva od 1 do n koja će minimizirati funkciju cilja:

$$Y = d_{i_1 i_2} + d_{i_2 i_3} + ... + d_{i_n i_1} \ . \tag{2.1}$$

Uz pretpostavku da je matrica D simetrična odnosno da vrijedi $d_{ij} = d_{ji}$, broj mogućih rješenja s obzirom na funkciju cilja iznosi $\frac{1}{2}(n-1)!$. Problem može biti zapisan i riješen kao linearni program [78]. Za definiranu matricu D, potrebno je odrediti varijable odluke x_{ij} koje minimiziraju iznos:

$$Q = \sum_{i,j} d_{ij} x_{ij} \ , \tag{2.2}$$

ako vrijedi:

$$x_{ii} = 0 \ , \tag{2.3}$$

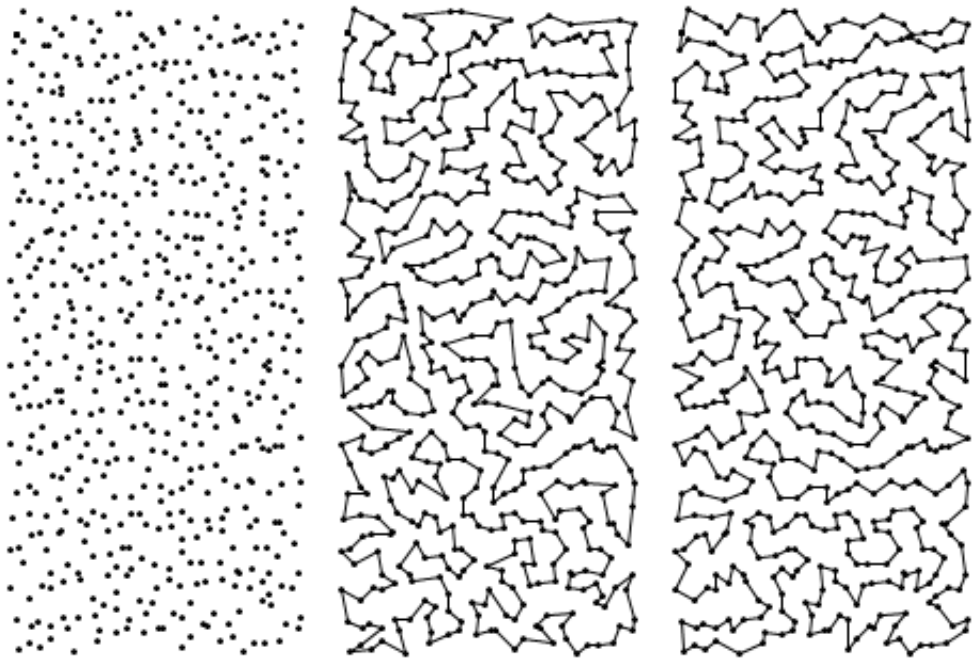
$$x_{ij} = 0, 1 \ , \tag{2.4}$$

$$\sum_i x_{ij} = \sum_j x_{ij} = 1 \ , \tag{2.5}$$

za bilo koji podskup $S = i_1, i_2, ..., i_r$ cijelih brojeva od 1 do n,

$$x_{i_1 i_2} + x_{i_2 i_3} + ... + x_{i_{r-1} i_r} + x_{i_r i_1} \begin{cases} < r \text{ , ako je } r < n \\ \leq n \text{ , ako je } r = n \end{cases} \ . \tag{2.6}$$

Zbog velikog broja ograničenja čak i za mali n, ovakav pristup nije prikladan za rješavanje većih problema. Unatoč nesagledivom broju mogućih rješenja danas su poznati algoritmi koji su u stanju pronaći optimalna rješenja i kod problema od više tisuća gradova [66]. Na slici 2.1 prikazana su dva moguća rješenja problema trgovačkog putnika veličine 575 gradova koja se po duljini ukupnog pređenog puta razlikuju za cca. 5%.



(a) Problem (b) Rješenje 1 (c) Rješenje 2

Slika 2.1. *Dva rješenja problema trgovačkog putnika veličine 575 gradova*

2.2 Problem usmjeravanja vozila s ograničenjima kapaciteta

Problem usmjeravanja vozila kolokvijalno se koristi kao općeniti naziv za ovu klasu problema, međutim, nazivom problem usmjeravanja vozila s ograničenjima kapaciteta (eng. *Capacitated Vehicle Routing Problem*, kratica CVRP) precizira se prisutnost ograničenja kapaciteta vozila u problemu [34],[107],[23]. Osnovni VRP problem ne mora uključivati ograničenja kapaciteta i zahtjeve za dostavom, primjerice kada problem ne predstavlja dostavu ili prikupljanje već poslove servisiranja, ali rute mogu imati ograničenu maksimalnu duljinu. Problem usmjeravanja vozila je NP-težak budući da poopćava problem trgovačkog putnika kada je zadano samo jedno vozilo neograničenog kapaciteta.

Prema [31], simetrični VRP definiran je na potpunom neusmjerenom grafu $G = (V, E)$. Skup $V = 0, ..., n$ je skup vrhova. Svaki vrh $i \in V \setminus \{0\}$ predstavlja korisnika s nenegativnom potražnjom q_i, dok vrh 0 predstavlja skladište. Svakom bridu $e \in E = \{(i, j) : i, j \in V, i < j\}$ pridružen je trošak putovanja c_e ili c_{ij}. Skupina od m vozila kapaciteta Q smještena je u skladištu. Kod simetričnog VRP problema potrebno je odrediti m ruta čiji ukupni trošak je minimalan uz uvjete: (1) svakog korisnika posjećuje samo jedno vozilo jednom rutom; (2) svaka ruta započinje i završava u skladištu; (3) ukupna potražnja svih korisnika u ruti ne premašuje kapacitet vozila Q; (4) duljina svake rute nije veća od postavljenog

ograničenja L. Rješenje se može promatrati kao skup od m obilazaka koji dijele zajednički vrh u skladištu. Asimetrični VRP definiran je na usmjerenom grafu $G = (V, E)$, gdje je skup bridova $E = \{(i, j) : i, j \in V, i \neq j\}$. U ovom slučaju rutu vozila predstavlja usmjereni obilazak vrhova.

Problem se može formulirati kao cjelobrojni linearni program u kojem za svaki brid $e \in E$ cjelobrojna varijabla x_e sadrži informaciju o broju prolazaka bridom. Neka je $r(S)$ minimalan broj vozila potreban da se posluže korisnici iz podskupa S. Vrijednost $r(S)$ može se odrediti rješavanjem odgovarajućeg problema pakiranja posuda (eng. *Bin Packing Problem*, kratica BPP) sa skupom predmeta S i posuda kapaciteta Q ili aproksimacijom pomoću izraza:

$$r(S) = \lceil \frac{\sum_{i \in S} q_i}{Q} \rceil \ . \tag{2.7}$$

Za $S \subset V$, neka je $\delta(S) = \{(i, j) : i \in S, j \notin S \ ili \ i \notin S, j \in S\}$. Ako je $S = \{i\}$ umjesto $\delta(\{i\})$ pisat će se $\delta(i)$. Prema CVRP formulaciji predloženoj u [74] potrebno je minimizirati:

$$\min \sum_{e \in E} c_e x_e \ , \tag{2.8}$$

uvažavajući ograničenja:

$$\sum_{e \in \delta(i)} x_e = 2, \quad i \in V \setminus \{0\} \ , \tag{2.9}$$

$$\sum_{e \in \delta(0)} x_e = 2m \ , \tag{2.10}$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S), \quad S \subseteq V \setminus \{0\}, S \neq \emptyset \ , \tag{2.11}$$

$$x_e \in \{0, 1\}, \quad e \notin \delta(0) \ , \tag{2.12}$$

$$x_e \in \{0, 1, 2\}, \quad e \in \delta(0) \ . \tag{2.13}$$

Ograničenje (2.9) osigurava da svaki korisnik bude posjećen jednom, a ograničenje (2.10) kreiranje m ruta. Ograničenje kapaciteta (2.11) nalaže povezanost

rješenja i očuvanje kapaciteta vozila forsirajući ulazak dovoljnog broja bridova u svaki podskup vrhova. Ograničenja (2.12) i (2.13) osiguravaju da je svaki brid između dva korisnika korišten maksimalno jednom, odnosno, da je svaki brid povezan sa skladištem korišten najviše dva puta. Brid može biti korišten dva puta u slučaju da vozilo poslužuje samo jednog korisnika.

Široko rasprostranjena alternativna formulacija temelji se na modelu problema particioniranja skupa (eng. *Set Partitioning Problem*, kratica SPP) koja sadrži potencijalno eksponencijalan broj binarnih varijabli [10]. Neka je $\mathcal{R} = \{R_1, ..., R_s\}$ skup svih mogućih ruta koja zadovoljavaju ograničenja, gdje je $s = |\mathcal{R}|$. Svaka ruta R_j ima odgovarajući trošak γ_j. Binarni koeficijent a_{ij} ima vrijednost 1 ako i samo ako je vrh i posjećen u ruti R_j. Binarna varijabla x_j, $j = 1, .., s$, ima vrijednost 1 ako i samo ako je ruta R_j odabrana u optimalnom rješenju. Prema ovoj formulaciji potrebno je minimizirati:

$$\min \sum_{j=1}^{s} \gamma_j x_j \ , \tag{2.14}$$

uvažavajući ograničenja:

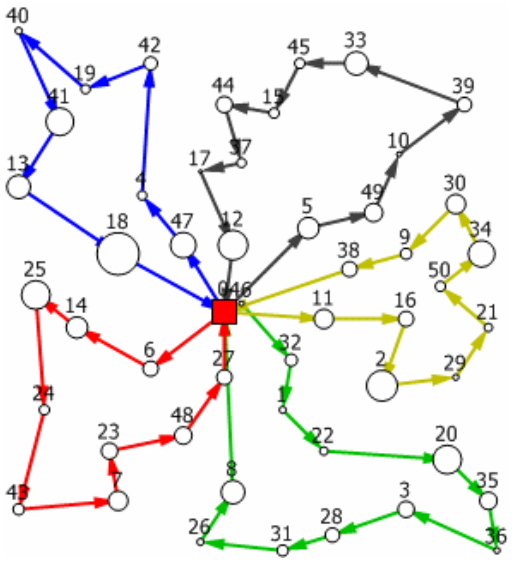
$$\sum_{j=1}^{s} a_{ij} x_j = 1, \quad i \in V \setminus \{0\} \ , \tag{2.15}$$

$$\sum_{j=1}^{s} x_j = m \ , \tag{2.16}$$

$$x_j \in \{0, 1\}, \quad j = 1, ..., s \ . \tag{2.17}$$

Ograničenje 2.15 nalaže da svaki korisnik i bude u točno jednoj ruti, a ograničenje 2.16 da bude odabrano m ruta. Budući da je zadovoljavanje ograničenja u rutama implicitno određeno definicijom skupa $\mathcal{R}$, ovaj model predstavlja opći model koji se može jednostavno proširiti dodatnim ograničenjima. Niz drugih formulacija CVRP problema može se pronaći u [107].

Na slici 2.2 prikazano je optimalno rješenje testnog CVRP problema *E-n51-k5* kojeg sačinjava 51 korisnik i 5 vozila [25]. Pravokutnik u sredini predstavlja skladište, a kružnice korisnike čiji su zahtjevi za dostavom proporcionalni opsegu.



Slika 2.2. *Optimalno rješenje CVRP problema E-n51-k5*

2.3 Problem usmjeravanja vozila s vremenskim prozorima

Kod problema usmjeravanja vozila s vremenskim prozorima (VRPTW) posluživanje svakog korisnika potrebno je započeti unutar zadanog vremenskog intervala ili prozora [102],[53]. Trajanje posluživanja unaprijed je poznato za svakog korisnika, a skladište također ima vrijeme otvaranja i zatvaranja između kojih vozila moraju obaviti posao i vratiti se natrag. VRPTW problem je NP-težak problem budući da generalizira CVRP ako je vrijeme otvaranja svih vremenskih prozora postavljeno na 0, a zatvaranja na $+\infty$. Pored toga, pronalazak izvedivog rješenja u kojem su sva ograničenja zadovoljena, ako je broj vozila fiksno zadan, predstavlja NP-potpun problem [30]. Za razliku od CVRP problema kod kojeg je cilj minimizirati ukupnu pređenu udaljenost (minimalan broj vozila može se unaprijed odrediti), kod VRPTW problema primarni cilj je minimizirati broj vozila, a sekundarni minimizirati ukupnu udaljenost ili vrijeme.

Prema [18], problem je definiran na grafu (N, A). Skup čvorova N sastoji se od skupa korisnika C i čvorova 0 i $n+1$ koji predstavljaju skladište. Broj korisnika $|C|$ označen je s n, a sami korisnici vrijednostima $1, 2, ..., n$. Skup lukova A predstavlja sve moguće veze između čvorova. Nijedan luk ne završava u čvoru 0 niti započinje u čvoru $n + 1$. Svakom luku $(i, j) \in A$ pridružen je trošak c_{ij} i vrijeme putovanja t_{ij} koje uključuje vrijeme posluživanja korisnika i. Skup vozila jednakih kapaciteta označen je s V. Svako vozilo ima kapacitet q, a svaki korisnik nenegativnu potražnju d_i, $i \in C$. Posluživanje svakog korisnika mora započeti unutar zadanog vremenskog prozora $[a_i, b_i]$, $i \in C$. Vozila moraju napustiti skladište unutar intervala $[a_0, b_0]$ i vratiti se natrag unutar $[a_{n+1}, b_{n+1}]$. Vozilu je dozvo-

ljeno doći na lokaciju korisnika prije otvaranja vremenskog prozora, no u tom slučaju mora čekati na otvaranje kako bi posluživanje moglo započeti. Dolazak kod korisnika nakon zatvaranja vremenskog prozora nije dozvoljen. Pretpostavka je da je svaka ruta počinje u trenutku 0, odnosno da vrijedi $a_0 = b_0 = 0$.

Model sadrži dva tipa varijabli odlučivanja. Varijabla odlučivanja x_{ij}^k definirana $\forall (i,j) \in A$, $\forall k \in V$ ima vrijednost 1 ako vozilo k vozi od čvora i do čvora j, dok u protivnom ima vrijednost 0. Varijabla odlučivanja s_i^k definirana $\forall i \in N$, $\forall k \in V$ predstavlja vrijeme kada je vozilo k, $k \in V$ započelo posluživanje korisnika i, $i \in C$. Ako vozilo k ne poslužuje korisnika i onda s_i^k nema značenje. Pretpostavka je da je $s_0^k = 0$, $\forall k$ i da s_{n+1}^k predstavlja vrijeme povratka vozila k u skladište. Cilj je odrediti skup ruta uz koje će trošak biti minimalan. Jedno vozilo može imati samo jednu rutu, a svakog korisnika poslužuje samo jedno vozilo. Rute moraju zadovoljiti ograničenja kapaciteta vozila i vremenske prozore posluženih korisnika. VRPTW problem može se matematički formulirati kao:

$$\min \sum_{k \in V} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \ , \tag{2.18}$$

uz ograničenja:

$$\sum_{k \in V} \sum_{j \in N} x_{ij}^k = 1, \quad \forall i \in C \ , \tag{2.19}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ij}^k \leq q, \quad \forall k \in V \ , \tag{2.20}$$

$$\sum_{j \in N} x_{0j}^k = 1, \quad \forall k \in V \ , \tag{2.21}$$

$$\sum_{i \in N} x_{i,n+1}^k = 1, \quad \forall k \in V \ , \tag{2.22}$$

$$\sum_{i \in N} x_{ih}^k - \sum_{j \in N} x_{hj}^k = 0, \quad \forall h \in C, \forall k \in V \ , \tag{2.23}$$

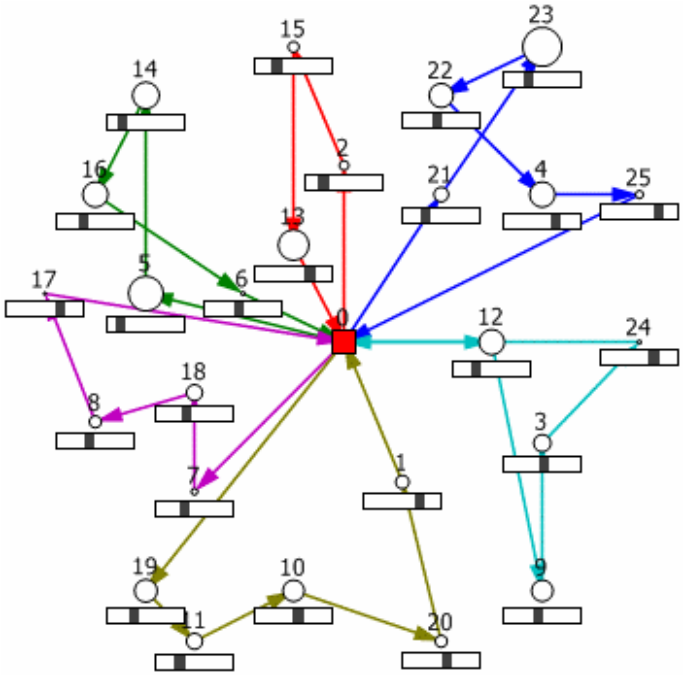
$$x_{ij}^k (s_i^k + t_{ij} - s_j^k) \leq 0, \quad \forall (i,j) \in A, \forall k \in V \ , \tag{2.24}$$

$$a_i \leq s_i^k \leq b_i, \quad \forall i \in N, \forall k \in V \ , \tag{2.25}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in A, k \in V \ . \tag{2.26}$$

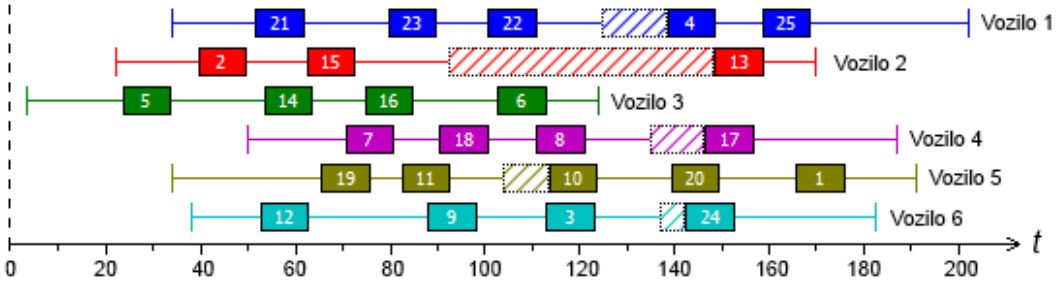
Prema funkciji (2.18) cilj optimizacije je minimizirati ukupni trošak svih vozila. Ograničenje (2.19) osigurava da svakom korisniku bude pridruženo jedno vozilo, a ograničenje (2.20) da kapacitet svakog vozila ne bude premašen. Prema ograničenju (2.21), svako vozilo k napušta čvor 0 jednom, a prema ograničenju (2.22), vraća se u čvor $n+1$. Ograničenje (2.23) osigurava da vozilo napušta čvor h, $h \in C$ ako i samo ako ono uđe u taj čvor. Ograničenje (2.24) onemogućuje slučaj dolaska vozila k do korisnika j prije vremena $s_i^k + t_{ij}$ ako putuje od i do j. Ograničenje (2.25) osigurava zadovoljenje svih vremenskih prozora, a ograničenje (2.26) cjelobrojnost. Budući da je kod ovog problema primarni cilj optimizacije minimizirati broj vozila, a sekundarni minimizirati udaljenost, potrebno je na odgovarajući način vrednovati prazne rute odnosno neiskorištena vozila. Ako se luku $(0, n+1)$ pridruži dovoljno visok negativan trošak, ukupni trošak će biti manji što je više neiskorištenih vozila.

Na slici 2.3 prikazano je optimalno rješenje VRPTW problema $R105.25$ od 25 korisnika u kojem se koristi minimalan broj vozila i prevaljuje minimalna udaljenost [102]. Kao i kod CVRP problema, skladište je prikazano pravokutnikom u sredini, dok kružnice predstavljaju korisnike čiji polumjeri označavaju njihovu relativnu potražnju. Pravokutnici koji se nalaze ispod kružnica predstavljaju vremenski horizont čija širina odgovara razlici između vremena otvaranja i zatvaranja skladišta. Osjenčani dio unutar pravokutnika predstavlja relativan položaj i širinu vremenskog prozora tijekom kojeg se korisnik mora početi posluživati.



Slika 2.3. *Optimalno rješenje VRPTW problema R105.25*

Na slici 2.4, isto rješenje problema s prethodne slike prikazano je u vremenskoj domeni. Ravna crta označava trajanje vožnje, a pravokutnici ispunjeni bojom trajanje posluživanja kod označenih korisnika (trajanje posluživanja jednako je za sve korisnike). Pravokutnici ispunjeni kosim crtama predstavljaju vrijeme čekanja u slučajevima kada je vozilo do korisnika stiglo prije otvaranja vremenskog prozora a_i. Vrijeme polaska pomaknuto je svim vozilima za odgovarajući vremenski interval kako bi se izbjeglo čekanje kod prvog korisnika u ruti, odnosno, kako bi vozilo stiglo kod korisnika u vrijeme otvaranja njegovog vremenskog prozora.



Slika 2.4. *Vremenski prikaz optimalnog rješenja VRPTW problema R105.25*

2.4 Višeskladišni problem usmjeravanja vozila

Kod višeskladišnog problema usmjeravanja vozila (MDVRP) korisnike je moguće poslužiti iz više od jednog skladišta gdje svako skladište raspolaže odgovarajućim brojem vozila jednakih kapaciteta. Prema definiciji izloženoj u [29], neka je skup skladišta označen s $\mathcal{D}$, a skup korisnika s $\mathcal{C}$. Svakom skladištu $i \in \mathcal{D}$ pridružen je skup od m_i vozila, a svakom korisniku $j \in \mathcal{C}$ potražnja d_j. Pretpostavka je da su sva vozila jednakog kapaciteta Q i da im je duljina rute ograničena na maksimalnih T jedinica. Problem se definira na grafu $G = (V, E)$ gdje je $V = \mathcal{D} \cup \mathcal{C}$ i $E = \{\{i, j\} : i, j \in V, i \text{ i } j \text{ nisu oboje u } \mathcal{D}\}$. Svakom bridu $e \in E$ pridružen je trošak putovanja c_e (vrijeme ili duljina). Cilj je odabrati podskup vozila i konstruirati rute koje će zadovoljiti ograničenja kapaciteta vozila i maksimalne duljine, uz uvjet da se svakog korisnika posjeti samo jednom, te da ukupni troškovi budu minimalni. Višeskladišni problem usmjeravanja vozila je NP-težak budući da poopćava VRP problem ako je $|\mathcal{D}| = 1$.

2.4.1 Formulacija modelom toka vozila

Za svako skladište $i \in \mathcal{D}$ i korisnika $j \in \mathcal{C}$ definira se binarna varijabla y_{ij} čija je vrijednost jednaka 1 ako i samo ako je korisnik j poslužen jednom rutom koja počinje u skladištu i. Za svaki brid $e \in E$ definira se binarna varijabla x_e čija je

vrijednost jednaka 1 ako i samo ako bridom e prolazi vozilo koje u ruti posjećuje barem dva korisnika. Za skup korisnika $S \subseteq \mathcal{C}$, neka je donja granica za broj vozila s obzirom na kapacitet:

$$r(S) = \lceil \frac{\sum_{j \in S} d_j}{Q} \rceil \ . \tag{2.27}$$

S obzirom na ograničenje maksimalne duljine rute neka je donja granica za broj vozila označena s $\rho(S)$. Izračun $r(S)$ se može obaviti u konstantnom vremenu, dok izračun $\rho(S)$ zahtijeva rješavanje m-TSP problema s ograničenjem maksimalne duljine rute što predstavlja NP-težak problem. Za podskup vrhova grafa $U \subset V$ neka je $\delta(U)$ rez na podskupu U koji predstavlja podskup bridova koji imaju jedan krajnji vrh u U. Za podskup bridova $F \subseteq E$ definiran je $x(F) = \sum_{e \in F} x_e$, i ako je $F \subseteq \delta(\mathcal{D})$, $y(F) = \sum_{e \in F} y_e$. Formulacija toka vozila za MDVRP problem je slijedeća:

$$\min \sum_{e \in E} c_e x_e + 2 \sum_{i \in \mathcal{D}, j \in \mathcal{C}} c_{ij} y_{ij} \ , \tag{2.28}$$

uz ograničenja:

$$x(\delta \{j\}) + 2y(\delta(\{j\}) \cap \delta(\mathcal{D})) = 2 \quad j \in \mathcal{C} \ , \tag{2.29}$$

$$x(\delta \{i\}) + 2y(\delta(\{i\})) \leq 2m_i \quad i \in \mathcal{D} \ , \tag{2.30}$$

$$x(\delta \{S\}) + 2y(\delta(S) \cap \delta(\mathcal{D})) \geq 2 \max \{r(S), \rho(S)\} \quad S \subseteq \mathcal{C} \ , \tag{2.31}$$

$$x(\delta(S)) \geq 2[x(\delta(\{h\}) \cap \delta(\mathcal{D}')) + x(\delta(\{j\}) \cap \delta(\mathcal{D} \backslash \mathcal{D}'))] \quad S \subseteq \mathcal{C}, h, j \in S, \mathcal{D}' \subset \mathcal{D} \ , \tag{2.32}$$

$$y_e \in \{0, 1\} \quad e \in \delta(\mathcal{D}) \ , \tag{2.33}$$

$$x_e \in \{0, 1\} \quad e \in E \ . \tag{2.34}$$

Funkcijom cilja nastoji se minimizirati ukupne troškove. Ograničenje (2.29) osigurava da svaki korisnik bude posjećen samo jednom, a ograničenje (2.30) da kod svakog skladišta bude iskorišteno najviše m_i vozila. Ograničenje maksimalne duljine rute i kapaciteta (2.31) nalaže da najmanje $\max\{r(S), \rho(S)\}$ vozila poslužuje skup korisnika S. Ograničenje (2.32) osigurava da rute imaju isto početno i završno skladište, dok ograničenja (2.33) i (2.34) osiguravaju da su varijable x_e i y_e binarne.

2.4.2 Formulacija modelom problema particioniranja skupa

Za svaki $i \in \mathcal{D}$ i $j \in \mathcal{C}$ definirana je binarna varijabla y_{ij} čija je vrijednost jednaka 1 ako i samo ako je korisnik j jedini u ruti koja započinje u skladištu i i čiji je trošak $2c_{ij}$. Neka je Ω skup svih ruta koje posjećuju dva ili više korisnika i koje zadovoljavaju ograničenja kapaciteta vozila i maksimalne duljine rute. Za svako skladište $i \in \mathcal{D}$ pripadajući podskup ruta koje počinju i završavaju u njemu označen je s Ω_i. Za svaku rutu $l \in \Omega$ definirana je binarna varijabla θ_l čija je vrijednost jednaka 1 ako i samo ako je ruta l odabrana u optimalnom rješenju. Trošak rute c_l predstavlja sumu troškova putovanja po svim lukovima korištenim u ruti. Za svakog korisnika $j \in \mathcal{C}$ i rutu $l \in \Omega$ definirana je varijabla a_j^l čija je vrijednost jednaka broju posjeta korisniku j u ruti l. Za svaki podskup skladišta $D \subseteq \mathcal{D}$ i podskup korisnika $C \subseteq \mathcal{C}$ neka je $y(D : C) = \sum_{i \in D} \sum_{j \in C} y_{ij}$. Formulacija MDVRP problema modelom particioniranja skupa je slijedeća:

$$\min \sum_{l \in \Omega} c_l \theta_l + 2 \sum_{i \in \mathcal{D}, j \in \mathcal{C}} c_{ij} y_{ij} \ , \tag{2.35}$$

uz ograničenja:

$$\sum_{l \in \Omega} a_j^l \theta_l + y(\mathcal{D} : \{j\}) = 1 \quad j \in \mathcal{D} \ , \tag{2.36}$$

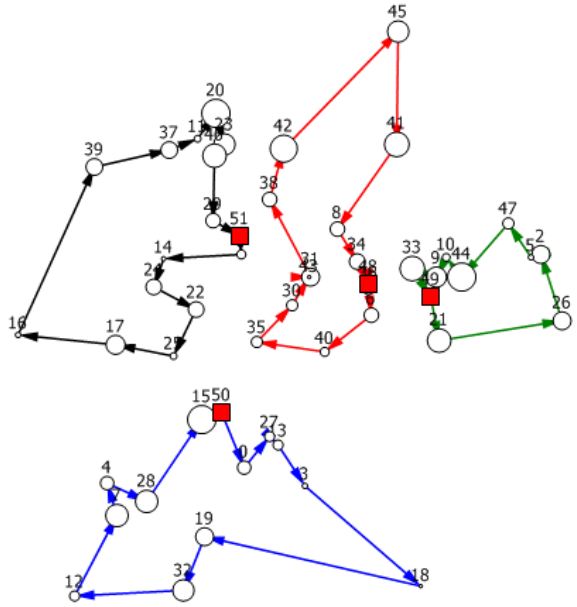
$$\sum_{l \in \Omega_i} \theta_l + y(\{j\} : \mathcal{C}) \leq m_i \quad i \in \mathcal{D} \ , \tag{2.37}$$

$$y_{ij} \in \{0, 1\} \quad i \in \delta(\mathcal{D}) \ , \tag{2.38}$$

$$\theta_l \in \{0, 1\} \quad l \in \Omega \ . \tag{2.39}$$

Ograničenje (2.36) osigurava da svaki korisnik bude posjećen samo jednom, a ograničenje (2.37) da broj ruta koje počinju i završavaju u skladištu i ne može biti veći od raspoloživog broja vozila m_i. Ograničenja (2.38) i (2.39) osiguravaju da

su varijable y_{ij} i θ_l binarne. Ograničenja kapaciteta vozila i maksimalne duljine rute su implicitno uključena u definiciju skupa ruta Ω. Na slici 2.5 prikazano je optimalno rješenje testnog MDVRP problema PR01 [32].



Slika 2.5. *Optimalno rješenje MDVRP problema PR01*

2.5 Ostala proširenja osnovnog problema

U prethodnim odlomcima opisane su tri varijante VRP problema koje su vrlo dobro istražene i koje su često testni poligon za razvoj i testiranje metaheurističkih postupaka rješavanja. Postoji niz drugih proširenja osnovnog modela s kojima se specifični procesi dostave i prikupljanja mogu preciznije i potpunije opisati. Među ključne čimbenike po kojima se VRP modeli razlikuju spadaju:

1. Broj skladišta
   - jedno skladište
   - više skadišta
2. Vremenska ograničenja
   - s vremenskim prozorima
   - bez vremenskih prozora
3. Karakteristike vozila

- vozila su jednakih karakteristika (homogena flota)
- vozila su različitih karakteristika (heterogena flota)

4. Horizont planiranja
   - jednokratno
   - za dulji period (npr. tjedan ili mjesec)
5. Vrsta posluživanja
   - samo dostava ili samo prikupljanje
   - dostava i prikupljanje
     - prikupljanje na jednoj, dostava na drugoj lokaciji
     - dostava i prikupljanje na istoj lokaciji
     - prikupljanje na povratku u skladište
6. Završetak posluživanja
   - povratak u skladište
   - bez povratka u skladište

U nastavku su ukratko opisana neka od proširenja osnovnog problema usmjeravanja vozila koja se često pojavljuju u gospodarstvu i industriji. Podrazumijeva se da svako navedeno proširenje može imati više-skladišnu varijantu (prefiks *MD*), te uključivati vremenska ograničenja (sufiks *TW*). Također se različita ograničenja pojedinog proširenja mogu kombinirati s ograničenjima iz drugog proširenja (primjerice, višeskladišni problem prikupljanja i dostave s mješovitom flotom i vremenskim prozorima).

2.5.1 Problem usmjeravanja mješovite flote vozila

Klasični VRP problem podrazumijeva homogenu flotu vozila, odnosno jednak kapacitet kod svih vozila. Kod problema usmjeravanja mješovite flote vozila (eng. *Heterogeneous or Mixed Fleet Vehicle Routing Problem*, kratica HFVRP) vozila mogu imati različit kapacitet ili trošak korištenja [68]. Pod ovaj zajednički naziv ubrajaju se problem veličine i kombiniranja flote (eng. *Fleet Size and Mix*, kratica FSM), te problem usmjeravanja heterogene flote vozila (eng. *Heterogeneous Vehicle Routing Problem*, kratica HVRP). FSM problem podrazumijeva neograničen broj vozila i koristi se za strateško planiranje flote. Kod HVRP problema pak, potrebno je između konačnog broja raspoloživih vozila iz flote odabrati ona uz koja će ukupan trošak biti minimalan. U gospodarstvu, posebice u poslovima distribucije robe široke potrošnje flota dostavnih vozila se sastoji najčešće od vozila različitih karakteristika.

2.5.2 Problem periodičkog usmjeravanja vozila

Za razliku od klasičnog VRP problema kod kojeg se rute planiraju za jedan cjeloviti proces posluživanja odnosno jedan dan, kod problema periodičkog usmjeravanja vozila (eng. *Periodic Vehicle Routing Problem*, kratica PVRP) rute se planiraju unaprijed za dulji vremenski period, primjerice jedan tjedan ili mjesec [52]. Pored količine koju treba iskrcati ili ukrcati prilikom svakog posluživanja, za svakog korisnika poznata je i učestalost posluživanja koje treba obaviti u planiranom razdoblju (npr. dva puta unutar razdoblja od tjedan dana). Sve kombinacije dana u kojima je posluživanje moguće obaviti potrebno je generirati iz učestalosti i broja dana u planiranom razdoblju. Moguće je i eksplicitno zadati ili zabraniti pojedine kombinacije (npr. posluživanje dozvoljeno samo ponedjeljkom i četvrtkom ili srijedom i petkom). Rješavanje PVRP problema složenije je od rješavanja klasičnog VRP problema budući da je svakom pojedinom danu iz planiranog razdoblja potrebno pridružiti korisnike koji će se taj dan poslužiti, te riješiti VRP problem za svaki dan. Ovaj model može se u praksi primijeniti za organiziranje prikupljanja otpada, raspoređivanje servisnih timova koji obavljaju održavanja na terenu, raspoređivanje poštanskih službenika, organiziranje obilazaka klijenata trgovačkih putnika i dr.

2.5.3 Problem prikupljanja i dostave

Kod problema prikupljanja i dostave (eng. *Pickup and Delivery Problem*, kratica PDP) roba/teret se prikuplja na jednoj lokaciji i dostavlja na drugu [99]. Ako se umjesto robe radi o prijevozu putnika onda se problem može nazvati i problemom prijevoza na poziv (eng. *Dial-A-Ride*, kratica DARP). Za razliku od osnovnog VRP modela u kojem pojedinačni zahtjev za posluživanjem predstavlja jedna lokacija kod PDP problema svaki zahtjev za posluživanjem predstavljaju dvije lokacije. Na prvoj se obavlja prikupljanje, a na drugoj dostava. Rješavanje pored zadovoljenja ograničenja kapaciteta vozila podrazumijeva i zadovoljenje ograničenja prvenstva obilaska lokacija zahtjeva (prikupljanje se mora obaviti prije dostave) i povezanosti lokacija (istim vozilom mora se obaviti i prikupljanje i dostava). Između prikupljanja i dostave koji skupa čine jedan zahtjev može se obaviti i niz drugih prikupljanja i dostava ukoliko kapacitet vozila to dozvoljava. Primjerice, ako je zadano skladište S i zahtjevi $(p_1,\ d_1), (p_2, d_2), (p_3, d_3)$, valjana ruta može biti $S \to p_2 \to p_1 \to d_1 \to p_3 \to d_2 \to d_3 \to S$.

2.5.4 Problem usmjeravanja vozila s istovremenom dostavom i prikupljanjem

Kod problema usmjeravanja vozila s istovremenom dostavom i prikupljenjem (eng. *Vehicle Routing Problem with Simultaneous Delivery and Pickup*, kratica VRPSDP) neposredno nakon obavljanja dostave potrebno je ukrcati određenu

količinu robe / tereta na istoj lokaciji [104]. U praksi se ovaj problem javlja kada je nakon obavljanja dostave, od korisnika potrebno prikupiti i praznu ambalažu (npr. kod dostave pića, pekarskih proizvoda i dr.). Pored ograničenja kapaciteta potrebno je zadovoljiti i ograničenje koje nalaže da kod pojedinog korisnika dostavu i prikupljanje mora obaviti isto vozilo.

2.5.5 Problem usmjeravanja vozila s povratnim prikupljanjem

Kod problema usmjeravanja vozila s povratnim prikupljanjem (eng. *Vehicle Routing Problem with Backhauls*, kratica VRPB), roba/teret se prikuplja u povratku, nakon što se obavi sva dostava i isprazni teretni prostor vozila [97]. Vozila se ne moraju nužno vraćati istim putem i prikupljati robu/teret s istih lokacija na kojima su ranije obavila dostavu. Pored ograničenja prvenstva prema kojem se prvo moraju obaviti sve dostave kako bi moglo započeti prikupljanje, kapacitet vozila mora biti dostatan za sav teret koji se dostavlja i za teret koji se nakon dostave prikuplja. Pretpostavka je da nije dozvoljeno prikupljanje duž čitave rute bez prethodne dostave. U praksi se ovaj problem često javlja kod tvrtki čija dostavna vozila nakon isporuke robe kupcima moraju u skladište dopremiti robu od dobavljača.

2.5.6 Problem otvorenog usmjeravanja vozila

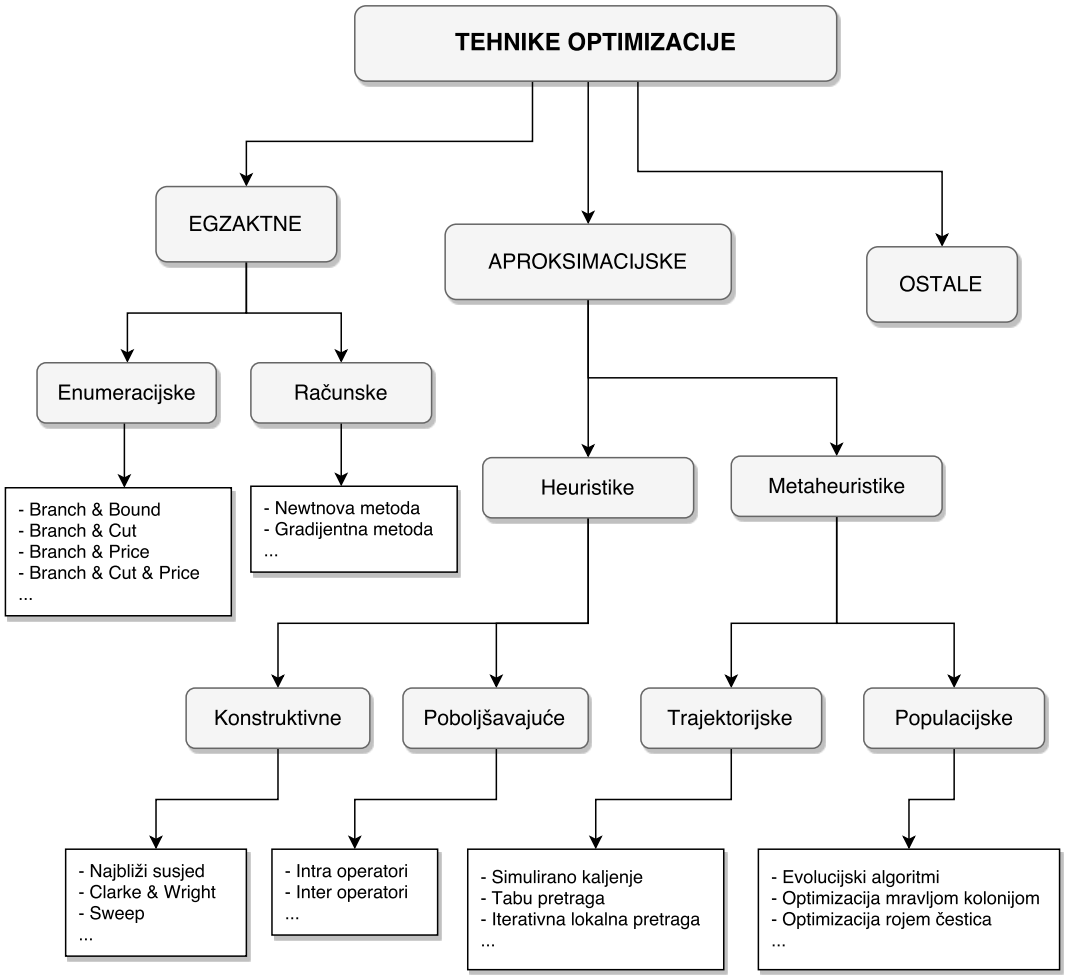
U slučaju problema otvorenog usmjeravanja vozila (eng. *Open Vehicle Routing Problem*, kratica OVRP) vozila se nakon posluživanja posljednjeg korisnika u ruti ne vraćaju u skladište [113]. Ovaj model pogodan je u slučajevima kada tvrtka distributer ne koristi vlastita vozila i vozače već koristi usluge vanjskih prijevoznika što uklanja potrebu povratka u skladište. Klasični VRP problem može se transformirati u OVRP problem ako se matrica udaljenosti modificira tako što će se za udaljenost između svakog korisnika i skladišta uzeti vrijednost 0. Udaljenost u obrnutom smjeru, od skladišta do svih korisnika, mora ostati nepromijenjena.

Poglavlje 3

Postupci rješavanja

Od kraja pedesetih godina prošlog stoljeća pa sve do danas, proveden je velik broj istraživanja problema usmjeravanja vozila te su predložene različite optimizacijske tehnike za njegovo rješavanje. Na slici 3.1 prikazana je gruba podjela postupaka koji se koriste za rješavanje problema iz područja kombinatorne optimizacije. U ovom istraživanju u fokus su stavljene aproksimacijske tehnike rješavanja, odnosno heuristički i metaheuristički algoritmi. Egzaktni algoritmi, iako sposobni pronaći optimalna rješenja, u praksi se mogu primijeniti samo za rješavanje problema manjih dimenzija zbog enormnog rasta računske složenosti s porastom broja varijabli. Heurističke algoritme odlikuje mogućnost brzog pronalaska rješenja problema, no zbog nedovoljnog istraživanja i eksploatacije prostora rješenja, pronađena rješenja su najčešće daleko od optimalnih s obzirom na ciljnu funkciju koja se minimizira ili maksimizira. Mogu se podijeliti na konstruktivne algoritme kojima se rješenje gradi ispočetka i postupke poboljšavanja postojećeg rješenja. Potonji se koriste kao operatori lokalnog pretraživanja, iterativnog postupka kojim se eksploatira susjedstvo postojećeg rješenja i koji rezultira pronalaskom lokalnog optimuma. Pojedine konstruktivne heuristike nakon što u prvoj fazi grupiraju korisnike (dodijele ih vozilima) podrazumijevaju naknadno korištenje postupaka za poboljšavanje rješenja s obzirom da rješenje konstruirano u prvoj fazi najčešće ne bude kvalitetno s obzirom na ciljnu funkciju. Metaheuristički algoritmi predstavljaju strategije pretrage kojima se tijekom pretraživanja pokušava obuhvatiti što širi prostor rješenja, a pomnije istražiti samo ona područja koja obećavaju pronalazak kvalitetnih rješenja. Često koriste konstruktivne heurističke algoritme za generiranje početnog/početnih rješenja, ali i heuristike iterativnog poboljšavanja rješenja u hibridnim varijantama. Mogu se podijeliti na trajektorijske algoritme kod kojih se pretraga prostora rješenja vodi samo preko jednog, aktualnog rješenja, te na populacijske, kod kojih pretraga podrazumijeva istovremeno pretraživanje prostora oko većeg broja rješenja koja čine populaciju. Heuristički i metaheuristički algoritmi ne jamče pronalazak optimalnih rješenja,

no u stanju su relativno brzo pronaći visoko kvalitetna rješenja i na problemima od više stotina pa i tisuća korisnika. U odlomcima koji slijede, napravljen je pregled poznatih heurističkih i metaheurističkih postupaka i tehnika rješavanja. Više informacija o egzaktnim postupcima rješavanja može se pronaći u radovima [107],[9].



Slika 3.1. *Tehnike rješavanja optimizacijskih problema*

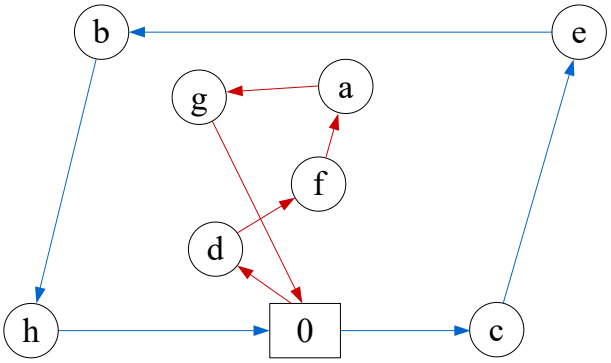
3.1 Konstruiranje početnog rješenja

Konstruktivne heuristike postepeno grade izvedivo rješenje problema, vodeći računa o ukupnom trošku i ne služeći se pri tome nikakvim postupcima poboljšavanja [75]. Karakterizira ih "pohlepan" način rada s obzirom da se u svakom koraku algoritma rješenje nadograđuje uz minimalan trošak. Takav pristup dugoročno, do kraja postupka konstruiranja, najčešće vodi ka lošem rješenju. Rute

se mogu konstruirati serijski i paralelno. Serijsko konstruiranje podrazumijeva konstruiranje ruta jedne za drugom, od početka do kraja. Kod paralelnog konstruiranja, korisnici se mogu dodavati ili umetati u bilo koju rutu koja zadovoljava ograničenja.

3.1.1 Heuristika najbližeg susjeda

Klasični primjer "pohlepnog" algoritma je heuristika najbližeg susjeda (eng. *Nearest Neighbour Heuristic*, kratica NNH) kod kojeg se rute konstruiraju serijski, jedna za drugom. U svakoj iteraciji algoritma u rutu se dodaje neposluženi korisnik koji je najbliži trenutnoj lokaciji vozila uz uvjet da je kapacitet vozila dostatan. U trenutku kada vozilu ponestane raspoloživog kapaciteta, ruta se zatvara, angažira se novo vozilo, a postupak ponavlja dok god ima korisnika koji nisu posluženi. Na slici 3.2 prikazano je rješenje CVRP problema konstruirano heuristikom najbližeg susjeda. Problem se sastoji od 8 lokacija na koje treba dostaviti po jednu jedinicu tereta te dva vozila koja raspolažu kapacitetom od 4 jedince.

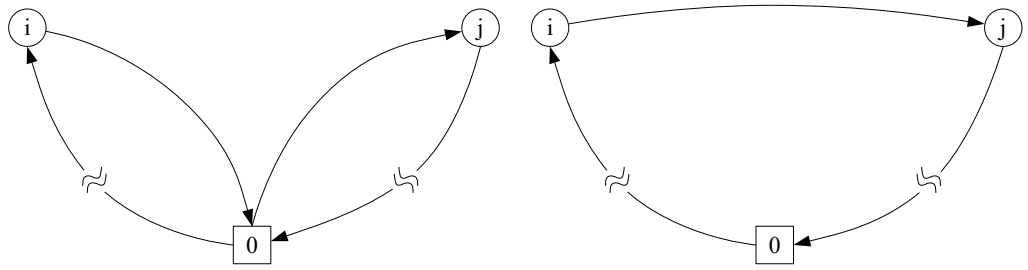


Slika 3.2. *Heuristika najbližeg susjeda*

Slika jasno ilustrira "pohlepnu" narav algoritma koji je gledajući samo jedan korak unaprijed konstruirao rješenje koje čak ni za ovako mali problem ne predstavlja optimum. Prva ruta označena crvenom bojom $(0, d, f, a, g, 0)$ bi u konačnici bila kraća da je u drugoj iteraciji umjesto bližeg korisnika f odabran udaljeniji korisnik g. Pored toga, forsiranje odabira najbližeg utječe i na raspodjelu korisnika među vozilima. Očigledno je da bi se minimalna udaljenost postigla drugačijom raspodjelom korisnika odnosno rutama $(0, d, g, b, h, 0)$ i $(0, c, e, a, f, 0)$. Unatoč spomenutim nedostacima, prednosti ovog algoritma su jednostavnost implementacije i brzina, te mogućnost jednostavne izvedbe stohastičke varijante koja može generirati više različitih izvedivih rješenja problema. Rješavanje VRPTW problema zahtjeva drugačiji kriterij odabira slijedećeg korisnika s obzirom na vremenska ograničenja. U radu [102], predložena je vremenski-ovisna heuristika najbližeg susjeda, a u radu [53] njezina stohastička varijanta.

3.1.2 Clarke & Wright algoritam ušteda

Jedan od najstarijih i najpoznatijih algoritama za rješavanje CVRP problema je *Clarke & Wright* algoritam ušteda [27]. U prvom koraku, kreira se ruta za svakog korisnika u problemu dodavanjem lukova između skladišta i korisnika u oba smjera. U nastavku se iterativnim postupkom po kriteriju maksimalne uštede dvije rute spajaju u jednu dok god to kapacitet vozila dozvoljava. Spajanje ruta obavlja se brisanjem završnog luka prve rute i početnog luka druge rute, te kreiranjem novog luka između zadnjeg korisnika prve rute i prvog korisnika druge rute na način kako je to prikazano na slici 3.3.



(a) Prije spajanja (b) Nakon spajanja

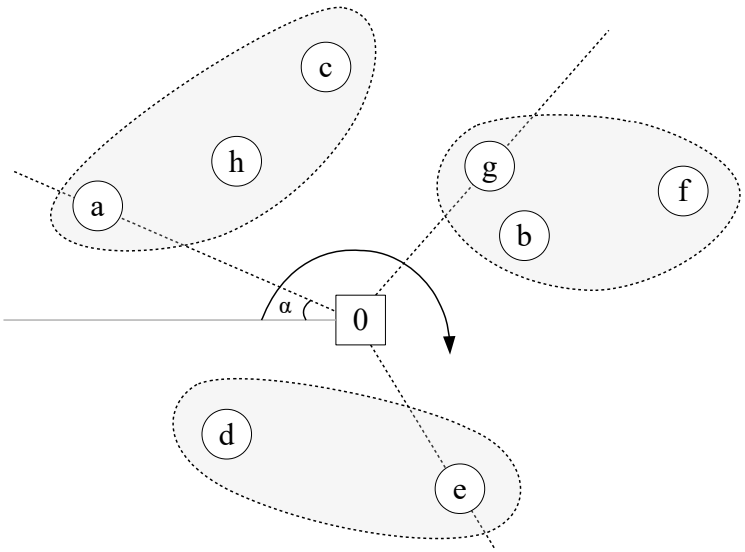
Slika 3.3. *Spajanje ruta u Clarke & Wright algoritmu*

Za svaki par ruta $(0, ..., i, 0)$ i $(0, j, ..., 0)$ gdje je $i, j = 1, ..., n$ i $i \neq j$ može se izračunati potencijalna ušteda $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ uz uvjet da je kapacitet vozila veći ili jednak ukupnoj potražnji svih korisnika iz obje rute. Postoji paralelna i slijedna inačica algoritma. U paralelnoj inačici, odabiru se dvije rute čijim će se spajanjem ostvariti najveća ušteda s_{ij}. U slijednoj inačici, odabrana ruta proširuje se spajanjem s drugima po kriteriju maksimalne uštede dok god ima raspoloživog kapaciteta. U trenutku kada se kapacitet vozila iscrpi, odabire se slijedeća ruta i spajanje se nastavlja. Postupak završava kada nestane mogućnosti za ostvarenje novih ušteda. Pregled poboljšanih verzija *Clarke & Wright* algoritma za rješavanje CVRP problema može se pronaći u [75]. U radu [102] opisana je proširena varijanta algoritma za rješavanje VRPTW problema u kojoj se prilikom spajanja ruta pored ograničenja kapaciteta vozila u obzir uzimaju i vremenska ograničenja.

3.1.3 Sweep algoritam

Među klasične konstruktivne algoritme za rješavanje CVRP problema spada i *Sweep* algoritam u kojem se neposluženi korisnici dodaju u rute prema kriteriju najmanjeg polarnog kuta u odnosu na zamišljeni pravac koji prolazi kroz skladište i rotira se u smjeru kazaljke na satu [61], slika 3.4. Rute se konstruiraju

serijski, dodajući korisnike s najmanjim polarnim kutom u trenutnu rutu dok god vozilo ima raspoloživog kapaciteta. Kada raspoloživi kapacitet ne bude dovoljan za prihvat ni jednog od preostalih neposluženih korisnika, ruta se zatvara povratkom u skladište te se angažira novo vozilo. Postupak se ponavlja dok god ima neposluženih korisnika.



Slika 3.4. *Grupiranje korisnika kod Sweep algoritma*

S obzirom da se prilikom dodavanja korisnika ne minimizira udaljenost već polarni kut α, konstruirano rješenje najčešće je loše s obzirom na ukupnu udaljenost koja kod CVRP problema predstavlja ciljnu funkciju. Zbog toga se naknadno rute najčešće poboljšavaju nekim od iterativnih postupaka lokalnog pretraživanja poput *2-Opt* heuristike, ili se rekonsturiraju rješavanjem pojedinačnih TSP problema sastavljenih od skladišta i korisnika ruta. Uz primjenu potonjeg koraka, ova metoda je primjer dvofaznog, *prvo-grupiraj-zatim-rutiraj*, pristupa rješavanju VRP problema. Vremenski orijentirana *Sweep* heuristika prilagođena rješavanju VRPTW problema predložena je također u radu [102].

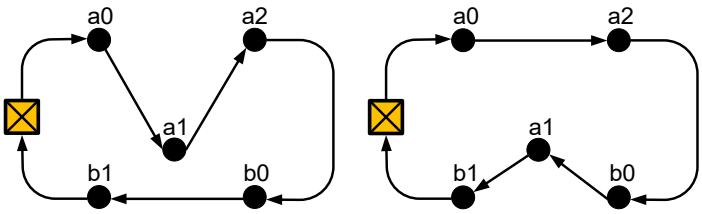
3.2 Lokalno pretraživanje

Lokalno pretraživanje heuristički je postupak kojim se pretražuje susjedstvo postojećeg rješenja s ciljem pronalaska novog, boljeg rješenja. Pretraga se provodi operatorima koji evaluiraju promjene u ciljnoj funkciji koje će nastati ako se na postojećem rješenju naprave određene modifikacije. Razmatraju se samo modifikacije nakon kojih će sva ograničenja i dalje ostati zadovoljena. Ciljna funkcija koja se minimizira unutar lokalne pretrage obično je udaljenost. U susjedstvu

može postojati više rješenja koja su bolja od postojećeg pa je potrebno koristiti ogovarajuću strategiju prihvaćanja poboljšanja. Najčešće korištene strategije su prihvaćanje prvog pronađenog i prihvaćanje ukupno najboljeg poboljšanja [19],[53],[50]. Strategijom prihvaćanja prvog poboljšanja u pravilu se dobivaju lošiji rezultati, ali je izvršavanje brže budući da nije potrebno do kraja pretražiti cijelo susjedstvo. Ipak, kod određenih problema izvršavanje može biti i sporije ako se tijekom izvršavanja prihvaća jako velik broj manjih poboljšanja. U trenutku kada operatori lokalne pretrage ne pronađu rješenje bolje od postojećeg, znači da se pretraga zaustavila u lokalnom optimumu. Veličina susjedstva koje se pretražuje ovisi o broju i vrsti operatora koji se primjenjuju. U nastavku je dan pregled operatora koji modificiraju jednu rutu (*Intra* operatori) i operatora koji istovremeno modificiraju dvije rute (*Inter* operatori) [24].

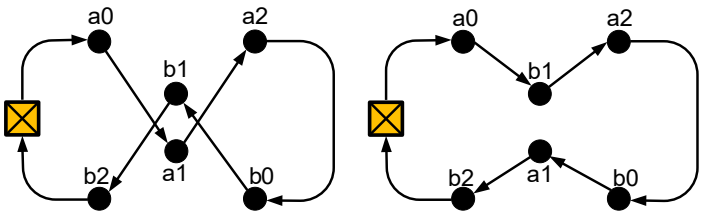
3.2.1 Intra operatori

Ova skupina operatora traga za poboljšanjima koja se repozicioniranjem korisnika mogu napraviti na jednoj ruti. Premještanje korisnika uzrokuje promjenu vremena dolazaka i odlazaka kod svih ili kod dijela korisnika pa je kod rješavanja VRPTW problema prije prihvaćanja promjene potrebno provjeriti da li će vremenska ograničenja ostati zadovoljena. Ograničenja kapaciteta nije potrebno provjeravati budući da se izmjene provode na istoj ruti. Različiti načini na koji *Inter* operatori modificiraju rutu prikazani su na slikama 3.5, 3.6, 3.7 i 3.8.



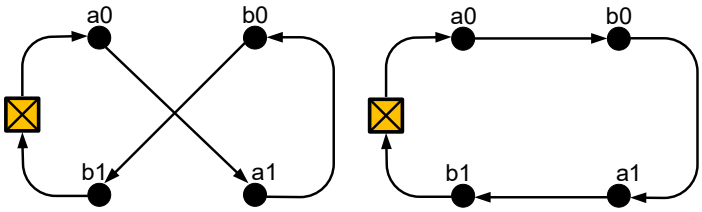
(a) Prije izmjene (b) Nakon izmjene

Slika 3.5. *Intra-relocate operator*



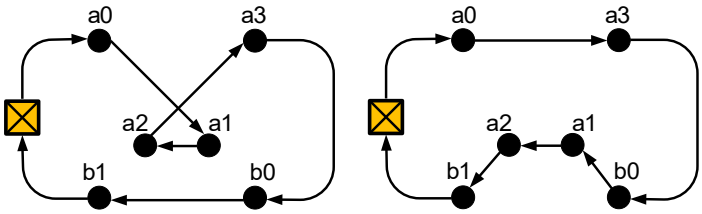
(a) Prije izmjene (b) Nakon izmjene

Slika 3.6. *Intra-exchange operator*



(a) Prije izmjene (b) Nakon izmjene

Slika 3.7. *2-Opt operator*



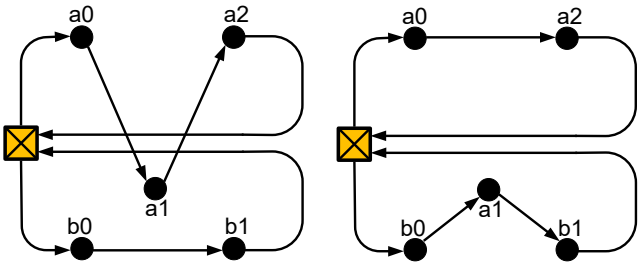
(a) Prije izmjene (b) Nakon izmjene

Slika 3.8. *Or-Opt operator*

Kod *Intra-relocate* operatora, korisnik se s jedne pozicije u ruti premješta na drugu. Kod *Intra-exchange* operatora, dva korisnika u ruti međusobno zamjenjuju mjesta. Operatorom *2-Opt* rješenje se poboljšava inverzijom jednog segmenta rute, dok se *Or-Opt* operatorom ruta skraćuje razmještanjem tri povezana korisnika. *2-Opt* operator mijenja smjer odabranog segmenta pa je kod asimetričnih problema potrebno u obzir uzeti duljinu segmenta u obrnutom smjeru.

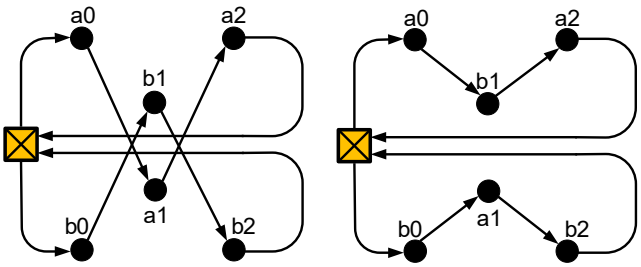
3.2.2 Inter operatori

Inter operatori rješenje poboljšavaju modificirajući istovremeno dvije rute premještanjem korisnika ili njihovom međusobnom razmjenom. Principi funkcioniranja različitih *Inter* operatora prikazani su na slikama 3.9, 3.10, 3.11, 3.12 i 3.13.



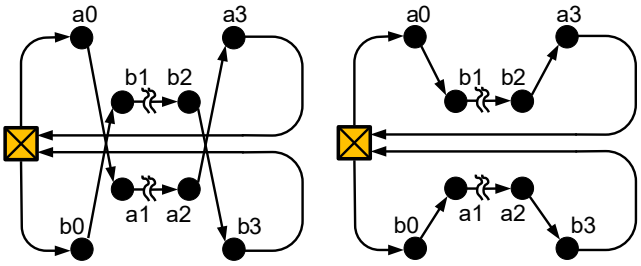
(a) Prije izmjene (b) Nakon izmjene

Slika 3.9. *Inter-relocate operator*



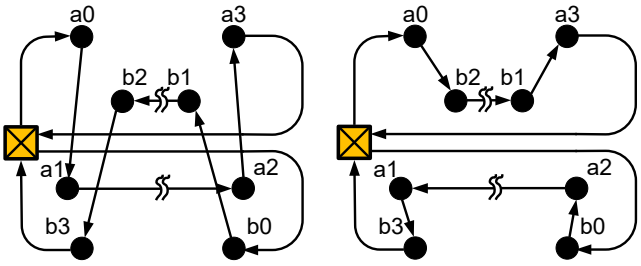
(a) Prije izmjene (b) Nakon izmjene

Slika 3.10. *Inter-exchange operator*



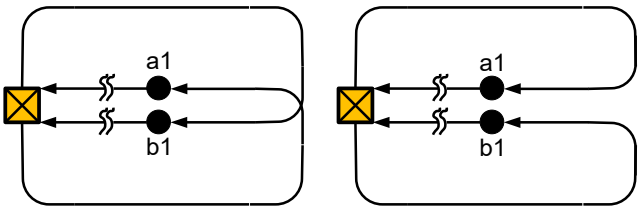
(a) Prije izmjene (b) Nakon izmjene

Slika 3.11. *Cross-exchange operator*



(a) Prije izmjene (b) Nakon izmjene

Slika 3.12. *ICross-exchange operator*



(a) Prije izmjene (b) Nakon izmjene

Slika 3.13. *2-Opt* operator*

Inter-relocate operator izbacuje korisnika iz jedne rute i umeće ga u drugu. *Inter-exchange* operator međusobno zamjenjuje dva korisnika iz različitih ruta. *Cross-exchange* i *ICross-exchange* razmjenjuju segmente ruta pri čemu segmenti mogu imati različit broj korisnika. *ICross-exchange* operator okreće redoslijed obilaska korisnika kod jednog ili kod oba segmenta. Ova dva operatora računski su kompleksnija od ostalih, no u stanju su jednim potezom napraviti velike promjene na rješenju. Operator *2-Opt** presijeca obje rute na pola i pokušava kombinirati njihove početne i završne segmente. S obzirom da se kod *Inter* operatora teret premješta iz jednog u drugo vozilo, potrebno je provjeriti da li će modifikacija ruta uzrokovati prekoračenje kapaciteta vozila. Zbog većeg broja kombinacija koje se ispituju, *Inter* operatori računski su puno složeniji od *Intra* operatora.

3.3 Metaheuristike

Metaheuristike su postupci rješavanja optimizacijskih problema koje odlikuje sposobnost brzog pronalaska visoko kvalitetnih rješenja. Rješenja koja se dobiju konstruktivnim heuristikama mogu se dalje poboljšati postupkom lokalne pretrage ali samo do razine lokalnog optimuma koji se postiže operatorima poboljšanja koji se u postupku primjenjuju. Metaheuristike pretražuju puno širi prostor rješenja što je omogućeno privremenim/povremenim prihvaćanjem i lošijih rješenja. Dvije najvažnije značajke metaheurističkih algoritama su mehanizmi diverzifikacije i intenzifikacije. Diverzifikacijom se nastoji istražiti što veći dio prostora rješenja, a intenzifikacijom eksploatirati pojedino područje tragajući za lokalnim optimumima. Potrebno je postići određenu ravnotežu između ta dva mehanizma kako bi cijeli postupak bio brz i učinkovit. Nije dobro previše se zadržavati na određenom području u prostoru rješenja i pretjerano ga eksploatirati ako ono ne obećava pronalazak kvalitetnih rješenja. S druge strane, nije dobro ni prebrzo napustiti neko područje bez da se dovoljno eksploatira jer se u blizini mogu nalaziti kvalitetna rješenja. Od 70-ih godina prošlog stoljeća razvijen je velik broj metaheurističkih algoritama koji se uspješno primjenjuju za rješavanje optimizacijskih problema u mnogim znanstvenim područjima i industriji. Neki od njih oponašaju procese koji se odvijaju u prirodi poput evolucijskih algoritama, simuliranog kaljenja ili optimizacije mravljom kolonijom. Evolucijski i mravlji algoritmi pretragu vode preko više rješenja koja tvore populaciju, dok se kod simuliranog kaljenja prostor rješenja pretražuje mijenjajući uvijek jedno rješenje. Drugi algoritmi poput tabu pretrage koriste memoriju u koju spremaju prethodno napravljene korake što se koristi za izbjegavanje njihovog kasnijeg ponavljanja. S ciljem povećanja učinkovitosti, metaheuristike se često hibridiziraju s drugim optimizacijskim tehnikama [17]. Kategorizirana bibliografija znanstvenih radova o metaheurističkim algoritmima korištenim za rješavanje problema usmjeravanja vozila može se pronaći u radovima [20],[57]. U nastavku je dan pregled poznatih metaheuristika koje se često koriste za rješavanje VRP problema.

3.3.1 Evolucijski algoritmi

Evolucijski algoritmi (eng. *Evolutionary Algorithm*, kratica EA) su metaheuristike inspirirane teorijom evolucije u kojima se prostor rješenja pretražuje oponašanjem načela prirodne selekcije na populaciji jedinki koje predstavljaju kodirana rješenja optimizacijskog problema koji se rješava. Tijekom zadanog broja generacija jedinke se mijenjaju operatorima križanja i mutacije. Operator selekcije veće izglede za preživljavanje ili prijenos genetskog materijala u iduću generaciju daje jedinkama s boljom vrijednošću dobrote. Postoji velik broj različitih varijanti evolucijskih algoritama, a među najpoznatije spadaju genetski algoritmi, evolucijske strategije, evolucijsko i genetsko programiranje te memetički algoritmi. Generički evolucijski (genetski) algoritam prikazan je pseudokôdom u algoritmu 1.

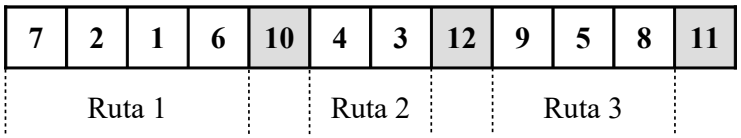
Algoritam 1 Evolucijski algoritam

```
1:  procedure EA(n, g, p_m)
2:      P ← GenerirajPocetnuPopulaciju(n)
3:      Evaluiraj(P)
4:      s_min ← Najbolji(P)
5:      for i = 1 → g do
6:          P* ← ∅
7:          for j = 1 → n do
8:              r ← SelektirajRoditelje(P)
9:              P*[j] ← Rekombiniraj(r[1], r[2])
10:             if rnd() < p_m then
11:                 Mutiraj(P*[j])
12:             end if
13:             Evaluiraj(P*[j])
14:         end for
15:         s ← Najbolji(P*)
16:         if f(s) < f(s_min) then
17:             s_min ← s
18:         end if
19:         Zamijeni(P, P*)
20:     end for
21:     return s_min
22: end procedure
```

Kod implementacije je potrebno odrediti način na koji se kodiraju rješenja, postupak njihove evaluacije, način odabira roditelja prilikom križanja i mutacije, te način zamjene starih jedinki novima. Sve nabrojeno bitno utječe na performanse i učinkovitost algoritma te ovisi o vrsti optimizacijskog problema koji se rješava i cilju(evima) optimizacije.

Kodiranje rješenja

Kada rješenje optimizacijskog problema predstavlja sekvencu ili redoslijed kao što je to slučaj kod TSP i VRP problema, klasičan način kodiranja binarnim nizom nije prikladan za korištenje budući da operatori varijacije mogu narušiti valjanost rješenja. Rješenja VRP problema najčešće se kodiraju permutacijom cijelih brojeva koja predstavlja redoslijed obilaska korisnika. U radu [1], autori uz indekse korisnika u permutaciju dodaju oznake za početak/kraj svake rute. Te oznake su indeksi vozila uvećani za broj korisnika čime je očuvana struktura permutacije i omogućena primjena klasičnih operatora varijacije koji se koriste kod rješavanja TSP problema. Slika 3.14 prikazuje kodirano rješenje problema koje se sastoji od 9 korisnika i 3 vozila.



Slika 3.14. *Rješenje VRP problema kodirano permutacijom*

U slučaju da se u rješenju određeno vozilo ne koristi, oznake za početak/kraj rute u permutaciji će se nalaziti jedna uz drugu. U radovima [7] i [53] autori rješenje kodiraju skupinom kromosoma koji predstavljaju pojedinačne rute vozila. U radu [15], u prvih k elemenata permutacije zapisani su prvi korisnici u rutama, gdje je k broj vozila u problemu. U postupku dekodiranja, preostalih $n - k$ korisnika ubacuje se u već započete rute heurističkim postupkom. U radu [94] rješenja CVRP problema kodiraju se kao permutacije ali bez oznaka početaka i krajeva ruta. Rute se određuju u postupku dekodiranja brzom egzaktnom procedurom *Split* koja optimalno grupira korisnike s obzirom na kapacitet i redoslijed zapisan permutacijom. Ovakav način kodiranja-dekodiranja svrstava se u pristup *prvo-rutiraj-onda-grupiraj* rješavanja, a navedeni algoritam je u vrijeme objave bio među najboljima. Kod nekih implementacija, izbjegava se eksplicitno kodiranje rješenja u kromosome, a operatori varijacije primjenjuju se izravno na rješenjima problema za čiju se pohranu koriste složenije memorijske strukture [13].

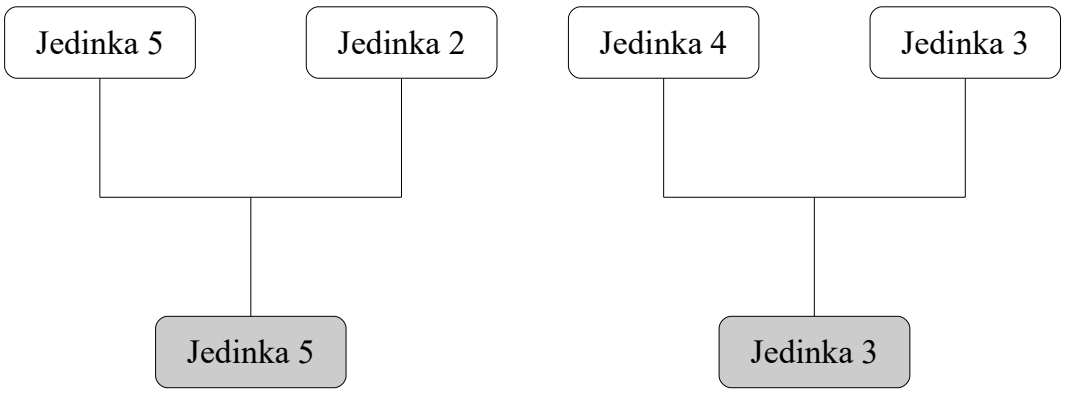
Evaluacija dobrote

Dobrota jedinke (eng. *fitness*) određuje njezine izglede da bude odabrana za reprodukciju ili kopirana u iduću generaciju. Kod CVRP problema za dobrotu se može uzeti ukupna pređena udaljenost s obzirom da se minimalan broj vozila može unaprijed odrediti. No, kod rješavanja VRPTW problema potrebno je prvo minimizirati broj vozila, a tek potom ukupnu udaljenost. Smanjenje udaljenosti

pretragu ne vodi nužno prema područjima u kojima se nalaze rješenja s minimalnim brojem vozila. Osim na postupak evaluacije, ova činjenica utječe i na ostale elemente evolucijskog algoritma. Primjerice, u radu [13] paralelno ko-evoluiraju dvije populacije. U jednoj se minimizira broj vozila, a u drugoj ukupna udaljenost. Kada se pronađe rješenje koje koristi manji broj vozila od dotada najboljeg, kopira se u drugu populaciju. U radu [15], populacija može sadržavati i nepotpuna rješenja, a funkcija dobrote izračunava se na osnovu broja nerutiranih korisnika i pređene udaljenosti. U radu [108], također ko-evoluiraju dvije populacije, ali jedna s izvedivim, druga s neizvedivim rješenjima kod kojih ograničenja nisu zadovoljena. Dobrota se može evaluirati i leksikografski kao u radovima [7], [67] uspoređujući rješenja prema više kriterija među kojima kod minimizacije broja vozila bitnu ulogu igra gruba procjena kašnjenja koje bi se pojavilo kad bi se ruta s najmanjim brojem korisnika eliminirala, a njezini korisnici umetnuli u ostale rute. Ukupan broj generacija može se podijeliti na dva dijela i u prvom optimizirati broj vozila, a u drugom, udaljenost kao u radu [53].

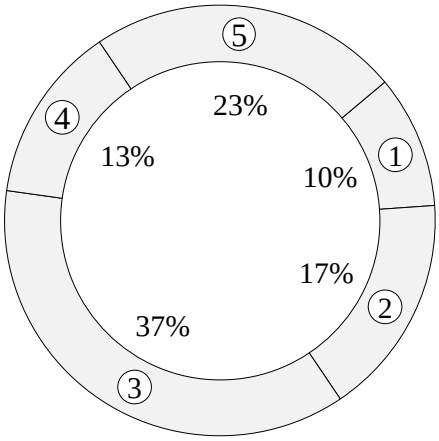
Selekcija

Dva često korištena i jednostavna postupka selekcije su turnirska selekcija (eng. *Tournament Selection*) i selekcija pomoću kotača ruleta (eng. *Roulette Wheel Selection*). Kod turnirske selekcije, dvije ili više jedinki odabiru se iz populacije slučajnim odabirom, te im se uspoređuje dobrota. Jedinka s najboljom vrijednošću dobrote postaje pobjednik turnira. Što je veći broj sudionika turnira to će veći biti i selekcijski pritisak. Najjednostavnija varijanta turnirske selekcije je binarna turnirska selekcija kod koje se natječu samo dva protukandidata. Na slici 3.15 prikazan je ishod dva binarna turnira proveden na jedinkama populacije koja broji pet članova i čije su vrijednosti dobrote redom: 10, 17, 37, 13, 23. U oba turnira kandidati su odabrani slučajno, a pobijedio je onaj s boljom vrijednošću dobrote.



Slika 3.15. *Binarna turnirska selekcija*

Kod selekcije pomoću kotača ruleta, vjerojatnost odabira jedinke proporcionalna je njezinoj dobroti. Slika 3.16 vizualizira kotač ruleta za populaciju iz prethodnog primjera. Na zamišljenom ruletu površina žlijeba određuje vjerojatnost da će se u njemu loptica zaustaviti odnosno da će jedinka koju predstavlja biti odabrana.



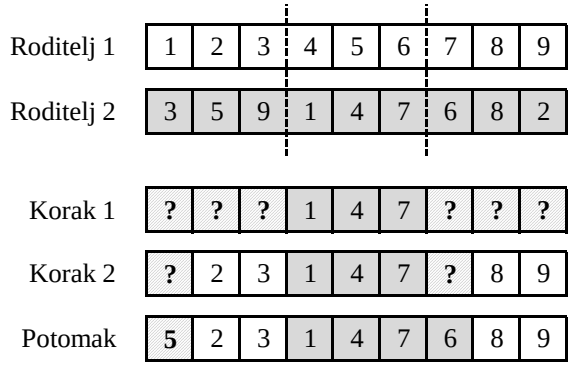
Slika 3.16. *Selekcija pomoću kotača ruleta*

Implementacija ovog postupka zahtjeva izračun zbroja dobrote svih jedinki u populaciji, generiranje slučajne vrijednosti između 0 i dobivenog zbroja, te ponovni prolaz populacije uz zbrajanje dobrote. U trenutku kada zbroj bude veći od slučajno generirane vrijednosti, postupak se zaustavlja odabirom trenutne jedinke. Selekcijski pritisak može se kontrolirati skaliranjem dobrote [62]. Nedostatak rulet selekcije je preveliko favoriziranje najbolje jedinke u slučajevima kada je vrijednost njezine dobrote mnogo bolja nego kod ostalih jedinki u populaciji. Alternativa je selekcija rangiranjem koja koristi isti postupak, no vjerojatnost odabira proporcionalna je poziciji jedinke u ukupnom poretku populacije s obzirom na dobrotu. Primjenom rangiranja selekcijski pritisak se umanjuje, a glavni nedostatak postupka je potreba za sortiranjem populacije u svakoj generaciji.

Križanje

Križanje je operator varijacije koji generira nove jedinke kombinirajući genetski materijal jedinki roditelja. Implementacija ovisi o načinu na koji se rješenja problema kodiraju u kromosome. Primjerice, kod binarnog kodiranja križanje se može izvesti presijecanjem kromosoma roditelja na slučajno odabranoj poziciji i sastavljanjem potomka od suprotnih dijelova kromosoma oba roditelja. Ako su rješenja kodirana permutacijama što je najčešći slučaj kod rješavanja TSP ili VRP problema, koriste se specijalizirani operatori križanja poput djelomično usklađenog križanja (eng. *Partially Matched Crossover*, kratica PMX), slijednog

križanja (eng. *Order-based Crossover*, kratica OX) i cikličkog križanja (eng. *Cycle Crossover*, kratica CX) [62],[84]. Slika 3.17 prikazuje primjer križanja provedenog PMX operatorom. Nakon što se slučajnim odabirom odrede dvije točke prekida, u kromosom potomka prvo se kopira srednji segment kromosoma drugog roditelja (korak 1). Potom se iz lijevog i desnog dijela kromosoma prvog roditelja na iste pozicije kopiraju svi geni koji već nisu umetnuti u srednji dio kromosoma potomka (korak 2). Generiranje potomka završava popunjavanjem preostalih pozicija pomoću zamjenskih gena ($4 \leftrightarrow 1, 5 \leftrightarrow 4, 6 \leftrightarrow 7$). Na prvu poziciju, gen 1 iz prvog roditelja ne može biti ubačen pošto se već nalazi u srednjem dijelu kromosoma. Budući da je njegov zamjenski gen 4 također ubačen u srednji dio, na prvu poziciju se ubacuje gen 5, zamjenski gen od 4. Istim postupkom, na sedmu poziciju ubacuje se preostali gen 6. Rezultat križanja uvijek je ispravno kodiran kromosom, a zamjenom uloga roditelja može se dobiti još jedna, različita jedinka.



Slika 3.17. *Križanje dvaju jedinki PMX postupkom*

PMX križanje nastoji očuvati roditeljske pozicije gena u kromosomu djeteta gdje god je to moguće. Kod CX križanja potomak nasljeđuje sve pozicije gena od roditelja, a kod OX križanja nastoji se sačuvati njihov redoslijed. Navedeni operatori su "slijepi" što znači da prilikom generiranja potomka ne koriste nikakvo znanje o problemu. Složeniji operatori poput križanja zasnovanog na rekombinaciji bridova (eng. *Edge Recombination Crossover*, kratica ERX) i križanja zasnovanog na sklapanju bridova (eng. *Edge Assembly Crossover*, kratica EAX) koriste informacije o problemu tijekom generiranja potomka (udaljenosti između lokacija), te se uz njih u pravilu postižu bolji rezultati nego s klasičnim operatorima. ERX i EAX križanja predložena su za potrebe rješavanja problema trgovačkog putnika u radovima [111] i [89]. ERX križanje autori koriste za rješavanje CVRP problema u radu [1]. EAX križanje koristi se također za rješavanje CVRP problema u radu [86], a za rješavanje VRPTW problema u radovima [87],[88],[90]. Kod rješavanja CVRP i VRPTW problema, primjena bilo kojeg od nabrojanih operatora križanja može rezultirati neizvedivim rješenjima u kojima nisu zadovoljena sva ograničenja, pa se ona trebaju popravljati dodatnim postupcima ili drugačije vrednovati prilikom evaluacije.

Mutacije

Mutacije su vrlo važan element svakog evolucijskog algoritma a predstavljaju povremene i slučajne varijacije u genima jedinke. Iako najčešće pogoršavaju dobrotu jedinke, doprinose raznovrsnosti genetskog materijala u populaciji i sprječavaju prerano zasićenje populacije istim ili vrlo sličnim jedinkama. Kod nekih varijanti evolucijskog algoritma, mutacije se koriste kao jedini operator varijacije. Kao i kod križanja, implementacija mutacije ovisi o načinu kodiranja rješenja. Kod binarnog kodiranja, mutacija može predstavljati zamjenu vrijednosti bita na slučajno odabranoj poziciji u kromosomu. Kod kodiranja permutacijom, za mutacije se najčešće koriste jednostavne operacije prikazane na slici 3.18, [106],[1].

(a) Jedinka

(b) Mutacija relokacijom gena

(c) Mutacija zamjenom gena

(d) Mutacija inverzijom segmenta

Slika 3.18. *Različite varijante operatora mutacije*

Slika 3.18(b) prikazuje mutaciju u kojoj se gen sa slučajno odabrane pozicije u kromosomu premješta na drugu slučajno odabranu poziciju. Na slici 3.18(c), prikazana je mutacija međusobnom zamjenom gena koji se nalaze na dvama slučajno odabranim pozicijama, a na slici 3.18(d), mutacija inverzijom slučajno odabranog segmenta permutacije. S ciljem povećanja učinkovitosti u nekim radovima se u svojstvu mutacije koristi postupak lokalne pretrage [94]. U radovima [69],[14],[53], koriste se specijalizirane mutacije koje doprinose minimizaciji broja vozila kod rješavanja VRPTW problema.

Zamjena jedinki

Na kraju svake generacije, nakon što završi reproduktivni ciklus, između starih jedinki iz populacije i novih jedinki potomaka, potrebno je odabrati one koje će preživjeti i nastaviti proces evolucije u slijedećoj generaciji. Pri tome se mogu koristiti različite strategije. Primjerice, moguće je zamijeniti sve stare jedinke novima ili uzeti najbolje jedinke iz obje populacije. U sklopu zamjene, često se

koristi načelo elitizma odnosno kopiranje najbolje jedinke u svaku slijedeću generaciju. Time se sprječava iščezavanje najbolje jedinke iz populacije te istovremeno pojačava intenzifikacija pretrage područja rješenja u kojem se ono nalazi.

3.3.2 Simulirano kaljenje

Simulirano kaljenje (eng. *Simulated Annealing*, kratica SA) jedna je od najstarijih metaheurističkih metoda koja oponaša termodinamičke procese koji se odvijaju u metalurgiji prilikom kaljenja metala [83],[72],[44],[53]. Kaljenjem se nastoji postići stanje minimalne unutarnje energije koja se dobije formiranjem pravilne kristalne rešetke. Metal se zagrijava na temperaturu taljenja, nakon čega se polako hladi. Na visokim temperaturama unutarnja struktura materijala je stohastički organizirana, a što je veća temperatura veće je i gibanje atoma. Brzim spuštanjem temperature (eng. *quenching*) atomi ostaju zarobljeni u položaju u kojem su se zatekli, te formiraju metastabilne strukture koje predstavljaju energetski lokalni optimum. Da bi se dobio energetski globalni optimum u postupku se koristi sporo hlađenje ili kaljenje (eng. *annealing*) koje ostavlja dovoljno vremena atomima da formiraju pravilne kristalne strukture. Opisana tehnika može se preslikati i upotrijebiti za rješavanje optimizacijskih problema koristeći temperaturu kao ključni parametar koji kontrolira diverzifikaciju i intenzifikaciju pretrage. Veća temperatura znači veću diverzifikaciju ili istraživanje cjelokupnog prostora rješenja, a manja temperatura veću intenzifikaciju ili eksploataciju uskog područja rješenja u kojem se pretraga nalazi. Diverzifikacija podrazumijeva prihvaćanje lošijih rješenja i udaljavanje od susjedstva trenutnog rješenja.

Struktura algoritma

Pseudokôd simuliranog kaljenja prikazan je u algoritmu 2. Na osnovu početnog rješenja s, određuju se početna i završna temperatura kaljenja. U glavnoj petlji, pri svakoj temperaturi odvija se ciklus kaljenja kroz k iteracija što treba osigurati dovoljno promjena potrebnih za uspostavljanje "termodinamičke ravnoteže". Unutar ciklusa kaljenja, u svakoj iteraciji, nasumično se odabire susjedno rješenje s'' do kojeg se može doći jednostavnim modificiranjem trenutnog rješenja s'. Ukoliko je susjedno rješenje bolje od trenutnog, bezuvjetno se prihvaća i prenosi u iduću iteraciju. U protivnom, odluka o prihvaćanju susjednog rješenja ovisi o vjerojatnosti koja se dobije pomoću funkcije *VjerojatnostPrihvacanja*. Ta se vjerojatnost izračunava prema Boltzmannovoj distribuciji (3.1), a ovisi o trenutnoj temperaturi T_i i vrijednosti ΔE koja predstavlja razliku u funkciji cilja između susjednog rješenja s'' i trenutnog rješenja s'. Boltzmanova konstanta u ovom se slučaju zanemaruje.

$$p = e^{-\frac{\Delta E}{T_i}} \tag{3.1}$$

Ukoliko je slučajno generirani broj čija je vrijednost unutar intervala [0..1] manji ili jednak dobivenoj vjerojatnosti p, susjedno rješenje se prihvaća i prenosi u iduću iteraciju. U protivnom, trenutno rješenje s' ostaje nepromijenjeno i postupak se ponavlja u idućoj iteraciji. Što je temperatura veća, veća je i vjerojatnost da će se prihvatiti znatno lošije rješenje. Pri niskim temperaturama samo dovoljno kvalitetna rješenja imaju izglede biti prihvaćena. Na kraju svakog ciklusa kaljenja poziva se funkcija za smanjenje temperature, nakon čega započinje novi ciklus kaljenja.

Algoritam 2 Osnovni algoritam simuliranog kaljenja

```
1:  procedure SA(s, n, k)
2:      i ← 1
3:      T_0 ← PocetnaTemperatura(s)
4:      T_n ← ZavrsnaTemperatura(s)
5:      T_i ← T_0
6:      s' ← s
7:      repeat
8:          for j = 1 → k do
9:              s'' ← SusjednoRjesenje(s')
10:             if f(s'') < f(s') then
11:                 s' ← s''
12:             else
13:                 p ← VjerojatnostPrihvacanja(s', s'', T_i)
14:                 if Random(0, 1) ≤ p then
15:                     s' ← s''
16:                 end if
17:             end if
18:         end for
19:         i ← i + 1
20:         T_i ← Ohladi(i, n, T_0, T_n)
21:     until UvjetPrekida(i, n, T_i, T_n)
22:     return s'
23: end procedure
```

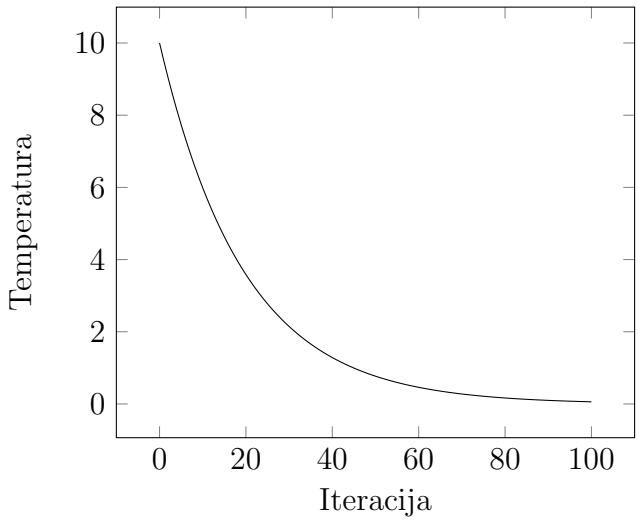
Podešavanje parametara

Dimenzije problema i prostorni raspored korisnika bitno utječu na ciljnu funkciju, pa različiti problemi zahtijevaju prilagodbu i podešavanje početne i završne temperature kao i odabir odgovarajuće funkcije hlađenja kako bi se mogli učinkovito riješiti. Početna temperatura T_0 treba biti takva da u početku kaljenja vjerojatnost prihvaćanja lošijih susjednih rješenja prema izrazu (3.1) bude vrlo velika. S druge strane, završna temperatura T_n mora biti toliko mala da se

pri kraju rješavanja prihvaćaju samo bolja ili neznatno lošija rješenja. Smjernice za određivanje početne i završne temperature mogu se pronaći u [43],[53]. Prijelaz iz stanja "visoke" temperature u "nisku", ključan je za uravnoteženje diverzifikacije/intenzifikacije pretrage. Naglim spuštanjem temperature ne ostavlja se dovoljno vremena za diverzifikaciju pa potencijalno dobra područja prostora rješenja ostaju neistražena. S druge strane, presporim hlađenjem pretjerano se eksploatiraju područja koja često ne sadrže kvalitetna rješenja što može usporiti cijeli postupak ili ostaviti manje vremena za intenzivno pretraživanje područja oko najboljih rješenja pronađenih u završnoj fazi. Jedan od raširenijih načina smanjivanja temperature je primjena geometrijskog pravila:

$$T_{i+1} = \alpha \cdot T_i \ , \tag{3.2}$$

gdje je α konstanta čija se vrijednost obično kreće između 0.8 i 0.99. Slika 3.19 prikazuje opadanje temperature kroz 100 ciklusa hlađenja za parametre $T_0 = 10$ i $\alpha = 0.95$.

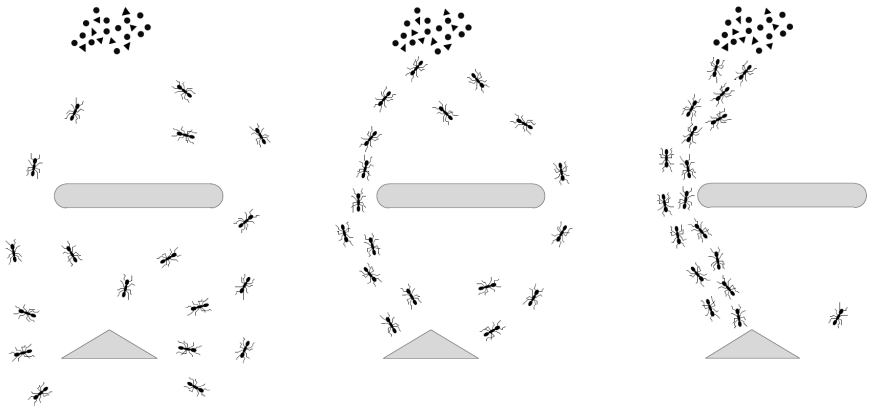


Slika 3.19. *Krivulja opadanja temperature kod geometrijskog hlađenja*

Uz ovakav program hlađenja potrebno je vrijeme izvršavanja prilagoditi dimenzijama problema koji se rješava podešavanjem broja ciklusa hlađenja, broja iteracija u svakom ciklusu i konstante α. Program hlađenja moguće je prilagoditi maksimalnom broju iteracija ili ograničenju trajanja izvršavanja izraženom u sekundama. Više informacija o različitim programima hlađenja može se pronaći [81],[53],[79]. Kod rješavanja problema usmjeravanja vozila, simulirano kaljenje često se koristi u kombinaciji s drugim (meta)heuristikama kao glavni mehanizam pomoću kojega se vodi pretraga [100],[11],[96],[91].

3.3.3 Optimizacija mravljom kolonijom

Algoritmi optimizacije mravljom kolonijom (eng. *Ant Colony Optimization*, kratica ACO) predloženi su ranih devedesetih u radovima [38],[28],[37]. Temelje se na principima neizravne komunikacije između jedinki iste vrste putem tragova koje ostavljaju u okolišu [40]. Takav oblik komunikacije omogućava postizanje kompleksnih, samo-organiziranih ponašanja unatoč nedostatku sposobnosti planiranja, inteligencije i pamćenja kod samih jedinki [53]. Primjerice, vrsta argentinskog mrava *Iridomyrmex humilis* u stanju je pronaći najkraći put između izvora hrane i mravinjaka neizravno komunicirajući putem feromona [64],[35]. Proces uspostave najkraćeg puta ilustriran je na slici 3.20.



(a) Početak traganja (b) Uspostava dva puta (c) Odabir kraćeg

Slika 3.20. *Najkraći put kao rezutltat neizravne komunikacije putem feromona*

U početku potrage za hranom što prikazuje slika 3.20(a), mravi se gibaju u svim smjerovima i istražuju područje u blizini mravinjaka, ostavljajući za sobom feromonski trag na tlu. Prvi mravi koji samostalno pronađu hranu ponijeti će dio hrane natrag u mravinjak, vraćajući se putem koji su prethodno označili i ostavljajući novi feromonski trag čija jačina ovisi o kvaliteti i količini hrane koju su pronašli. Kada ostali mravi naiđu na veću koncentraciju feromona, postoji velika vjerojatnost da će početi slijediti taj trag, slika 3.20(b). Budući da se hrana brže transportira kraćim putem koji vodi s lijeve strane prepreke, na njemu će se s vremenom nakupiti veća koncentracija feromona pa će velika većina mravi u konačnici prometovati najkraćim putem, slika 3.20(c). Izvorni ACO algoritam nazvan je "mravlji sustav" (eng. *Ant System*, kratica AS) i primijenjen je za rješavanje problema trgovačkog putnika. Mravi su simulirani softverskim agentima koji za razliku od prirodnih mrava nisu "slijepi" i imaju sposobnost pamćenja. To znači da mogu određivati udaljenosti između lokacija i na kraju putovanja rekonstruirati cijeli put. Okoliš u kojem se kreću i ostavljaju tragove predstavljen je matricom feromona veličine $n \times n$, gdje n predstavlja broj lokacija, a svaki

element matrice (i, j) količinu feromona na putu između lokacija i i j. Pronalazak hrane i povratak u mravinjak predstavlja konstruiranje izvedivog rješenja, a uspostavljanje najkraćeg puta između hrane i mravinjaka pronalazak optimalnog rješenja problema. U izvornom radu predložene su tri varijante mravljeg sustava, *Ant-density*, *Ant-quantity* i *Ant-cycle*, koje se razlikuju po načinu ažuriranja feromonskog traga. Kod prve dvije varijante feromon se ažurira tijekom konstruiranja rješenja, nakon što mrav doputuje na lokaciju j s lokacije i. Pri tome se kod *Ant-density* algoritma vrijednost elementa (i, j) u matrici uvećava za konstantnu količinu feromona, a kod *Ant-quantity* algoritma, za količinu feromona koja je obrnuto proporcionalna udaljenosti između i i j. Kod *Ant-cycle* varijante, feromonski trag ažurira se tek nakon što mrav konstruira čitavo rješenje pri čemu se na sve korištene lukove u rješenju dodaje količina feromona obrnuto proporcionalna duljini ukupnog pređenog puta. Ova metoda pokazala se boljom od prethodne dvije zbog toga što količina feromona kojom se ažurira matrica ovisi o kvaliteti cijelog rješenja, a ne samo o duljini pojedinačnih lukova što posebno dolazi do izražaja kod rješavanja složenijih problema poput VRPTW problema gdje je potrebno zadovoljiti velik broj ograničenja. Algoritam 3 prikazuje pseudokod metode mravljeg sustava *Ant-Cycle*.

Algoritam 3 Mravlji sustav

```
1:  procedure AS(n, i_max, α, β, ρ)
2:      mravi ← InicijalizirajAgente(n)
3:      η ← InicijalizirajMatricuUdaljenosti()
4:      S ← KonstruirajPocetnaRjesenja(n, η)
5:      τ ← InicijalizirajMatricuFeromona(S)
6:      s_min = Najbolji(S)
7:      for i = 1 → i_max do
8:          for j = 1 → n do
9:              KonstruirajRjesenje(mravi[j], τ, η, α, β)
10:             if f(mravi[j]) < f(s_min) then
11:                 s_min ← mravi[j]
12:             end if
13:         end for
14:         Ispari(τ, ρ)
15:         Azuriraj(τ, mravi)
16:     end for
17:     return s_min
18: end procedure
```

Nakon što se inicijalizira n agenata i matrica udaljenosti između svih lokacija u problemu, konstruira se n različitih rješenja pomoću neke od konstruktivnih heurističkih metoda. Na temelju lukova korištenih u rješenjima, inicijalizira se

matrica feromona τ. Broj iteracija glavne petlje određen je parametrom i_{max}, a u svakoj iteraciji umjetni mravi konstruiraju rješenja problema. Pri tome se oslanjaju na postojeće tragove feromona koji su se akumulirali u prethodnim iteracijama, ali i na "vid" pomoću kojeg mogu odrediti udaljenosti između lokacija. Parametri α i β određuju utjecaj feromona i "vida" na postupak konstruiranja rješenja. Ukoliko mrav pronađe najbolje rješenje do tog trenutka, ono se sprema u varijablu s_{min}. Nakon što faza konstruiranja novih rješenja završi, poziva se funkcija $Ispari$ koja simulira isparavanje feromona. Razina feromona smanjuje se u cijeloj matrici, pri čemu parametar ρ određuje postotak smanjenja. Nakon isparavanja, matrica se ažurira dodavanjem novih količina feromona samo na lukove koji su korišteni u konstruiranim rješenjima. Nakon isteka posljednje iteracije, algoritam vraća najbolje pronađeno rješenje s_{min}.

Konstruiranje rješenja

Ključni dio svakog ACO algoritma je konstruktivna heuristika kojom se u svakoj iteraciji iznova konstruiraju rješenja problema. Podrazumijeva se da je heuristika prilagođena problemu koji se rješava i da uvažava postavljena ograničenja. U svakoj iteraciji glavnog algoritma agenti stohastički konstruiraju rješenja kombinirajući "vid" i informacije iz matrice feromona u kojoj je zapisana učestalost korištenja pojedinih lukova. Matrica feromona predstavlja znanje i iskustvo čitave kolonije. Postupak konstruiranja rješenja problema opisan je algoritmom 4.

Algoritam 4 Konstruiranje rješenja

```
1:  procedure KonstruirajRjesenje(mrav, τ, η, α, β)
2:      Resetiraj(mrav)
3:      C ← Nerutirani()
4:      while |C| > 0 do
5:          J ← Kandidati(mrav, C)
6:          if |J| = 0 then
7:              break
8:          end if
9:          j ← Odaberi(mrav, J, τ, η, α, β)
10:         Dodaj(mrav, j)
11:         Izbaci(C, j)
12:     end while
13: end procedure
```

Postupak započinje resetiranjem agenta i kreiranjem liste C koja sadrži korisnike koje treba rutirati, a završava kada se ta lista isprazni. Kandidati za iduće odredište izdvajaju se u listu J pomoću funkcije $Kandidati$. Implementacija ove funkcije ovisi o vrsti problema budući da u datom trenutku treba izdvojiti samo one kandidate koji zadovoljavaju postavljena ograničenja. Primjerice, kod CVRP

problema kandidati su svi korisnici čija je potražnja manja ili jednaka preostalom kapacitetu vozila koje se rutira. U slučaju da kandidata nema, postupak se prekida. U suprotnom, kandidat j odabire se iz liste funkcijom *Odaberi*. Ako je trenutna lokacija u kojoj se mrav nalazi označena s i, odabir kandidata j ovisit će o vidljivosti η_{ij} i jačini feromonskog traga koji vodi do njega, τ_{ij}. Vidljivost η_{ij} jednaka je recipročnoj vrijednosti udaljenosti između dvije lokacije, $\eta_{ij} = \frac{1}{d_{ij}}$. Oslanjajući se na vidljivost, ubrzava se proces konstruiranja dobrih rješenja jer se preferiraju kraći lukovi. S druge strane, oslanjanjem na feromon τ_{ij}, crpi se iskustvo cijele kolonije akumulirano u prethodnim iteracijama. Vjerojatnost odabira korisnika j izračunava se uz pomoć izraza:

$$p_{ij} = \begin{cases} \frac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{k \in J} (\tau_{ik})^{\alpha} \cdot (\eta_{ik})^{\beta}} & \text{, ako } j \in J \\ 0 & \text{, ako } j \notin J \end{cases} . \tag{3.3}$$

Parametri α i β određuju relativnu važnost vidljivosti i tragova feromona prilikom odabira korisnika j. Dva krajnja slučaja su $\alpha = 0$, kada na konstruiranje rješenja utječu samo udaljenosti između kandidata te $\beta = 0$, kada se rješenje konstruira samo na temelju akumuliranog feromona. Odabrani kandidat j ubacuje se na kraj postojeće rute, a izbacuje iz liste nerutiranih korisnika C. Ako je korisnik j bio jedini mogući kandidat, ruta aktivnog vozila zatvara se dodavanjem skladišta, nakon čega se aktivira slijedeće nekorišteno vozilo. Postupak se ponavlja dok god ima nerutiranih korisnika.

Inicijalizacija feromona

Matrica feromona inicijalizira na temelju n početnih rješenja dobivenih konstruktivnim heurističkim algoritmom. Za svaki korišteni luk u rješenju, element matrice koji predstavlja taj luk uvećava se za količinu feromona koja je obrnuto proporcionalna funkciji cilja za to rješenje. Lukovima koji nisu korišteni niti u jednom početnom rješenju, postavlja se vrijednost τ_{min} koja se kod VRP problema može grubo odrediti pomoću izraza:

$$\tau_{min} = \frac{1}{f(s_w) \cdot n} , \tag{3.4}$$

gdje je $f(s_w)$ vrijednost funkcije cilja za najlošije generirano početno rješenje, a n broj mrava. Minimalna količina feromona τ_{min} ne smije biti premala jer će se u tom slučaju pretraga usmjeriti prema prvim konstruiranim rješenjima koja su daleko od optimalnog. Isto tako, vrijednost τ_{min} ne smije biti ni prevelika jer će mravi u prvim iteracijama previše lutati, umanjujući tako učinkovitost algoritma i vrijednost potencijalno dobrih tragova koji su rezultat konstruktivne heurističke metode.

Isparavanje feromona

Iako se u prirodi isparavanje feromona odvija sporo i ne utječe previše na sposobnost mrava da pronađu najkraći put, kod ACO algoritama ono igra vrlo važnu ulogu. Sa sporim isparavanjem ili bez njega, algoritam ima tendenciju vrlo brzo zaustaviti se u lošem lokalnom optimumu, te pojačavati ranije uspostavljeni put novim količinama feromona iz iteracije u iteraciju. Ubrzavanjem isparavanja omogućava se zaboravljanje loših segmenata rješenja i bijeg iz lokalnog optimuma. Količina feromona u svim elementima matrice smanjuje se pomoću izraza (3.5), gdje vrijednost parametra ρ predstavlja postotak za koji se feromon smanjuje.

$$\tau_{ij} = (1 - \rho)\tau_{ij} \tag{3.5}$$

Ažuriranje feromona

Nakon isparavanja, matrica feromona ažurira se dodavanjem novog sloja samo na lukove po kojima su se mravi kretali. Količina novog feromona koji se dodaje ovisi i o vrijednosti ciljne funkcije za svako pojedino rješenje. Nove vrijednosti elemenata u matrici feromona mogu se izračunati prema izrazu:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \ , \tag{3.6}$$

gdje je m broj mravi, a $\Delta\tau_{ij}^{k}$ količina feromona koju mrav k ispušta na luk (i, j). U slučaju da mrav u rješenju ne koristi luk (i, j) neće ni ispustiti feromon na njega, dok će u suprotnom ispuštena količina biti obrnuto proporcionalna ciljnoj funkciji za kompletno rješenje:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{f(s_k)} \text{, ako luk } (i,j) \in s_k \\ 0 \quad \text{, ako luk } (i,j) \notin s_k \end{cases} \ , \tag{3.7}$$

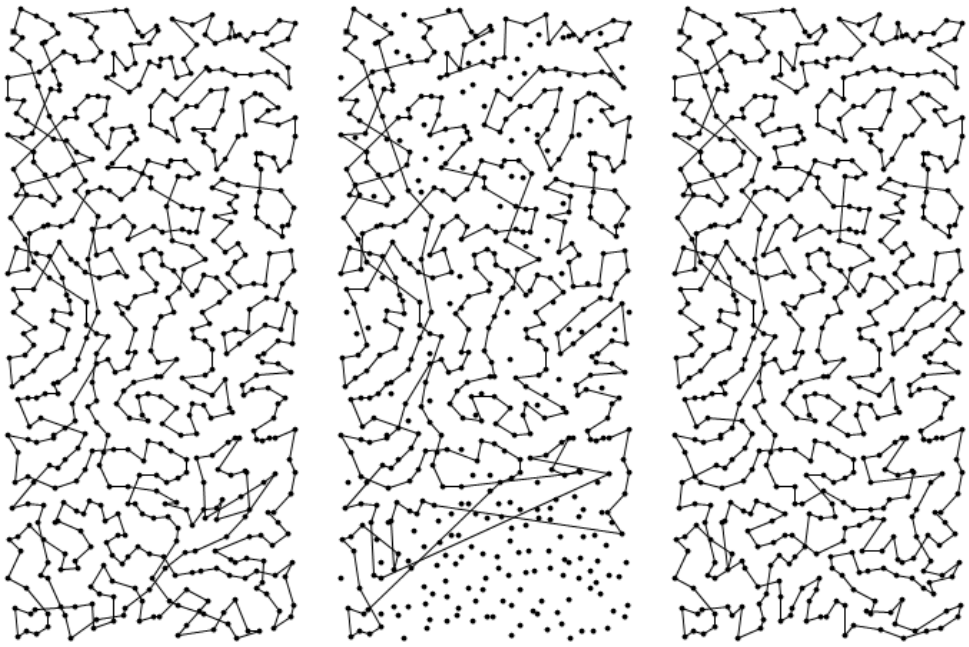
gdje je s_k rješenje koje je mrav k konstruirao, a Q fiksni parametar. Može se primijetiti da osim kvalitete rješenja, količina feromona koja će se dodati na luk (i, j) ovisi i o učestalosti njegova korištenja. To znači da će feromonski trag na luku biti jači što ga više mravi koristi.

Proširenja osnovnog algoritma

S vremenom su se pojavile različite inačice ACO algoritama koje se u prvom redu razlikuju po načinu ažuriranja i upravljanja feromonom [40]. Među značajnije spadaju elitistički mravlji sustav [39], mravlji sustav zasnovan na rangiranju [22], $\mathcal{MAX} - \mathcal{MIN}$ mravlji sustav [103], te sustav mravlje kolonije koji je svojevremeno bio među najboljim algoritmima za rješavanje VRPTW problema [55].

3.4 Pretraživanje velikog susjedstva

Pretraživanje velikog susjedstva (eng. *Large Neighborhood Search*, kratica LNS) postupak je baziran na lokalnom pretraživanju predložen za rješavanje problema usmjeravanja vozila u radu [101]. Sličan koncept nazvan principom uništavanja i popravljanja (eng. *Ruin and Recreate*, kratica *R&R*) predlažu autori otprilike u isto vrijeme u radu [100]. Dok je kod većine algoritama susjedstvo eksplicitno definirano operatorima poboljšavanja poput ranije opisanih *Intra* i *Inter* operatora, kod LNS algoritma susjedstvo je implicitno definirano jednom operacijom uništavanja i popravljanja rješenja [92]. Slika 3.21 ilustrira princip uništavanja i popravljanja na problemu trgovačkog putnika.



(a) Početno (b) Uništeno (c) Popravljeno

Slika 3.21. *Rezultat operacije uništavanja i popravljanja TSP problema*

Način na koji se rješenje uništava bitno utječe na učinkovitost algoritma pa tako autori u adaptivnoj inačici LNS algoritma koriste sedam različitih kriterija po kojima se korisnici izbacuju iz ruta [91]. Postupak popravljanja kompleksniji je i računski zahtjevniji od postupka uništavanja pa je bitno da bude brz i učinkovit. U spomenutom radu, autori koriste dvije heuristike za umetanje. Operacijom uništavanja i popravljanja može se zaobići dosta lokalnih optimuma što pojačava intenzitet pretrage prostora rješenja. Jedna od praktičnih prednosti LNS algoritma je mogućnost jednostavne prilagodbe novim ograničenjima s obzirom da je potrebno modificirati samo postupak popravljanja kojim se korisnici umeću u rute. Zbog navedenog, koncept uništavanja i popravljanja odabran je kao glavni mehanizam pretraživanja prostora rješenja u ovom istraživanju.

Poglavlje 4

Hibridni stanični evolucijski algoritam

U ovom poglavlju opisan je razvijeni hibridni stanični evolucijski algoritam za rješavanje problema usmjeravanja vozila (kratica HCEA, od eng. *Hybrid Cellular Evolutionary Algorithm*). Ograničavanje interakcije jedinki iz populacije samo na neposredno susjedstvo u prostornoj matrici za cilj ima postizanje boljeg kompromisa između istraživanja (diverzifikacije pretrage) i eksploatacije (intenzifikacije pretrage) prostora rješenja u odnosu na klasične evolucijske algoritme. Poput mnogih algoritama za rješavanje VRP problema koji su se pojavili u novije vrijeme, predloženi stanični evolucijski algoritam nosi atribut hibridni budući da u radu koristi određene mehanizme koji nisu predviđeni u tradicionalnom ili kanonskom obliku evolucijskog algoritma. Stanični genetski algoritam koji je poslužio kao polazna točka razvoja HCEA algoritma prezentiran je na početku poglavlja. U odjeljku koji slijedi objašnjen je koncept pretraživanja prostora rješenja na kojem se algoritam temelji, način na koji su rješenja kodirana u jedinke, te razlozi razdvajanja postupka rješavanja u dvije faze. Nadalje, opisani su svi koraci HCEA algoritma: inicijalizacija populacije, evaluacija dobrote jedinki, postupak selekcije jedinki za reprodukciju, operatori križanja i mutacije, te način zamjene populacije odnosno odabir jedinki preko kojih će se evolucija nastaviti u idućoj generaciji. Naposljetku, prezentirani su rezultati ostvareni na standardnim testnim zadacima te je napravljena usporedba s najboljim poznatim rezultatima drugih autora.

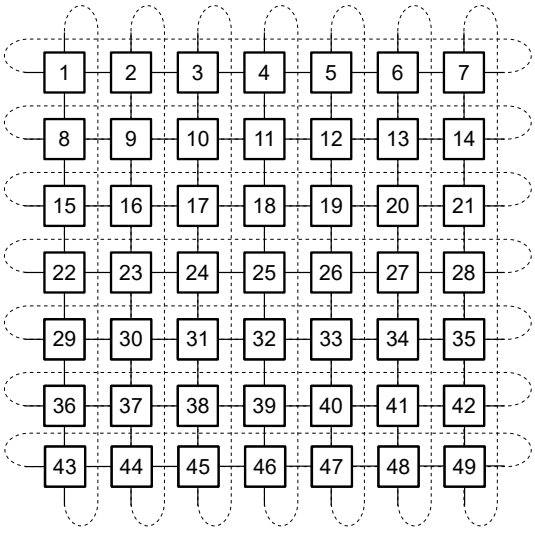
4.1 Stanični genetski algoritam

Stanični genetski algoritam (eng. *Cellular Genetic Algorithm*, kratica cGA) podvrsta je evolucijskog algoritma u kojem su raznolikost u populaciji i istraživanje prostora rješenja poboljšani zahvaljujući postojanju malih preklapajućih susjedstava [1]. Evolucija iz perspektive jedinke koja je u interakciji samo s obližnjim

susjedima rezultira finim raspršivanjem genetskog materijala (rješenja ili dijelova rješenja) kroz populaciju. Time se usporava nagla konvergencija koja se manifestira zasićenjem populacije jednakim ili jako sličnim jedinkama. Pojačano istraživanje posljedica je efekta izolacije putem udaljenosti i stvaranja grozdova koji sadrže slične jedinke [4]. Granice tih grozdova nisu jasno definirane već se neprestano mijenjaju uslijed selekcijskog pritiska i propagacije dobrih jedinki. Grozdovi s boljim jedinkama brzo koloniziraju obližnja područja s lošim jedinkama, a s vremenom i širenjem dospijevaju i do udaljenih dijelova populacije. Eksploatacija određenog područja u prostoru rješenja odvija se unutar samih grozdova primjenom operatora križanja i mutacije te opcionalno, primjenom procedura lokalnog pretraživanja na pojedinoj jedinki.

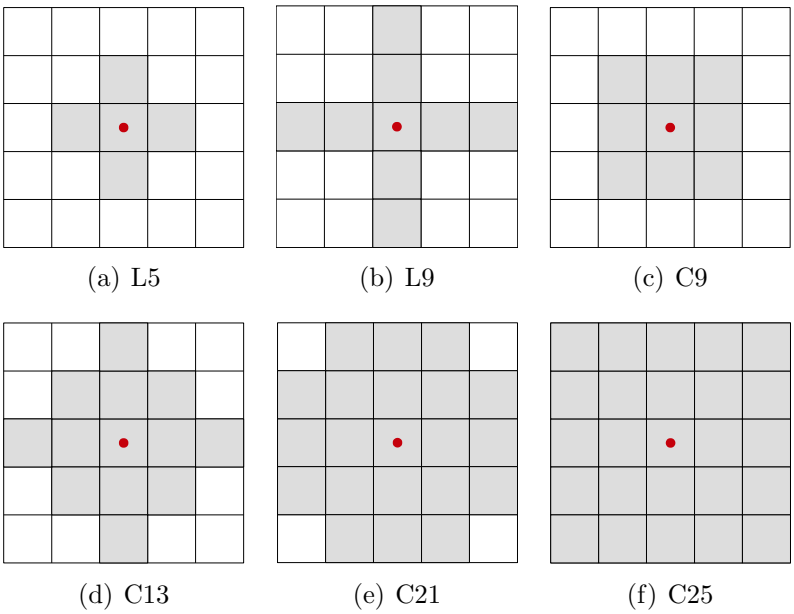
4.1.1 Toroidalna topologija i tipovi susjedstva

Kod staničnog genetskog algoritma svaka jedinka iz populacije zauzima jedinstvenu poziciju u toroidalnoj mreži. Jedinke mogu biti u interakciji samo s obližnjim susjedima, a toroidalna topologija osigurava povezanost jedinki s rubova mreže s jedinkama koje se nalaze na suprotnim stranama. Time je osigurano da svaka jedinka ima jednak broj susjeda. Slika 4.1 prikazuje međusobnu povezanost 49 jedinki iz populacije koje su raspoređene na dvodimenzionalnoj toroidalnoj mreži dimenzija 7×7.



Slika 4.1. *Jedinke populacije raspoređene na torodialnoj mreži*

Dva najčešće korištena tipa susjedstva su L5 susjedstvo koje se još naziva i *Von Neumann*-ovo ili NEWS susjedstvo (od eng. *North, East, West, South*), te C9 susjedstvo, poznato i kao *Moore*-ovo susjedstvo [4]. L5, C9 i još nekoliko često korištenih tipova susjedstva prikazano je na slici 4.2.

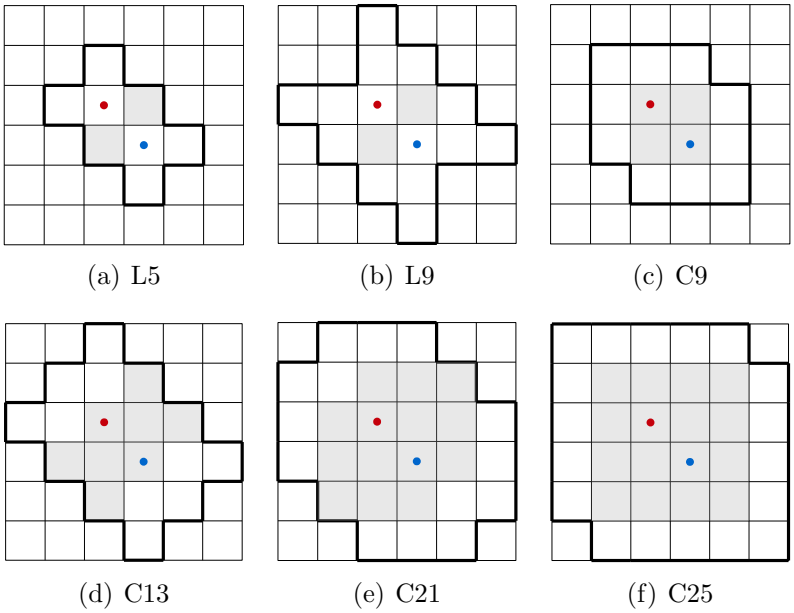


(a) L5 (b) L9 (c) C9

(d) C13 (e) C21 (f) C25

Slika 4.2. *Najčešće korišteni tipovi susjedstva*

Veća susjedstva povećavaju razinu migracija u mreži. Pretpostavka je da je susjedstvo svake jedinke u populaciji iste veličine i oblika. Slika 4.3 prikazuje preklapanje susjedstava od dvije susjedne jedinke.



(a) L5 (b) L9 (c) C9

(d) C13 (e) C21 (f) C25

Slika 4.3. *Preklapanje susjedstava dvaju susjednih jedinki*

4.1.2 Opis algoritma

Pseudokôd staničnog genetskog algoritma prikazan je u algoritmu 5. Maksimalan broj generacija zadan je parametrom g_{max} pri čemu jedna generacija predstavlja kompletan reproduktivni ciklus. Veličina i oblik susjedstva zadani su parametrom N_t, dok parametri w i h određuju dimenzije dvodimenzionalne toroidalne mreže, a time i veličinu populacije $|P| = w \times h$. Učestalost primjene operatora križanja i mutacija zadana je parametrima p_x i p_m, a primjena načela elitizma određena je vrijednošću binarnog parametra E.

Algoritam 5 Stanični genetski algoritam

```
1: procedure cGA(g_max, N_t, w, h, p_x, p_m, E)
2:     P ← GenerirajPocetnuPopulaciju(w, h)
3:     Evaluiraj(P)
4:     s_min ← Najbolji(P)
5:     for i = 1 → g_max do
6:         P* ← ∅
7:         for x = 1 → w do
8:             for y = 1 → h do
9:                 if Random() < p_x then
10:                    r_1 ← Selektiraj(P, N_t, x, y)
11:                    r_2 ← Selektiraj(P, N_t, x, y)
12:                    P*[x, y] ← Krizaj(r_1, r_2)
13:                else
14:                    r ← Selektiraj(P, N_t, x, y)
15:                    P*[x, y] ← r
16:                end if
17:                if Random() < p_m then
18:                    Mutiraj(P*[x, y])
19:                end if
20:                Evaluiraj(P*[x, y])
21:            end for
22:        end for
23:        Zamijeni(P, P*)
24:        s ← Najbolji(P)
25:        if f(s) < f(s_min) then
26:            s_min ← s
27:        end if
28:        if E = true then
29:            ZamijeniNajgoreg(P, s_min)
30:        end if
31:    end for
32:    return s_min
33: end procedure
```

U prvom koraku algoritma, stohastičkim konstruiranjem početnih rješenja inicijalizira se aktivna populacija P koja se pohranjuje u dvodimenzionalnoj matrici zadanih dimenzija. Sve jedinke iz populacije evaluiraju se, a najbolje ocijenjena sprema u varijablu s_{min}. Na početku svake generacije (iteracije glavne petlje) priprema se nova populacija P^* u koju će se privremeno pohraniti svi potomci proizašli iz trenutne generacije. Slijedi dvostruka ugniježđena petlja kojom se prolazi kroz sve pozicije u toroidalnoj mreži. Vjerojatnost da će na pojedinoj poziciji potomak nastati križanjem iznosi p_x. U slučaju križanja, funkcija *Selektiraj* odabire dva roditelja iz susjedstva trenutne pozicije (x, y) čija je veličina i oblik zadana parametrom N_t. Potomak dobiven razmjenom informacija od roditeljskih jedinki sprema se u novu populaciju P^* na poziciju (x, y). U slučaju da se križanje ne provede (vjerojatnost za to iznosi $1 - p_x$), u novu populaciju P^* na poziciju (x, y) kopira se jedinka iz populacije P odabrana istim postupkom selekcije koji se koristi kod križanja. Potomak $P^*[x, y]$ će mutirati s vjerojatnošću p_m, te će na kraju obavezno biti evaluiran. Nakon što postupak reprodukcije završi, jedinke aktivne populacije P zamjenjuju se potomcima s istih pozicija iz populacije P^*. Kriteriji zamjene mogu biti različiti, a dva ekstrema su: bezuvjetna zamjena svih starih jedinki novima, te zamjena samo lošijih jedinki boljima iz nove populacije. U privremenu varijablu s izdvaja se najbolja jedinka iz generacije te se globalno najbolja jedinka s_{min} ažurira ukoliko je jedinka s bolja. Ako se primjenjuje načelo elitizma odnosno bezuvjetnog prijenosa najbolje jedinke u svaku iduću generaciju, tada se najgora jedinka u populaciji zamjenjuje s s_{min}.

4.1.3 Sinkrona i asinkrona zamjena jedinki

Ovisno o strategiji zamjene starih jedinki novima, algoritam može imati sinkronu ili asinkronu implementaciju. Kod sinkrone implementacije, aktivna populacija iz koje se odabiru roditelji se ne mijenja dok ne završi kompletan ciklus reprodukcije jedne generacije. Drugim riječima, svi potomci su kreirani ravnopravno, u jednom trenutku. Kod asinkrone implementacije, jedinke iz aktivne populacije zamjenjuju se tijekom trajanja ciklusa reprodukcije. To znači da neke jedinke koje su potencijalni roditelji mogu iščeznuti iz populacije i prije nego što budu odabrane. Algoritam 5 opisan u prethodnom odlomku koristi sinkronu zamjenu jedinki. Prema [59],[4], osim strategije zamjene jedinki (sinkrono ili asinkrono), redoslijed kojim se zamjene obavljaju kod asinkrone implementacije također utječe na ponašanje algoritma. Postoji nekoliko različitih metoda zamjene jedinki kod asinkrone implementacije cGA algoritma:

1. **Linearni prolaz** (eng. *Line Sweep*, kratica LS) - Jedinke se zamjenjuju redak po redak, stupac po stupac, u najjednostavnijem slučaju odmah nakon dobivanja potomka u svakoj iteraciji ugniježđene petlje.

2. **Fiksno zadani nasumični prolaz** (eng. *Fixed Random Sweep*, kratica FRS) - Slučajnim odabirom generirana permutacija pozicija koja predsta-

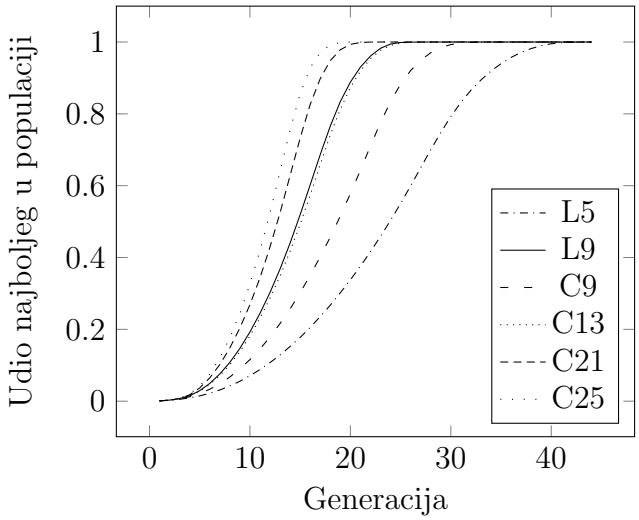
vlja redoslijed provedbe zamjena. Redoslijed se ne mijenja tijekom trajanja algoritma.

3. **Novi nasumični prolaz** (eng. *New Random Sweep*, kratica NRS)- Redoslijed provedbe zamjena u svakoj generaciji se iznova generira slučajnim odabirom.

4. **Jednoliki odabir** (eng. *Uniform Choice*, kratica UC) - Svaka slijedeća pozicija u populaciji na kojoj će se jedinka zamijeniti, bira se slučajnim odabirom. To znači da zamjena svih jedinki nije zagarantirana, dok s druge strane postoji mogućnost zamjene pojedine pozicije više puta u jednoj generaciji.

4.1.4 Utjecaj mreže i susjedstva na selekcijski pritisak

Selekcijski pritisak ključan je čimbenik kojim se kod populacijskih algoritama može ugrubo postići odgovarajuća ravnoteža između procesa eksploatacije i istraživanja prostora rješenja [4]. Kod staničnih evolucijskih algoritama veće susjedstvo znači i veća preklapanja sa susjedstvima obližnjih pozicija što uz fiksno zadane dimenzije toroidalne mreže vodi ka bržoj propagaciji boljih jedinki. Razina selekcijskog pritiska može se utvrditi izvođenjem pojednostavljene verzije staničnog evolucijskog algoritma u kojem se operatori križanja i mutacije ne koriste, već se u iduću generaciju nepromijenjene kopiraju jedinke odabrane operatorom selekcije. Vrijeme preuzimanja (eng. *takeover time*) predstavlja broj generacija potreban da najbolja jedinka kloniranjem zauzme sva mjesta u populaciji. Što je kraće vrijeme preuzimanja to je veći selekcijski pritisak. Na slici 4.4 prikazan je selekcijski pritisak uz različite tipove susjedstva na mreži dimenzija 32x32.



Slika 4.4. *Utjecaj susjedstva na selekcijski pritisak u mreži dimenzija 32x32*

Jedinkama na slučajno odabranim pozicijama vrijednost dobrote postavljena je redom na vrijednosti [1..1024]. Kao operator selekcije korištena je binarna turnirska selekcija, a provedeno je 100 neovisnih testova za svaki tip susjedstva. Krivulje prikazuju prosječan udio najbolje jedinke u populaciji tijekom generacija. U radu [98], autori zaključuju da na selekcijski pritisak pored veličine (broja jedinki) bitno utječe i oblik susjedstva (raspored jedinki). To se najbolje može primijetiti na krivuljama tipova susjedstva L9 i C13. Iako je broj jedinki kod susjedstva C13 veći gotovo za 50% od broja jedinki kod tipa L9, selekcijski pritisci su im vrlo slični. Ta anomalija posebno dolazi do izražaja ako se usporedi razlika u selekcijskom pritisku između tipova susjedstva L9 i C9 koji imaju jednak broj jedinki. Ako se usporedi oblik tipova susjedstva L9 i C13 s tipom C9 može se pretpostaviti da je jačina selekcijskog pritiska u vezi s polumjerom najmanje kružnice koja opisuje susjedstvo. No budući da bi za neka susjedstva taj polumjer bio identičan (npr. L9 i C13), autori za potrebe utvrđivanja selekcijskog pritiska definiraju posebnu mjeru prostorne raspršenosti uzorka točaka nazvanu polumjer susjedstva:

$$rad = \sqrt{\frac{\sum (x_i - \bar{x})^2 + \sum (y_i - \bar{y})^2}{n}} \ , \tag{4.1}$$

gdje je:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \ , \tag{4.2}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \ , \tag{4.3}$$

Polumjer susjedstva *rad* polumjer je kružnice s centrom u točki $(\bar{x}, \bar{y})$, gdje su $\bar{x}$ i $\bar{y}$ prosječne x i y koordinate od n točaka koje sačinjavaju uzorak (susjedstvo). U tablici 4.1 prikazani su polumjeri za svaki navedeni tip susjedstva.

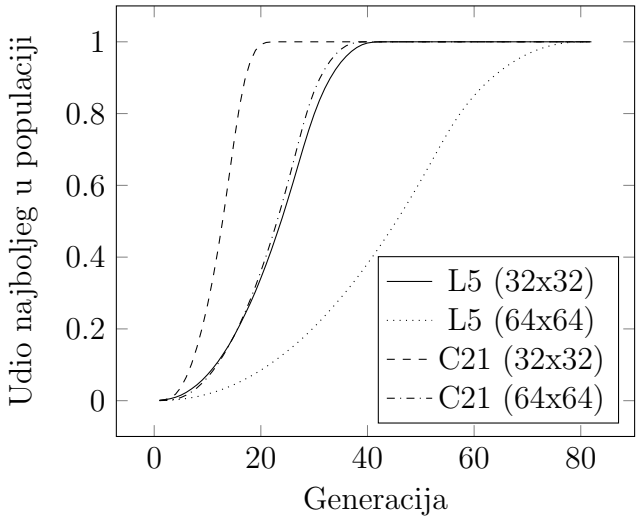
Tablica 4.1. *Polumjeri različitih tipova susjedstva*

Tip susjedstva	Polumjer
L5	0.8944
L9	1.4907
C9	1.1547
C13	1.4676
C21	1.7995
C25	2.0000

Nadalje, u istom radu autori zaključuju da je za kontrolu selekcijskog pritiska kod staničnog evolucijskog algoritma ključan omjer između polumjera susjedstva i dimenzija toroidalne mreže:

$$omjer_{cEA} = \frac{rad_{susjedstvo}}{rad_{mreza}} \ , \tag{4.4}$$

te kao primjer navode sličnost selekcijskog pritiska kod L5 tipa susjedstva na toroidalnoj mreži dimenzija 32x32 i C21 susjedstva na mreži dimenzija 64x64, što se može primijetiti na slici 4.5.



Slika 4.5. *Utjecaj dimenzija toroidalne mreže na selekcijski pritisak*

Omjer susjedstva L5 na mreži dimenzija 32x32 je 0.0685, a omjer susjedstva C21 na mreži dimenzija 64x64, vrlo sličnih 0.0689. Polumjer mreže izračunava se na isti način kao i polumjer susjedstva, putem izraza (4.1), s tim da se u obzir uzimaju koordinate svih pozicija u mreži. U tablici 4.2 prikazani su polumjeri za nekoliko različitih dimenzija kvadratne mreže.

Tablica 4.2. *Polumjeri različitih dimenzija kvadratne mreže*

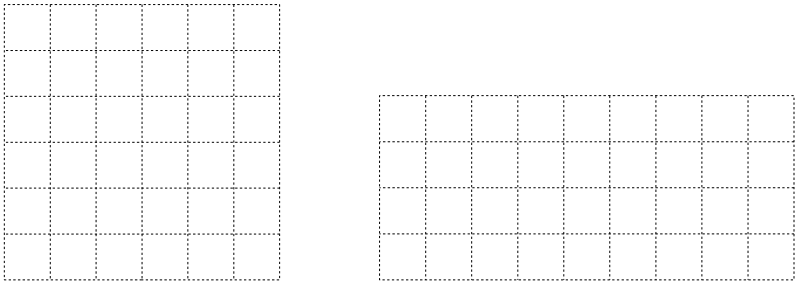
Dimenzije	Polumjer
8x8	3.2404
16x16	6.5192
32x32	13.0576
64x64	26.1247
128x128	52.2542

U tablici 4.3 prikazane su vrijednosti omjera polumjera svih tipova susjedstva i svih polumjera mreže kvadratnog oblika iz prethodne tablice.

Tablica 4.3. *Omjeri polumjera susjedstva i kvadratne mreže različitih dimenzija*

Tip susjedstva	8x8	16x16	32x32	64x64	128x128
L5	0.2760	0.1372	0.0685	0.0342	0.0171
L9	0.4600	0.2287	0.1142	0.0571	0.0285
C9	0.3563	0.1771	0.0884	0.0442	0.0221
C13	0.4529	0.2251	0.1124	0.0562	0.0281
C21	0.5553	0.2760	0.1378	0.0689	0.0344
C25	0.6172	0.3068	0.1532	0.0766	0.0383

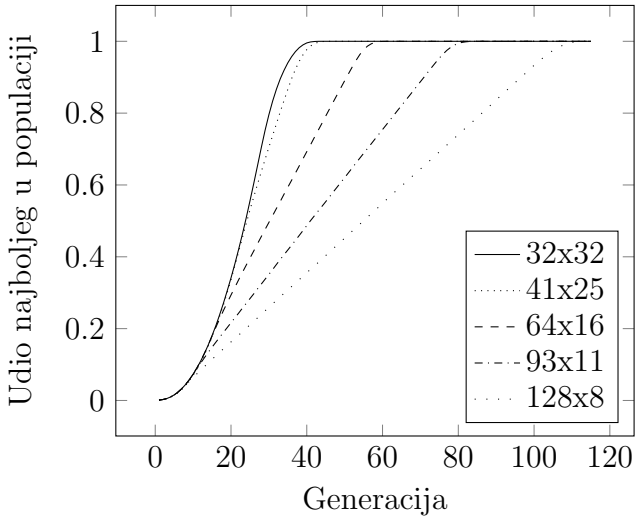
U radu [6], autori proširuju koncept omjera na toroidalne mreže koje mogu biti i pravokutnog oblika, te zaključuju da osim veličine i oblika susjedstva, te veličine mreže, oblik mreže također bitno utječe na kvalitetu pretrage prostora rješenja i selekcijski pritisak. Na slici 4.6 prikazana su dva različita oblika toroidalne mreže za populaciju od 36 jedinki kod kojih za susjedstvo L5 $omjer_{cEA}$ iznosi 0.3703, odnosno 0.3179.



(a) Mreža 6x6 (b) Mreža 9x4

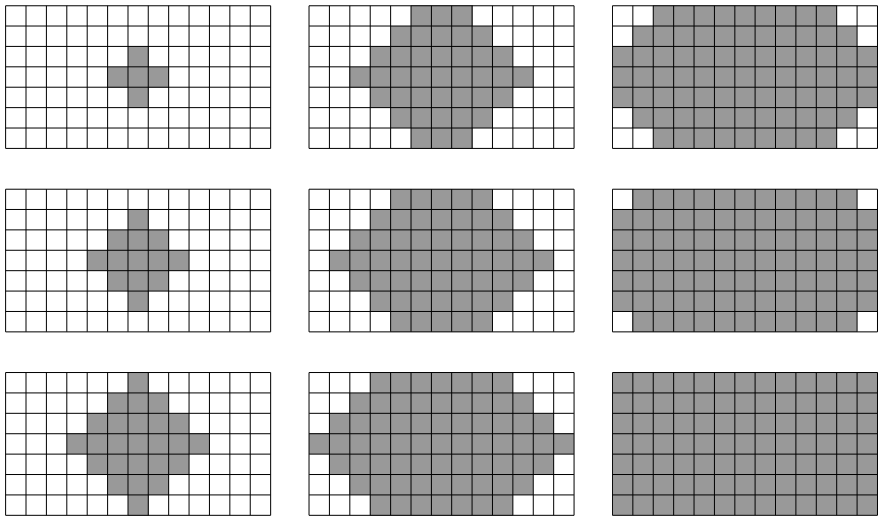
Slika 4.6. *Dva različita oblika mreže za populaciju veličine 36 jedinki*

Što je pravokutnik koji predstavlja mrežu uži, polumjer mreže biti će veći, a to znači da će $omjer_{cEA}$ u konačnici biti manji. Manji omjer znači i manji selekcijski pritisak što za posljedicu ima pojačano istraživanje prostora rješenja. S druge strane, jača eksploatacija vezana je uz veće omjere i mreže kvadratnog oblika. Ovu činjenicu autori koriste u radovima [6],[2] za dinamičko mijenjanje oblika mreže i omjera tijekom izvođenja algoritma s ciljem povećanja učinkovitosti pretrage prostora rješenja. Slika 4.7 prikazuje razlike u konvergenciji između mreže kvadratnog oblika i mreža pravokutnog oblika različitih dimenzija, a koje sadrže jednak ili približno jednak broj jedinki (u slučaju 41x25=1025 i 93x11=1023).



Slika 4.7. *Konvergencija kod kvadratnog i pravokutnog oblika mreže uz L5 susjedstvo*

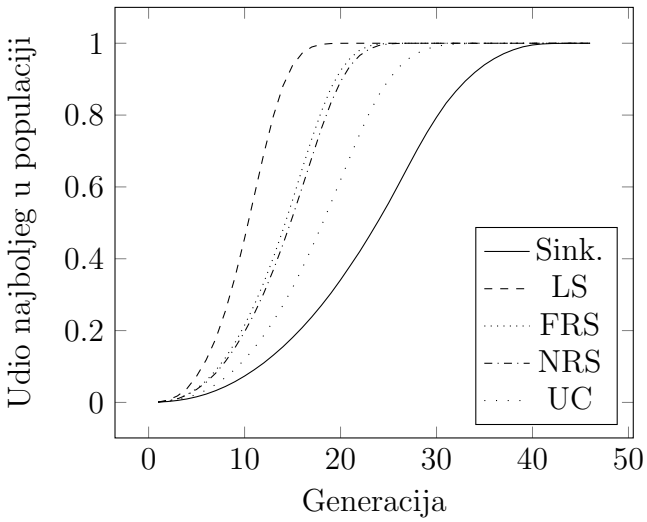
Kloniranje najbolje jedinke odvija se kvadratnom brzinom dok se ista ne proširi do rubova užeg dijela mreže, nakon čega se prostire linearnom brzinom do rubova šireg dijela mreže. Širenje se na kraju usporava i završava popunjavanjem preostalih pozicija koje se nalaze u kutovima mreže. Ove tri faze "preuzimanja populacije" vizualizirane su na slici 4.8, a matematička definicija krivulje rasta može se pronaći u radovima [58],[60] i knjizi [4].



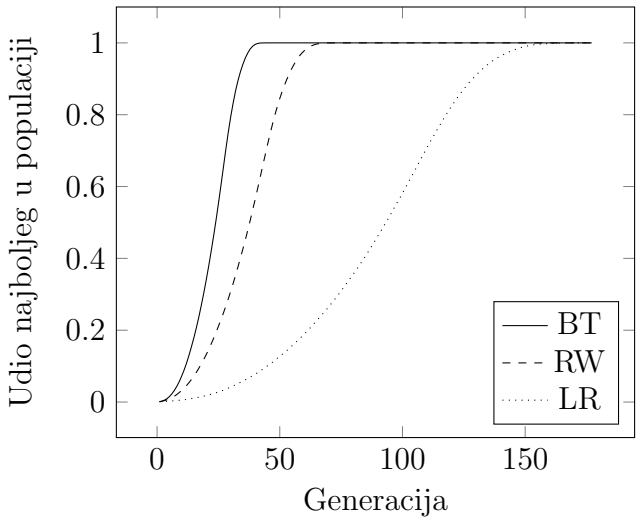
(a) Kvadratni rast (b) Linearni rast (c) Zasićenje

Slika 4.8. *Faze širenja najbolje jedinke kroz toroidalnu mrežu pravokutnog oblika*

Pored veličine i oblika toroidalne mreže, te veličine i oblika susjedstva, na selekcijski pritisak uvelike utječu i način zamjene jedinki u populaciji te korišteni operator selekcije. Slika 4.9 prikazuje utjecaj načina zamjene jedinki, a slika 4.10 utjecaj korištenog operatora selekcije (binarna turnirska selekcija - BT, selekcija kotačem ruleta - RW, linearno rangiranje - LR) na mreži veličine 32x32 uz tip susjedstva L5.



Slika 4.9. *Utjecaj načina zamjene jedinki na konvergenciju*



Slika 4.10. *Utjecaj operatora selekcije na konvergenciju*

Više informacija o utjecaju načina zamjene jedinki na selekcijski pritisak kod staničnih evolucijskih algoritama može se pronaći u radovima [5],[41],[58],[60].

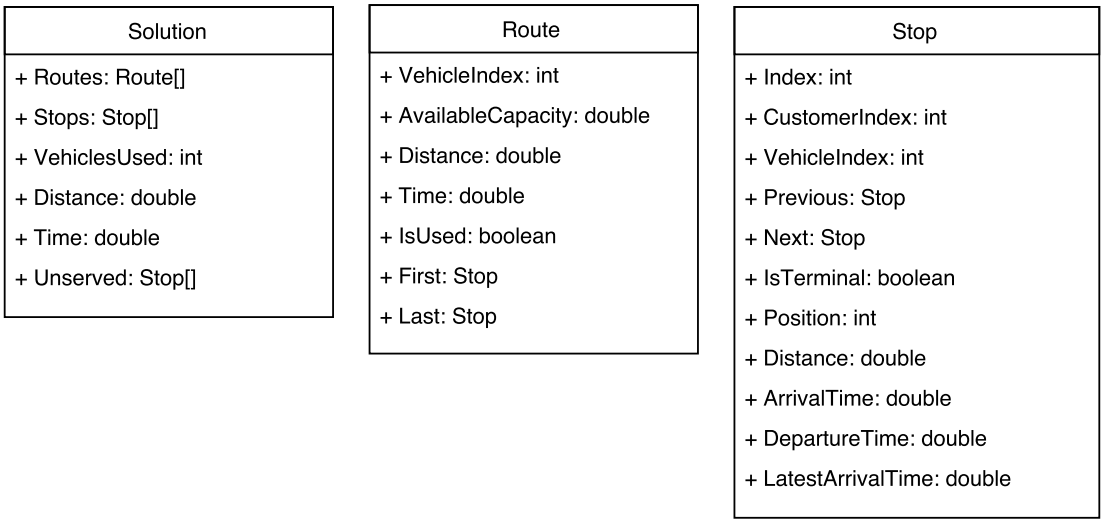
4.2 Implementacija algoritma

U dostupnoj literaturi, može se pronaći svega nekoliko radova u kojima se problem usmjeravanja vozila rješava pomoću staničnog evolucijskog algoritma. To je relativno malen broj u usporedbi s količinom radova koji su se proteklih deset godina objavili na temu rješavanja različitih varijanti ovoga problema evolucijskim algoritmima. U radovima [1],[3],[42],[85] autori prezentiraju hibridni cGA algoritam za rješavanje CVRP problema po učinkovitosti usporediv s ostalim *state-of-the-art* metodama rješavanja iz tog perioda, a autori u radu [70], cGA algoritam za rješavanje VRPTW problema. Iako su vrlo dobri, rezultate iz potonjeg treba uzeti s određenom dozom rezerve s obzirom da su prosječna rješenja ostvarena na pojedinim skupovima Solomonovih testnih zadataka bolja od prosjeka najboljih rješenja objavljenih na referentnoj web stranici za ovu vrstu problema [82]. Autori ne daju dovoljno informacija o implementaciji algoritma kao ni detaljna rješenja testnih zadataka, pa se njihova točnost ne može provjeriti. Uz iznimku rada [42], svi navedeni algoritmi rješenja kodiraju permutacijama i koriste operatore križanja i mutacije tipične za problem trgovačkog putnika. Implementacija evolucijskog algoritma za rješavanje CVRP problema ne može se izravno primijeniti i za rješavanje VRPTW problema zbog dvostrukog optimizacijskog cilja (broj vozila i ukupna udaljenost) te zbog vremenskih ograničenja koja često bivaju narušena primjenom klasičnih operatora varijacije. Evolucijski algoritmi za rješavanje VRPTW problema zbog toga često koriste specifične procedure za popravljanje rješenja ili specijalizirane operatore križanja i mutacija koji uvijek producirаju izvedivo rješenje, ili se pak u funkciji evaluacije za svako ograničenje koje nije zadovoljeno jedinka kažnjava, odnosno njezina dobrota uvećava po nekom kriteriju. Jedan od uvjeta postavljenih pred ovdje predloženi algoritam je taj da bude sposoban učinkovito rješavati i CVRP i VRPTW probleme bez posebnih prilagodbi i modifikacija, te da se na jednostavan način može proširiti dodatnim ograničenjima potrebnim za rješavanje ostalih varijanti VRP problema i problema iz prakse. Zbog toga je u ovom istraživanju kao glavni mehanizam varijacije odabran princip uništavanja i popravljanja, odnosno postupak pretraživanja velikog susjedstva opisan u prethodnom poglavlju. Ovakav koncept sa sobom nosi različite implikacije na implementaciju cEA algoritma koja odstupa od kanonskog oblika te mu time daje atribut "hibridni".

4.3 Kodiranje jedinki

Odabir principa djelomičnog uništavanja i popravljanja kao osnovnog mehanizma varijacije bitno je utjecao na način kodiranja i evaluacije rješenja u ovdje implementiranom algoritmu. Budući da se nakon djelomičnog uništavanja rješenja postupkom popravljanja ono ne može uvijek u cijelosti popraviti, u populaciji mogu egzistirati i nepotpuna rješenja. Rješenja ostaju nepotpuna kada ograni-

čenja ne dozvoljavaju umetanje preostalih korisnika u postojeće rute. Na ovaj način osigurano je da su sva ograničenja kapaciteta i vremenskih prozora uvijek zadovoljena, i kod potpunih i kod nepotpunih rješenja. Rješenje problema predstavlja jednu jedinku u toroidalnoj mreži te sadrži sve bitne informacije o korištenim vozilima i njihovim rutama, pređenoj udaljenosti, raspoloživom kapacitetu, vremenima dolazaka kod korisnika i odlazaka od njih, itd. Sve operacije križanja i mutacija provode se izravno na rješenjima čime se izbjegava eksplicitno kodiranje u kromosome. Sličan pristup koriste i autori u radovima [69],[12],[13]. Dok je kod kodiranja rješenja permutacijom najčešće potrebno evaluirati cijeli kromosom nakon primjene operatora varijacije, izravna manipulacija rješenjima omogućuje re-evaluaciju samo dijelova rješenja koja su pogođena promjenama što pozitivno utječe na performanse algoritma. Na slici 4.11 prikazan je dijagram klasa kojima se opisuje jedna jedinka ili rješenje problema.



Slika 4.11. *Dijagram klasa koje se koriste za pohranu rješenja*

Objekt klase *Solution* pored informacija o broju korištenih vozila, ukupnoj pređenoj udaljenosti i utrošenom vremenu, sadrži skup ruta (klasa *Route*), skup svih lokacija koje se mogu pojaviti u rutama (klasa *Stop*), te skup neposluženih korisnika. Objekt klase *Route* predstavlja rutu određenog vozila, a sadrži podatke o raspoloživom kapacitetu, pređenoj udaljenosti, utrošenom vremenu, te početnoj i završnoj lokaciji. Završna lokacija predstavlja dodatnu kopiju skladišta. Rute se pohranjuju koristeći podatkovnu strukturu povezanih listi. Od početne (atribut *First*) ili završne (atribut *Last*) lokacije u ruti, može se proći čitavom rutom uz pomoć atributa *Previous* i *Next* iz klase *Stop*. Svaka lokacija sadrži informacije o ruti u kojoj se nalazi i poziciji unutar nje, vremenu dolaska i odlaska, te najkasnijem mogućem vremenu dolaska s obzirom na vremenska ograničenja nadolazećih lokacija u ruti. Posljednja informacija koristi se za ubrzavanje postupka umetanja neposluženih korisnika u rute.

4.4 Inicijalizacija populacije

Populacija je strukturirana kao matrica $M(w, h)$ gdje w predstavlja širinu odnosno broj stupaca, a h visinu ili broj redaka u matrici. Prije početka simulacije procesa evolucije za svako polje konstruira se početno rješenje koristeći modificiranu varijantu vremenski-orijentirane heuristike najbližeg susjeda (eng. *Time-Oriented Nearest Neighborhood Heuristics*, kratica TONNH) [102],[53]. Ova konstruktivna heuristika u stanju je za kratko vrijeme konstruirati velik broj različitih rješenja problema, a dodatna prednost joj je mogućnost određivanja utjecaja vremena nauštrb udaljenosti prilikom konstrukcije ruta uz pomoć nekoliko parametara. To je čini pogodnom za konstruiranje rješenja za VRPTW ali i za CVRP probleme. Postupak konstruiranja jednog rješenja prikazan je algoritmom 6.

Algoritam 6 Vremenski orijentirana heuristika najbližeg susjeda

```
1: procedure TONNH(δ1, δ2, δ3, η)
2:     v ← NekoristenoVozilo()
3:     repeat
4:         c ← NerutiraniKorisnik(δ1, δ2, δ3, η)
5:         if c = −1 then
6:             VratiUSkladiste(v)
7:             v ← NekoristenoVozilo()
8:         else
9:             PosluziKorisnika(v, c)
10:        end if
11:    until SviPosluzeni()
12: end procedure
```

Rute se konstruiraju serijski, jedna za drugom, dodavanjem korisnika u započetu rutu dok god to ograničenja kapaciteta i vremenskih prozora dozvoljavaju. Ulazni parametri $\delta_1, \delta_2, \delta_3, \eta$ koji se prosljeđuju funkciji *NerutiraniKorisnik* kontroliraju postupak odabira svakog slijedećeg korisnika. Kako bi bio uzet u obzir kao kandidat za dodavanje u rutu iza korisnika i, nerutirani korisnik j mora zadovoljiti ograničenje kapaciteta vozila:

$$q_j \leq Q - \sum_{i=1}^{n} q_i \ , \tag{4.5}$$

gdje je q količina tereta koju korisniku treba dostaviti, Q kapacitet vozila čija se ruta trenutno konstruira, a n broj korisnika u ruti. Nadalje, potrebno je zadovoljiti vremensko ograničenje:

$$t_{a_j} \leq t_{l_j} \ , \tag{4.6}$$

gdje je t_{l_j} najkasnije dozvoljeno vrijeme dolaska kod korisnika j, dok t_{a_j} predstavlja vrijeme dolaska ako bi se do korisnika j došlo izravno iz i:

$$t_{a_j} = t_{b_i} + t_{s_i} + t_{ij} \ . \tag{4.7}$$

Vrijeme početka posluživanja korisnika t_{b_i} jednako je vremenu dolaska t_{a_i} u slučaju da je vozilo stiglo nakon otvarnja vremenskog prozora t_{e_i}, ili vremenu otvaranja ako je stiglo ranije:

$$t_{b_i} = \max(t_{a_i}, t_{e_i}) \ . \tag{4.8}$$

t_{s_i} predstavlja trajanje posluživanja, a t_{ij} vrijeme putovanja između i i j. Posljednji uvjet koji treba zadovoljiti je povratak u skladište prije njegova zatvaranja:

$$t_{b_j} + t_{s_j} + t_{j0} \leq t_{l_0} \ . \tag{4.9}$$

Od svih kandidata koji zadovoljavaju navedena ograničenja, za dodavanje iza posljednjeg korisnika u ruti i odabrati će se korisnik j koji minimizira funkciju:

$$c_{ij} = \delta_1 \cdot max(0, d_{ij} + \lambda) + \delta_2 T_{ij} + \delta_3 v_{ij} \ , \tag{4.10}$$

gdje d_{ij} predstavlja udaljenost između korisnika i i j, λ stohastičku komponentu ili pogrešku kojom se ta udaljenost modificira, T_{ij} vremenski razmak između početka posluživanja korisnika j i završetka posluživanja korisnika i, a v_{ij} vrijeme preostalo do zatvaranja vremenskog prozora t_{l_j}:

$$\lambda = \eta d_{max} \cdot (-1 + 2 \cdot rnd()) \ , \tag{4.11}$$

$$T_{ij} = t_{b_j} - (t_{b_i} + t_{s_i}) \ , \tag{4.12}$$

$$v_{ij} = t_{l_j} - (t_{b_i} + t_{s_i} + t_{ij}) \ . \tag{4.13}$$

Maksimalan iznos pogreške kojom se udaljenost modificira, određen je parametrom η koji predstavlja postotak najveće udaljenosti između dva korisnika u problemu, d_{max}, čime se iznos pogreške grubo prilagođava problemu koji se rješava. Prilikom svake evaluacije izraza (4.11), funkcija $rnd()$ generira pseudo-slučajnu vrijednost iz intervala $[0..1\rangle$ što u postupak unosi stohastičnost i osigurava da uzastopnim izvršavanjem algoritam konstruira različito rješenje. Koeficijenti δ_1, δ_2, δ_3 koriste se za dodjeljivanje težine svakoj od tri komponente funkcije c_{ij} i vrijedi:

$$\delta_1 + \delta_2 + \delta_3 = 1 \ , \tag{4.14}$$

$$\delta_1 \geq 0, \delta_2 \geq 0, \delta_3 \geq 0 \ . \tag{4.15}$$

Kod rješavanja VRPTW problema, konstruiranje početnog rješenja za svaku poziciju u toroidalnoj matrici koristi slučajno generirane vrijednosti parametara $\delta_1, \delta_2, \delta_3$ čiji je zbroj jednak 1. Kod rješavanja CVRP problema uzimaju se vrijednosti $\delta_1 = 1, \delta_2 = 0, \delta_3 = 0$ s obzirom da vremenska ograničenja nisu zadana i ovdje se tretiraju kao interval $[0..\infty]$. Vrijednost parametra η utvrđena je empirijski i iznosi 0.01. U trenutku kada se nijedan od preostalih korisnika ne može dodati u rutu zbog ograničenja kapaciteta ili vremenskih prozora (uvjet $c = -1$, redak 5 algoritma 6), ruta se zatvara vraćanjem vozila u skladište te se aktivira novo vozilo. Naposljetku, kada se svi korisnici dodaju u rute postupak konstruiranja rješenja završava.

4.5 Evolucija u dvije faze

Za razliku od rješavanja CVRP problema gdje je cilj minimizirati ukupnu pređenu udaljenost, kod rješavanja VRPTW problema primarni je cilj minimizirati broj vozila a tek potom minimizirati ukupnu udaljenost. U ovdje predloženoj implementaciji HCEA algoritma CVRP problemi transformiraju se u VRPTW probleme postavljanjem vremenskih prozora svim korisnicima i skladištu na interval $[0..+\infty]$. Rješavanje ovako transformiranih CVRP problema olakšava činjenica da se minimalan broj vozila može odrediti unaprijed pomoću izraza (2.7). Kod VRPTW problema, pronalazak izvedivog rješenja s minimalnim brojem vozila predstavlja NP-potpun problem te je minimalan broj vozila subjekt optimizacije. Vrednovanje dobrote rješenja ili jedinke može se provoditi leksikografski gdje se boljim rješenjem smatra ono u kojem se koristi manji broj vozila, ili, u slučaju da rješenja koriste jednak broj vozila, ono kod kojeg je ukupna udaljenost manja. Alternativno, u funkciji dobrote ukupna udaljenost se može zbrojiti s produktom broja korištenih vozila i konstante koja je za red veličine veća od udaljenosti. Iako se čini logičnim vrednovati rješenja VRPTW problema na ovaj način, minimizacija udaljenosti koja predstavlja sekundarni cilj optimizacije pretragu često usmjerava prema područjima u kojima se nalaze rješenja koja koriste broj vozila veći od optimalnog. Drugim riječima, da bi se problem riješio uz manji broj vozila ponekad je potrebno prevaliti veću udaljenost, a takva rješenja mogu se nalaziti jako daleko od područja prema kojima pretraga vodi minimiziranjem ukupne udaljenosti. Brojna istraživanja pokazala su da su najučinkovitiji algoritmi koji postupak rješavanja VRPTW problema dijele u dvije faze gdje se u prvoj pokušava pronaći rješenje s minimalnim brojem vozila, a potom u drugoj minimizirati

ukupna pređena udaljenost [67],[11][91],[87],[88],[16]. Često se pri tome primjenjuju potpuno različiti algoritmi za prvu i drugu fazu. I u ovom istraživanju, implementirani hibridni stanični evolucijski algoritam pretragu prostora rješenja dijeli u dvije faze. Jedna od karakteristika HCEA algoritma je ta da se tijekom čitavog trajanja postupka broj korištenih vozila održava jednakim kod svih jedinki u populaciji. U fazi minimizacije broja vozila, pokušava se pronaći potpuno rješenje koje koristi jedno vozilo manje od najboljeg potpunog rješenja koje je pronađeno do tog trenutka. Svaki put kada se u tome uspije, rješenja iz populacije djelomično se uništavaju brisanjem ruta kako bi se broj korištenih vozila uskladio s ciljanim koji je ponovo za jedan manji od najboljeg pronađenog rješenja. U svakom rješenju koje ima višak ruta briše se ona s najmanjim brojem korisnika, a u slučaju da je takvih više, briše se ona čiji se korisnici po statistici lakše umeću u druge rute. U fazu minimizacije ukupne udaljenosti prelazi se nakon protoka trećine od maksimalnog broja generacija ili nakon što se pronađe potpuno rješenje koje koristi teoretski minimalan broj vozila (gledano po kapacitetu i ukupnoj potražnji). Prilikom prelaska u drugu fazu, broj korištenih vozila konačno se fiksira postavljanjem na broj vozila koji se koristi u najboljem rješenju iz prve faze. Razmotrene su dvije inačice algoritma, sa sinkronom i asinkronom zamjenom jedinki. Kod inačice sa sinkronom zamjenom jedinki, jedinke iz aktivne populacije bivaju zamijenjene novima tek nakon što se obavi kompletan reproduktivni ciklus jedne generacije. Kod asinkrone inačice algoritma, jedinke se zamjenjuju tijekom trajanja jednog reproduktivnog ciklusa. Načini asinkrone zamjene mogu biti različiti i ovdje su razmotrene četiri varijante. Algoritam 7 prikazuje postupak rješavanja sa sinkronom zamjenom jedinki, a algoritam 8, postupak s asinkronom zamjenom jedinki. Ulazni parametri obaju algoritama prikazani su u tablici 4.4. U nastavku su detaljnije opisane obje inačice algoritma.

Tablica 4.4. *Ulazni parametri HCEA algoritma*

Parametar	Opis
g_{max}	maksimalan broj iteracija glavne petlje
w	širina toroidalne matrice
h	visina toroidalne matrice
N_t	tip susjedstva koje koristi operator selekcije
p_x	učestalost primjene operatora križanja
p_m	učestalost primjene operatora mutacije
x_{rc}	maksimalan broj korisnika koji se izbacuje kod križanja
m_{rc}	maksimalan broj korisnika koji se izbacuje kod mutacija
E	primjena načela elitizma
R_t	način zamjene jedinki u populaciji[1]

[1]Koristi se samo u asinkronoj inačici algoritma

4.5.1 Sinkrona inačica HCEA algoritma

Varijable v_{min} i v_{best} koriste se u fazi minimizacije broja vozila. Na početku postupka, varijabla v_{min} inicijalizira se funkcijom *NajmanjiMoguciBrojVozila* koja pomoću izraza (2.7) izračunava minimalan broj vozila potreban da se problem riješi s obzirom na kapacitet vozila i ukupnu potražnju korisnika. Taj minimum neće biti moguće uvijek postići zbog vremenskih ograničenja, no u slučajevima kada se postigne, algoritam će znati da odmah može nastaviti s drugom fazom i minimizacijom ukupne udaljenosti. Varijabla v_{best} predstavlja najmanji broj vozila uz koji je algoritam pronašao izvedivo i potpuno rješenje. Populacija P implementirana je kao dvodimenzionalna matrica koja ima w stupaca h redaka. Elementi matrice predstavljaju individualna rješenja problema ili jedinke, a inicijaliziraju se prethodno opisanom modificiranom varijantom vremenski orijentirane konstruktivne heuristike. Najbolje rješenje pohranjuje se u varijablu s_{best} (linija 6). U slučaju da su u najboljem rješenju posluženi svi korisnici, vrijednost varijable v_{best} se postavlja na broj vozila koji se koristi u s_{best} (linija 8). Funkcijom *FiksirajBrojVozila* svim rješenjima u populaciji briše se određen broj ruta kako bi broj korištenih vozila bio za jedan manji od v_{best} ili minimalan, ovisno o tome je li odmah na početku pronađeno rješenje koje koristi v_{min} vozila (linija 9). Navedena funkcija redom briše rute koje sadrže najmanje korisnika, a izbačeni korisnici se nakon brisanja pokušavaju umetnuti u preostale rute procedurom popravljanja koja je opisana u nastavku. Ukoliko najbolje rješenje iz početne populacije koristi minimalan broj vozila, algoritam odmah prelazi u fazu minimizacije ukupne udaljenosti (linije 10-12). U svakoj iteraciji glavne petlje priprema se matrica P' u koju će privremeno biti spremljene jedinke koje nastanu u reproduktivnom ciklusu trenutne generacije. Reprodukcija se obavlja unutar dvije ugniježđene petlje kojima se prolazi svaka pozicija u matrici (linije 16-31). Operator križanja primjenjuje se s vjerojatnošću p_x (linije 19-22). Funkcijom $Selektiraj$, iz susjedstva pozicije (x, y) odabiru se dvije jedinke koje će imati ulogu roditelja pri čemu se matrica P tretira kao toroidalna mreža. Oblik i veličina susjedstva zadani su parametrom N_t. Operacija križanja biti će uspješna samo ako roditelji predstavljaju različita rješenja problema. Detaljno objašnjenje postupka križanja dato je u nastavku. U slučaju da potomak nije generiran operatorom križanja, jedinka s pozicije (x, y) se klonira (linije 23-25), nakon čega se s vjerojatnošću p_m primjenjuje operator mutacije. Nova jedinka sprema se u zamjensku populaciju P', a postupak se ponavlja dok se ne obrade sve pozicije u matrici. Nakon što reproduktivni ciklus završi, najbolja jedinka s' iz populacije P' uspoređuje s najboljim pronađenim rješenjem s_{best} i ono se po potrebi ažurira (linije 32-35). Funkcija *Zamijeni* obavlja zamjenu jedinki iz aktivne populacije P jedinkama iz populacije P' koje se nalaze na istim pozicijama u matrici (linija 36). Postupak zamjene bitno utječe na ponašanje algoritma i detaljnije je opisan u nastavku. Dok traje faza minimizacije broja vozila, na kraju svake generacije provjerava se je li pronađeno potpuno rješenje (linije 38-44). Ako je to slučaj, provjerava

Algoritam 7 Sinkroni hibridni stanični evolucijski algoritam

```
1: procedure HCEA_sync(g_max, w, h, N_t, p_x, p_m, x_rc, m_rc, E)
2:     faza ← 1
3:     v_min ← NajmanjiMoguciBrojVozila()
4:     v_best ← UkupanBrojVozila()
5:     P ← GenerirajPocetnaRjesenja(w, h)
6:     s_best ← Najbolji(P)
7:     if BrojNeposluzenih(s_best) = 0 then
8:         v_best ← BrojVozila(s_best)
9:         FiksirajBrojVozila(P, max(v_min, v_best − 1))
10:        if v_best = v_min then
11:            faza ← 2
12:        end if
13:    end if
14:    for g = 1 → g_max do
15:        P' = PripremiMatricuZaNovuGeneraciju(w, h)
16:        for x = 0 → w − 1 do
17:            for y = 0 → h − 1 do
18:                potomak ← ∅
19:                if rnd() < p_x then
20:                    roditelji ← Selektiraj(P, x, y, N_t)
21:                    potomak ← Krizaj(roditelji[0], roditelji[1], x_rc)
22:                end if
23:                if potomak = ∅ then
24:                    potomak ← Kloniraj(P[x][y])
25:                end if
26:                if rnd() < p_m then
27:                    Mutiraj(potomak, m_rc)
28:                end if
29:                P'[x][y] ← potomak
30:            end for
31:        end for
32:        s' ← Najbolji(P')
33:        if Bolje(s', s_best) then
34:            s_best ← s'
35:        end if
36:        Zamijeni(P, P')
37:        if faza = 1 then
38:            if BrojNeposluzenih(s') = 0 then
39:                v_best ← BrojVozila(s')
40:                if v_best = v_min then
41:                    faza ← 2
42:                else
43:                    FiksirajBrojVozila(P, v_best − 1)
44:                end if
45:            else if g ≥ g_max/3 then
46:                faza ← 2
47:                FiksirajBrojVozila(P, v_best)
48:            end if
49:        else if E = true then
50:            ZamijeniNajgoreg(P, s_best)
51:        end if
52:    end for
53:    return s_best
54: end procedure
```

Algoritam 8 Asinkroni hibridni stanični evolucijski algoritam

procedure $HCEA_{async}(g_{max}, w, h, N_t, p_x, p_m, x_{rc}, m_{rc}, E, R_t)$
  $faza \leftarrow 1$
  $v_{min} \leftarrow NajmanjiMoguciBrojVozila()$
  $v_{best} \leftarrow UkupanBrojVozila()$
  $P \leftarrow GenerirajPocetnaRjesenja(w, h)$
  $s_{best} \leftarrow Najbolji(P)$
  if $BrojNeposluzenih(s_{best}) = 0$ **then**
    $v_{best} \leftarrow BrojVozila(s_{best})$
    $FiksirajBrojVozila(P, max(v_{min}, v_{best} - 1))$
    if $v_{best} = v_{min}$ **then**
      $faza \leftarrow 2$
    end if
  end if
  $r \leftarrow \{0, 1, 2, 3, ..., |P| - 1\}$
  if $R_t = 1$ **or** $R_t = 2$ **then**
    $Promijesaj(r)$
  end if
  for $g = 1 \rightarrow g_{max}$ **do**
    for $i = 0 \rightarrow |P| - 1$ **do**
      if $R_t = 3$ **then**
        $x \leftarrow rnd(w)$
        $y \leftarrow rnd(h)$
      else
        $x \leftarrow r[i]$ **mod** w
        $y \leftarrow r[i]$ **div** w
      end if
      $potomak \leftarrow \varnothing$
      if $rnd() < p_x$ **then**
        $roditelji \leftarrow Selektiraj(P, x, y, N_t)$
        $potomak \leftarrow Krizaj(roditelji[0], roditelji[1], x_{rc})$
      end if
      if $potomak = \varnothing$ **then**
        $potomak \leftarrow Kloniraj(P[x][y])$
      end if
      if $rnd() < p_m$ **then**
        $Mutiraj(potomak, m_{rc})$
      end if
      if $Bolje(potomak, s_{best})$ **then**
        $s_{best} \leftarrow potomak$
      end if
      $Zamijeni(P, x, y, potomak)$
    end for
    if $faza = 1$ **then**
      $s' \leftarrow Najbolji(P)$
      if $BrojNeposluzenih(s') = 0$ **then**
        $v_{best} \leftarrow BrojVozila(s')$
        if $v_{best} = v_{min}$ **then**
          $faza \leftarrow 2$
        else
          $FiksirajBrojVozila(P, v_{best} - 1)$
        end if
      else if $g \geq g_{max}/3$ **then**
        $faza \leftarrow 2$
        $FiksirajBrojVozila(P, v_{best})$

```
55:             end if
56:         else if E = true then
57:             ZamijeniNajgoreg(P, s_best)
58:         end if
59:         if R_t = 2 then
60:             Promijesaj(r)
61:         end if
62:     end for
63:     return s_best
64: end procedure
```

se je li dosegnut minimalan broj vozila v_{min} i ukoliko jeste, algoritam prelazi u drugu fazu. U protivnom, funkcijom *FiksirajBrojVozila* svim se jedinkama broj ruta ponovo umanjuje za jedan i algoritam će u slijedećoj generaciji nastaviti s minimizacijom broja vozila (linija 43). Trajanje prve faze ograničeno je na trećinu od ukupnog broja generacija g_{max}. Ako prva faza završi istekom potrebnog broja generacija, prije nastavka s drugom fazom i minimizacijom ukupne udaljenosti potrebno je fiksirati broj vozila svim jedinkama na v_{best} budući da su u tom trenutku u populaciji sve jedinke nepotpune i koriste $v_{best} - 1$ vozila (linije 45-48). Načelo elitizma ili zamjena najgore jedinke najboljom na kraju svake generacije, primjenjuje se samo u drugoj fazi ako je logički parametar E istinit (linije 49-51). Nakon što se glavna petlja izvrši g_{max} puta, postupak završava i vraća najbolje pronađeno rješenje koje je spremljeno u varijabli s_{best}.

4.5.2 Asinkrona inačica HCEA algoritma

Kod asinkrone inačice, jedinke se mijenjaju tijekom trajanja reproduktivnog ciklusa jedne generacije. Algoritam 8 ima dodatni parametar R_t koji predstavlja metodu zamjene jedinki u populacijskoj matrici. Razmotrene su četiri varijante opisane u uvodnom dijelu poglavlja, pa parametar može imati jednu od vrijednosti prikazanih u tablici 4.5.

Tablica 4.5. *Dozvoljene vrijednosti parametra R_t*

R_t	Opis	Kratica
0	linearni prolaz	LS
1	fiksno zadani nasumični prolaz	FRS
2	novi nasumični prolaz	NRS
3	jednoliki odabir	UC

Linije 1-13 algoritma identične su kao kod sinkrone inačice. U liniji 14, inicijalizira se cjelobrojno polje r čija veličina odgovara broju jedinki u populaciji P. Polje predstavlja redoslijed pozicija po kojem će jedinke u matrici prolaziti

postupak reprodukcije i ažurirati je ako se koristi LS, FRS ili NRS metoda zamjene. Svaki element polja je pozicija u dvodimenzionalnoj matrici zapisana u linearnom obliku. U slučaju da se koristi FRS ili NRS metoda zamjene, elementi polja miješaju se funkcijom *Promijesaj* s ciljem dobivanja nasumično složenog redoslijeda (linije 15-17). Ako se koristi metoda zamjene jedinki linearnim prolazom (LS), inicijalni redoslijed ostati će nepromijenjen do kraja postupka, dok se kod zamjene jednolikim odabirom (UC) redoslijed neće ni koristiti. U svakoj iteraciji glavne petlje, reproduktivna petlja izvršit će se $|P|$ puta (linije 19-42). Ukoliko se koristi metoda zamjene jedinki jednolikim odabirom, pozicija (x, y) se generira slučajnim odabirom (linije 21-22), dok se u protivnom uzima trenutni element iz polja r na osnovu kojega se operacijom ostatka i cjelobrojnog dijeljenja linearna pozicija transformira u (x, y) (linije 24-25). Slijede linije 27-37 u kojima se odvija reprodukcija na isti način kao i kod sinkrone inačice algoritma. Ažuriranje najboljeg rješenja obavlja se unutar reproduktivnog ciklusa s obzirom da se ovdje ne koristi pomoćna matrica za pohranu novonastalih jedinki (linije 38-40). Isto vrijedi i za zamjenu stare jedinke novom (linija 41). Kriteriji po kojima se potomak prihvaća ili odbacuje opisani su u nastavku. Linije 43-55 odnose se na fazu minimizacije broja vozila, a linije 56-58 na primjenu načela elitizma. Objašnjenje dato u opisu sinkrone inačice algoritma vrijedi i ovdje. Na kraju svake generacije, u slučaju da se koristi metoda zamjene jedinki NRS, redoslijed r se ponovo miješa. Nakon isteka zadanog broja generacija g_{max}, algoritam vraća najbolje pronađeno rješenje s_{best}.

4.6 Evaluacija jedinki

Može se primijetiti da u algoritmima 7 i 8 nema posebnih naredbi koje bi inicirale evaluaciju jedinki kao što je to uobičajeno u implementacijama genetskih i evolucijskih algoritama. Kako je ranije spomenuto, jedinke ili rješenja problema nisu zapisane na klasičan način kao kromosomi, već se za njihovu pohranu koriste složene podatkovne strukture prikazane na slici 4.11. Sve operacije koje modificiraju jedinku poput križanja i mutacija, provode se izravno na rješenjima ažurirajući pri tome sve relevantne informacije koje su pogođene promjenama (kapacitet vozila, vremena dolazaka i odlazaka, listu neposluženih korisnika, udaljenost i dr). S obzirom da jedinke mogu predstavljati nepotpuna rješenja, potrebno je na određeni način napraviti razliku u vrednovanju potpunih i nepotpunih rješenja. Kažnjavanje nepotpunih jedinki uvećavanjem dobrote za red ili nekoliko redova veličine ovdje se nije pokazalo plodonosnim jer potencijalno dobre jedinke koje mogu postati potpune u nekoj od slijedećih generacija vrlo brzo iščezavaju iz populacije. Zbog toga se umjesto izračunavanja apsolutne vrijednosti dobrote numeričkom funkcijom, tijekom čitavog trajanja algoritma koristi leksikografska usporedba parova jedinki pri čemu se uspoređuje broj neposluženih korisnika, broj korištenih vozila, ukupna udaljenost te složenost popravljanja nepotpunog rješenja.

Algoritam 9 Funkcija za usporedbu dobrote dvaju jedinki

```
1: procedure BOLJE(s_1, s_2)
2:     u_1 ← BrojNeposluzenih(s_1)
3:     u_2 ← BrojNeposluzenih(s_2)
4:     if u_1 = 0 and u_2 = 0 then
5:         v_1 ← BrojVozila(s_1)
6:         v_2 ← BrojVozila(s_2)
7:         if v_1 < v_2 then
8:             return true
9:         else if v_1 = v_2 then
10:            d_1 ← Udaljenost(s_1)
11:            d_2 ← Udaljenost(s_2)
12:            if d_1 < d_2 then
13:                return true
14:            else
15:                return false
16:            end if
17:        else
18:            return false
19:        end if
20:    else
21:        if u_1 = 0 then
22:            return true
23:        else if u_2 = 0 then
24:            return false
25:        end if
26:        z_1 ← ZbrojNeuspjelihUbacivanja(s_1)
27:        z_2 ← ZbrojNeuspjelihUbacivanja(s_2)
28:        if z_1 < z_2 then
29:            return true
30:        else if z_1 = z_2 then
31:            if u_1 < u_2 then
32:                return true
33:            else if u_1 = u_2 then
34:                k_1 ← MinimalnoKasnjenje(s_1)
35:                k_2 ← MinimalnoKasnjenje(s_2)
36:                if k_1 < k_2 then
37:                    return true
38:                else
39:                    return false
40:                end if
41:            else
42:                return false
43:            end if
44:        else
45:            return false
46:        end if
47:    end if
48: end procedure
```

Postupak usporedbe prikazan algoritmom 9 rezultirati će istinom samo ako je rješenje s_1 bolje od rješenja s_2 prema postavljenim kriterijima. Posebno se tretira slučaj kada su oba rješenja potpuna (linije 4-19). Rješenje s_1 se smatra boljim ako koristi manje vozila od s_2 (linije 7-8), odnosno ako je ukupna pređena udaljenost manja u slučaju da je broj korištenih vozila jednak (linije 9-16). Ovakav način vrednovanja u potpunosti odgovara minimizaciji primarnog i sekundarnog cilja optimizacije kod VRPTW problema. Ako je s_1 potpuno rješenje, a s_2 nepotpuno, kao bolje uzima se s_1 (linije 21-25). U slučaju da su oba rješenja nepotpuna, uspoređuje se složenost popravljanja rješenja (linije 26-46). Prvi kriterij koji se pri tome razmatra je zbroj svih neuspjelih pokušaja ubacivanja neposluženih korisnika (funkcija *ZbrojNeuspjelihUbacivanja*) koji se izračunava pomoću izraza:

$$z = \sum_{i=1}^{|U|} I(U_i) \ , \tag{4.16}$$

gdje je U lista neposluženih korisnika u rješenju, a I statistički podatak koji predstavlja ukupan broj neuspjelih pokušaja ubacivanja korisnika U_i u rute (linije 26-27). Na osnovu ovog podatka grubo se procjenjuje složenost popravljanja rješenja odnosno "težina" umetanja svih korisnika iz liste U. Primjerice, ponekad će kroz par generacija biti lakše popraviti rješenje koje ima nekoliko neposluženih korisnika za koje statistika pokazuje da se lako umeću u rute, nego rješenje s jednim "zahtjevnijim" korisnikom kojega je često nemoguće ubaciti u rute. Atribut "zahtjevan" obično imaju korisnici s bitno većom potražnjom za dostavu od ostalih korisnika, prostorno izolirani korisnici ili pak korisnici s vrlo uskim vremenskim prozorima. Brojač neuspjelih umetanja ažurira se za svakog neposluženog korisnika na kraju procedure za popravljanje rješenja. Rješenje s_1 smatra se boljim ako ima manji zbroj neuspjelih pokušaja umetanja neposluženih korisnika (linije 28-29). Ako je zbroj jednak kod oba rješenja, kao bolje uzima se rješenje koje ima manje neposluženih korisnika (linije 31-32), dok se u slučaju da je taj broj isti, dalje uspoređuje ukupno minimalno kašnjenje koje bi se pojavilo kad bi se problem relaksirao i korisnici ubacili u rute (linije 33-43). Sličan pristup koriste autori u radu [67],[11] za potrebe eliminacije rute s najmanjim brojem korisnika u fazi minimizacije broja vozila. Funkcija *MinimalnoKasnjenje* zbroj minimalnih kašnjenja svih neposluženih korisnika izračunava pomoću izraza:

$$k = \sum_{i=1}^{|U|} D_{min}(U_i) \ , \tag{4.17}$$

gdje je D_{min} funkcija koja će dati vrijednost $+\infty$ ako se neposluženi korisnik U_i ne može umetnuti niti u jednu rutu bez prekoračenja ograničenja kapaciteta vozila, ili, minimalno kašnjenje koje bi nastalo umetanjem korisnika na najpovoljnije mjesto u najpovoljnijoj ruti s obzirom na vremenska ograničenja:

$$D_{min}(i) = \begin{cases} +\infty & , \forall r \in s : q_i > Q - \sum_{j=1}^{|r|} q_j \\ \min_{j \in C \setminus U} D(i,j) & , \text{inače} \end{cases} , \tag{4.18}$$

gdje r predstavlja rute korištenih vozila u rješenju s, q potražnju korisnika, Q kapacitet vozila, a C skup svih korisnika u problemu. Funkcija koja se minimizira $D(i,j)$ izračunava kašnjenje koje će se pojaviti ako se korisnik i umetne u rutu iza korisnika j:

$$D(i,j) = max(t_{d_j} + t_{ji} - t_{l_i}, 0) + max(t_{d_i} + t_{i(j+1)} - L_{(j+1)}) \ , \tag{4.19}$$

gdje je t_d vrijeme odlaska od korisnika, t vrijeme putovanja između dva korisnika, t_l najkasnije dozvoljeno vrijeme dolaska kod korisnika, a L najkasnije vrijeme dolaska uz koje će biti zadovoljena vremenska ograničenja svih korisnika do kraja rute. Vrijednost varijable L ažurira se za sve korisnike pogođene promjenama prilikom svake modifikacije rješenja. Treba imati na umu da i ova mjera predstavlja samo grubu procjenu kašnjenja budući da umetanje svakog korisnika može promijeniti mogućnosti umetanja preostalih neposluženih korisnika.

4.7 Odabir za reprodukciju

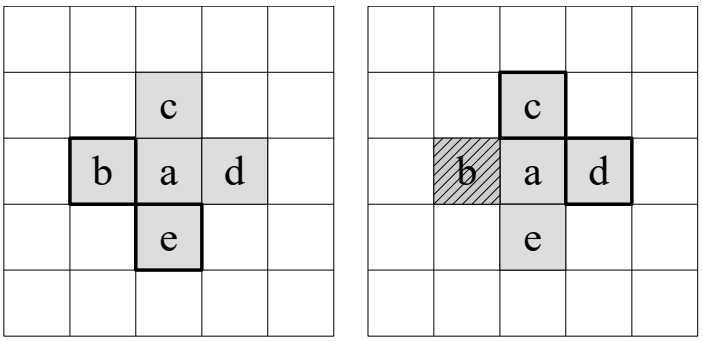
Istražene su tri metode selekcije: rulet selekcija, turnirska selekcija i selekcija linearnim rangiranjem. S obzirom da se evaluacijom jedinki u HCEA algoritmu ne izračunava apsolutna numerička vrijednost dobrote, primjena rulet selekcije kod koje je vjerojatnost odabira jedinke proporcionalna njezinoj dobroti, otpala je kao mogućnost. Kod selekcije linearnim rangiranjem, vjerojatnost odabira jedinke proporcionalna je njezinoj poziciji u ukupnom poretku jedinki iz susjedstva. Ova metoda ima slabiji selekcijski pritisak od turnirske selekcije pa samim time lošijim jedinkama koje su često nepotpuna rješenja pruža veće izglede za preživljavanje. Iako se ovo čini kao prednost u konceptu evolucijskog algoritma kod kojeg u populaciji skupa egzistiraju potpuna i nepotpuna rješenja, preliminarni testovi su pokazali da je selekcijski pritisak ipak preslab te da je potreban nekoliko puta veći broj generacija da bi učinkovitost algoritma bila na razini one koja se ostvari koristeći turnirsku selekciju. K tome još treba dodati činjenicu da je jedinke iz svakog susjedstva potrebno sortirati ili rangirati prije selekcije što bitno utječe na performanse algoritma. Iz navedenih razloga, za osnovni operator selekcije u HCEA algoritmu odabrana je binarna turnirska selekcija kojom se iz susjedstva pozicije (x, y), veličine i oblika zadanih parametrom N_t, odabiru jedinke roditelji koje će generirati potomka. Za prvog roditelja uzima se bolja od dvije slučajno

odabrane jedinke pri čemu se jedinke uspoređuju prethodno opisanom funkcijom. Postupak se ponavlja i kod odabira drugog roditelja, ali uz isključenu mogućnost ponovnog odabira prvog roditelja. Tablica 4.6 i slika 4.12 ilustriraju postupak odabira jedinki roditelja za potrebe križanja koristeći susjedstvo tipa L5.

Tablica 4.6. *Status jedinki susjedstva L5 prilikom odabira za križanje*

Jedinka	Neposluženih	Broj vozila	Udaljenost
a	1	10	833
b	0	10	1051
c	0	10	1012
d	0	10	1194
e	2	10	973

Prema funkciji *Bolje*, poredak jedinki od najbolje prema najlošijoj u ovom slučaju je c, b, d, a, e. U prvom krugu natjecanja, slučajno odabrane jedinke b i e uspoređuju se i jedinka b pobjeđuje. U drugom krugu natjecanja, ponovo se slučajnim odabirom biraju dvije jedinke, ali sada bez mogućnosti odabira jedinke b. Odabrane jedinke c i d uspoređuju se, a jedinka c izlazi kao pobjednik.



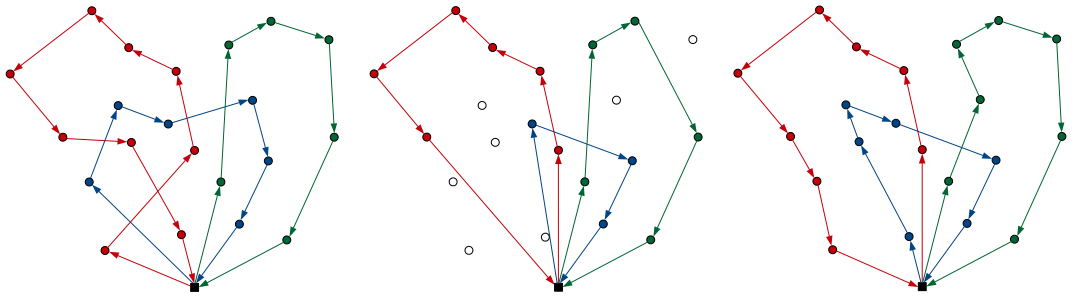
(a) Prvi krug natjecanja (b) Drugi krug natjecanja

Slika 4.12. *Vizualizacija primjera binarne turnirske selekcije*

Jedinka koja je izgubila u prvom krugu natjecanja može opet biti odabrana kao natjecatelj u drugom krugu (u ovom primjeru to je mogla biti jedinka e). Može se primijetiti da jedinka a koja se nalazi na središnjoj poziciji i prema kojoj se određuju granice susjedstva ne mora uopće biti odabrana. Međutim, u istom reproduktivnom ciklusu ona može biti odabrana prilikom obrade obližnjih pozicija u čijem se susjedstvu također nalazi. Evidentno je i da najlošija jedinka u susjedstvu nema izgleda postati roditelj. Binarna turnirska selekcija primjenjuje se samo kod odabira roditelja za križanje, a u slučaju da se križanje ne provede, klonira se jedinka s pozicije (x, y).

4.8 Operatori varijacije

Kao što je spomenuto u uvodnom dijelu, izuzev konstruktivne heuristike kojom se generiraju početna rješenja, jedinke se tijekom čitavog trajanja postupka rješavanja mijenjaju isključivo djelomičnim uništavanjem i popravljanjem. Postupkom uništavanja, određen broj korisnika izbacuje se iz ruta nakon čega se svi neposluženi korisnici pokušavaju umetnuti u rute postupkom popravljanja. Postoji više mogućih ishoda koji su rezultat ovakvog načina modifikacije rješenja. Primjerice, nakon djelomičnog uništavanja i popravljanja potpunog rješenja može se dobiti nepotpuno rješenje zbog nedostatka mogućnosti umetanja jednog ili više korisnika pri kraju postupka popravljanja. Može se dobiti novo potpuno rješenje koje je najčešće lošije, pogotovo ako se izvorno rješenje nalazi u blizini lokalnog optimuma. Rjeđe se uspijeva dobiti potpuno rješenje koje je bolje od izvornog. Može se dobiti rješenje identično izvornom što je čest slučaj ako se uništavanjem izbacuje mali broj korisnika. U slučaju da se modificira nepotpuno rješenje u najboljem slučaju može se dobiti novo potpuno rješenje. Primjer modifikacije rješenja CVRP problema postupkom djelomičnog uništavanja i popravljanja prikazan je na slici 4.13.



(a) Izvorno rješenje (b) Uništeno rješenje (c) Popravljeno rješenje

Slika 4.13. *Modifikacija rješenja postupkom djelimičnog uništavanja i popravljanja*

Izvorno rješenje prikazano na slici 4.13(a) djelomično je uništeno izbacivanjem trećine slučajno odabranih korisnika iz ruta čime se dobilo nepotpuno rješenje prikazano na slici 4.13(b). Rješenje je popravljeno heuristikom pohlepnog umetanja čime se dobilo novo rješenje koje je u ovom slučaju bolje od izvornog, slika 4.13(c). Operacija uništavanja i popravljanja rješenja u HCEA algoritmu ima tri uloge. U svojstvu križanja potiče istraživanje prostora rješenja u trenucima kada se u populaciji nalazi velik broj strukturalno različitih rješenja. S druge strane, u trenucima zasićenja populacije sličnim jedinkama postiže se efekt eksploatacije područja u kojem se rješenja nalaze. U svojstvu povremenih mutacija jedinki uloga operacije uništavanja i popravljanja je izbjegavanje lokalnih optimuma i diverzifikacija populacije.

4.8.1 Uništavanje rješenja

Kriterij odabira korisnika koji će se pri postupku uništavanja rješenja izbaciti iz ruta uvelike utječe na učinkovitost algoritma. Autori u radovima [101],[100],[91] predlažu nekoliko metoda odabira te naglašavaju važnost "povezanosti" korisnika koji se izbacuju pri čemu se povezanost odnosi na prostornu bliskost, slično vrijeme dolaska, početka posluživanja ili odlaska od korisnika, sličnu potražnju ili pak pripadnost istoj ruti. Da bi algoritam bio robustan potrebno je koristiti više kriterija prema kojima se korisnici izbacuju iz ruta. Broj korisnika koji se izbacuje također utječe na učinkovitost postupka. Ako je taj broj mali i/ili ako su odabrani korisnici "nepovezani", postoji velika vjerojatnost da će se popravljanjem dobiti identično rješenje jer će korisnici zbog nedostatka drugih mogućnosti jednostavno biti vraćeni u iste rute na pozicije iz kojih su izbačeni. S druge strane, broj korisnika koji se izbacuje ne smije biti ni prevelik jer je postupak popravljanja kritičan dio algoritma koji najviše utječe na performanse. Osim toga, što je veći broj izbačenih korisnika, to je i teže u potpunosti popraviti rješenje. Kriteriji odabira korisnika koji će se izbaciti u provedbi operacija križanja i mutacije opisani su u nastavku.

4.8.2 Popravljanje rješenja

Postupak popravljanja može biti izveden kao heuristika, egzaktni algoritam ili njihova kombinacija [101]. U radu [100], autori koriste jednostavnu heuristiku umetanja gdje se neposluženi korisnici slučajnim odabirom, jedan po jedan, ubacuju na najbolje mjesto u najboljoj ruti u obzirom na trošak umetanja. Može se koristiti više različitih heurističkih postupaka popravljanja kao u radu [91] u kojem autori koriste "pohlepnu" heuristiku umetanja i heuristiku "umetanja sa žaljenjem" (eng. *Regret Insertion Heuristics*, kratica RIH). Iako je sporija od pohlepne, potonja heuristika odabrana je kao osnovni postupak popravljanja u HCEA algoritmu zbog bolje učinkovitosti s obzirom na broj korisnika koje u uspijeva ubaciti natrag u rute. Umjesto da u svakoj iteraciji odabere potez za umetanje (korisnika, rutu i poziciju u ruti) prema kriteriju najmanjeg troška kako to radi pohlepna heuristika, RIH heuristika odabire potez za kojim će najviše žaliti ako ga odmah ne sprovede u djelo. Naime, kako se neposluženi korisnici jedan po jedan ubacuju u rute, smanjuju se mogućnosti ubacivanja preostalih korisnika. Primjerice, bolje je odmah ubaciti korisnika koji ima samo jednu mogućnost umetanja iako je trošak velik, nego korisnika kojemu je trošak umetanja u istu rutu minimalan, a ima na raspolaganju i druge rute kao opcije za umetanje. Neposluženi korisnik i koji će se slijedeći umetnuti na najpovljniju poziciju u najpovoljnijoj ruti s obzirom na trošak umetanja, odabire se uz pomoć izraza:

$$i = \arg\max_{i \in U}(\Delta f_i^2 - \Delta f_i^1) \ , \tag{4.20}$$

gdje je U lista neposluženih korisnika, Δf_i^2 promjena u vrijednosti ciljne funkcije koja će se dogoditi umetanjem korisnika i na drugo najbolje mjesto, a Δf_i^1 umetanjem na najbolje mjesto. Promjena vrijednosti ciljne funkcije ovdje predstavlja razliku između zbroja duljine dva nova luka koja će se pojaviti umetanjem korisnika i na poziciju k i starog luka $(k-1,k)$ koji će umetanjem nestati:

$$\Delta f_i = d_{(k-1)i} + d_{ik} - d_{(k-1)k} \ . \tag{4.21}$$

Ukoliko se korisnik i može umetnuti samo u jedno vozilo, uzima se da je $\Delta f_i^2 = +\infty$. U spomenutom radu, autori generaliziraju RIH postupak čime se omogućuje razmatranje n najboljih poteza umetanja. Zbog specifične implementacije i performansi, kod HCEA algoritma razmatra se samo razlika u trošku između drugog i prvog najboljeg poteza umetanja korisnika i.

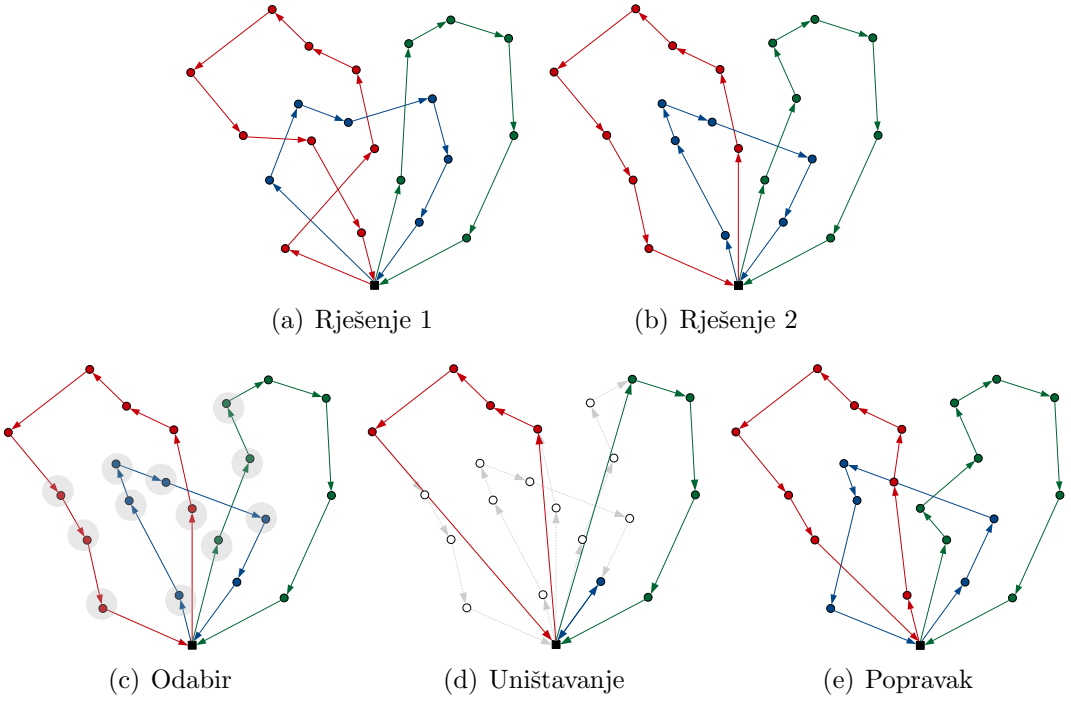
4.8.3 Ažuriranje rješenja

Nakon svake modifikacije jedinke, bilo da se radi o uništavanju ili popravljanju, ažuriraju se sve važne informacije o rješenju poput broja i liste neposluženih korisnika, broja korištenih vozila, ukupne pređene udaljenosti i utrošenog vremena. Ažuriranje rješenja ujedno znači i ažuriranje ruta vozila gdje su najvažnije informacije raspoloživi kapacitet vozila, pređena udaljenost, vrijeme završetka rute, te početna i završna lokacija u ruti. Ažuriranje pojedine rute obavlja se prolaskom kroz povezanu listu počevši od početne lokacije (atribut *First* klase *Route*). Ruta se može i djelomično ažurirati prolaskom liste od korisnika koji prethodi prvoj poziciji koja je mijenjana. Prilikom prolaska kroz listu, ažuriraju se informacije o zaustavljanju kod svakog pojedinog korisnika koje uključuju poziciju u ruti, vrijeme dolaska, vrijeme odlaska te najkasnije vrijeme dolaska do kojeg će biti zadovoljena vremenska ograničenja korisnika u dijelu rute koji slijedi. Ova posljednja informacija ubrzava provjeru mogućnosti ubacivanja neposluženih korisnika u rutu. Povezana lista u korištenoj ruti završava s još jednom dodatnom kopijom skladišta. Statistički podaci o neuspjelim pokušajima umetanja korisnika u rute koji se koriste prilikom evaluacije jedinke u izrazu (4.16), ažuriraju se odmah nakon provedbe postupka popravljanja.

4.8.4 Križanje

Dvije jedinke, prethodno odabrane kao roditelji iz susjedstva pozicije (x,y), postupkom križanja generiraju novu jedinku potomka. Za razliku od klasičnih genetskih algoritama kod kojih se dijelovi kromosoma "naslijepo" kombiniraju tako kreirajući novu jedinku, u HCEA algoritmu se operacija križanja provodi postupkom djelomičnog uništavanja i popravljanja pri čemu se razmjena informacija odvija samo u postupku uništavanja. Za postupak popravljanja koristi se ranije

opisana RIH heuristika. Nova jedinka kreira se kopiranjem ruta dominantnijeg (boljeg) roditelja nakon čega se djelomično uništava izbacivanjem korisnika za koje se utvrdi da im prethodni ili slijedeći korisnik u ruti nije jednak kod oba roditelja. Ovakav način razmjene informacija pri postupku uništavanja rješenja za cilj ima preispitati pozicije u rutama odabranih korisnika njihovim ponovnim umetanjem pomoću RIH heuristike. Pretpostavka je da je zbog selekcijskog pritiska i drugo roditeljsko rješenje kvalitetno te da sadrži dobre segmente. U početku evolucije, dok populacija sadrži velik broj različitih rješenja, razlike između rješenja su velike pa je i broj kandidata za izbacivanje velik. S vremenom, razlike između jedinki u susjedstvima postupno se smanjuju zbog konvergencije pretrage prema lokalnim optimumima pa se i prosječan broj kandidata za izbacivanje iz ruta postepeno smanjuje. Slika 4.14 ilustrira postupak križanja dvaju rješenja.



(a) Rješenje 1 (b) Rješenje 2

(c) Odabir (d) Uništavanje (e) Popravak

Slika 4.14. *Primjer križanja dva rješenja*

Generiranje potomka započinje kopiranjem boljeg rješenja i odabirom korisnika čija se relativna pozicija ne podudara kod oba roditelja, slika 4.14(c). Odabrani korisnici izbacuju se iz ruta, slika 4.14(d), a rješenje se popravlja umetanjem izbačenih korisnika u postojeće rute pomoću RIH heuristike, slika 4.14(e). U ovom primjeru jednako bi se rješenje dobilo i da je potomak kreiran kopiranjem lošijeg rješenja jer su se iz ruta izbacili svi korisnici po kojima se rješenja razlikuju. Međutim, kako je postupak popravljanja računski najzahtjevniji dio algoritma, maksimalan broj korisnika koji se može izbaciti ograničava se parametrom x_{rc}

HCEA algoritma. Često će broj korisnika koji predstavljaju razliku u rješenjima biti veći od ovog parametra, pogotovo u ranoj fazi evolucije, pa se zbog toga kao osnova za potomka uzima bolje rješenje od dva roditelja. Pseudokôd postupka križanja prikazan je u algoritmu 10.

Algoritam 10 Operator križanja u HCEA algoritmu

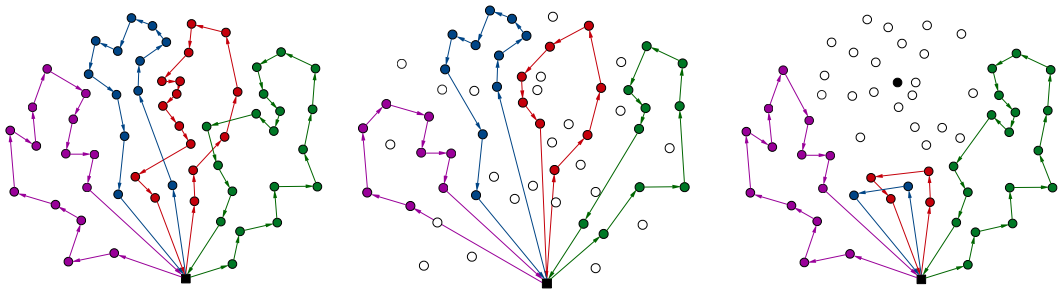
```
 1: procedure KRIZAJ(r_1, r_2, x_rc)
 2:     if Bolje(r_2, r_1) then
 3:         Zamijeni(r_1, r_2)
 4:     end if
 5:     R_c ← ∅
 6:     for i = 1 → UkupanBrojKorisnika() do
 7:         p_1 ← Prethodni(r_1, i)
 8:         p_2 ← Prethodni(r_2, i)
 9:         s_1 ← Slijedeci(r_1, i)
10:         s_2 ← Slijedeci(r_2, i)
11:         if i ∉ Neposluzeni(r_1) and (p_1 ≠ p_2 or s_1 ≠ s_2) then
12:             Dodaj(i, R_c)
13:         end if
14:     end for
15:     if |R_c| = 0 then
16:         return null
17:     end if
18:     potomak ← Kloniraj(r_1)
19:     n ← min(|R_c|, 1 + rnd(x_rc))
20:     if |R_c| > n then
21:         Promijesaj(R_c)
22:     end if
23:     for i = 1 → n do
24:         IzbaciKorisnika(potomak, R_c[i])
25:     end for
26:     Popravi(potomak)
27:     return potomak
28: end procedure
```

Parametri r_1 i r_2 predstavljaju rješenja roditelja. Linijama 2-4 osigurava se da je u r_1 pohranjeno bolje rješenje. Potencijalni kandidati za izbacivanje iz ruta spremaju se u listu R_c. Za svakog posluženog korisnika provjerava se da li su prethodni i slijedeći korisnik u ruti jednaki kod oba roditelja. Ukoliko to nije slučaj, korisnik se dodaje u listu kandidata za izbacivanje (linije 6-14). Kako s vremenom dolazi do zasićenja susjedstva i populacije jednakim ili sličnim jedinkama, r_1 i r_2 mogu predstavljati identična rješenja pa će lista R_c biti prazna. U tom slučaju, postupak križanja se obustavlja te se HCEA algoritmu vraća **null** vrijednost (linije 15-17). U protivnom, kreira se nova jedinka kloniranjem ili kopiranjem boljeg roditelja. Iz ruta će se izbaciti n korisnika, gdje je n slučajno

generirani broj iz intervala $[1..x_{rc}]$, ili ukupan broj kandidata ukoliko je on manji od slučajno generiranog broja (linija 19). Listu je potrebno promiješati ako u njoj ima više kandidata od broja koji se izbacuje tako da svi korisnici koji se nalaze u njoj imaju jednake izglede za izbacivanje (linije 20-22). Konačno, n korisnika izbacuje se iz ruta (linije 23-25), nakon čega se rješenje popravlja i vraća HCEA algoritmu (linije 26-27).

4.8.5 Mutacije

Povremene mutacije jedinki vrlo su važan element implementiranog algoritma s obzirom da sprječavaju preranu konvergenciju pretrage koja je neizbježna ukoliko se u postupku evolucije koristi samo operacija križanja. I za provedbu mutacija koristi se postupak uništavanja i popravljanja, ali uz dvije različite varijante odabira korisnika koje imaju jednaku vjerojatnost primjene. U prvoj, jednostavnijoj varijanti, korisnici se biraju slučajnim odabirom. U drugoj, kružnoj varijanti (eng. *radial*), prvi korisnik odabire se slučajno, a ostali korisnici po kriteriju najmanje udaljenosti od prvog. Kao i u slučaju križanja, maksimalan broj korisnika koji se može izbaciti iz ruta ograničen je parametrom m_{rc} HCEA algoritma, a rješenje se popravlja RIH heuristikom. Slika 4.15 prikazuje rezultat djelomičnog uništavanja rješenja slučajnim i kružnim odabirom korisnika.

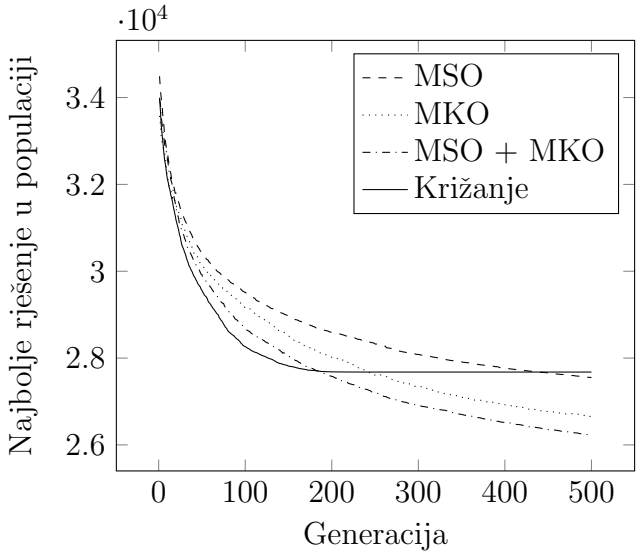


(a) Rješenje (b) Slučajni odabir (c) Kružni odabir

Slika 4.15. *Dvije varijante uništavanja rješenja prilikom mutacije*

U određenim situacijama obje varijante odabira korisnika imaju svoje prednosti. Primjerice, kada je prostorno izolirana grupa korisnika poslužena različitim vozilima, njihov odabir i izbacivanje kružnom varijantom može rezultirati poboljšanjem rješenja ukoliko se korisnici ponovnim umetanjem bolje raspodijele među vozilima. Mnogo je manja vjerojatnost izbacivanja te iste grupe korisnika varijantom uništavanja u kojoj se koristi slučajni odabir. S druge strane, izbacivanjem slučajno odabranih korisnika moguće je postići promjene na rješenjima koje nije moguće ostvariti kružnim izbacivanjem. Iako primarni cilj mutacije nije poboljšavanje rješenja već diverzifikacija populacije, algoritmu se povećava učinkovitost

ako se kombiniraju različiti postupci djelomičnog uništavanja rješenja. Slika 4.16 prikazuje utjecaj različitih postupaka uništavanja na učinkovitost HCEA algoritma.



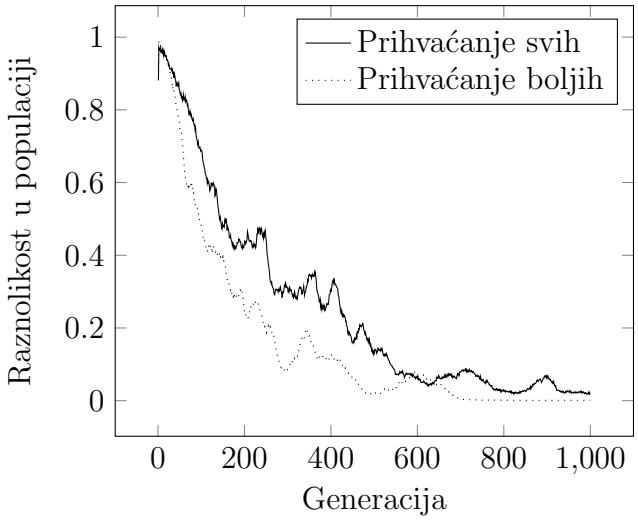
Slika 4.16. *Utjecaj postupka djelimičnog uništavanja rješenja na konvergenciju pretrage*

Na ordinati su prikazane vrijednosti ciljne funkcije (ukupna pređena udaljenost) najboljih rješenja u svakoj generaciji. Za svaki postupak uništavanja provedeno je deset neovisnih testova na CVRP problemu *tai385.dat*. Prikazane vrijednosti predstavljaju prosječne veličine. Problem je rješavan sinkronom inačicom HCEA algoritma koristeći toroidalnu matricu dimenzija 32×32 i L5 susjedstvo. Maksimalan broj korisnika koji se uništavanjem izbacivao iz ruta ograničen je parametrima $m_{rc} = x_{rc} = 25$. Rezultati testova provedenih postupkom rješavanja u kojem se kao jedini operator varijacije koristi mutacija (parametri $p_m = 1.0, p_x = 0.0$), jasno pokazuju važnost uloge postupka uništavanja i njegov utjecaj na učinkovitost algoritma. Primjena operatora mutacije u kojoj se korisnici iz ruta izbacuju slučajnim odabirom (MSO), na ispitanom je problemu dovela do lošijih rezultata od varijante koja koristi kružni odabir korisnika (MKO). Međutim, pogrešno bi bilo zaključiti da je MSO varijanta suvišna jer se kombinacija obje varijante (MSO + MKO) pokazala uspješnijom od pojedinačnih. Za očekivati je da bi rezultati bili još bolji kada bi se koristili i dodatni, problemski orijentirani postupci uništavanja rješenja poput onih predloženih u radovima [101],[96],[91], no kako je primarna uloga mutacije u HCEA algoritmu diverzifikacija populacije i izbjegavanje lokalnih optimuma, u ovom istraživanju primjenjuju se samo dvije opisane varijante uništavanja. Krivulja koja predstavlja konvergenciju pretrage ostvarenu isključivo primjenom križanja (parametri $p_m = 0.0, p_x = 1.0$) jasno pokazuje nužnost primjene mutacija u postupku rješavanja. Iako se uništavanje "razlika" između roditeljskih rješenja prilikom provedbe

križanja pokazalo najučinkovitijim u prvim generacijama evolucije, postupnim zasićenjem populacije sličnim jedinkama konvergencija počinje stagnirati da bi se u konačnici zaustavila zbog nedostatka različitih rješenja koja bi se mogla križati. Zbog toga je potrebno posebnu pažnju posvetiti podešavanju parametara koji kontroliraju učestalost i jačinu operatora križanja i mutacija.

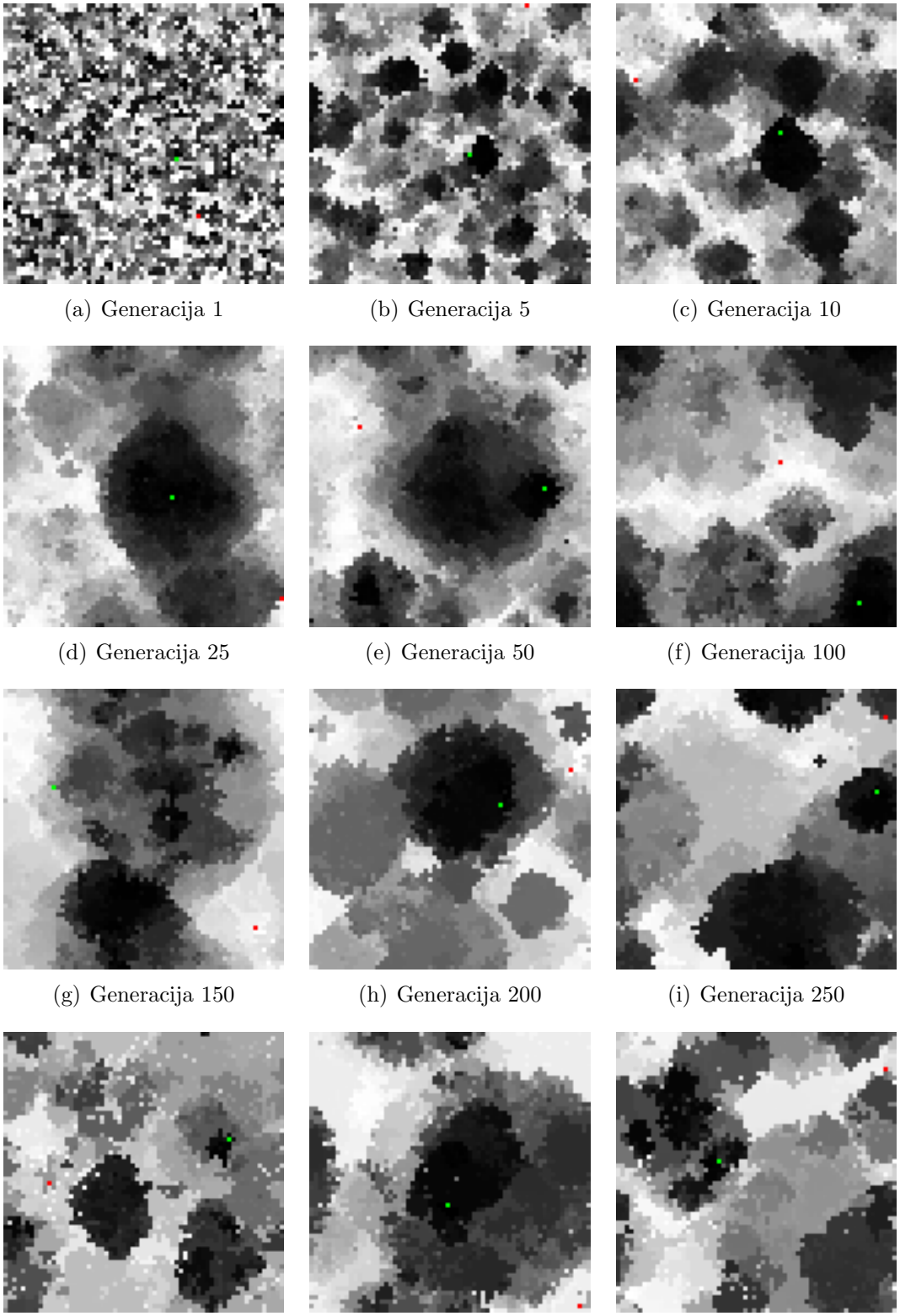
4.9 Odabir preživjelih

Na kraju svake iteracije algoritma potrebno je odabrati jedinke koje će preživjeti i nastaviti proces evolucije u slijedećoj generaciji. Način odabira preživjelih bitno utječe na konačni ishod algoritma. Svaki potomak bilo da je dobiven križanjem roditelja, bilo mutacijom, može zamijeniti samo staru jedinku koja se nalazi na istoj poziciji u toroidalnoj mreži. Ovdje je istražen utjecaj dvije najjednostavnije varijante prihvaćanja potomaka: prihvaćanje svih i prihvaćanje samo onih potomaka koji su bolji od stare jedinke. Slika 4.17 prikazuje utjecaj načina prihvaćanja potomaka na raznolikost u populaciji tijekom rješavanja CVRP problema *tai385* sinkronom varijantom HCEA algoritma.



Slika 4.17. *Raznolikost u populaciji uz dvije varijante prihvaćanja potomaka*

U pokusu su korišteni parametri: L5 susjedstvo, matrica 64×64, $p_x = 1.0$, $p_m = 0.01$, $x_{rc} = m_{rc} = 25, E = false$. Očekivano, prihvaćanjem svih potomaka postiže se veća raznolikost u populaciji. Sličan trend može se uočiti i kod rješavanja istog problema asinkronom inačicom algoritma. Za oba pokusa generiran je i vizualni prikaz populacije u različitim generacijama evolucije, slike 4.18 i 4.19. Svaki piksel predstavlja jedinku, a intenzitet kojim je osjenčan njezin poredak u populaciji. Poredak je određen linearnim rangiranjem koristeći funkciju *Bolje*.



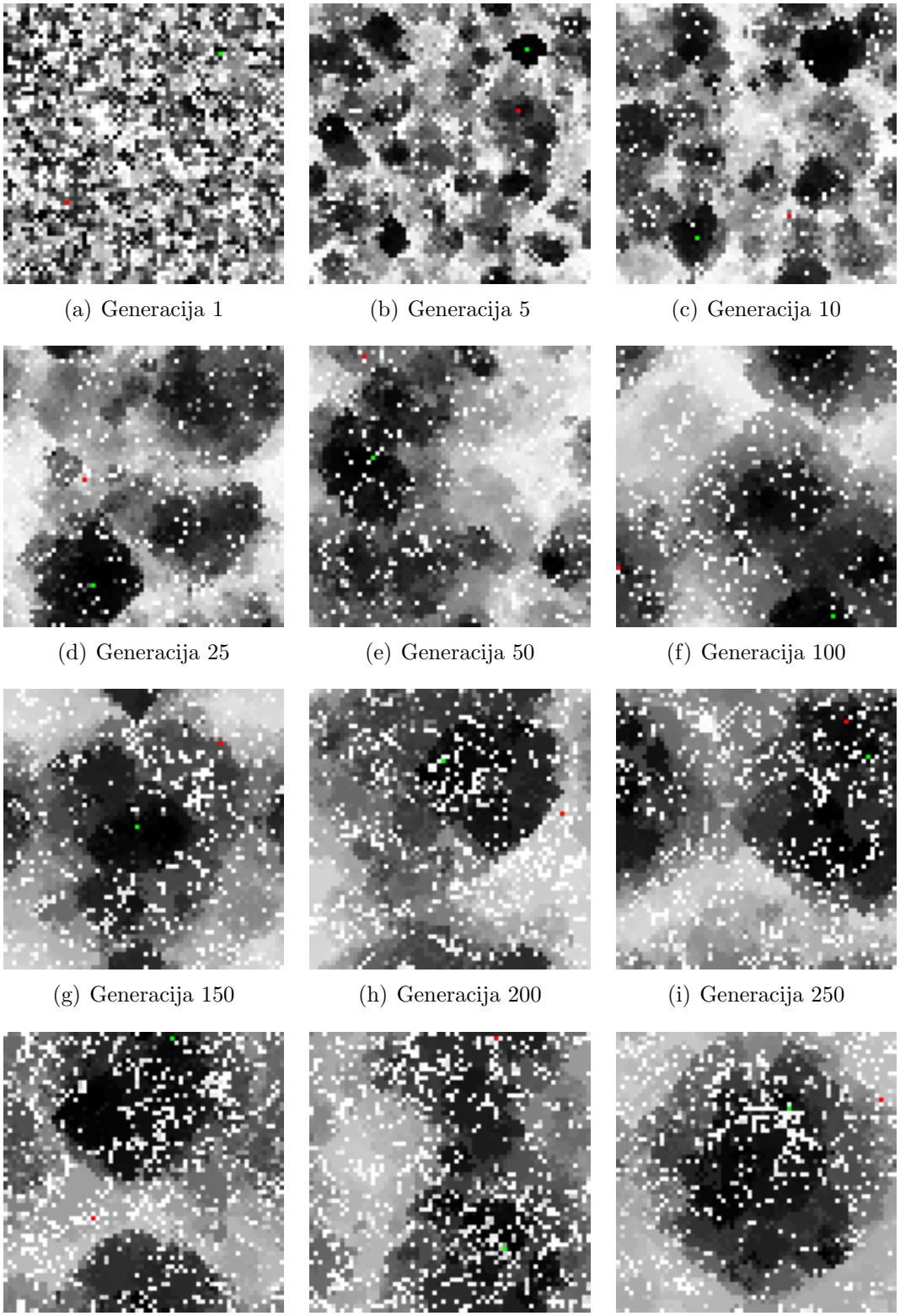
(a) Generacija 1 (b) Generacija 5 (c) Generacija 10

(d) Generacija 25 (e) Generacija 50 (f) Generacija 100

(g) Generacija 150 (h) Generacija 200 (i) Generacija 250

(j) Generacija 300 (k) Generacija 350 (l) Generacija 400

Slika 4.18. *Vizualizacija postupka rješavanja uz prihvaćanje samo boljih potomaka*



(a) Generacija 1 (b) Generacija 5 (c) Generacija 10

(d) Generacija 25 (e) Generacija 50 (f) Generacija 100

(g) Generacija 150 (h) Generacija 200 (i) Generacija 250

(j) Generacija 300 (k) Generacija 350 (l) Generacija 400

Slika 4.19. *Vizualizacija postupka rješavanja uz bezuvjetno prihvaćanje svih potomaka*

Što je piksel tamniji to je jedinka bolja. Zeleni piksel predstavlja najbolje rješenje u generaciji, a crveni najlošije. Na slikama se može primijetiti postupno stvaranje preklapajućih susjedstava koja nastaju propagacijom genetskog materijala boljih jedinki što je uzrokovano selekcijskim pritiskom. Kod pokusa u kojem se koristilo bezuvjetno prihvaćanje potomaka, na slikama se može primijetiti prisutnost svojevrsnog šuma kojeg čine nove jedinke koje su po dobroti znatno lošije od starih koje su zamijenile. Često se događa da se križanjem dvaju dobrih jedinki dobije nepotpuna, loša jedinka koja je među zadnjima u ukupnom poretku. Brojnost takvih jedinki najbolje ilustrira složenost postupka popravljanja i ukazuje na prostor za potencijalno poboljšanje HCEA algoritma. Iako postoji mogućnost gubitka najbolje jedinke iz populacije, bezuvjetnim prihvaćanjem svih potomaka u velikoj većini slučajeva postižu se bolji rezultati. Tablica 4.7 prikazuje rezultate testova provedenih na *Taillardovom* skupu testnih CVRP zadataka koristeći obje varijante prihvaćanja u sinkronoj inačici HCEA algoritma.

Tablica 4.7. *Usporedba strategije prihvaćanja svih i samo boljih jedinki*

Problem	Prihvaćanje svih	Prihvaćanje boljih
tai75a	1618,36	1618,36
tai75b	**1355,97**	1356,11
tai75c	**1291,01**	1291,66
tai75d	1365,42	1365,42
tai100a	**2049,43**	2051,31
tai100b	1942,18	1942,18
tai100c	1407,26	**1406,87**
tai100d	**1585,84**	1586,79
tai150a	**3056,15**	3058,56
tai150b	**2741,29**	2742,99
tai150c	**2369,78**	2371,77
tai150d	**2665,51**	2667,38
tai385	**25394,27**	25483,28

Rezultati predstavljaju prosječnu ukupnu udaljenost dobivenu na temelju 20 provedenih testova za svaki problem i varijantu prihvaćanja novih jedinki. Korišten je minimalan broj vozila kod svih problema. U testovima je korištena matrica dimenzija 32×32, L5 susjedstvo, broj generacija $g_{max} = 1000$, vjerojatnost križanja $p_x = 1.0$, vjerojatnost mutacije $p_m = 0.05$, te maksimalan broj korisnika koji se izbacuju iz ruta u postupku križanja i mutacije $x_{rc} = m_{rc} = 33$. Podebljane vrijednosti jasno ukazuju na bolju učinkovitost varijante u kojoj se bezuvjetno prihvaćaju sve jedinke. Ovakvi rezultati mogu se obrazložiti činjenicom da su i nepotpuna rješenja važna karika u evoluciji jer osim što doprinose raznolikosti u populaciji, već u slijedećoj generaciji mutacijom ili križanjem s drugim jedinkama mogu producirati potpunu i kvalitetnu jedinku. Iznimka u postupku odabira preživjelih je zamjena najlošije jedinke najboljom u slučaju da se primjenjuje načelo

elitizma. Kako se ovo načelo implicitno koristi uz varijantu prihvaćanja samo boljih jedinki, provedeni su dodatni testovi s ciljem utvrđivanja opravdanosti njegove primjene uz varijantu prihvaćanja svih jedinki. U tablici 4.8 prikazani su rezultati dobiveni na istom skupu problema koristeći iste parametre izuzev parametra E koji je bio predmet testiranja. Testirane su i sinkrona i asinkrona inačica algoritma u kojoj se koristila NRS strategija zamjene jedinki.

Tablica 4.8. *Utjecaj načela elitizma na učinkovitost pretrage*

		$HCEA_{sync}$		$HCEA_{async}$ (NRS)	
Problem	Br. voz.	Uz elitizam	Bez elitizma	Uz elitizam	Bez elitizma
tai75a	10	1618,36	1618,36	1618,36	1618,36
tai75b	9	1356,96	**1356,17**	1357,60	**1357,16**
tai75c	9	1291,25	**1291,09**	1291,82	**1291,09**
tai75d	9	1365,44	**1365,43**	1365,42	1365,42
tai100a	11	2050,21	**2049,27**	2051,96	**2051,70**
tai100b	11	1943,81	**1942,65**	1943,63	**1943,03**
tai100c	11	1408,21	**1406,37**	**1407,64**	1408,49
tai100d	11	1589,89	**1587,24**	1592,06	**1588,30**
tai150a	15	**3057,02**	3057,27	3056,53	**3056,00**
tai150b	14	2743,52	**2738,22**	2748,99	**2741,31**
tai150c	14	2375,85	**2370,52**	2380,83	**2376,12**
tai150d	14	**2666,44**	2672,13	**2669,82**	2671,90
tai385	46	25459,13	**25418,72**	**25384,06**	25484,75

Svaka vrijednost u tablici predstavlja prosječnu ukupnu udaljenost dobivenu na osnovu 20 neovisnih pokretanja algoritma. Korišteni broj vozila bio je minimalan u svakom dobivenom rješenju. Podebljane vrijednosti ukazuju na to da se bez primjene načela elitizma u većem broju slučajeva postižu bolji rezultati. Za razliku od klasičnih evolucijskih i genetskih algoritama kod kojih primjena elitizma bitno doprinosi učinkovitosti pretrage, ovdje to nije slučaj. Ovakav ishod testova može se objasniti činjenicom da su jedinke iz neposrednog susjedstva najbolje pronađene jedinke njoj vrlo slične ili jednake te da zbog djelovanja selekcijskog pritiska pomažu zadržati njezine karakteristike u populaciji i bez posebnog mehanizma za njihovo očuvanje.

4.10 Podešavanje parametara

S obzirom na velik broj parametara HCEA algoritma i njihovu međuovisnost, proveden je niz dodatnih testova s ciljem utvrđivanja vrijednosti uz koje će omjer kvalitete dobivenih rješenja i vremena izvršavanja biti najpovoljniji. U tablici 4.9 prikazane su empirijski utvrđene vrijednosti uz koje algoritam relativno brzo

uspijeva pronaći rješenja na problemima dimenzija do 200 korisnika koja su po kvaliteti unutar nekoliko postotaka od najboljih poznatih ili optimalnih rješenja.

Tablica 4.9. *Empirijski utvrđene vrijednosti parametara HCEA algoritma*

Parametar	g_{max}	w	h	N_t	p_x	p_m	x_{rc}	m_{rc}	E	R_t
Vrijednost	1000	32	32	L5	1.0	0.05	33	33	false	NRS[2]

Vrijednosti prikazane u tablici predstavljaju okvirne, grube postavke koje će se daljim ispitivanjem pokušati poboljšati. Za postizanje jednako kvalitetnih rezultata na problemima većih dimenzija utvrđeno je da je potrebno povećati broj generacija g_{max}, ali i broj korisnika koji se prilikom križanja ili mutacija izbacuju iz rješenja, x_{rc} i m_{rc}. Naime, ako se rješenja djelomično uništavaju izbacivanjem malog broja korisnika, popravljanjem nije moguće napraviti dovoljno jake strukturalne promjene koje bi spriječile prerano zaustavljanje pretrage u lošem lokalnom optimumu. Zbog toga je broj generacija i broj korisnika koji se uništavanjem izbacuju iz rješenja potrebno prilagoditi dimenzijama problema koji se rješava. U nastavku su prikazani rezultati ispitivanja utjecaja različitih vrijednosti ostalih parametara iz tablice na postupak rješavanja.

4.10.1 Dimenzije kvadratne matrice

U tablici 4.10 prikazani su rezultati ostvareni koristeći sinkronu inačicu HCEA algoritma uz različite dimenzije kvadratne matrice. Svaka vrijednost u tablici predstavlja prosječnu ukupnu udaljenost dobivenu na osnovu 20 neovisnih rješavanja problema iz Taillardovog skupa testnih zadataka. Da bi usporedba bila ravnopravna, broj generacija podešen je tako da maksimalan broj reproduktivnih operacija bude jednak kod svih dimenzija, pa je tako za matrice većih dimenzija određen proporcionalno manji broj generacija. Podebljane vrijednosti u tablici pokazuju da je na ovom skupu testnih zadataka algoritam u prosjeku najuspješniji ako je populacija smještena u toroidalnu matricu dimenzija 128×128. Iznimka je najveći CVRP problem od 385 korisnika na kojem su uz ove dimenzije matrice postignuti bitno lošiji rezultati nego kod matrica manjih dimenzija. Ovakav ishod može se objasniti činjenicom da je potrebno više vremena za propagaciju kvalitetnih jedinki kroz čitavu površinu matrice odnosno činjenicom da mnoga posjećena područja u prostoru rješenja ostaju nedovoljno istražena zbog sporije interakcije preklapajućih susjedstava. Može se stoga zaključiti da je selekcijski pritisak postignut matricom 128×128 i uz L5 susjedstvo nedovoljno jak za probleme ove veličine što potvrđuje pretpostavku da je za rješavanje većih problema potrebno koristiti veći broj generacija.

[2]Koristi se samo u asinkronoj inačici algoritma

Tablica 4.10. *Utjecaj dimenzija kvadratne matrice na učinkovitost algoritma*

Prob.	Br. voz.	8x8/32K	16x16/8K	32x32/2K	64x64/500	128x128/125
tai75a	10	1618,64	1618,46	**1618,36**	**1618,36**	**1618,36**
tai75b	9	1362,95	1360,08	1355,71	1355,42	**1355,19**
tai75c	9	1295,08	1291,34	1291,09	**1291,01**	**1291,01**
tai75d	9	1365,63	1365,46	**1365,42**	**1365,42**	**1365,42**
tai100a	11	2056,87	2052,74	2049,81	2047,41	**2042,67**
tai100b	11	1946,89	1943,29	1941,73	1940,22	**1939,9**
tai100c	11	1417,46	1409,56	1406,63	**1406,2**	**1406,2**
tai100d	11	1597,3	1592,84	1586,05	1582,63	**1580,93**
tai150a	15	3061,23	3060,15	3056,88	3055,37	**3055,34**
tai150b	14	2750,16	2750,16	2737,52	2734,85	**2728,79**
tai150c	14	2413,42	2391,58	2374,36	2366,1	**2364,72**
tai150d	14	2688,43	2670,44	2662,04	2652,54	**2647,22**
tai385	46	25498,87	25491,21	**25295,83**	25328,74	26613,28

S ciljem utvrđivanja utjecaja selekcijskog pritiska na rješavanje problema većih dimenzija, provedeni su dodatni testovi na problemu *tai385*. Za iste dimenzije matrica i broj iteracija kao u prethodnom testu, ispitani su i ostali tipovi susjedstva određeni parametrom N_t koji također utječu na selekcijski pritisak. Prosječni rezultati prikazani su u tablici 4.11.

Tablica 4.11. *Utjecaj dimenzija matrice i tipa susjedstva na rješavanje problema tai385*

N_t	8x8/32K	16x16/8K	32x32/2K	64x64/500	128x128/125
L5	25498,87	25491,21	**25295,83**	25328,74	26613,28
L9	25632,17	25480,27	25480,59	**25256,6**	26352,08
C9	25434,79	25529,8	**25285,68**	25304,11	26455,98
C13	25563,67	25533,5	**25323,45**	25325,63	26374,77
C21	25651,33	25440,93	25429,72	**25290,15**	26275,21
C25	25492,24	25527,18	25336,57	**25289,79**	26269,56

Može se uočiti korelacija između najboljih postignutih rezultata i omjera polumjera susjedstva i dimenzija matrice koji aproksimira selekcijski pritisak. Prema omjerima prikazanim u tablici 4.3 u uvodnom dijelu poglavlja, učinkovitost algoritama veća je što je omjer bliži intervalu [0,05...0,12]. Uz najveću matricu (128×128) algoritam je najmanje učinkovit u svim provedenim testovima, no jasno su vidljiva poboljšanja prosječnih rezultata kako se omjer približava navedenom intervalu. Za pretpostaviti je da bi rezultati bili bolji uz jednak broj iteracija kad bi se koristilo susjedstvo većeg polumjera ili na drugi način pojačao selekcijski pritisak, primjerice uvođenjem dodatnih etapa natjecanja u operatoru turnirske selekcije. Na temelju rezultata iz prethodne tablice proveden je dodatni test s većim brojem generacija. Za usporedbu su odabrane su matrice dimenzija

32×32 i 128×128, te tip susjedstva uz koji je u prethodnom testu postignut najbolji rezultat. Prosječne udaljenosti prikazane u tablici 4.12 potvrđuju pretpostavku da će najveća ispitana matrica polučiti bolje rezultate ako se broj generacija proporcionalno poveća u oba slučaja.

Tablica 4.12. *Usporedba dvaju dimenzija matrica uz povećan broj generacija*

N_t	Matrica	Br. generacija	Udaljenost (avg)
C9	32x32	8000	25244,55
C25	128x128	500	**25116,28**

Rezultati provedenih testova jasno ukazuju na prednosti staničnog evolucijskog algoritma. Povećanjem broja jedinki u populaciji moguće je povećati učinkovitost što kod klasičnih genetskih i evolucijskih algoritama nije uvijek slučaj. Kod slijedne inačice algoritma implementirane u sklopu ovog istraživanja, povećanje populacije znači duže vrijeme izvršavanja te veći utrošak radne memorije. No, kako su stanični algoritmi u svojoj naravi paralelni, HCEA algoritam nameće se kao idealan kandidat za paralelizaciju na višejezgrenim procesorima. Tu posebno treba istaknuti mogućnost paralelne izvedbe na grafičkim procesorskim jedinicama za opću namjenu (eng. *General Purpose Graphics Processing Unit*, kratica GPGPU) čime bi se postiglo višestruko ubrzanje izvršavanja [51], a time i omogućilo korištenje većih populacija uz prihvatljivo vrijeme izračuna.

4.10.2 Dimenzije pravokutne matrice

Oblik toroidalne matrice također utječe na selekcijski pritisak. Uz matrice pravokutnog oblika postiže se manji selekcijski pritisak nego uz matrice kvadratnog oblika. Što je oblik matrice uži, manji je i pritisak čime se povećava istraživanje prostora rješenja. Na najvećem problemu iz Taillardovog skupa testnih zadataka provedeno je po 20 neovisnih testova za sve tipove susjedstva koristeći pravokutne matrice različitih dimenzija i uz različit broj generacija. Tablica 4.13 prikazuje prosječne rezultate ostvarene sinkronom inačicom HCEA algoritma.

Tablica 4.13. *Utjecaj matrice pravokutnog oblika na rješavanje problema tai385*

N_t	41x25/2K	64x16/2K	128x8/2K	82x50/500	128x32/500	256x16/500
L5	**25272,97**	**25257,86**	**25246,53**	25262,15	25356,74	25417,2
L9	25411,29	25299,26	25289,83	25315,74	**25207,74**	**25301,09**
C9	25353,34	25317,35	25277,71	25287,75	25235,95	25318,59
C13	25364,98	25456,7	25282,24	**25212,24**	25285,37	25342,61
C21	25299,28	25390,98	25397,76	25319,95	25258,05	25313,46
C25	25501,06	25289,79	25324,45	25290,14	25275,27	25323,19

Rezultati dobiveni korištenjem pravokutnog oblika matrice bolji su od onih dobivenih korištenjem kvadratnog oblika matrice jednake površine. Kao i u prethodnom testu, najbolji rezultati postignuti su uz veće dimenzije matrice i uz proporcionalno manji broj generacija. Uže matrice imaju veći polumjer što znači da je omjer polumjera susjedstva i dimenzija koji utječe na selekcijski pritisak manji, što rezultira povećanjem istraživanja prostora rješenja. Populacija kod prve tri dimenzije u tablici broji 1024 (1025) jedinki te se kod njih najbolji rezultati postižu uz najmanje susjedstvo L5. S proporcionalnim povećanjem dimenzija matrice (posljednja tri stupca) odnosno broja jedinki u populaciji na približno 4096, selekcijski pritisak dvostruko se smanjuje, pa su očekivano najbolji rezultati postignuti korištenjem susjedstava većeg polumjera koja pojačavaju selekcijski pritisak (L9 i C13).

4.10.3 Strategija zamjene jedinki

U tablici 4.14 prikazani su rezultati ostvareni sinkronom i asinkronom inačicom HCEA algoritma na Taillardovom skupu testnih zadataka koristeći vrijednosti parametara uz koje su u prethodnom testu ostvareni najbolji rezultati.

Tablica 4.14. *Usporedba učinkovitosti sinkrone i asinkrone inačice HCEA algoritma*

Problem	SYNC	LS	FRS	NRS	UC
tai75a	**1618,36**	**1618,36**	**1618,36**	**1618,36**	**1618,36**
tai75b	**1355,39**	1356,44	1355,82	1355,67	1355,78
tai75c	**1291,01**	1291,17	1291,09	1291,09	1291,09
tai75d	**1365,42**	**1365,42**	**1365,42**	**1365,42**	**1365,42**
tai100a	**2046,7**	2052,76	2048,41	2048,13	2048,58
tai100b	1940,27	1942,87	1941,16	1940,51	**1940,24**
tai100c	**1406,2**	1408,87	1406,43	1406,3	1406,37
tai100d	**1582,75**	1591,86	1583,25	1585,33	1586,55
tai150a	3055,73	3061,86	3055,83	3056,04	**3055,71**
tai150b	2734,14	2743,23	**2733,27**	2735,45	2733,54
tai150c	**2366,79**	2384,59	2368,45	2367,53	2368,25
tai150d	**2660,81**	2677,28	2667,64	2661,75	2661,83
tai385	25260,85	25398,47	**25253,6**	25275,28	25277,55

Iako se sinkrona inačica HCEA algoritma na većem broju problema pokazala boljom, može se zaključiti da nema velikih razlika između učinkovitosti sinkrone i asinkrone inačice. Iznimka je strategija zamjene jedinki linearnim prolazom (LS) uz koji su postignuti najlošiji rezultati. Rezultati postignuti uz FRS, NRS i UC strategiju zamjene jedinki ohrabruju odabir asinkrone inačice algoritma za paralelnu implementaciju na GPGPU platformi budući da se komunikacija s glavnim procesom može izbjeći ili značajno smanjiti u odnosu na sinkronu inačicu.

4.10.4 Učestalost i jačina križanja i mutacija

U tablici 4.15 prikazani su rezultati serije testova kojima je ispitan utjecaj različitih učestalosti primjene operatora križanja i mutacija (parametri p_x i p_m), te njihove jakosti koja se određuje maksimalnim brojem korisnika koji se izbacuju iz ruta (parametri x_{rc} i m_{rc}). Rezultati predstavljaju prosječne ukupne udaljenosti dobivene na osnovu 20 neovisnih izvršavanja sinkrone inačice HCEA algoritma na problemu *tai385*. U svim testovima izvršavanje je ograničeno na 500 generacija, korištene su dimenzije matrice 128×32 i L9 susjedstvo te strategija bezuvjetnog prihvaćanja svih jedinki bez primjene načela elitizma.

Tablica 4.15. *Utjecaj postavki križanja i mutacije na rješavanje problema tai385*

Serija testova	p_x	p_m	x_{rc}	m_{rc}	Udalj. (avg)	t [s]
A	1,00	0,01	33	33	25428,91	342
	1,00	0,10	33	33	25241,61	401
	1,00	0,20	33	33	25172,41	456
	1,00	0,30	33	33	25166,01	501
	1,00	0,50	33	33	**25076,66**	620
B	1,00	0,05	33	33	25267,66	349
	0,90	0,10	33	33	25279,04	326
	0,75	0,25	33	33	**25241,75**	373
	0,50	0,50	33	33	25355,26	448
	0,25	0,75	33	33	26072,60	584
C	1,00	0,10	33	10	25556,38	339
	1,00	0,10	33	20	25294,08	351
	1,00	0,10	33	50	**25169,80**	420
	1,00	0,10	33	75	25219,25	502
	1,00	0,10	33	100	25233,94	599
	1,00	0,10	33	150	25244,51	858
D	1,00	0,25	33	10	25439,14	323
	1,00	0,05	33	100	**25327,88**	419
E	1,00	0,10	10	33	25575,40	277
	1,00	0,10	20	33	25356,90	332
	1,00	0,10	50	33	**25211,20**	636
	1,00	0,10	75	33	25255,26	1100
	1,00	0,10	100	33	25270,95	1867
	1,00	0,10	150	33	25460,66	4823

U prvoj seriji testova (A) ispitan je utjecaj različitih učestalosti mutacije uz konstantnu, empirijski utvrđenu jakost. Može se primijetiti da je konačan rezultat bolji što je učestalost mutacija veća. Češće mutacije doprinose održavanju veće raznolikosti u populaciji što pretragu gura naprijed. Uz učestalost mutacije od 0.5 postignuti su najbolji rezultati, no algoritam radi sporije zbog većeg broja

operacija izbacivanja i umetanja korisnika koje se naprave u jednoj generaciji. U drugoj seriji testova (B), ispitan je utjecaj koji operator križanja ima na postupak pretrage u odnosu na operator mutacije. Može se zaključiti da se sa smanjenjem učestalosti križanja postižu lošiji rezultati čak i uz povećanu učestalost mutacija. U trećoj seriji testova (C), ispitan je utjecaj parametra m_{rc} koji određuje maksimalnu jačinu mutacije gdje jačina podrazumijeva broj korisnika koji se uništavanjem izbacuje iz ruta. Kada su mutacije slabijeg intenziteta, pojačava se eksploatacija područja u kojem se rješenja nalaze. Na testiranom problemu koji broji 385 korisnika testovi pokazuju da je najpovoljnija vrijednost parametra m_{rc} u blizini intervala [50..75] što grubo odgovara ograničenju od maksimalnih 60 korisnika koje autori izbacuju iz ruta u algoritmu adaptivnog pretraživanja velikog susjedstva [91]. Valja obratiti pozornost i na trajanje izvršavanja algoritma. Algoritam je uz najjaču mutaciju više nego dva puta sporiji od mutacije s kojom su postignuti najbolji rezultati. U slijedećoj seriji testova (D), ispitano je da li učestalije mutacije slabijeg intenziteta više doprinose pretrazi od jakih mutacija koje su manje učestale. U oba testa ostvareni su rezultati lošiji od rezultata trećeg testa iz serije (C) čije se vrijednosti parametara p_m i m_{rc} nalaze negdje između ove dvije krajnosti. Posljednjom serijom testova (E) ispitan je utjecaj parametra za određivanje maksimalnog intenziteta križanja, x_{rc}. Rezultati potvrđuju zaključak testova iz serije (C) da je algoritam najučinkovitiji ako je broj korisnika koji se uništavanjem izbacuje iz rješenja ograničen na 50. S obzirom na stopostotnu učestalost operacije križanja algoritam se s povećanjem vrijednosti ovog parametra značajno usporava.

4.10.5 Automatsko podešavanje parametara

Da bi algoritam s približno jednakom učinkovitošću bio sposoban rješavati probleme različitih dimenzija potrebno je prilagoditi vrijednosti određenih parametara problemu. Ispitivanja su pokazala da parametri jačine križanja i mutacije x_{rc} i m_{rc} bitno utječu na konačan rezultat pretrage ali i vrijeme izvršavanja. Broj korisnika koji se prilikom uništavanja rješenja izbacuje iz ruta treba biti dovoljno velik da se u populaciju unesu bitne promjene, no ne smije biti prevelik s obzirom na ograničenu uspješnost i računsku složenost procedure za popravljanje rješenja. U svim testovima koji slijede, jačina križanja i mutacija određivat će se automatski prema izrazu:

$$x_{rc} = m_{rc} = 3\sqrt{n} \ , \tag{4.22}$$

gdje je n broj lokacija u problemu (korisnici i skladišta). Iako i dimenzije toroidalne matrice utječu na trajanje izvršavanja zbog specifične implementacije (veći memorijski zahtjevi, implementacija u .NET programskoj okolini) kao i zbog većeg prosječnog broja operacija izbacivanja i umetanja korisnika uslijed veće

raznolikosti, u nastavku će se koristiti vrijednosti uz koje su postignuti najbolji rezultati u prethodnim testovima. Odabrane vrijednosti svih parametara HCEA algoritma prikazane su u tablici 4.16.

Tablica 4.16. *Vrijednosti parametara HCEA algoritma za probleme do 400 korisnika*

Parametar	g_{max}	w	h	N_t	p_x	p_m	x_{rc}	m_{rc}	E
Vrijednost	500	128	32	L9	1.0	0.1	$3\sqrt{n}$	$3\sqrt{n}$	false

Prikazane vrijednosti predstavljaju grube postavke algoritma s obzirom da su utvrđene na temelju relativno malog uzorka od 20 ponavljanja po problemu i skupu parametara. Isto tako, sve moguće kombinacije vrijednosti parametara nisu ispitane, niti je ispitivanje provedeno na svim vrstama i dimenzijama problema. Parametri bi se morali preispitati i u slučaju unaprjeđenja postupka popravljanja. Broj parametra, njihova međuovisnost i potreba za podešavanjem pojedinom problemu mogu se izdvojiti kao najveći nedostatak HCEA algoritma.

4.11 Analiza dobivenih rezultata

Učinkovitost HCEA algoritma ispitana je na više skupova testnih zadataka. Za ispitivanje učinkovitosti na CVRP problemima korišteni su Augeratovi skupovi A, B i P s problemima dimenzija 16-101 korisnika [8], Christofidesov i Eilonov skup E s problemima dimenzija 22-101 korisnika [25], Christofidesov, Mingozzijev i Tothov skup s problemima dimenzija 50-199 korisnika [26], Fisherov skup F s problemima dimenzija 45-135 korisnika [48], Taillardov skup s problemima dimenzija 75-385 korisnika [105], te Goldenov skup s problemima veličine od 240 do 480 korisnika [63]. Za ispitivanje učinkovitosti na VRPTW problemima, korišten je Solomonov skup testnih problema veličine 100 korisnika [102], te Gehring & Hombergerov skup problema veličine 200 korisnika [56]. Algoritam je implementiran u C# programskom jeziku, a svi problemi rješavani su na računalu opremljenim procesorom Intel i7-6700HQ radnog takta od 2.6 GHz, te 16 GB radne memorije. Algoritam je implementiran slijedno, a svaki test koristio je samo jednu jezgru procesora tijekom čitavog trajanja izvršavanja. U tablicama 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24 prikazani su rezultati ostvareni sinkronom inačicom HCEA algoritma na testnim CVRP problemima, a u tablicama 4.25 i 4.26 rezultati ostvareni na testnim VRPTW problemima. Svaki problem riješen je 5 puta, a prikazani su prosječni i najbolji ostvareni rezultati, prosječno trajanje rješavanja te optimalno ili najbolje poznato rješenje drugih autora. Optimalni ili najbolji poznati rezultati preuzeti su s web stranica [33],[82] u trenucima dovršavanja ove disertacije.

Tablica 4.17. *Rezultati testova - CVRP problemi, Augerat, skup A*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
A-n32-k5	32	5	787,08	787,08	36,4	784	0,39%
A-n33-k5	33	5	662,11	662,11	21,6	661	0,17%
A-n33-k6	33	6	742,69	742,69	27,6	742	0,09%
A-n34-k5	34	5	785,61	780,94	26,1	778	0,38%
A-n36-k5	36	5	802,13	802,13	28,7	799	0,39%
A-n37-k5	37	5	672,47	672,47	29,3	669	0,52%
A-n37-k6	37	6	950,85	950,85	31,0	949	0,19%
A-n38-k5	38	5	733,95	733,95	29,7	730	0,54%
A-n39-k5	39	5	828,99	828,99	28,8	822	0,84%
A-n39-k6	39	6	833,20	833,20	26,9	831	0,26%
A-n44-k6	44	6	938,18	938,18	43,5	937	0,13%
A-n45-k6	45	6	944,88	944,88	32,0	944	0,09%
A-n45-k7	45	7	1146,77	1146,77	53,2	1146	0,07%
A-n46-k7	46	7	917,72	917,72	38,6	914	0,41%
A-n48-k7	48	7	1076,97	1074,34	32,6	1073	0,12%
A-n53-k7	53	7	1012,25	1012,25	38,3	1010	0,22%
A-n54-k7	54	7	1171,68	1171,68	51,3	1167	0,40%
A-n55-k9	55	9	1074,46	1074,46	43,2	1073	0,14%
A-n60-k9	60	9	1355,80	1355,80	58,7	1354	0,13%
A-n61-k9	61	9	1040,79	1040,34	35,4	1034	0,61%
A-n62-k8	62	8	1295,37	1293,94	39,6	1288	0,46%
A-n63-k9	63	9	1622,14	1622,14	47,1	1616	0,38%
A-n63-k10	63	10	1313,46	1313,46	72,3	1314	0,04%
A-n64-k9	64	9	1406,43	1400,63	51,2	1401	0,03%
A-n65-k9	65	9	1184,39	1183,31	41,4	1174	0,79%
A-n69-k9	69	9	1167,56	1167,56	39,1	1159	0,73%
A-n80-k10	80	10	1766,50	1766,50	85,6	1763	0,20%

Tablica 4.18. *Rezultati testova - CVRP problemi, Augerat, skup B*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
B-n31-k5	31	5	676,09	676,09	29,9	672	0,60%
B-n34-k5	34	5	789,84	789,84	25,5	788	0,23%
B-n35-k5	35	5	956,29	956,29	26,5	955	0,14%
B-n38-k6	38	6	807,88	807,88	25,6	805	0,36%
B-n39-k5	39	5	553,16	553,16	30,7	549	0,75%
B-n41-k6	41	6	833,66	833,66	37,5	829	0,56%
B-n43-k6	43	6	746,81	746,69	25,8	742	0,63%
B-n44-k7	44	7	914,96	914,96	33,3	909	0,65%
B-n45-k5	45	5	753,96	753,96	32,3	751	0,39%
B-n45-k6	45	6	680,44	680,44	28,6	678	0,36%
B-n50-k7	50	7	744,23	744,23	54,7	741	0,43%
B-n50-k8	50	8	1315,38	1315,38	49,9	1312	0,26%
B-n51-k7	51	7	1035,04	1035,04	33,2	1032	0,29%
B-n52-k7	52	7	749,97	749,97	46,9	747	0,40%
B-n56-k7	56	7	712,92	712,92	52,9	707	0,83%
B-n57-k7	57	7	1157,73	1157,73	41,6	1153	0,41%
B-n57-k9	57	9	1602,29	1602,29	64,7	1598	0,27%
B-n63-k10	63	10	1506,88	1506,88	43,1	1496	0,72%
B-n64-k9	64	9	868,19	868,19	51,8	861	0,83%
B-n66-k9	66	9	1325,04	1324,57	56,0	1316	0,65%
B-n67-k10	67	10	1039,27	1039,27	67,5	1032	0,70%
B-n68-k9	68	9	1277,01	1276,20	50,9	1272	0,33%
B-n78-k10	78	10	1228,09	1227,90	61,0	1221	0,56%

Tablica 4.19. *Rezultati testova - CVRP problemi, Augerat, skup P*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
P-n16-k8	16	8	451,34	451,34	19,6	450	0,30%
P-n19-k2	19	2	212,66	212,66	14,7	212	0,31%
P-n20-k2	20	2	217,42	217,42	15,4	216	0,65%
P-n21-k2	21	2	212,71	212,71	17,2	211	0,80%
P-n22-k2	22	2	217,85	217,85	13,5	216	0,85%
P-n22-k8	22	8	600,83	600,83	22,3	603	0,36%
P-n23-k8	23	8	531,17	531,17	24,4	529	0,41%
P-n40-k5	40	5	461,73	461,73	34,9	458	0,81%
P-n45-k5	45	5	512,79	512,79	35,9	510	0,54%
P-n50-k7	50	7	559,86	559,86	28,6	554	1,05%
P-n50-k8	50	8	635,40	634,85	32,2	631	0,61%
P-n50-k10	50	10	702,59	699,56	34,6	696	0,51%
P-n51-k10	51	10	741,50	741,50	36,6	741	0,07%
P-n55-k7	55	7	570,27	570,27	39,3	568	0,40%
P-n55-k10	55	10	697,89	696,83	44,3	694	0,41%
P-n55-k15	55	15	998,99	990,13	50,8	989	0,11%
P-n60-k10	60	10	748,07	748,07	49,4	744	0,54%
P-n60-k15	60	15	971,58	971,58	62,6	968	0,37%
P-n65-k10	65	10	795,66	795,66	62,0	792	0,46%
P-n70-k10	70	10	831,70	829,93	44,0	827	0,35%
P-n76-k4	76	4	599,03	598,20	62,8	593	0,87%
P-n76-k5	76	5	634,10	633,97	63,1	627	1,10%
P-n101-k4	101	4	692,10	691,29	141,5	681	1,49%

Tablica 4.20. *Rezultati testova - CVRP problemi, Christofides & Eilon*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
E-n22-k4	22	4	375,28	375,28	17,0	375	0,07%
E-n23-k3	23	4	568,56	568,56	21,1	569	0,08%
E-n30-k3	30	3	535,80	535,80	25,9	534	0,34%
E-n33-k4	33	4	837,67	837,67	21,6	835	0,32%
E-n51-k5	51	5	524,61	524,61	36,6	521	0,69%
E-n76-k7	76	7	689,16	687,60	63,4	682	0,81%
E-n76-k8	76	8	741,71	740,66	66,4	735	0,76%
E-n76-k10	76	10	836,89	835,26	46,1	830	0,63%
E-n76-k14	76	14	1026,68	1026,60	53,2	1021	0,55%
E-n101-k8	101	8	827,36	826,14	112,5	815	1,35%
E-n101-k14	101	14	1088,38	1083,27	97,1	1067	1,50%

Tablica 4.21. *Rezultati testova - CVRP problemi, Christofides, Mingozzi & Toth*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
vrpnc1	50	5	524,61	524,61	39,2	524,61	0,00%
vrpnc2	75	10	837,92	835,26	48,4	835,26	0,00%
vrpnc3	100	8	826,87	826,14	120,9	826,14	0,00%
vrpnc4	150	12	1031,80	1028,42	141,9	1028,42	0,00%
vrpnc5	199	16	1318,08	1311,65	175,5	1291,29	1,55%
vrpnc11	120	7	1042,12	1042,12	114,7	1042,12	0,00%
vrpnc12	100	10	819,56	819,56	116,9	819,56	0,00%

Tablica 4.22. *Rezultati testova - CVRP problemi, Fisher*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Optimum	Δ [%]
F-n45-k4	45	4	723,54	723,54	49,1	724	0,06%
F-n72-k4	72	4	241,97	241,97	60,9	237	2,06%
F-n135-k7	135	7	1163,48	1162,96	175,6	1162	0,08%

Tablica 4.23. *Rezultati testova - CVRP problemi, Taillard*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Opt./n.p.	Δ [%]
tai75a	75	10	1618,36	1618,36	81,4	1618,36	0,00%
tai75b*	75	9	1355,59	1355,18	76,0	1344,62	0,78%
tai75c	75	9	1291,01	1291,01	94,4	1291,01	0,00%
tai75d	75	9	1365,42	1365,42	83,2	1365,42	0,00%
tai100a	100	11	2048,43	2047,90	102,4	2041,34	0,32%
tai100b	100	11	1940,33	1939,90	110,2	1939,90	0,00%
tai100c	100	11	1406,20	1406,20	105,8	1406,20	0,00%
tai100d	100	11	1582,51	1580,46	97,9	1580,46	0,00%
tai150a	150	15	3055,28	3055,23	187,2	3055,23	0,00%
tai150b	150	14	2728,98	2727,99	168,2	2727,03	0,04%
tai150c*	150	14	2363,85	2362,69	165,0	2358,66	0,17%
tai150d	150	14	2649,92	2645,39	196,1	2645,39	0,00%
tai385*	385	46	25250,22	25184,76	717,4	24366,41	3,25%

Tablica 4.24. *Rezultati testova - CVRP problemi, Golden*

Problem	Br. kor.	Br. voz.	Udalj. (avg)	Udalj. (min)	t [s] (avg)	Opt./n.p.	Δ [%]
Golden09	255	14	587,65	585,61	504,6	579,71	1,01%
Golden10	323	16	749,69	746,56	833,4	736,26	1,38%
Golden11*	399	17	1305,53	1261,35	3107,4	912,84	27,63%
Golden12	483	19	1128,14	1125,58	2355,5	1102,69	2,03%
Golden13	252	26	871,59	867,63	259,4	857,19	1,20%
Golden14*	320	29	1124,56	1117,40	506,2	1080,55	3,30%
Golden15	396	33	1368,61	1364,24	649,3	1337,92	1,93%
Golden16*	480	36	2159,55	1839,29	2483,8	1612,50	12,33%
Golden17	240	22	709,59	708,67	288,9	707,76	0,13%
Golden18	300	27	1005,38	1000,34	301,3	995,13	0,52%
Golden19	360	33	1379,18	1373,00	589,5	1365,60	0,54%
Golden20	420	38	1838,60	1829,96	643,2	1817,59	0,68%

Optimalni ili najbolji poznati rezultati na sva tri Augeratova skupa, Christofidesovom i Eilovnom skupu te na Fisherovom skupu CVRP problema, ostvareni su egzaktnim algoritmima koji euklidske udaljenosti zaokružuju na najbliže cijele brojeve. Kako se u HCEA algoritmu koristi dvostruka preciznost kod izračuna udaljenosti, u tablicama se mogu uočiti mala odstupanja od optimalnih rješenja. Pojedini problemi dolaze u dvije varijante, s minimalnim brojem vozila i s jednim vozilom više. Potonji su izostavljeni iz tablica s obzirom da HCEA algoritam prilikom rješavanja CVRP problema uvijek koristi minimalan broj vozila. Također, kod nekih problema optimum predstavlja minimalnu udaljenost koja je ostvarena koristeći veći broj vozila od minimalnog. Takvi problemi su u tablicama označeni sa simbolom zvjezdice (*). Najveće odstupanje kod Augeratovih problema iznosi 1.49%, kod Christofidesovih i Eilonovih problema 1.50%, kod Christofidesovih, Mingozzijevih i Tothovih 1.55%, kod Fisherovih problema 2.06%, kod Taillardovih 3.25%. Kod navedenih skupova problema, najveća odstupanja su postignuta na najvećim problemima. Iznimka je Goldenov skup problema kod kojeg su na dva problema postignuti rezultati bitno lošiji od najboljih poznatih (Golden11 i Golden16). Kod ove skupine problema korisnici su raspoređeni u geometrijski pravilnu mrežu s jednakim udaljenostima između susjednih korisnika. Zahtjevi za dostavom su također specifično distribuirani u prostoru što otežava popravljanje rješenja RIH heuristikom kada se koristi minimalan broj vozila.

Tablica 4.25. *Rezultati na Solomonovim problemima od 100 korisnika*

	$HCEA_{sync}$ (avg)		$HCEA_{sync}$ (min)		t [s]	Najbolje poznato		Δ
Prob.	**Br. voz.**	**Udalj.**	**Br. voz.**	**Udalj.**	**(avg)**	**Br. voz.**	**Udalj.**	**[%]**
C101	10	828,94	10	828,94	72,2	10	828,94	0,00%
C102	10	828,94	10	828,94	74,8	10	828,94	0,00%
C103	10	828,06	10	828,06	89,3	10	828,06	0,00%
C104	10	824,78	10	824,78	100,4	10	824,78	0,00%
C105	10	828,94	10	828,94	81,8	10	828,94	0,00%
C106	10	828,94	10	828,94	80,7	10	828,94	0,00%
C107	10	828,94	10	828,94	66,1	10	828,94	0,00%
C108	10	828,94	10	828,94	74,3	10	828,94	0,00%
C109	10	828,94	10	828,94	70,3	10	828,94	0,00%
AVG	**10,00**	**828,38**	**10,00**	**828,38**	**78,9**	**10**	**828,38**	**0,00%**
SUM	**90,00**	**7455,40**	**90,00**	**7455,40**	**710,0**	**90,00**	**7455,42**	
C201	3	591,56	3	591,56	64,1	3	591,56	0,00%
C202	3	591,56	3	591,56	74,2	3	591,56	0,00%
C203	3	591,17	3	591,17	75,5	3	591,17	0,00%
C204	3	593,96	3	593,93	97,0	3	590,60	0,56%
C205	3	588,88	3	588,88	67,0	3	588,88	0,00%
C206	3	588,49	3	588,49	71,4	3	588,49	0,00%
C207	3	588,29	3	588,29	72,4	3	588,29	0,00%
C208	3	588,32	3	588,32	76,2	3	588,32	0,00%
AVG	**3,00**	**590,28**	**3,00**	**590,27**	**74,7**	**3**	**589,86**	**0,07%**
SUM	**24,00**	**4722,23**	**24,00**	**4722,20**	**597,9**	**24,00**	**4718,87**	
R101	19	1650,97	19	1650,80	92,4	19	1650,80	0,00%
R102	17	1486,60	17	1486,12	116,2	17	1486,12	0,00%
R103	13	1293,07	13	1292,68	115,2	13	1292,68	0,00%
R104	9,2	1003,73	9	1007,31	134,5	9	1007,31	0,00%
R105	14	1388,60	14	1377,11	93,0	14	1377,11	0,00%
R106	12	1258,84	12	1252,62	108,4	12	1252,03	0,05%
R107	10	1110,01	10	1104,66	125,8	10	1104,66	0,00%
R108	9	964,55	9	961,55	150,4	9	960,88	0,07%
R109	11	1198,90	11	1197,42	116,6	11	1194,73	0,22%
R110	10	1134,36	10	1119,14	131,2	10	1118,84	0,03%
R111	10	1097,44	10	1096,73	132,0	10	1096,72	0,00%
R112	9	1002,69	9	982,14	160,4	9	982,14	0,00%
AVG	**11,93**	**1215,81**	**11,92**	**1210,69**	**123,0**	**11,92**	**1210,34**	**0,03%**
SUM	**143,20**	**14589,76**	**143,00**	**14528,26**	**1476,2**	**143,00**	**14524,02**	
R201	4	1266,02	4	1258,59	145,9	4	1252,37	0,49%
R202	3	1195,31	3	1191,70	191,9	3	1191,70	0,00%
R203	3	949,43	3	941,08	218,0	3	939,50	0,17%
R204	2	827,09	2	825,52	173,5	2	825,52	0,00%
R205	3	1000,42	3	994,43	201,8	3	994,43	0,00%
R206	3	912,94	3	912,75	176,6	3	906,14	0,72%
R207	2	893,33	2	893,33	164,2	2	890,61	0,30%
R208	2	734,35	2	727,69	162,1	2	726,82	0,12%
R209	3	923,47	3	914,48	189,8	3	909,16	0,58%
R210	3	963,91	3	959,26	200,7	3	939,37	2,07%
R211	2	903,13	2	899,69	171,0	2	885,71	1,55%
AVG	**2,73**	**960,86**	**2,73**	**956,23**	**181,4**	**2,73**	**951,03**	**0,54%**
SUM	**30,00**	**10569,41**	**30,00**	**10518,52**	**1995,7**	**30,00**	**10461,33**	
RC101	14	1696,95	14	1696,95	103,9	14	1696,95	0,00%
RC102	12	1554,75	12	1554,75	111,4	12	1554,75	0,00%
RC103	11	1267,24	11	1262,02	105,0	11	1261,67	0,03%
RC104	10	1153,50	10	1139,50	109,2	10	1135,48	0,35%
RC105	13	1631,66	13	1629,44	102,9	13	1629,44	0,00%
RC106	11	1424,73	11	1424,73	118,3	11	1424,73	0,00%
RC107	11	1230,97	11	1230,48	137,2	11	1230,48	0,00%
RC108	10	1151,95	10	1146,66	130,5	10	1139,82	0,60%
AVG	**11,50**	**1388,97**	**11,50**	**1385,56**	**114,8**	**11,50**	**1384,17**	**0,10%**
SUM	**92,00**	**11111,75**	**92,00**	**11084,52**	**918,4**	**92,00**	**11073,32**	
RC201	4	1415,15	4	1413,52	118,5	4	1406,94	0,47%
RC202	3	1414,22	3	1395,77	172,2	3	1365,65	2,16%
RC203	3	1071,57	3	1054,28	203,7	3	1049,62	0,44%
RC204	3	800,96	3	798,61	220,9	3	798,46	0,02%
RC205	4	1316,86	4	1304,52	140,3	4	1297,65	0,53%
RC206	3	1178,91	3	1166,51	168,3	3	1146,32	1,73%
RC207	3	1074,98	3	1061,84	203,8	3	1061,14	0,07%
RC208	3	853,26	3	838,91	215,4	3	828,14	1,28%
AVG	**3,25**	**1140,74**	**3,25**	**1129,24**	**180,4**	**3,25**	**1119,24**	**0,89%**
SUM	**26,00**	**9125,92**	**26,00**	**9033,96**	**1443,1**	**26,00**	**8953,92**	
SUM	**405,20**	**57574,46**	**405,00**	**57342,86**	**7141,2**	**405,00**	**57186,88**	**0,27%**

Tablica 4.26. *Rezultati na Gehring-Hombergerovim problemima od 200 korisnika*

	$HCEA_{sync}$ (avg)		$HCEA_{sync}$ (min)		t [s]	Najbolje poznato		Δ
Prob.	**Br. voz.**	**Udalj.**	**Br. voz.**	**Udalj.**	**(avg)**	**Br. voz.**	**Udalj.**	[%]
C1_2_1	20	2704,57	20	2704,57	256,4	20	2704,57	0,00%
C1_2_2	18	2947,92	18	2944,67	173,4	18	2917,89	0,91%
C1_2_3	18	2720,76	18	2719,62	191,9	18	2707,35	0,45%
C1_2_4	18	2648,43	18	2645,08	232,7	18	2643,31	0,07%
C1_2_5	20	2702,05	20	2702,05	282,3	20	2702,05	0,00%
C1_2_6	20	2701,04	20	2701,04	294,2	20	2701,04	0,00%
C1_2_7	20	2701,04	20	2701,04	249,7	20	2701,04	0,00%
C1_2_8	19	2783,83	19	2775,48	272,6	19	2775,48	0,00%
C1_2_9	18	2687,83	18	2687,83	163,0	18	2687,83	0,00%
C1_2_10	18	2663,49	18	2645,08	189,1	18	2643,51	0,06%
AVG	**18,90**	**2726,09**	**18,90**	**2722,64**	**230,5**	**18,90**	**2718,41**	**0,16%**
SUM	**189,00**	**27260,93**	**189,00**	**27226,44**	**2305,4**	**189**	**27184,07**	
C2_2_1	6	1931,44	6	1931,44	175,0	6	1931,44	0,00%
C2_2_2	6	1863,16	6	1863,16	196,0	6	1863,16	0,00%
C2_2_3	6	1781,82	6	1776,96	437,9	6	1775,08	0,11%
C2_2_4	6	1726,16	6	1705,03	459,4	6	1703,43	0,09%
C2_2_5	6	1879,31	6	1879,31	197,4	6	1878,85	0,02%
C2_2_6	6	1857,39	6	1857,35	217,5	6	1857,35	0,00%
C2_2_7	6	1849,46	6	1849,46	208,5	6	1849,46	0,00%
C2_2_8	6	1823,89	6	1823,88	224,9	6	1820,53	0,18%
C2_2_9	6	1843,35	6	1836,99	249,5	6	1830,05	0,38%
C2_2_10	6	1811,91	6	1806,58	237,3	6	1806,58	0,00%
AVG	**6,00**	**1836,79**	**6,00**	**1833,02**	**260,3**	**6,00**	**1831,59**	**0,08%**
SUM	**60,00**	**18367,88**	**60,00**	**18330,16**	**2603,4**	**60**	**18315,93**	
R1_2_1	20	4839,24	20	4798,38	212,8	20	4784,11	0,30%
R1_2_2	18	4097,62	18	4060,12	177,1	18	4039,86	0,50%
R1_2_3	18	3409,66	18	3388,34	182,8	18	3381,96	0,19%
R1_2_4	18	3092,21	18	3078,16	209,6	18	3057,81	0,66%
R1_2_5	18	4209,58	18	4179,71	154,1	18	4107,86	1,72%
R1_2_6	18	3629,67	18	3620,98	176,9	18	3583,14	1,04%
R1_2_7	18	3185,48	18	3154,71	194,4	18	3150,11	0,15%
R1_2_8	18	2977,61	18	2965,25	199,2	18	2951,99	0,45%
R1_2_9	18	3837,82	18	3822,46	168,2	18	3760,58	1,62%
R1_2_10	18	3352,03	18	3319,62	183,6	18	3301,18	0,56%
AVG	**18,20**	**3663,09**	**18,20**	**3638,77**	**185,9**	**18,20**	**3611,86**	**0,74%**
SUM	**182,00**	**36630,94**	**182,00**	**36387,72**	**1858,7**	**182**	**36118,60**	
R2_2_1	4	4751,66	4	4665,86	318,7	4	4483,16	3,92%
R2_2_2	4	3733,15	4	3695,38	357,9	4	3621,20	2,01%
R2_2_3	4	2964,88	4	2953,23	361,7	4	2880,62	2,46%
R2_2_4	4	2029,65	4	1992,94	478,9	4	1981,29	0,58%
R2_2_5	4	3409,67	4	3392,93	267,0	4	3366,79	0,77%
R2_2_6	4	3001,02	4	2957,76	376,8	4	2913,03	1,51%
R2_2_7	4	2483,09	4	2466,92	481,0	4	2451,14	0,64%
R2_2_8	4	1881,94	4	1852,50	447,0	4	1849,87	0,14%
R2_2_9	4	3148,93	4	3135,30	326,9	4	3092,04	1,38%
R2_2_10	4	2719,39	4	2694,36	379,3	4	2654,97	1,46%
AVG	**4,00**	**3012,34**	**4,00**	**2980,72**	**379,5**	**4,00**	**2929,41**	**1,72%**
SUM	**40,00**	**30123,37**	**40,00**	**29807,18**	**3795,1**	**40**	**29294,11**	
RC1_2_1	18	3819,81	18	3693,02	156,5	18	3602,80	2,44%
RC1_2_2	18	3332,50	18	3261,51	169,4	18	3249,05	0,38%
RC1_2_3	18	3043,82	18	3022,16	179,2	18	3008,33	0,46%
RC1_2_4	18	2885,73	18	2873,33	201,3	18	2851,68	0,75%
RC1_2_5	18	3521,56	18	3455,30	174,0	18	3371,00	2,44%
RC1_2_6	18	3445,62	18	3380,85	170,8	18	3324,80	1,66%
RC1_2_7	18	3258,11	18	3225,98	183,2	18	3189,32	1,14%
RC1_2_8	18	3122,65	18	3098,80	190,2	18	3083,93	0,48%
RC1_2_9	18	3129,53	18	3104,31	183,7	18	3081,13	0,75%
RC1_2_10	18	3037,59	18	3021,37	172,9	18	3000,30	0,70%
AVG	**18,00**	**3259,69**	**18,00**	**3213,66**	**178,1**	**18,00**	**3176,23**	**1,16%**
SUM	**180,00**	**32596,92**	**180,00**	**32136,61**	**1781,1**	**180**	**31762,34**	
RC2_2_1	6	3195,45	6	3163,66	405,6	6	3099,53	2,03%
RC2_2_2	5	2898,69	5	2844,52	629,3	5	2825,24	0,68%
RC2_2_3	4	2703,85	4	2683,55	490,1	4	2601,87	3,04%
RC2_2_4	4	2107,49	4	2088,71	562,6	4	2038,56	2,40%
RC2_2_5	4	2981,59	4	2922,25	435,7	4	2911,46	0,37%
RC2_2_6	4	3032,53	4	2988,47	403,5	4	2873,12	3,86%
RC2_2_7	4	2627,17	4	2595,58	361,9	4	2525,83	2,69%
RC2_2_8	4	2385,81	4	2345,49	413,5	4	2292,53	2,26%
RC2_2_9	4	2271,75	4	2221,08	353,4	4	2175,04	2,07%
RC2_2_10	4	2092,82	4	2065,24	391,1	4	2015,60	2,40%
AVG	**4,30**	**2629,71**	**4,30**	**2591,85**	**444,7**	**4,30**	**2535,88**	**2,16%**
SUM	**43,00**	**26297,14**	**43,00**	**25918,54**	**4446,8**	**43**	**25358,78**	
SUM	**694,00**	**171277,18**	**694,00**	**169806,64**	**16790,5**	**694**	**168033,83**	**1,04%**

Za ispitivanje učinkovitosti HCEA algoritma na VRPTW problemima korišteni su Solomonovi i Gehring & Hombergerovi skupovi testnih zadataka dimenzija 100 i 200 korisnika. Ovi skupovi mogu se podijeliti u podskupine C, R i RC s obzirom na prostorni raspored korisnika, te na podskupine 1 i 2 s obzirom na prosječan broj korisnika koji se mogu poslužiti jednom rutom. Kod C problema, korisnici su grupirani u grozdove, dok su kod R skupine prostorno distribuirani slučajnim odabirom. Problemi iz RC skupine djelomično su grupirani u grozdove, a djelomično distribuirani slučajnim odabirom. Kod skupine 1, jednom rutom se u prosjeku može poslužiti manji broj korisnika nego što je to slučaj kod skupine 2. Prema ovoj podjeli, ukupno postoji šest različitih tipova problema (C1, R1, RC1, C2, R2, RC2). Na Solomonovim problemima od 100 korisnika, HCEA algoritam je ukupno gledajući ostvario rezultate koji su 0.27% lošiji od najboljih poznatih rezultata. Najveća odstupanja se mogu primijetiti kod skupina R2 (0.54%) i RC2 (0.89%). Od ukupno 56 testnih zadataka, za njih 32 (57%) algoritam je uspio pronaći najbolje poznato rješenje unutar tri minute u bar jednom od pet pokušaja. Kod Gehring & Hombergerovih problema veličine 200 korisnika odstupanja od najboljih poznatih rezultata iznose 1.04%. Od 60 problema, za njih 11 (18%) su pronađena najbolja poznata rješenja i to samo kod C1 i C2 skupine. Kao i kod Solomonovih testnih zadataka i ovdje se najveća odstupanja mogu primijetiti na skupinama R2 (1.72%) i RC2 (2.16%). Ako se izuzme skupina problema C2 koja se pokazala jednostavnom za riješiti, može se zaključiti da HCEA algoritam ima manju učinkovitost na problemima iz podskupine 2 kod kojih se koristi mali broj vozila čijim se rutama poslužuje veliki broj korisnika.

Poglavlje 5

Rješavanje problema iz stvarnog svijeta

Problem usmjeravanja vozila u praksi se svakodnevno javlja kod tvrtki koje se bave distribucijom prehrambenih proizvoda, pića, robe široke potrošnje, poštanskih pošiljaka, novina, goriva i sl. Da bi se predloženi HCEA algoritam mogao praktično primijeniti s ciljem smanjenja transportnih troškova tvrtke koja obavlja distribuciju, potrebno je prikupiti i pripremiti podatke, utvrditi model VRP problema, modelirati cestovnu mrežu potpunim usmjerenim grafom i po potrebi prilagoditi algoritam i parametre algoritma problemu koji se rješava [53]. U uvodnom dijelu navedeni su najvažniji podaci koje je nužno prikupiti za modeliranje praktičnog VRP problema te je ukazano na česta proširenja koja se mogu pojaviti u praksi. Napravljen je pregled različitih mogućnosti dobivanja koordinata lokacija skladišta i korisnika, te je ukazano na važnost njihove preciznosti. Nadalje, navedeni su važni elementi na koje treba obratiti pozornost kod izračuna matrica udaljenosti, te je ukazano na važnost preciznosti predviđanja trajanja putovanja. Na kraju poglavlja, opisano je nekoliko studija slučaja hrvatskih tvrtki za koje su uspješno modelirani i riješeni VRP problemi, te je napravljena usporedba resursa koji bi se utrošili (broj vozila i pređena udaljenost) provedbom rješenja dobivenih HCEA algoritmom i stvarno potrošenih resursa.

5.1 Prikupljanje podataka

Da bi se problem distribucije iz gospodarstva ili industrije mogao modelirati kao VRP problem, potrebno je prikupiti sve bitne podatke vezane uz posao distribucije koji treba obaviti. Ti podaci uključuju informacije o lokaciji i radnom vremenu skladišta, lokacijama i zahtjevima korisnika kojima treba dopremiti robu, te podatke o vozilima koja su raspoloživa za obavljanje distribucije. Svaka tvrtka može imati i određene specifičnosti u procesu distribucije koje zahtijevaju proširе-

nje modela VRP problema uvođenjem dodatnih ograničenja. Prikupljeni podaci moraju biti kompletni i što precizniji da bi rute vozila koje su rezultat optimizacije bile provedive u praksi i minimalno odstupale od zahtjeva distributera i korisnika kojima se roba dostavlja. Tri su skupa podataka od kojih se svaki VRP problem sastoji: skladište(a), vozila i korisnici. U problemu mora biti definirano barem jedno skladište, a važne informacije koje treba prikupiti su:

- Lokacija
  - Adresa
  - Poštanski broj i mjesto
  - Geografska duljina i širina
- Radno vrijeme
  - Vrijeme otvaranja
  - Vrijeme zatvaranja

Ako postoji više skladišta iz kojih se obavlja distribucija pri čemu se korisnike može poslužiti iz bilo kojeg skladišta, problem se može modelirati kao višeskladišni problem usmjeravanja vozila (MDVRP). No, ukoliko su korisnici vezani uz pojedino skladište, potrebno je modelirati više pojedinačnih VRP problema, za svako skladište i pripadajuće korisnike posebno. Kod višeskladišnog problema, vozila su obično vezana uz skladište. Osnovni podaci o vozilu uključuju:

- Kapacitet teretnog prostora
  - Nosivost [kg]
  - Obujam [m^3]
  - Broj paletnih mjesta [kom]
  - Broj teretnih jedinica [kom]
- Radno vrijeme vozača
- Lokacija polaska
- Lokacija povratka

Kapacitet teretnog prostora u praksi je ograničen po više kriterija. Primjerice, kod distribucije robe široke potrošnje pored ograničene nosivosti potrebno je paziti da obujam teretnog prostora bude dostatan za svu robu koja se transportira. Određene vrste robe mogu biti lagane, ali zauzimati puno prostora pa je

potrebno zadovoljiti oba ograničenja. Roba se često slaže na palete pa kapacitet vozila može predstavljati i broj paleta koje je moguće smjestiti u teretni prostor. Najjednostavniji je slučaj kada se distribuira roba jednakih dimenzija. U tom slučaju kapacitet se može izraziti maksimalnim brojem komada koji se može smjestiti u teretni prostor. Vrlo bitno ograničenje je radno vrijeme vozača jer bez njega najbolje rješenje može sadržavati rute čija će izvedba trajati duže od zakonski dozvoljenog radnog vremena. Lokacija polaska i povratka je najčešće skladište, no u slučaju problema otvorenog usmjeravanja vozila (OVRP), lokacije povratka nema jer se vozila ne vraćaju u skladište nakon obavljenog posla. S druge strane, lokacija povratka ne mora nužno biti skladište nego može biti udaljena garaža, lokacija za odlaganje prazne ambalaže, paleta i sl. Podaci o korisnicima sadrže:

- Lokacija
  - Adresa
  - Poštanski broj i mjesto
  - Geografska duljina i širina
- Narudžba
  - Masa [kg]
  - Obujam [m^3]
  - Zauzimanje palete [%]
  - Broj teretnih jedinica [kom]
- Vremenska ograničenja
  - Najranije vrijeme početka iskrcaja
  - Najkasnije vrijeme početka iskrcaja
  - Predviđeno trajanje iskrcaja

Narudžba predstavlja ukupnu količinu robe koju je potrebno dopremiti do lokacije. Može biti izražena kao ukupna masa, obujam, udio u zauzimanju jedne palete, broj teretnih jedinica (ako se radi o jednolikom teretu) ili kombinacija ovih veličina. Korisnici mogu zahtijevati poseban termin unutar kojeg treba dopremiti robu, a koji se zadaje najranijim i najkasnijim dozvoljenim vremenom za početak iskrcaja. Predviđeno trajanje iskrcaja predstavlja aproksimaciju koja ovisi o količini naručene robe i specifičnostima pojedine lokacije (vrijeme potrebno za parkiranje, obavljanje administrativnih poslova kod korisnika i sl.). Ovo su najvažnija ograničenja nužna za modeliranje praktičnog VRP problema. Ovisno o kompleksnosti procesa distribucije, u praksi se može pojaviti i niz drugih ograničenja. U današnje vrijeme većina navedenih podataka pohranjena je u informacijskom sustavu tvrtke što otvara mogućnosti za automatizaciju postupka pripreme i modeliranja problema.

5.1.1 Geokodiranje lokacija

Ukoliko su prikupljene samo adrese lokacija, potrebno je provesti postupak geokodiranja kojim se točna koordinata (geografska širina i duljina) dobiva pretraživanjem kartografske baze podataka. Za ovu svrhu mogu se iskoristiti mrežni servisi poznatih globalnih tvrtki (*Google Maps*, *Microsoft Bing Maps*), GIS alati (eng. *Geographic Information System*) i zemljovidi zajednice otvorenog kôda (*OpenStreetMaps*) ili pak komercijalna rješenja. Za potrebe geokodiranja i modeliranja praktičnih problema u ovom istraživanju korišten je GIS alat i zemljovid Republike Hrvatske tvrtke Mireo d.d. [114]. Koordinate je moguće prikupiti i odlaskom na svaku pojedinu lokaciju s uređajem koji posjeduje prijamnik globalnog navigacijskog satelitskog sustava (eng. *Global Navigation Satellite System*, kratica GNSS) poput pametnog telefona ili tableta. Kako sve više tvrtki koristi uređaje za daljinski nadzor vozila, koordinate korisnika mogu se izvući iz povijesnih podataka. Kod nekih tvrtki predstavnici prodaje obilaze sve korisnike na dnevnoj, tjednoj ili mjesečnoj bazi te ih se može angažirati za prikupljanje koordinata. Koordinate lokacija moraju biti što preciznije s obzirom da pogreška već od par desetaka metara može stvoriti probleme u realizaciji izračunate rute. Slika 5.1 prikazuje dvije pogrešno geokodirane lokacije s pogreškom od približno 50m.

(a) Izvangradska sredina (b) Gradska sredina

Slika 5.1. *Primjeri netočnog geokodiranja s pogreškom od 50m*

Križić predstavlja pogrešno geokodiranu, a kružnica točnu lokaciju dostavnog mjesta. Strelice pokazuju usmjerenost ulica. Dok se u izvangradskoj sredini ili manjem mjestu prikazanom na slici 5.1(a) pogreška od 50m može tolerirati, u gradskoj sredini s gustom mrežom prometnica i pripadajućih pravila prometovanja prikazanih na slici 5.1(b), ovaj stupanj pogreške može izazvati niz problema. Najkraći ili najbrži put između prikazane lokacije i svih ostalih neće odgovarati stvarnosti što će utjecati na postupak rješavanja i rute koje se dobiju kao rezultat. Kada bi se vozilo pridržavalo isplanirane rute i stiglo na problematičnu lokaciju,

moralo bi pronaći novi put do točnog mjesta iskrcaja. Dok u izvangradskoj sredini to ne predstavlja problem, u gradskoj sredini će vozilo zbog jednosmjernih ulica i zabrana skretanja često morati prevaliti veći put i potrošiti dodatno vrijeme. To posebno dolazi do izražaja u vrijeme prometnih gužvi. Odstupanje od vremenskog plana može uzrokovati probleme kod realizacije ostatka rute (kašnjenje na isporuku, plaćanje kazni zbog kašnjenja, neisporuka zbog kraja radnog vremena, prekovremeni rad vozača i dr.).

5.1.2 Generiranje potpunog usmjerenog grafa

Na temelju prikupljenih podataka o lokacijama skladišta i korisnika, potrebno je kreirati potpuni usmjereni graf kojim će se opisati cestovna mreža koja povezuje svaki par korisnika. Graf se može modelirati matricom čiji je broj redaka i stupaca jednak broju lokacija u problemu (skladišta i korisnici). Svaki element matrice predstavlja duljinu najkraćeg ili najbržeg puta između dvije lokacije. Ako su zadana i vremenska ograničenja (radno vrijeme vozača i termini isporuke robe), potrebna je i komplementarna matrica vremena putovanja između svakog para lokacija. Najkraći ili najbrži put može se izračunati pomoću GIS alata i zemljovida koji sadrži bitne informacije vezane za mrežu cestovnih prometnica (usmjerenost ulica, zabranjena skretanja, obavezni smjerovi i dr.). Iako se izračunom dobiju detaljne informacije o putu, za modeliranje i rješavanje VRP problema dovoljno je koristiti ukupnu udaljenost i trajanje putovanja. Detaljne informacije o trasi puta koriste se nakon optimizacije za ispis ili prikaz kompletne rute na zemljovidu ili navigacijskom uređaju. Matrice udaljenosti i vremena putovanja u praksi su asimetrične jer duljina puta i vrijeme potrebno da se prevali put između dvije lokacije ne moraju biti jednaki u oba smjera. Ako se rješavanjem VRP problema želi minimizirati ukupna udaljenost, matrica udaljenosti mora zadovoljavati uvjet (5.1), a ako se želi minimizirati ukupno vrijeme, matrica vremena mora zadovoljavati uvjet (5.3).

$$d_{ac} \leq d_{ab} + d_{bc} \tag{5.1}$$

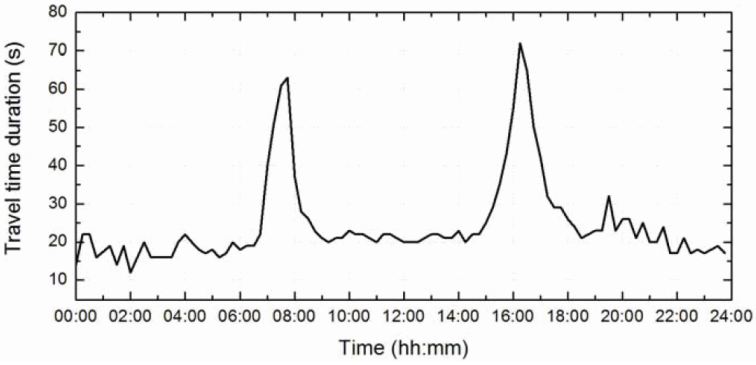
$$t_{ac} \leq t_{ab} + t_{bc} \tag{5.2}$$

Ovim ograničenjima osigurano je da put između lokacije a i c u matrici nema kraću ili bržu alternativu koja ide preko trećeg korisnika. Većina GIS alata omogućava postavljanje ograničenja prilikom izračuna puteva. Primjerice, moguće je odrediti tip vozila za koji se put izračunava te ograničiti prosječnu brzinu vožnje na određenim kategorijama prometnica. Prometovanje velikih kamiona u stvarnosti je ograničeno različitim propisima (obavezno korištenje obilaznica, ograničenja mase, visine i dr.) posebice u urbanim sredinama pa najkraći ili najbrži put

između dvije lokacije ne mora odgovarati putu predviđenom za prolazak osobnim automobilom ili manjim dostavnim vozilom. Kod rješavanja VRPTW problema treba preferirati najbrže puteve između lokacija budući da je vrijeme često bitniji čimbenik od udaljenosti. Kraći putevi mogu biti puno sporiji od duljih puteva koji koriste prometnice većeg ranga poput obilaznica i autocesta, što u konačnici može zahtijevati veći broj vozila za realizaciju distribucije.

5.1.3 Približan izračun trajanja putovanja

Vrijeme potrebno za prolazak izračunatog puta predstavlja grubu procjenu dobivenu zbrajanjem trajanja prolaska svakog cestovnog segmenta od kojeg se put sastoji. GIS alat i zemljovid korišten za potrebe ovog rada cestovne segmente dijeli na 16 kategorija (8 gradskih i 8 izvangradskih). Svakoj kategoriji pridružena je prosječna brzina ustanovljena na temelju povijesnih podataka o kretanjima vozila prikupljenih uz pomoć GNSS uređaja. U stvarnosti brzine vožnje variraju ovisno o dobu dana pa tako isti put u vrijeme gužvi (odlazak na posao u jutarnjim satima i povratak s posla u popodnevnim satima) traje duže nego inače. Na slici 5.2, prikazano je prosječno trajanje prolaska jednog cestovnog segmenta u Zagrebu u različitim razdobljima dana [80]. Rezultati su dobiveni analizom podataka koji su prikupljeni praćenjem više stotina vozila pomoću sustava za daljinski nadzor vozila kroz duži period.



Slika 5.2. *Vrijeme prolaska jednog segmenta Jadranskog mosta tijekom radnog dana*[1]

Ako se želi povećati vremenska preciznost odvijanja planirane rute u stvarnosti, problem se može modelirati kao vremenski ovisan problem usmjeravanja vozila s vremenskim prozorima (eng. *Time-Dependent Vehicle Routing Problem with Time Windows*, kratica TDVRPTW) koji je proširenje VRPTW problema u kojem trajanje putovanja između korisnika ovisi o vremenu u kojem se putovanje odvija [36],[45],[73],[49]. Kod ovog tipa problema vrijeme putovanja između korisnika i i j izračunava se posebnom funkcijom:

[1]Slika preuzeta iz [80]

$$t_{ij} = Vrijeme(i, j, t_c)\ , \tag{5.3}$$

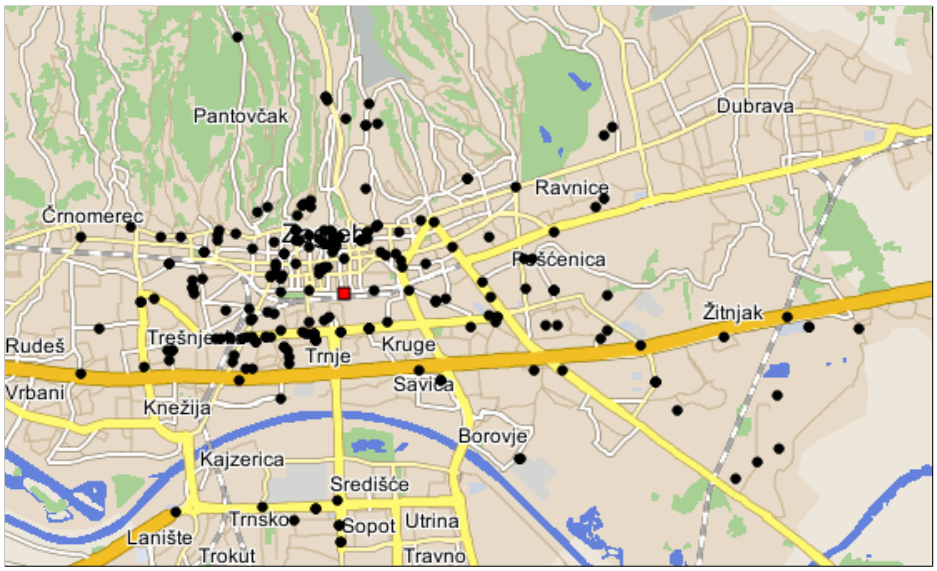
gdje t_c predstavlja vrijeme polaska od korisnika i. Cijeli vremenski horizont dijeli se na više segmenata u kojima se koristi odgovarajući faktor za korekciju brzine na putu između i i j na osnovu koje se potom izračunava vrijeme putovanja. Distribucija dnevnih brzina može varirati na pojedinim područjima i prometnicama, pa je za praktičnu primjenu ovog modela potrebno imati precizne podatke o razdiobi brzina kretanja na cestovnoj mreži promatranog područja.

5.2 Studije slučaja

Praktična primjenjivost predloženog HCEA algoritma ispitana je na skupu problema hrvatskih tvrtki koji su korišteni u prethodnom istraživanju [53]. Gdje je bilo potrebno, osnovni model problema proširen je novim ograničenjima kako bi optimizirane rute bile u skladu sa specifičnostima poslovanja svake pojedine tvrtke. Da bi se mogle izračunati potencijalne uštede koje bi se ostvarile uvođenjem softvera za planiranje i optimizaciju ruta distribucije, kod nekih tvrtki stvarno vožene rute dostavnih vozila snimljene su uz pomoć GPS uređaja, a gdje to nije bilo moguće, prikupljene su informacije o vozilima koja su se koristila u distribuciji promatranog dana, te broj pređenih kilometara koji je pročitan iz tahografa. Matrice udaljenosti i trajanja putovanja kojima se modelira cestovna mreža najkraćih (najbržih) putova generirane su uz pomoć GIS alata i zemljovida Republike Hrvatske tvrtke Mireo d.d. iz Zagreba [114]. Tvrtke čiji su problemi obrađeni bave se različitim djelatnostima poput dostave poštanskih pošiljaka, distribucije tiskovina, farmaceutskih proizvoda, pekarskih proizvoda te distribucije robe široke potrošnje.

5.2.1 Dostava poštanskih pošiljaka

Podaci prikupljeni od poštanskog operatera predstavljaju problem dostave i prikupljanja poštanskih pošiljaka kod 225 velikih korisnika na užem području Grada Zagreba. Dostava se obavlja u jutarnjim satima pomoću 16 vozila, a prikupljanje u popodnevnim satima pomoću 4 vozila. S obzirom da se radi o dva odvojena VRP problema, u razmatranje je uzet samo problem jutarnje dostave. Svaki veliki korisnik s poštanskim operaterom ima ugovoreni vremenski prozor predviđen za isporuku pismovnih pošiljaka i paketa. Budući da se radi o velikim tvrtkama, ministarstvima, gradskim uredima, pismovne pošiljke se dostavljaju u poštanskim vrećama čije su prosječne dimenzije približno jednake dimenzijama paketa standardne veličine i koje su uzete kao jedinstvena teretna jedinica. Za dostavu se koristi homogeni vozni park u kojem vozila imaju maksimalan kapacitet od 20 poštanskih vreća/paketa, a geografski raspored korisnika prikazan je na slici 5.3.



Slika 5.3. *Veliki korisnici poštanskog operatera na području Grada Zagreba*

Postojeće rute dostavnih vozila snimljene su uz pomoć GPS uređaja za potrebe geokodiranja lokacija korisnika, te za kasniju usporedbu s optimiziranim rutama. Analiza odvoženih ruta pokazala je da ugovorena vremena za isporuku pošiljaka nisu u cijelosti zadovoljena. Kako je poštanski operater dužan korisniku platiti kaznu za svako kašnjenje, cilj provedene studije bio je između ostalog pokazati opravdanost i isplativost uvođenja poštanskih sandučića na dostavnim mjestima pomoću kojih bi se proširili vremenski prozori i umanjila kašnjenja. Na temelju GPS snimaka utvrđeno je da je prosječno trajanje posluživanja iznosilo 3 minute, prosječna brzina kretanja vozila 21.5 km/h, te da je za dostavu korišteno 16 vozila koja su ukupno prevalila 240.79 km. Također, provjerena je preciznost zemljovida simulacijom odvoženih ruta čime je utvrđeno odstupanje od cca. 3% ukupne pređene udaljenosti. Utvrđeno je da su uzrok odstupanja pogreške GPS mjerenja, neažurnost zemljovida na pojedinim mjestima, te ljudski faktor (kršenje prometnih propisa tijekom vožnje poput polukružnog okretanja na nedozvoljenim mjestima, korištenje nedozvoljenih prečaca i dr.). Problem je modeliran kao klasični VRPTW s homogenim voznim parkom, ugovorenim vremenskim prozorima za isporuku pošiljaka, te vremenom posluživanja od 3 minute kod svih korisnika. U tablici 5.1 prikazana je usporedba najboljeg rezultata iz prethodnog istraživanja [53] i rezultata ostvarenog pomoću HCEA algoritma.

Tablica 5.1. *Rezultati optimizacije problema dostave poštanskih pošiljaka*

-	Snimljeno	SA	HCEA	Δ [%]	Ušteda [%]
Broj vozila	16	12	12	0,0%	-25,0%
Udaljenost [km]	240,79	216,18	204,66	-5,3%	-15,0%

Radi ravnopravne usporedbe, trajanje rješavanja problema sinkronom inačicom HCEA algoritma ograničeno je na pet minuta kao u prethodnom istraživanju, a izračun je proveden na identičnom računalu. Korištene vrijednosti parametara HCEA algoritma prikazane su u tablici 5.2.

Tablica 5.2. *Parametri korišteni za optimizaciju dostave poštanskih pošiljki*

Parametar	g_{max}	w	h	N_t	p_x	p_m	x_{rc}	m_{rc}	E
Vrijednost	500	128	32	L9	1.0	0.1	$3\sqrt{n}$	$3\sqrt{n}$	false

Na ovom problemu HCEA algoritam pokazao se učinkovitijim od simuliranog kaljenja s kojim je u prethodnom istraživanju pronađeno najbolje rješenje. Broj vozila je isti kod oba algoritma, dok je ukupna udaljenost smanjena za 5.3%. Rezultati pokazuju da je razmotreni posao distribucije poštanskih pošiljaka bilo moguće obaviti koristeći 4 vozila manje uz 15% manju ukupnu pređenu udaljenost.

5.2.2 Dostava tiskovina

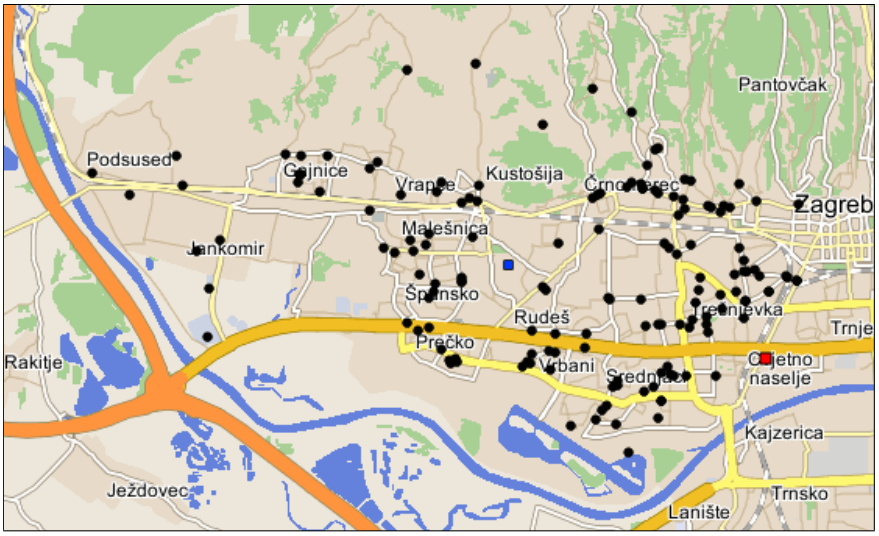
Za distributera tiskovina iz Zagreba provedena je analiza učinkovitosti postojećih ruta za distribuciju. Prikupljeni su podaci koji uključuju 153 prodajna mjesta na području zapadnog dijela Zagreba. Distribucija tiskovina započinje u 4:00h kada vozila izlaze iz tiskare, a u praksi se navedeno područje poslužuje sa šest vozila koja ukupno prijeđu 218 km. Sva vozila imaju nosivost 1500 kg, no prema sugestiji distributera nosivost je u izračunima ograničena na 1100 kg. Svako prodajno mjesto ima ograničeno vrijeme do kojeg se dostava mora izvršiti, a količina koju treba dostaviti izražena je u kilogramima. Procijenjeno trajanje iskrcaja aproksimirano je na osnovu količine koja se dostavlja prema pravilima prikazanim u tablici 5.3.

Tablica 5.3. *Aproksimacija trajanja iskrcaja na osnovu potražnje*

Potražnja [kg]	Trajanje [min]
$0 < q \leq 30$	1
$30 < q \leq 50$	2
$50 < q \leq 70$	3
$70 < q \leq 90$	4
$90 < q \leq 110$	5
$110 < q$	6

Nakon što obave iskrcaj, vozači su na svakom prodajnom mjestu dužni preuzeti remitendu odnosno neprodane primjerke tiskovina od prethodnog(ih) dana.

Iako se čini da je problem potrebno modelirati kao problem prikupljanja i dostave (eng. *Pickup & Delivery Problem*), količina koja se preuzima uvijek je manja ili u najgorem slučaju jednaka količini koja se iskrcava, pa se preuzimanje remitende može zanemariti. Problem je modeliran kao VRPTW no s modificiranim lokacijama povratka vozila. Naime, nakon obavljene dostave vozila odlaze iskrcati remitendu na lokaciju koja je od tiskare udaljena 6.6 km. Slika 5.4 prikazuje geografsku rasprostranjenost prodajnih mjesta u zapadnom dijelu Grada Zagreba. Lokacija skladišta označena je kvadratom crvene boje, a lokaciju za iskrcaj remitende kvadratom plave boje.



Slika 5.4. *Prodajna mjesta distributera tiskovina u zapadnom području Grada Zagreba*

Modelirani VRPTW problem riješen je pomoću sinkrone inačice HCEA algoritma uz iste parametre i ograničenje vremena izvršavanja kao u prethodnom slučaju. U tablici 5.4 prikazan je najbolji rezultat iz prethodnog istraživanja ostvaren genetskim algoritmom [53] i rezultat ostvaren HCEA algoritmom.

Tablica 5.4. *Rezultati optimizacije problema dostave tiskovina*

-	Snimljeno	GA	HCEA	Δ [%]	Ušteda [%]
Broj vozila	6	4	4	0,0%	-33,3%
Udaljenost [km]	218	103,07	96,55	-6,3%	-55,7%

Na zahtjev distributera modeliran je dodatni VRPTW problem koji pokriva scenarij u kojem početak dostave kasni zbog čekanja na završetak tiskanja određenog tjednika. U datom scenariju za početak dostave uzeto je vrijeme 4:45h. U tablici 5.5 prikazan je najbolji rezultat iz prethodnog istraživanja ostvaren simuliranim kaljenjem i rezultat ostvaren HCEA algoritmom.

Tablica 5.5. *Rezultati optimizacije modificiranog problema dostave tiskovina*

-	Snimljeno	GA	HCEA	Δ [%]	Ušteda [%]
Broj vozila	6	6	5	-16,7%	-16,7%
Udaljenost [km]	218	124,14	135,00	8,7%	-38,1%

U oba slučaja može se primijetiti poboljšanje rezultata iz prethodnog istraživanja. Ukupne uštede u pređenoj udaljenosti jasno ukazuju na kompleksnost problema i nedostatke ručnog planiranja distribucije u gustoj mreži prometnica i uz uska vremenska ograničenja.

5.2.3 Dostava robe široke potrošnje

Tijekom realizacije komercijalnog projekta i uvođenja modula za optimizaciju ruta dostavnih vozila u tvrtki Orbico d.o.o., distributera robe široke potrošnje, prikupljeni su podaci za jedan dan dostave koji uključuju geografske koordinate 289 kupaca, njihove narudžbe, te podatke o 29 vozila raspoloživih za dostavu. Promatrano skladište pokriva područje središnje i sjeverne Hrvatske. Geografska distribucija korisnika na promatrani dan dostave prikazana je na slici 5.5.

Slika 5.5. *Dostavna mjesta na području središnje i sjeverne Hrvatske*

Za razliku od prethodna dva problema kod kojih se sva dostavna mjesta nalaze u gradskoj sredini, kod ovog problema obuhvaćeno je šire geografsko područje koje pored gradskih uključuje lokalne i županijske ceste te autoceste. Zbog fizičkog ograničenja vozila na maksimalnu brzinu od 90 km/h, kod izračuna matrice vremena putovanja korigiran je parametar prosječne brzine na autocestama kako

bi optimizirane rute bile provedive u praksi uz što manja vremenska odstupanja. Tablica 5.6 prikazuje heterogeni vozni park raspoloživ za dostavu promatranog dana.

Tablica 5.6. *Heterogeni vozni park raspoloživ za dostavu*

Nosivost [kg]	Obujam [m^3]	Br. paleta	Broj vozila
1200	6	2,5	5
2500	13	4,5	8
2800	8	4,5	6
3000	8	4,5	3
3500	13	4,5	6
6000	14	12,0	1

Kapacitet svakog vozila određuje njegova maksimalna nosivost, obujam teretnog prostora i broj paleta koji u njega stane. Analogno tomu, svaka narudžba sadrži podatke o ukupnoj masi, obujmu, te zauzimanju jedinične palete u postocima. Dio kupaca robu može zaprimiti tijekom cijelog radnog vremena, a dio unutar točno specificiranih vremenskih prozora. Za svaku dostavnu lokaciju poznato je fiksno vrijeme zaustavljanja (vrijeme potrebno za parkiranje i obavljanje administrativnih poslova, čekanje na red za iskrcaj kod većih kupaca) koje je utvrđeno anketiranjem vozača. Ukupno vrijeme posluživanja zbroj je fiksnog vremena zaustavljanja i procjene vremena potrebnog za iskrcaj narudžbe koje se izračunava na osnovu količine i broja artikala. Radno vrijeme vozača ograničeno je na 8h, no budući da je svaki vozač prije same dostave dužan obaviti određene predradnje u skladištu koje traju 1h, trajanje dostave ograničeno je na 7h. Nakon što završi dostavu, većina vozila vraća se u skladište, no nekolicina njih završava svoje rute u jednoj od dvije dodatne lokacije za povratak koje su udaljene više desetaka kilometara od skladišta. Zbog navedenog ali i činjenice da tvrtka ima na raspolaganju višak vozila, konstruktivna heuristika modificirana je na način da se vozila angažiraju slučajnim odabirom. Naime, zbog specifičnog prostornog rasporeda korisnika, odabir vozila koja se koriste u rješenju uvelike utječe na konačni rezultat. U tablici 5.7 prikazan je najbolji rezultat postignut u prethodnom istraživanju genetskim algoritmom [53], te rezultat postignut sinkronom inačicom HCEA algoritma uz iste postavke kao u prethodnim primjerima.

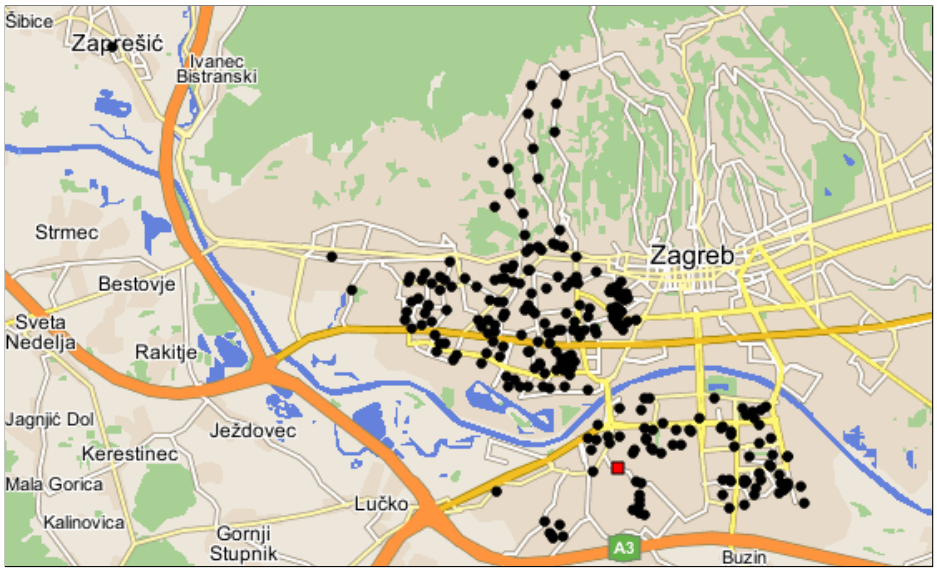
Tablica 5.7. *Rezultati optimizacije problema dostave robe široke potrošnje*

-	GA	HCEA	Δ [%]
Broj vozila	15	15	0.0%
Udaljenost [km]	3073,72	2823,90	-8,1%

Budući da su promatranog dana rute bile optimizirane, u tablici nema podatka o uštedi koja bi se ostvarila u odnosu na ručno planiranu dostavu. Uvođenjem softvera za optimizaciju dostave, ukupni troškovi na godišnjoj razini u navedenoj tvrtki bitno su smanjeni, direktno kroz smanjenje potrošnje goriva i broja korištenih vozila, te indirektno, unaprjeđenjem cjelokupnog procesa dostave što je u konačnici rezultiralo ukidanjem nekoliko manjih skladišta i uvođenjem sustava izravnog prekrcaja robe (eng. *Cross-Docking*)[54],[109].

5.2.4 Dostava pekarskih proizvoda

Od tvrtke koja se bavi proizvodnjom kruha, peciva i srodnih prehrambenih proizvoda, prikupljeni su podaci o prodajnim mjestima u jugozapadnom dijelu Zagreba i vozilima koja se koriste za dostavu na tom području. Dostava se obavlja pomoću 12 vozila različitih kapaciteta koja svakodnevno poslužuju 250 prodajnih mjesta čiji je prostorni raspored prikazan na slici 5.6. Sva vozila polaze i vraćaju se u skladište koje je na slici označeno kvadratom crvene boje.



Slika 5.6. *Prodajna mjesta proizvođača pekarskih proizvoda u Zagrebu*

Na svako dostavno mjesto potrebno je isporučiti određenu količinu proizvoda koja je izražena u broju standardiziranih košara koje se koriste za njihov transport. Kapacitet teretnog prostora vozila također je izražen u broju košara. Dostavna mjesta imaju specificiran vremenski prozor unutar kojeg dostavu treba obaviti. Ograničenje najranijeg vremena dostave postoji samo kod onih prodajnih mjesta na kojima dostavljač nema gdje ostaviti robu prije nego se prodajno mjesto otvori. Servisno vrijeme, odnosno trajanje iskrcaja zaokruženo je na 5 minuta za svako prodajno mjesto. Specifičnost ovog VRPTW problema su vremena

polazaka iz skladišta koja su unaprijed definirana za svako vozilo. Budući da proizvodni pogon ne može istovremeno opskrbiti sva vozila potrebnom količinom proizvoda, vozila se već od ranojutarnjih sati pune jedno po jedno i izlaze na teren obavljati dostavu. Nakon što završe dostavu, vraćaju se natrag u skladište. U tablici 5.8 prikazana su definirana vremena izlaska iz skladišta i kapaciteti vozila.

Tablica 5.8. *Kapaciteti vozila i vremena polaska iz skladišta*

Vozilo	Kapacitet	Vrijeme polaska
1	432	2:40
2	408	2:55
3	384	3:35
4	384	4:00
5	384	4:15
6	384	4:20
7	384	4:30
8	384	4:30
9	384	4:50
10	384	5:10
11	384	5:15
12	288	5:25

Budući da je broj vozila predefiniran redoslijedom polazaka iz skladišta, cilj optimizacije kod ovog praktičnog VRPTW problema je minimizirati ukupan pređeni put. Ukupna udaljenost koju prijeđe svih 12 vozila koristeći postojeće, ručno definirane rute, iznosi 523 km. U tablici 5.9 prikazan je najbolji rezultat ostvaren simuliranim kaljenjem u prethodnom istraživanju [53], te rezultat ostvaren ovdje predloženom sinkronom inačicom HCEA algoritma. Parametri HCEA algoritma jednaki su kao i u prethodnim primjerima, a vrijeme izvršavanja također ograničeno na pet minuta.

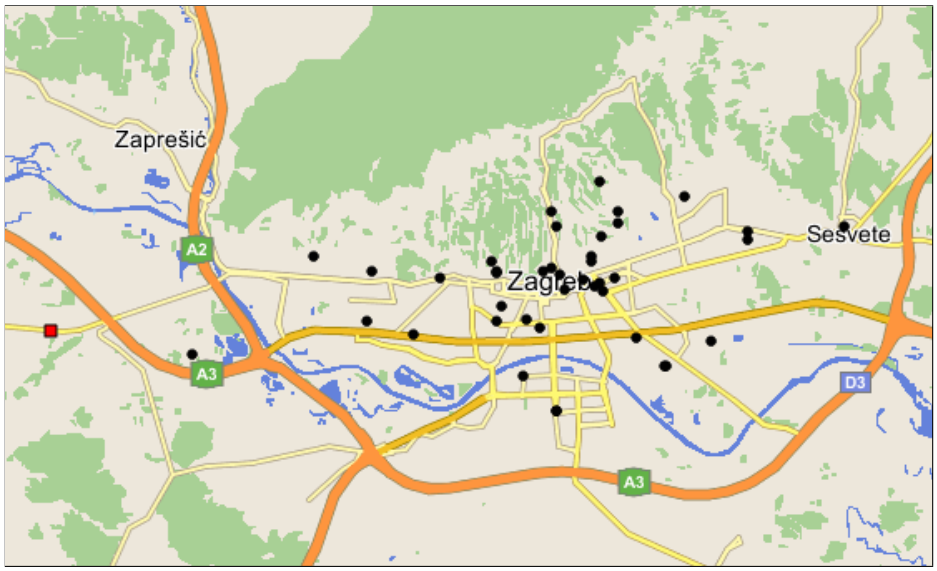
Tablica 5.9. *Rezultati optimizacije problema dostave pekarskih proizvoda*

-	Snimljeno	SA	HCEA	Δ [%]	Ušteda [%]
Broj vozila	12	12	12	0,0%	0,0%
Udaljenost [km]	523	254,16	242,45	-4,6%	-53,6%

Kao i kod prethodnih problema, HCEA algoritam pokazao se učinkovitijim. Rezultati ukazuju na to da se optimizacijom ruta dostavnih vozila u navedenoj tvrtki može ostvariti velika ušteda u pređenoj udaljenosti te potvrđuje da su statične rute koje se postepeno proširuju novim prodajnim mjestima kako tvrtka raste i širi poslovanje, daleko od optimalnih.

5.2.5 Dostava farmaceutskih proizvoda

Za potrebe pilot projekta rađenog za distributera farmaceutskih proizvoda izvršeno je snimanje postojećeg postupka planiranja jutarnje dostave na području Grada Zagreba. Prikupljeni podaci uključuju 41 zaprimljenu narudžbu, te kapacitete 8 vozila koja su u promatranom danu bila raspoloživa za dostavu. Na slici 5.7 prikazan je prostorni raspored dostavnih mjesta, te skladište koje je označeno kvadratom crvene boje.



Slika 5.7. *Lokacije za dostavu farmaceutskih proizvoda na širem području Zagreba*

Putem informacijskog sustava, dispečer u skladištu dobiva na uvid narudžbe zaprimljene do 19h, nakon čega slaže rute oslanjajući se na vlastito iskustvo i plan grada. Na osnovu složenih ruta, artikli iz svake narudžbe pakiraju se u kutije i ukrcavaju u vozila koja dostavu započinju ujutro u 7:30h. Rute moraju biti takve da se sva vozila stignu vratiti u skladište do 13:00h. Svako dostavno mjesto ima svoj standardni vremenski prozor unutar kojeg se dostava mora obaviti, a procjenu trajanja iskrcaja obavlja dispečer na osnovu iskustva i količine naručene robe. Budući da tvrtka narudžbe distribuira u netipiziranim paketima koji se izrađuju u skladištu, bilo je nužno odrediti kapacitet vozila u paketima prosječnih dimenzija, te preračunati dimenzije svake narudžbe u broj takvih paketa. Preračunavanje je obavljao dispečer koji poznaje cjelokupan asortiman tvrtke na temelju svih artikala iz narudžbe. Cilj projekta bio je pokazati stupanj (ne)učinkovitosti ručnog planiranja ruta, te izračunati optimizirane rute za promatrani dan. Da bi se stvarno vožene rute mogle usporediti s optimiziranima, prikupljeni su podaci iz sustava za daljinski nadzor vozila koji sadrže kompletne putanje vozila korištenih za dostavu promatranog dana. Tablica 5.10 prikazuje podatke o duljini i trajanju odvoženih ruta dobivene analiziranjem GPS zapisa.

Tablica 5.10. *Udaljenosti i trajanje stvarnih ruta distribucije farmaceutskih proizvoda*

Vozilo	Udaljenost	Vrijeme vožnje	Vrijeme posluž.	Ukupno
1	78.0 km	2:55:44	1:19:38	4:15:22
2	94.3 km	2:35:50	1:19:05	3:54:55
3	95.8 km	2:09:44	0:18:17	2:28:01
4	65.0 km	2:26:56	0:47:23	3:14:19
5	48.9 km	2:11:06	1:11:35	3:22:41
$\sum$	**382.0 km**	**12:19:20**	**4:55:58**	**17:15:18**

Modelirani VRPTW problem riješen je pomoću sinkrone inačice HCEA algoritma uz ranije definirane vrijednosti parametara. Vrijeme izračuna ograničeno je na jednu minutu. Ostvareni rezultat je prikazan u tablici 5.11 skupa s najboljim rezultatom iz prethodnog istraživanja koji je ostvaren algoritmom optimizacije mravljom kolonijom [53].

Tablica 5.11. *Rezultati optimizacije problema dostave farmaceutskih proizvoda*

-	Snimljeno	ACO	HCEA	Δ [%]	Ušteda [%]
Broj vozila	5	4	4	0,0%	-20,0%
Udaljenost [km]	382	214,06	213,22	-0,4%	-44,2%

Rezultati optimizacije ruta ukazuju na složenost zadaće koju dispečer obavlja čak i kod ovako malog broja lokacija. Kao i kod prethodnih primjera, evidentan je veliki prostor za uštede i racionalizaciju procesa distribucije što bi se moglo ostvariti uvođenjem softvera za optimizaciju ruta dostavnih vozila.

Poglavlje 6

Zaključak

U ovoj doktorskoj disertaciji istražen je problem planiranja ruta skupine vozila koja ima zadaću obaviti dostavu ili prikupljanje robe na određenom geografskom području. S ovim problemom susreće se velik broj tvrtki u gospodarstvu i industriji, a njegovim je rješavanjem moguće značajno smanjiti transportne troškove. Problem spada u kategoriju teških problema iz područja kombinatorne optimizacije i u literaturi se naziva problemom usmjeravanja vozila (VRP). Napravljen je pregled važnijih varijanti problema koje se razlikuju prema broju i vrsti ograničenja te ciljevima optimizacije. Kod CVRP problema optimizacija podrazumijeva minimizaciju ukupne pređene udaljenosti, dok je kod VRPTW problema potrebno prvo minimizirati broj vozila, a tek potom ukupnu udaljenost.

S obzirom na kompleksnost problema koja raste s brojem varijabli, egzaktni algoritmi nisu primjenjivi u praksi zbog dugotrajnog računanja. Umjesto njih mogu se koristiti heuristički i metaheuristički algoritmi koji ne jamče pronalazak optimalnih rješenja, no u stanju su relativno brzo pronaći rješenja koja su po kvaliteti blizu optimalnih. Istraženi su često korišteni heuristički postupci za konstruiranje početnih rješenja, te operatori koji se koriste u postupku lokalnog pretraživanja. Uspješnost lokalnog pretraživanja ograničena je jer se pretraga zaustavlja u lokalnom optimumu čim se iscrpe sve mogućnosti daljnjeg poboljšavanja rješenja uz primjenu istih operatora. Kako bi se omogućio pronalazak boljih rješenja potrebno je koristiti naprednije tehnike pretraživanja.

Metaheuristički algoritmi pretražuju puno širi prostor rješenja od heurističkih što je omogućeno povremenim prihvaćanjem i lošijih rješenja. Za uspješnost ovih algoritama ključna je uspostava povoljnog omjera diverzifikacije i intenzifikacije pretrage. Diverzifikacijom se pretraga širi na neistražena područja, dok se intenzifikacijom pokušavaju eksploatirati pojedina područja s ciljem pronalaska lokalnih optimuma. Istraženo je nekoliko metaheurističkih algoritama koji su se prema dosadašnjim istraživanjima pokazali vrlo uspješnima u rješavanju ove vrste problema: evolucijski algoritmi, simulirano kaljenje, optimizacija mravljom

kolonijom te pretraživanje velikog susjedstva.

Detaljno je istražena stanična inačica evolucijskog algoritma (cEA) kod koje svaka jedinka u populaciji zauzima određenu poziciju u toroidalnoj matrici. S obzirom da jedinke mogu biti u interakciji samo s jedinkama koje se nalaze na obližnjim pozicijama, pretraga rezultira stvaranjem preklapajućih susjedstava bez jasno definiranih granica. Susjedstva predstavljaju različita područja u prostoru rješenja koja se eksploatiraju primjenom operatora varijacije. Uslijed djelovanja selekcijskog pritiska, susjedstva s kvalitetnim jedinkama šire njihove karakteristike na obližnje pozicije, dok jedinke iz "loših" susjedstava postupno iščzavaju iz populacije zbog prodora boljih jedinki. Istražen je utjecaj koji na selekcijski pritisak imaju različiti operatori selekcije, dimenzije i oblik toroidalne matrice, veličina i oblik susjedstva te različite strategije zamjene starih jedinki novima.

Predložen je i implementiran hibridni stanični evolucijski algoritam (HCEA) za rješavanje CVRP i VRPTW problema. Stanični evolucijski algoritam hibridiziran je postupcima uništavanja i popravljanja koji se koriste u metaheuristici pretraživanja velikog susjedstva. Implementirane su sinkrona i asinkrona varijanta algoritma koje se razlikuju po načinu zamjene starih jedinki novima. S obzirom na dva cilja optimizacije kod VRPTW problema, postupak rješavanja podijeljen je u dvije faze. U prvoj se pokušava minimizirati broj vozila, a u drugoj ukupna pređena udaljenost. Broj vozila održava se konstantnim kod svih jedinki (rješenja) u populaciji tijekom čitavog postupka rješavanja.

Umjesto eksplicitnog kodiranja u kromosome, rješenja se pohranjuju u složene podatkovne strukture koje sadrže sve bitne informacije o rješenju problema (preostali kapaciteti vozila, pređena udaljenost, vremena dolazaka i odlazaka i dr). Rute su zapisane povezanim listama koje su omogućile efikasnu manipulaciju rješenjima u operatorima križanja i mutacije. Za inicijalizaciju populacije korištena je stohastička varijanta vremenski orijentirane heuristike najbližeg susjeda. S obzirom da se postupkom popravljanja ne može uvijek u potpunosti popraviti rješenje, u populaciji mogu egzistirati i nepotpune jedinke pri čemu su ograničenja vremena i kapaciteta uvijek zadovoljena. Predložena je složena leksikografska funkcija za ocjenu dobrote jedinke koja u obzir uzima broj neposluženih korisnika, broj korištenih vozila, ukupnu pređenu udaljenost, te složenost popravljanja nepotpunih rješenja.

Istražena su tri operatora selekcije jedinki prilikom reprodukcije: selekcija pomoću kotača ruleta, binarna turnirska selekcija i selekcija linearnim rangiranjem. Najboljom opcijom pokazala se binarna turnirska selekcija kod koje je odabir jedinki ograničen samo na jedinke iz susjedstva pozicije koja se u tom trenutku obrađuje.

U operatorima varijacije (križanja i mutacije) korišten je isključivo postupak djelomičnog uništavanja i popravljanja rješenja kojim se, ovisno o kontekstu, postižu tri efekta. Kada se u susjedstvu nalaze strukturalno različita rješenja, operatorom križanja pojačava se istraživanje prostora rješenja. Kada se u susjedstvu nalaze slične jedinke, njihovim križanjem se postiže efekt eksploatacije područja

u kojem se nalaze. Povremenim mutacijama, izbjegava se zaustavljanje pretrage u lokalnim optimumima i povećava raznolikost u populaciji.

U svojstvu mutacije korištene su dvije varijante uništavanja rješenja, uništavanje slučajnim odabirom korisnika i kružno uništavanje oko slučajno odabranog korisnika. Iako su uočene razlike u njihovoj učinkovitosti, utvrđeno je da je algoritam učinkovitiji ako se koristi kombinacija obje varijante uništavanja uz jednaku vjerojatnost primjene jedne i druge. U operatoru križanja korišten je novopredloženi postupak uništavanja rješenja koji se pokazao vrlo učinkovitim, posebice u početnim fazama pretrage dok je raznolikost u populaciji na visokoj razini. Potomak nasljeđuje karakteristike boljeg roditelja te se potom uništava izbacivanjem korisnika za koje se utvrdi da im relativna pozicija u ruti (prethodni i slijedeći korisnik) nije jednaka kao kod drugog roditelja. Popravljanjem se preispituje njihova izvorna pozicija u rutama jer drugi roditelj, iako je lošiji, može biti u blizini područja u kojem se nalaze bolja rješenja. Za postupak popravljanja odabrana je heuristika umetanja sa žaljenjem (RIH) koja se, unatoč mnogo većoj učinkovitosti od heuristike pohlepnog umetanja, pokazala kritičnim dijelom algoritma s obzirom na performanse.

Istražene su dvije varijante odabira jedinki koje će preživjeti i nastaviti proces evolucije i slijedećoj generaciji. Varijanta bezuvjetnog prihvaćanja svih novih jedinki pokazala se uspješnijom u većini testova od varijante prihvaćanja samo boljih jedinki. Analizirana je raznolikost u populaciji kod obje varijante te je utvrđeno da se ona duže vrijeme održava na većoj razini kod bezuvjetnog prihvaćanja svih novih jedinki. Napravljena je vizualizacija populacije u pojedinim generacijama evolucije te je utvrđeno da postupak uništavanja i popravljanja često rezultira nepotpunim rješenjima. Nepotpuna rješenja važna su karika u evoluciji jer u nekoj od slijedećih generacija mogu doprinijeti u produciranju potpune i kvalitetne jedinke. Ispitan je utjecaj primjene načela elitizma te je utvrđeno da kod staničnog evolucijskog algoritma ono nema utjecaj kao kod klasičnih evolucijskih algoritama zbog toga što karakteristike najbolje jedinke ostaju zadržane u populaciji zahvaljujući susjednim jedinkama i selekcijskom pritisku.

Proveden je niz testova s ciljem utvrđivanja najboljih vrijednosti parametara HCEA algoritma s obzirom na vrijeme izvršavanja i prosječne rezultate koji se postižu na odabranom skupu testnih zadataka. Na Taillardovom skupu CVRP problema utvrđena je prednost korištenja matrice većih dimenzija, ali i uočena potreba za jačim selekcijskim pritiskom i većim brojem generacija kod rješavanja većih problema. Uz pravokutne oblike matrice postignuti su nešto bolji rezultati od kvadratnih oblika iste površine što je potvrdilo teorijsku pretpostavku da se istraživanje prostora rješenja pojačava kako se oblik matrice sužava. Uspoređena je učinkovitost sinkrone i asinkrone inačice HCEA algoritma i na ispitanom skupu testnih zadataka nije utvrđena značajna razlika u učinkovitosti između njih. Serijom testova ispitan je utjecaj učestalosti i intenziteta križanja i mutacija. Testovi su pokazali da veća učestalost mutacija doprinosi većoj raznolikosti u populaciji što pozitivno utječe na krajnji ishod pretrage, ali uz povećanje trajanja izvrša-

vanja algoritma zbog većeg broja umetanja korisnika RIH heuristikom koji se u prosjeku obavi. Utvrđeno je i to da je na ispitanom problemu algoritam najučinkovitiji uz maksimalnu učestalost križanja. Intenzitet križanja i mutacije predstavlja maksimalan broj korisnika koji se uništavanjem izbacuje iz ruta. S obzirom na ograničenu učinkovitost i računsku kompleksnost RIH heuristike, provedeni su testovi s ciljem utvrđivanja ograničenja maksimalnog broja korisnika koji se izbacuje. Na ispitanom problemu utvrđeno je da je algoritam najučunkovitiji ako se taj broj kreće u rasponu 50-75. Uz veće vrijednosti, algoritam se značajno usporava te mu se smanjuje učinkovitost budući da RIH heuristika često ne uspijeva u potpunosti popraviti rješenje. Kako je utvrđeni raspon prevelik kod rješavanja manjih problema, predložena je funkcija kojom se maksimalan broj korisnika koji se uništavanjem izbacuje iz ruta automatski izračunava na osnovu broja korisnika u problemu.

Učinkovitost HCEA algoritma ispitana je na nekoliko skupova testnih CVRP i VRPTW zadataka. Kod CVRP problema, izuzev Goldenovog skupa testnih zadataka specifičnih po prostornom rasporedu korisnika, kod kojih algoritam nije u svim slučajevima uspio pronaći kvalitetna rješenja, najveće odstupanje u odnosu na najbolje poznato rješenje iznosi 3.25% kod Taillardovog problema od 385 korisnika. Kod VRPTW problema, učinkovitost je ispitana na testnim zadacima dimenzija 100 i 200 korisnika te su rezultati pokazali da je algoritam za relativno kratko vrijeme (nekoliko minuta) sposoban pronaći rješenja koja od najboljih poznatih rezultata u prosjeku odstupaju 0.27% kod problema od 100 korisnika, te u prosjeku 1.04% kod problema od 200 korisnika. Problemi iz podskupine 2 kod kojih se malim brojem vozila poslužuje velik broj korisnika pokazali su se težima od zadataka iz podskupine 1 kod kojih se koristi veći broj vozila.

Opisani su koraci koje je potrebno provesti da bi se problem distribucije iz gospodarstva ili industrije mogao modelirati kao VRP problem. Navedeni su ključni podaci koje je potrebno prikupiti te dodatna ograničenja koja se često javljaju u praksi. Opisani su postupci prikupljanja geografskih koordinata lokacija i objašnjena važnost njihove preciznosti. Opisan je postupak modeliranja mreže cestovnih prometnica matricom najkraćih putova i ukazano na probleme koji mogu nastati kao posljedica grube procjene trajanja putovanja. Obrađeno je nekoliko studija slučaja hrvatskih tvrtki koje se bave različitim granama gospodarstva (dostava poštanskih pošiljaka, dostava tiskovina, distribucija robe široke potrošnje, pekarskih proizvoda i farmaceutskih proizvoda). Podaci prikupljeni u prethodnim istraživanjima iskorišteni su za modeliranje praktičnih VRP problema, te su isti riješeni pomoću sinkrone inačice HCEA algoritma. Rezultati su pokazali da se primjenom metaheurističkih algoritama isti posao dostave ili distribucije može obaviti uz značajno manje transportne troškove. U pojedinim slučajevima uštede su veće od 30% u broju korištenih vozila, te veće od 50% u ukupnoj pređenoj udaljenosti. Predloženi HCEA algoritam poboljšao je rezultate koji su na istim problemima ostvareni u prethodnom istraživanju koristeći simulirano kaljenje, genetski algoritam i optimizaciju mravljom kolonijom.

Postoji niz mogućnosti za poboljšanje predloženog HCEA algoritma. U prvom redu to su dodatne varijante uništavanja rješenja kod kojih bi se iz ruta izbacivali "povezani" korisnici. Kod operatora križanja također se može primijeniti ovo pravilo u slučaju kada se rješenja razlikuju u većoj mjeri. Nadalje, stanični evolucijski algoritmi u svojoj su naravi paralelni pa ih je relativno jednostavno prilagoditi za rad na višeprocesorskim sustavima što bi omogućilo ubrzanje izvršavanja ili korištenje matrica većih dimenzija uz jednako vrijeme izračuna kao u sekvencijalnoj inačici algoritma. Algoritam bi se znatno unaprijedio kada bi se poboljšala i/ili ubrzala procedura popravljanja koja je trenutno računski najzahtjevniji dio algoritma. Također, unaprjeđenje postupka prihvaćanja novih rješenja pomoglo bi u održavanju dovoljno velike raznolikosti u populaciji. Konačno, automatizacijom parametara algoritma koji bi bili prilagođeni dimenzijama problema, postigla bi se veća robusnost. Kao najveći nedostatak predloženog algoritma može se izdvojiti brojnost parametara i njihova međusobna ovisnost ali i ovisnost o dimenzijama problema koji se rješava.

Literatura

[1] E. Alba and B. Dorronsoro. Solving the vehicle routing problem by using cellular genetic algorithms. *Lecture Notes in Computer Science*, 3004:11–20, 2004.

[2] E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142, 2005.

[3] E. Alba and B. Dorronsoro. Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm. *Information Processing Letters*, 98(6):225–230, 2006.

[4] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Springer-Verlag, Berlin, Germany, 2008.

[5] E. Alba, M. Giacobini, M. Tomassini, and S. Romero. Comparing synchronous and asynchronous cellular genetic algorithms. In J. J. M. Guervós, P. Adamidis, H-G. Beyer, H-P. Schwefel, and J-L. Fernández-Villacañas, editors, *Parallel Problem Solving from Nature - PPSN VII: 7th International Conference Granada, Spain, September 7-11, 2002 Proceedings*, pages 601–610, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[6] E. Alba and J. M. Troya. Cellular evolutionary algorithms: Evaluating the influence of ratio. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18-20, 2000 Proceedings*, pages 29–38, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[7] G. B. Alvarenga, R. M. de Abreu Silva, and R. M. Sampaio. A hybrid algorithm for the vehicle routing problem with time window. *INFOCOMP Journal of Computer Science*, 4(2):9–16, June 2005.

[8] P. Augerat. *Approche polyédrale du probleme de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1995.

[9] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

[10] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

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

[12] J. Berger, M. Barkaoui, and O. Bräysy. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Working paper, Defense Research Establishment Valcartier, Canada, 2001.

[13] J. Berger, M. Barkaoui, and O. Bräysy. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *Information Systems and Operational Research*, 41:179–194, 2003.

[14] J. Berger and M. Salois. A hybrid genetic algorithm for the vehicle routing problem with time windows. Technical report, Defence Research and Development Canada - Valcartier, 2004.

[15] J. L. Blanton and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 452–459, San Francisco, CA, 1993. Morgan Kaufmann Publishing.

[16] Miroslaw Blocho and Zbigniew J. Czech. A parallel memetic algorithm for the vehicle routing problem with time windows. In *Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013, Compiegne, France, October 28-30, 2013*, pages 144–151, 2013.

[17] C. Blum and A. Roli. *In C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels, editors, Hybrid Metaheuristics – An Emerging Approach to Optimization*, chapter Hybrid Metaheuristics: An Introduction. Springer, 2008.

[18] O. Bräysy and M. Gendreau. Tabu search heuristics for the vehicle routing problem with time windows. *TOP*, 10(2):211–237, 2002.

[19] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[20] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[21] O. Bräysy, P. Nakari, W. Dullaert, and P. Neittaanmaki. An optimization approach for communal home meal delivery service: A case study. *Journal of Computational and Applied Mathematics*, 232(1):46–53, 2009.

[22] B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank-based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38, 1999.

[23] T. Carić. Unapređenje organizacije transporta primjenom heurističkih metoda. Doktorska disertacija, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2004.

[24] T. Carić, A. Galić, J. Fosin, H. Gold, and A. Reinholz. *Vehicle Routing Problem*, chapter A Modelling and Optimization Framework for Real-World Vehicle Routing Problems, pages 15–34. I-Tech, Vienna, Austria, 2008.

[25] N. Christofides and S. Eilon. An algorithm for the vehicle routing dispatching problem. *Operations Research Quaterly*, 20:309–318, 1969.

[26] N. Christofides, A. Mingozzi, and P. Toth. *In N. Christofides, A. Mingozzi, P. Toth and C. Sandi, editors, Combinatorial Optimization*, chapter The vehicle routing problem, pages 315–338. Wiley, Chichester, 1979.

[27] G. Clarke and J. V. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[28] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of ECAK91 - European Conference on Artificial Life*, Paris, France, 1991.

[29] C. Contardo and R. Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146, 2014.

[30] J. F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. *In P. Toth and D. Vigo, editors, The Vehicle Routing Problem*, chapter VRP with Time Windows. SIAM Monographs on Discrete Mathematics and Applications, 3600 University City Science Center, Philadelphia, PA 19104-2688, 2002.

[31] J. F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. *In C. Barnhart and G. Laporte, editors, Handbook in Operations Research and Management Science, Vol. 14*, chapter Vehicle Routing. Elsevier B.V., Berlin, 2007.

[32] J.F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 2017.

[33] CVRPLIB. Capacitated vehicle routing problem library. http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. pristupljeno 17.08.2017.

[34] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, October 1959.

[35] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of lnsect Behavior*, 3(2):159–168, 1990.

[36] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185:1174–1191, 2008.

[37] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[38] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical report, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[39] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):1–13, 1996.

[40] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

[41] B. Dorronsoro, E. Alba, M. Giacobini, and M. Tomassini. The influence of grid shape and asynchronicity on cellular evolutionary algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 2, pages 2152–2158. IEEE, 2004.

[42] B. Dorronsoro, D. Arias, F. Luna, A. J. Nebro, and E. Alba. A grid-based hybrid cellular genetic algorithm for very large scale instances of the cvrp. In *Proceedings of the 21st European Conference on Modelling and Simulation, Prague, Czech Republic, June 4th - 6th, 2007*, pages 759–765, 2007.

[43] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization – Methods and Case Studies*. Springer, 2006.

[44] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal Of Optimization Theory And Applications*, 45(1):41–51, 1985.

[45] M. A. Figliozzi. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E*, 48:616–636, 2012.

[46] M. Filipec, D. Škrlec, and S. Krajcar. Darwin meets computers: New approach to multiple depot capacitated vehicle routing problem. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 421–426, 1997.

[47] M. Filipec, D. Škrlec, and S. Krajcar. Genetic algorithm approach for multiple depot capacitated vehicle routing problem solving with heuristic improvements. *International Journal of Modelling & Simulation*, 20(4):320–328, 2000.

[48] M. L. Fisher. Optimal solution of vehicle routing problems using minimum K-trees. *Operations Research*, 42:626–642, 1994.

[49] J. Fosin. *Metoda rješavanja vremenski ovisnoga problema usmjeravanja vozila zasnovana na profilima brzina*. PhD thesis, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2016.

[50] J. Fosin, T. Carić, and E. Ivanjko. Vehicle routing optimization using multiple local search improvements. *Automatika*, 55(2):124–132, 2014.

[51] J. Fosin, D. Davidović, and T. Carić. A GPU implementation of local search operators for symmetric travelling salesman problem. *Promet - Traffic & Transportation*, 25(3):225–234, 2013.

[52] P. M. Francis, K. R. Smilowitz, and M. Tzur. *In B. Golden, S. Raghavan, and E. Wasil, editors, The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter The Period Vehicle Routing Problem and its Extensions. Springer, New York, USA, 2008.

[53] A. Galić. Metaheurističke metode rješavanja problema usmjeravanja vozila s vremenskim prozorima. Znanstveni magistarski rad, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2012.

[54] A. Galić, T. Carić, and J. Fosin. The case study of implementing the delivery optimization system at a fast-moving consumer goods distributor. *Promet - Traffic & Transportation*, 25(6):595–603, 2013.

[55] L. M. Gambardella, E. Taillard, and G. Agazii. *New Ideas In Optimization*, chapter MACS-VRPTW: A multiple ant colony system for vehicle routing problem with time windows, pages 63–76. McGraw-Hill, London, UK, 1999.

[56] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. *Proc. of EUROGEN*, 99:57 – 64, 1999.

[57] M. Gendreau, J-Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen. *In B. Golden, S. Raghavan, and E. Wasil, editors, The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. Springer, 2008.

[58] M. Giacobini, E. Alba, A. Tettamanzi, and M. Tomassini. Modeling selection intensity for toroidal cellular evolutionary algorithms. In D. Kalyanmoy, editor, *Genetic and Evolutionary Computation - GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*, pages 1138–1149, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[59] M. Giacobini, E. Alba, and M. Tomassini. Selection intensity in asynchronous cellular evolutionary algorithms. In *Genetic and Evolutionary Computation - GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12-16, 2003. Proceedings, Part I*, pages 955–966, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[60] M. Giacobini, M. Tomassini, A. G. B. Tettamanzi, and E. Alba. Selection intensity in cellular evolutionary algorithms for regular lattices. *IEEE Transactions on Evolutionary Computation*, 9(5):489–505, 2005.

[61] B. Gillett and L. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22:340–349, 1974.

[62] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison Wesley Longman, Inc., 1989.

[63] B. L. Golden, E. A. Wasil, J. P. Kelly, and I. M. Chao. *In T. G. Crainic and G. Laporte, editors, Fleet Management and Logistics*, chapter Metaheuristics in Vehicle Routing, pages 33–56. Boston: Kluwer, 1998.

[64] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[65] M. Gulić and D. Jakobović. Evolution of vehicle routing problem heuristics with genetic programming. In *MIPRO, Proceedings of the 36th International Convention*, pages 988–992, Opatija, Croatia, 2013.

[66] K. Helsgaun. LKH. http://www.akira.ruc.dk/ keld/research/LKH/. pristupljeno 07.04.2014.

[67] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Information Systems and Operational Research*, 37:297–318, 1999.

[68] S. Irnich, M. Schneider, and D. Vigo. *In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, Second Edition*, chapter Four Variants of the Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, USA, 2014.

[69] Potvin J.-Y. and S. Bengio. The vehicle routing problem with time windows - part II: Genetic search. *INFORMS Journal on Computing*, 8:165–172, 1996.

[70] I. Kamkar, M. Poostchi, and M. R. Akbarzadeh Totonchi. *In Xiao-Zhi Gao and António Gaspar-Cunha and Mario Köppen and Gerald Schaefer and Jun Wang, editors, Soft Computing in Industrial Applications: Advances in Intelligent and Soft Computing 75*, chapter A Cellular Genetic Algorithm for Solving the Vehicle Routing Problem with Time Windows, pages 263–270. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[71] K.Ghoseiri and S. F. Ghannadpour. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10:1096–1107, 2010.

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

[73] S. Kritzinger, K. F. Doerner, R. F. Hartl, G. Kiechle, H. Stadler, and S. S. Manohar. Using traffic information for time-dependent vehicle routing. *Procedia - Social and Behavioral Sciences*, 39:217 – 229, 2012.

[74] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 2014.

[75] G. Laporte and F. Semet. *In P. Toth and D. Vigo, editors, The Vehicle Routing Problem*, chapter Classical Heuristics for the Capacitated VRP. Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688, 2002.

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

[77] F. Li, B. Golden, and E. Wasil. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research*, 34(10):2918–2930, 2007.

[78] S. Lin. Computer solutions of the traveling salesman problem. *Bell Systems Journal*, 44(10):2245–2269, 1965.

[79] B.T. Luke. Simulated annealing cooling schedules. http://www.btluke.com/simanf1.html. pristupljeno 03.09.2017.

[80] H. Marković. Mining spatio-temporal data for travel time estimation in urban traffic networks. Master's thesis, University of Zagreb, Faculty of Electrical Engineering And Computing, 2007.

[81] J. F. D. Martín and J. M. R. Sierra. *In J. R. R. Dopico, J. D. de la Calle, and A. P. Sierra, editors, Encyclopedia of Artificial Intelligence*, chapter A Comparison of Cooling Schedules for Simulated Annealing. Information Science Reference, 701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, 2009.

[82] SINTEF Applied Mathematics. Transportation optimization portal. http://www.sintef.no/projectweb/top/. pristupljeno 19.05.2017.

[83] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1091, June 1953.

[84] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2002.

[85] A. Munawar, M. Wahib, M. Munetomo, and K. Akama. Implementation and optimization of cGA+LS to solve capacitated VRP over Cell/B.E. *International Journal of Advancements in Computing Technology*, 1(2):16–28, 2009.

[86] Y. Nagata. Edge assembly crossover for the capacitated vehicle routing problem. In *Evolutionary Computation in Combinatorial Optimization: 7th European Conference, EvoCOP 2007, Valencia, Spain, April 11-13, 2007. Proceedings*, pages 142–153, 2007.

[87] Y. Nagata. Efficient evolutionary algorithm for the vehicle routing problem with time windows: edge assembly crossover for the VRPTW. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007*, pages 1175–1182, 2007.

[88] Y. Nagata, O. Bräysy, and W. Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research*, 37:724–737, April 2010.

[89] Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 450–457, 1997.

[90] J. Nalepa and M. Blocho. Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows. *Soft Computing*, 20(6):2309–2327, 2016.

[91] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Comput. Oper. Res.*, 34(8):2403–2435, 2007.

[92] D. Pisinger and S. Ropke. *In M. Gendreau and J-Y. Potvin, editors, Handbook of Metaheuristics*, chapter Large Neighborhood Search, pages 399–419. Springer US, Boston, MA, 2010.

[93] J-Y. Potvin. State-of-the art review - evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548, 2009.

[94] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[95] K. Puljić and R. Manger. Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18(2):359–375, 2013.

[96] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[97] S. Ropke and D. Pisinger. A unified heuristic for vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006.

[98] J. Sarma and K. A. De Jong. An analysis of the effects of neighborhood size and shape on local selection algorithms. In *Parallel Problem Solving from Nature - PPSN IV, International Conference on Evolutionary Computation. The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26, 1996, Proceedings*, pages 236–244, 1996.

[99] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):11–29, 1995.

[100] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139 – 171, April 2000.

[101] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming - CP98, 4th International Conference, Pisa, Italy, October 26-30, 1998, Proceedings*, pages 417–431, 1998.

[102] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[103] T. Stützle and H. Hoos. The max-min ant system and local search for the traveling salesman problem. *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 309–314, 1997.

[104] A. Subramanian, L. S. Ochi, and L. dos A. F. Cabral. An efficient ils heuristic for the vehicle routing problem with simultaneous pickup and delivery. Technical Report RT 07/08, Relatório Técnico, Universidade Federal Fluminense, 2008.

[105] E. Taillard. Éric Taillard web page. http://mistic.heig-vd.ch/taillard/problemes.dir/vrp.dir/vrp.html, 2017. pristupljeno 17.08.2017.

[106] K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, 15(3):281–295, 2001.

[107] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[108] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2014.

[109] D. Vilić. Integracija programskog rješenja za optimizaciju ruta "OPTiRUT" u poslovni sustav distribucijske tvrtke Orbico d.o.o. Diplomski rad, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2013.

[110] T. Weise, A. Podlich, and C. Gorldt. *Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms*, volume 250, pages 29–53. Springer-Verlag, Berlin, Germany, 2009.

[111] L. D. Whitley, T. Starkweather, and D. Fuquay:. *Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator*, pages 133–140. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[112] M. Woch and P. Łebkowski. Sequential simulated annealing for the vehicle routing problem with time windows. *Decision Making in Manufacturing and Services*, 3(1-2):87–100, 2009.

[113] E. E. Zachariadis and C. T. Kiranoudis. An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Computers & Operations Research*, 37(4):712–723, 2010.

[114] Mireo d.d. Zagreb. Mireo web page. http://www.mireo.hr, 2017. pristupljeno 25.08.2017.

Popis slika

Popis tablica

Popis algoritama

Popis kratica

ACO	*Ant Colony Optimization* Optimizacija mravljom kolonijom
AS	*Ant System* Mravlji sustav
BT	*Binary Tournament Selection* Binarna turnirska selekcija
cEA	*Cellular Evolutionary Algorithm* Stanični evolucijski algoritam
cGA	*Cellular Genetic Algorithm* Stanični genetski algoritam
CVRP	*Capacitated Vehicle Routing Problem* Problem usmjeravanja vozila s ograničenjima kapaciteta
CX	*Cycle Crossover* Cikličko križanje
DARP	*Dial-a-ride Problem* Problem prijevoza na poziv
EA	*Evolutionary algorithm* Evolucijski algoritam
EAX	*Edge Assembly Crossover* Križanje zasnovano na sklapanju bridova
ERX	*Edge Recombination Crossover* Križanje zasnovano na rekombinaciji bridova
FRS	*Fixed Random Sweep* Fiksno zadani nasumični prolaz
FSM	*Fleet Size and Mix Problem* Problem veličine i kombiniranja flote
GA	*Genetic Algorithm* Genetski algoritam
GIS	*Geographic Information System* Geografski informacijski sustav

GNSS	*Global Navigation Satellite System* Globalni navigacijski satelitski sustav
GPS	*Global Positioning System* Sustav globalnog pozicioniranja
GPGPU	*General-purpose Graphics Processing Unit* Grafička procesorska jedinica za opću namjenu
HCEA	*Hybrid Cellular Evolutionary Algorithm* Hibridni stanični evolucijski algoritam
HFVRP	*Heterogeneous or Mixed Fleet Vehicle Routing Problem* Problem usmjeravanja mješovite flote vozila
HVRP	*Heterogeneous Vehicle Routing Problem* Problem usmjeravanja heterogene flote vozila
ILS	*Iterated Local Search* Iterativna lokalna pretraga
LNS	*Large Neighborhood Search* Pretraživanje velikog susjedstva
LR	*Linear Ranking Selection* Selekcija linearnim rangiranjem
LS	*Linear Sweep* Linearni prolaz
m-TSP	*Multiple Traveling Salesman Problem* Višestruki problem trgovačkog putnika
MDVRP	*Multi-depot Vehicle Routing Problem* Višeskladišni problem usmjeravanja vozila
MSO	*Randomly Selected Customer Removal Mutation* Mutacija izbacivanjem slučajno odabranih korisnika
MKO	*Radially Selected Customer Removal Mutation* Mutacija izbacivanjem kružno odabranih korisnika
NNH	*Nearest Neighbor Heuristic* Heuristika najbližeg susjeda
NRS	*New Random Sweep* Novi nasumični prolaz
OVRP	*Open Vehicle Routing Problem* Problem otvorenog usmjeravanja vozila
OX	*Order-based Crossover* Slijedno križanje
PDP	*Pickup and Delivery Problem* Problem prikupljanja i dostave
PMX	*Partially Matched Crossover* Djelomično usklađeno križanje
PVRP	*Periodic Vehicle Routing Problem* Problem periodičkog usmjeravanja vozila

R&R	*Ruin and Recreate* Djelomično uništavanje i popravljanje
RIH	*Regret Insertion Heuristic* Heuristika umetanja sa žaljenjem
RVRP	*Rich Vehicle Routing Problem* Obogaćeni problem usmjeravanja vozila
RW	*Roulette Wheel Selection* Selekcija pomoću kotača ruleta
SA	*Simulated Annealing* Simulirano kaljenje
SDVRP	*Split Delivery Vehicle Routing Problem* Problem usmjeravanja vozila s podijeljenom dostavom
SPP	*Set Partitioning Problem* Problem particioniranja skupa
TDVRPTW	*Time-dependent Vehicle Routing Problem with Time Windows* Vremenski ovisan problem usmjeravanja vozila s vremenskim ograničenjima
TONNH	*Time-oriented Nearest Neighbor Heuristic* Vremenski-orijentirana heuristika najbližeg susjeda
TSP	*Traveling Salesman Problem* Problem trgovačkog putnika
UC	*Uniform Choice* Jednoliki odabir
VRP	*Vehicle Routing Problem* Problem usmjeravanja vozila
VRPB	*Vehicle Routing Problem with Backhauls* Problem usmjeravanja vozila s povratnim prikupljanjem
VRPPD	*Vehicle Routing Problem with Pickup and Delivery* Problem usmjeravanja vozila s prikupljanjem i dostavom
VRPSD	*Vehicle Routing Problem with Stochastic Demands* Problem usmjeravanja vozila sa stohastičkom potražnjom
VRPSDP	*Vehicle Routing Problem with Simultaneous Delivery and Pickup* Problem usmjeravanja vozila s istovremenom dostavom i prikupljanjem
VRPTW	*Vehicle Routing Problem with Time Windows* Problem usmjeravanja vozila s vremenskim ograničenjima
$\mathcal{MAX} - \mathcal{MIN}$	$\mathcal{MAX} - \mathcal{MIN}$ *Ant System* $\mathcal{MAX} - \mathcal{MIN}$ mravlji sustav

Životopis

Ante Galić rođen je 14. listopada 1978. godine u Širokom Brijegu (Bosna i Hercegovina) gdje je završio osnovnu školu i gimnaziju. Godine 1997. upisao je redoviti studij na Fakultetu prometnih znanosti Sveučilišta u Zagrebu, smjer pošta i telekomunikacije gdje je diplomirao 2004. godine i stekao zvanje diplomiranog inženjera prometa. U lipnju iste godine zaposlio se na Fakultetu prometnih znanosti Sveučilišta u Zagrebu kao mladi istraživač na tehnologijskom projektu Ministarstva znanosti i tehnologije "CRO-GRID Aplikacije - Optimizacija organizacije transporta" (STIRP-42/2002, TP098/2003-34, 2004-2006). U listopadu 2004. godine upisao je poslijediplomski znanstveni magistarski studij "Tehničko-tehnološki sustavi u prometu i transportu", a u prosincu 2007. godine, doktorski studij "Tehnološki sustavi u prometu i transportu" na Fakultetu prometnih znanosti Sveučilišta u Zagrebu. U suradničko zvanje asistenta iz područja tehničkih znanosti, polje tehnologija prometa i transport, izabran je u prosincu 2006. godine. Magistrirao je u prosincu 2012. obranivši znanstveni magistarski rad pod naslovom "Metaheurističke metode rješavanja problema usmjeravanja vozila s vremenskim prozorima". Do 2014. godine kao asistent na Fakultetu prometnih znanosti bio je nastavno angažiran na kolegijima Algoritmi i programiranje, Baze podataka, Napredne baze podataka i Lokacijski i navigacijski sustavi. Autor je i koautor više znanstvenih radova te je sudjelovao i izlagao na više međunarodnih kongresa i skupova. Upisan je u registar znanstvenika Ministarstva znanosti i tehnologije Republike Hrvatske pod matičnim brojem 291731. Autor je softvera za optimizaciju ruta dostavnih vozila "OPTiRUT" koji je uspješno implementiran u tvrtki Orbico, d.o.o. iz Zagreba. Od 2015. zaposlen je u poduzeću GCR d.o.o. iz Širokog Brijega i trenutno se bavi razvojem komercijalnog softvera za rješavanje različitih varijanti problema usmjeravanja vozila. Oženjen je i otac jednog djeteta. Aktivno se služi engleskim jezikom.

Curriculum Vitae

Ante Galić was born on 14th of October, 1978. at Široki Brijeg (Bosnia and Herzegovina) where he attended and finished elementary and high school. In 1997. he continued his education at the Faculty of Transport and Traffic Sciences, University of Zagreb, at the department of Post and Telecommunications where he graduated in 2004. and earned his dipl. ing. degree. In June 2004. he started to work at the Faculty of Transport and Traffic Sciences, University of Zagreb, as young researcher on a technological project supported by Ministry of Science and Technology "CRO-GRID Applications - Optimization of Transport Organisation" (STIRP-42/2002, TP098/2003-34, 2004-2006). In October 2004. he continued his education at master studies "Technical and Technological Systems in Transport and Traffic", and in December, 2007. doctoral studies "Technological Systems in Transport and Traffic" both at the Faculty of Transport and Traffic Sciences, University of Zagreb. He became a research assistant in area of technical sciences, field of transport and traffic technology, in December, 2006. He earned his master degree in December, 2012. by defending thesis titled "Metaheuristic methods for solving vehicle routing problem with time windows". Until 2014. as research assistant at the Faculty of Transport and Traffic Sciences he participated in teaching of the following courses: Algorithms and Programming, Database Systems, Advanced Database Systems, and Location and Navigation Systems. He is author and coauthor of several scientific papers, and he participated and presented his scientific works at several international conferences. He is author of software for optimization of delivery routes "OPTiRUT" which was successfully implemented in company Orbico d.o.o. from Zagreb. Since 2015. he works at company GCR d.o.o. from Široki Brijeg and is currently developing commercial software for solving different types of vehicle routing problems. He is married and father of one child. He speaks English fluently.

Popis objavljenih djela

1. A. Galić, T. Carić, and J. Fosin. The case study of delivery optimization system implementation at fast-moving consumer goods distributor. *Promet - Traffic & Transportation*, 25(6):595-603, 2013.

2. F. Taner, A. Galić, and T. Carić. Solving practical vehicle routing problem with time windows using metaheuristic algorithms. *Promet-Traffic & Transportation*. 24(4):343-351, 2012.

3. A. Galić. Metaheurističke metode rješavanja problema usmjeravanja vozila s vremenskim prozorima. Znanstveni magistarski rad, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2012.

4. T. Carić, A. Galić, J. Fosin, H. Gold, and A. Reinholz. *Vehicle Routing Problem*, chapter A modelling and optimization framework for real-world vehicle routing problems. pages 15-34. I-Tech, Vienna, Austria, 2008.

5. T. Carić, J. Fosin, A. Galić, H. Gold, and A. Reinholz. Empirical analysis of two different metaheuristics for real-world vehicle routing problems. *Lecture Notes in Computer Science*. LNCS 4771, pages 31-44, Berlin, Heidelberg, 2007.

6. A. Galić, T. Carić, J. Fosin, I. Ćavar, and H. Gold. Distributed solving of the VRPTW with coefficient weighted time distance and lambda local search heuristics. *Proceedings of the 29th International Convention MIPRO*, pages 247-252, Opatija, Croatia, 2006.

7. A. Galić, T. Carić, and H. Gold. MARS - A programming language for solving vehicle routing problems. *Recent Advances in City Logistics - Proceedings of the 4th International Conference on City Logistics*. pages 47-57. Elsevier, Netherlands, 2006.

8. T. Carić, A. Galić, and H. Gold. Solving a practical newspaper delivery problem. *INFORMS Annual Meeting / Pittsburgh : INFROMS, 2006*. pages 198-198, Pittsburgh, USA, 2006.

9. A. Galić, and T. Carić, and J. Fosin. Distributed solving of bakery products delivery problem. *Joint Conference Of The Canadian Operational Research Society And Optimization Days.* pages 59-60, Montreal, Canada, 2006.

10. T. Carić, H. Gold, and A. Galić. Interactive programming Environment for solving vehicle routing problem. *Proceedings of the European Congress on Intelligent Transport Systems : ITS in Europe 2004 : Moving Towards an Integrated Europe.* Budapest, Hungary, 2004.