UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Nataša Đ. Kovač

METAHEURISTIČKI PRISTUP REŠAVANJU JEDNE KLASE OPTIMIZACIONIH PROBLEMA U TRANSPORTU

doktorska disertacija

Beograd, 2018.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Nataša Đ. Kovač

METAHEURISTIC APPROACH FOR SOLVING ONE CLASS OF OPTIMIZATION PROBLEMS IN TRANSPORT

Doctoral Dissertation

Belgrade, 2018.

Mentor:

prof. dr Zorica Stanimirović, vanredni profesor
Univerzitet u Beogradu, Matematicki fakultet

Članovi komisije:

dr Tatjana Davidović, naučni savetnik
Matematički institut, SANU

prof. dr Miodrag Živković, redovni profesor
Univerzitet u Beogradu, Matematicki fakultet

prof. dr Milan Dražić, vanredni profesor
Univerzitet u Beogradu, Matematicki fakultet

Datum odbrane: _____

Mojoj voljenoj porodici za podršku i strpljenje

Zahvalnice

Svom mentoru, prof. dr Zorici Stanimirović želim da se zahvalim na konstruktivnim sugestijama i beskonačnoj podršci tokom ovog dugog putovanja. Dr Tatjani Davidović se zahvaljujem na stručnoj pomoći u kreiranju ideja vodilja i korisnim savetima tokom izrade ove teze. Iako zvanično nije mogla da ima tu ulogu, uvek ću je smatrati za svog drugog mentora. Poštovana Zorice, poštovana Tatjana, ovaj rad ne bi bio moguć bez Vaše sposobnosti da mi konstantno stimulišete kreativnost i maštu. Osećam najdublje poštovanje za Vas kao za naučnice, nastavnike i moje prijateljice. Čast je i privilegija raditi uz Vašu pomoć. Dali ste mi snagu, nadu i veru, onda kad sam mislila da je sve izgubljeno. Gurale ste me napred i učinile da ova disertacija ugleda svetlost dana.

Članovima komisije se zahvaljujem što su imali strpljenja da koriguju početnu verziju rukopisa i što su na taj način doprineli da rad bude kvalitetniji i sadržajniji.

Želim da se zahvalim i dr Stevanu Kordiću koji je bio moja inspiracija u toku izrade teze. Hvala ti na svim diskusijama koje smo vodili u proteklih par godina. Tvoji korisni saveti i tvoj uticaj su nemerljivi, iako ti to nikada nećeš priznati ni sebi ni drugima. Zahvaljujem se mojim prijateljima, mojoj skrivenoj snazi, koji su me podržali kad je život odlučio da stavi prepreke na put kojim idem.

Ne bih bila tu gde jesam bez podrške i ljubavi moje porodice. Sve vam dugujem, i nadam se da i vi to znate. Nema tih reči sa kojima bih mogla iskazati svu zahvalnost koju osećam. Podelili ste sa mnom i besane noći, i kišne dane, i osmeh i tugu... Trpeli ste moju narav i pored svega me voleli, čak i kad ja sebe nisam. Dragi Dejane, najbolji si sin na celom svetu, moje si sve. Volim te više od života, nikad ne zaboravi.

Naslov disertacije: Metaheuristički pristup rešavanju jedne klase optimizacionih problema u transportu

Rezime: Problem dodele vezova obuhvata nekoliko važnih odluka koje je potrebno doneti da bi se dosegla maksimalna efikasnost luke. U luci, menadžeri terminala treba da dodele slobodne vezove brodovima koji su najavili dolazak. Neophodno je da se ta dodela izvrši na način koji će voditi ka optimalnoj vrednosti zadate funkcije cilja. Kako su čak i jednostavnije varijante problema alokacije vezova NP-teške, metaheuristički pristup u rešavanju ovog problema je mnogo pogodniji od egzaktnih metoda, jer metaheuristike pronalaze kvalitetna rešenja u razumno kratkom vremenu izvršavanja. Ova disertacija razmatra problem hibridne alokacije vezova sa fiksnim vremenom obrade brodova i sa ciljem minimizacije troškova (engl. *Minimum Cost Hybrid Berth Allocation Problem*-MCHBAP) i to u dve varijante: statičkoj (MCHBAP) i dinamičkoj (engl. *Dynamic Minimum Cost Hybrid Berth Allocation Problem*-DMCHBAP). U obe varijante problema, zadatak je minimizovati funkciju cilja koja se sastoji od više komponenata: troškova pozicioniranja, troškova ubrzavanja broda, troškova čekanja broda i troškova kašnjenja u obradi brodova. Imajući u vidu da je brzina pronalaženja visoko kvalitetnih rešenja od presudnog značaja za dizajniranje efikasnog i pouzdanog sistema podrške odlučivanju u kontejnerskom terminalu, metaheurističke metode predstavljaju prirodan izbor za rešavanje MCHBAP-a i DMCHBAP-a. U ovoj doktorskoj disertaciji, za obe vatijante problema razmatrani su sledeći metaheuristički pristupi: dve varijante optimizacije kolonijom pčela (engl. *Bee Colony Optimization*-BCO), dve varijante evolutivnog algoritma (engl. *Evolutionary Algorithm*-EA) i četiri varijante metode promenljivih okolina (engl. *Variable Neighborhood Search*-VNS). Sve metaheuristike su testirane na istancama problema iz literature koje su dobijene na osnovu realnih podataka, kao i na generisanim instancama većih dimenzija koje ranije nisu razmatrane u literaturi. Dobijeni rezultati predloženih metaheuristika su međusobno upoređeni, a izvršeno je i poređenje sa rezultatima egzaktih metoda koje su ugrađene u rešavač CPLEX. Analiza dobijenih rezultata pokazuje da su na realnim instancama sve metaheuristike uspele da pronađu optimalno rešenje problema u kratkom vremenu izvršavanja. Egzaktni algoritam, usled nedostatka vremena ili memorijskog prostora nije mogao da reši kompjuterski generisane instance dok su metaheuristike prilikom svakog izvršavanja u kratkom vremenu izvršavanja dale visoko kvalitetna rešenja. Analizom eksperimentalnih rezultata, može se zaključiti

da predloženi metaheuristički pristupi predstavljaju pogodne metode za rešavanje MCHBAP-a, DMCHBAP-a i sličnih problema u pomorskom transportu.

Rezultati prikazani u ovoj disertaciji predstavljaju doprinos oblastima kombinatorne optimizacije, operacionih istraživanja, metaheurističkih metoda i problemu dodele vezova u kontejnerskim terminalima.

Ključne reči: Kontejnerski terminal, Dodela vezova brodovima, Optimizacija, Metaheuristički pristup, Penali, Minimizacija troškova

Naučna oblast: Matematika

Uža naučna oblast: Optimizacija, Operaciona istraživanja

Dissertation title: Metaheuristic approach for solving one class of optimization problems in transport

Abstract: Berth Allocation Problem incorporates some of the most important decisions that have to be made in order to achieve maximum efficiency in a port. Terminal manager of a port has to assign incoming vessels to the available berths, where they will be loaded/unloaded in such a way that some objective function is optimized. It is well known that even the simpler variants of Berth Allocation Problem are NP-hard, and thus, metaheuristic approaches are more convenient than exact methods, because they provide high quality solutions in reasonable computational time. This study considers two variants of the Berth Allocation Problem: Minimum Cost Hybrid Berth AllocationProblem (MCHBAP) and Dynamic Minimum Cost Hybrid Berth AllocationProblem (DMCHBAP), both with fixed handling times of vessels. Objective function to be minimized consists of the following components: costs of positioning, speeding up or waiting of vessels, and tardiness of completion for all vessels. Having in mind that the speed of finding high-quality solutions is of crucial importance for designing an efficient and reliable decision support system in container terminal, metaheuristic methods represent the natural choice when dealing with MCHBAP and DMCHBAP. This study examines the following metaheuristic approaches for both types of a given problem: two variants of the Bee Colony Optimization (BCO), two variants of the Evolutionary Algorithm (EA), and four variants of Variable Neighborhood Search (VNS). All metaheuristics are evaluated and compared against each other and against exact methods integrated in commercial CPLEX solver on real-life instances from the literature and randomly generated instances of higher dimensions. The analysis of the obtained results shows that on real-life instances all metaheuristics were able to find optimal solutions in short execution times. Randomly generated instances were out of reach for exact solver due to time or memory limits, while metaheuristics easily provided high-quality solutions in short CPU time in each run. The conducted computational analysis indicates that metaheuristics represent a promising approach for MCHBAP and similar problems in maritime transportation.

The results presented in this paper represent a contribution to the fields of combinatorial optimization, operational research, metaheuristic methods, and berth allocation problem in the container terminals.

Keywords: Container terminal, Assignment of berths to vessels, Optimization, Metaheuristic approach, Penalties, Cost minimization

Research area: Mathematics

Research sub-area: Optimization, Operational research

Sadržaj

Glava 1

Uvod

Problemi optimizacije

Optimizacija je proces u kome se pronalazi ekstremna vrednost neke funkcije pri zadatim uslovima. *Problemi globalne optimizacije* bave se pronalaženjem globalnog optimuma za zadatu funkciju nad skupom ulaznih promenljivih uz definisan skup ograničenja. Skup $D \subseteq S$ definiše skup kojim se opisuju ograničenja i naziva se *prostor pretraživanja* a njegovi elementi s su *dopustiva rešenja* problema optimizacije. *Kombinatorna optimizacija* je grana globalne optimizacije kod koje je skup dopustivih rešenja D konačan ili prebrojiv. Neka je funkcija f definisana nad skupom S. U opštem slučaju se problem globalne optimizacije može zapisati na sledeći način:

$$\min\{f(s) : s \in D\} \tag{1.1}$$

gde je $f(s)$ funkcija koju treba minimizovati a s označava dopustivo rešenje. Dopustivo rešenje $s^* \in D$ problema (1.1) je *optimalno* ako važi

$$(\forall s \in D) \quad f(s^*) \leq f(s) \tag{1.2}$$

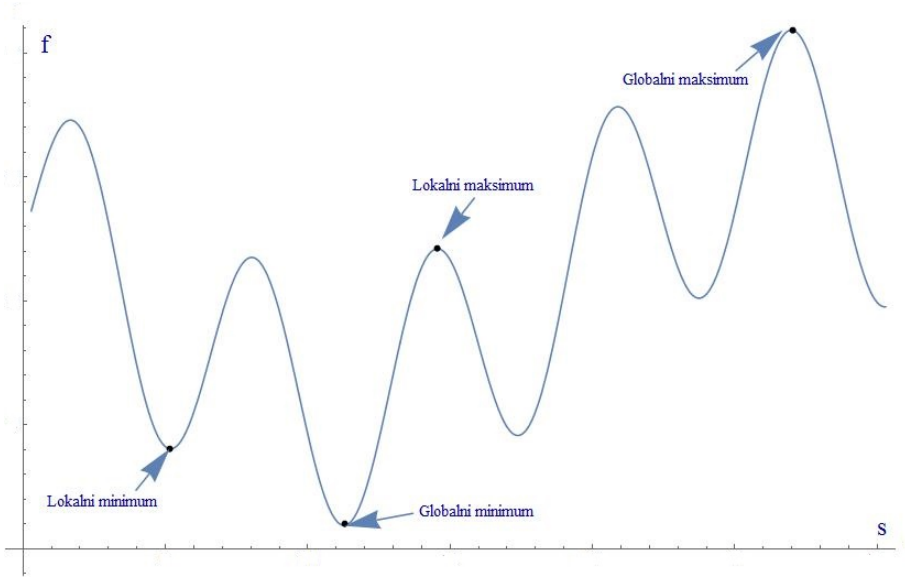
Analogno se definiše maksimizacioni problem i uz iste oznake se može zapisati na sledeći način:

$$\max\{f(s) : s \in D\} \tag{1.3}$$

Rešenje s^* problema (1.3) je optimalno ako važi

$$(\forall s \in D) \quad f(s^*) \geq f(s) \tag{1.4}$$

Funkcija f koju je potrebno minimizovati ili maksimizovati se naziva *funkcija cilja*. Funkcije koje najčešće treba minimizovati definišu cenu, dužinu puta, procesorsko vreme, udaljenost, itd., dok se u slučaju funkcija koje opisuju profit, efikasnost, kapacitet, broj objekata najčešće zahteva maksimizacija. Cilj problema optimizacije je pronaći jedno ili više optimalnih rešenja, ukoliko je to moguće, a u suprotnom, odrediti jedno ili više rešenja koja su bliska optimumu i nazivaju se *suboptimalna rešenja*. Kako je dopustiv skup rešenja u kombinatornoj optimizaciji diskretan i prebrojiv, da bi lokalni minimum bio korektno definisan, neophodno je prvo uvesti pogodnu metriku, koja meri udaljenost između rešenja. Ove metrike zavise od konkretnog problema koji se razmatra i mogu se iskoristiti za definisanje strukture okolina u prostoru pretrage. *Lokalni optimum* je rešenje problema optimizacije koje ima minimalnu ili maksimalnu vrednost funkcije cilja u poređenju sa okolnim dopustivim rešenjima definisanim datom metrikom. *Globalni optimum* je optimalno rešenje koje doseže minimalnu ili maksimalnu vrednost na skupu svih dopustivih rešenja. Na slici 1.1 je prikazan primer funkcije cilja sa više lokalnih minimuma i maksimuma. Više o problemima optimizacije se može naći u [23, 46, 133, 169].



Slika 1.1: Primeri lokalnih optimuma za problem globalne optimizacije

Metode optimizacije

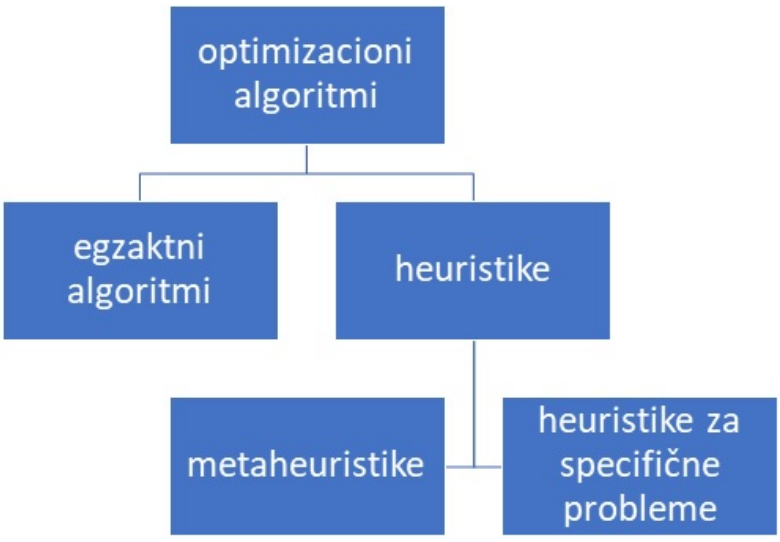
Metode za rešavanje problema optimizacije mogu se podeliti u tri osnovne grupe: *egzaktne metode*, *heuristike* i *metaheuristike*. Egzaktne metode uvek pronalaze optimalno rešenje s^* pod pretpostavkom neograničenih resursa memorije i vremena. Najčešće korišćene egzaktne metode u optimizaciji su: dinamičko programiranje (engl. *Dynamic Programming*), metoda grananja i ograničavanja (engl. *Branch & Bound*), metoda odsecajućih ravni (engl. *Cutting Plane method*), metoda grananja i odsecanja (engl. *Branch & Cut*), metoda grananja i ocenjivanja (engl. *Branch & Price*), pretraga u širinu (engl. *Breadth First Search*), pretraga u dubinu (engl. *Depth First Search*) i druge. Više o egzaktnim algoritmima može da se nađe u [128].

Za razliku od egzaktnih metoda koje teorijski pronalaze optimalno rešenje, heurističke metode ne mogu da dokažu optimalnost dobijenog rešenja. Međutim, ukoliko su adekvatno prilagođene razmatranom problemu, heurističke metode u kratkom vremenu izvršavanja dostižu poznato optimalno resenje s^* ili rešenje $s^{'}$ koje je blisko optimlanom. Efikasnost heurističkih metoda u dobijanju kvalitentnih rešenja je njihova prednost u odnosu na egzaktne metode. Pojam heuristike je prvi put uveo Polya 1945. godine [140], a prve heurističke metode su razvijene sedamdesetih godina prošlog veka, kada su predložene heurističke metode za neke specifične probleme iz oblasti nauke i tehnike [12]. Neke od najpoznatih heurističkih metoda su pohlepni algoritmi, podeli-pa-vladaj metoda, metode bazirane na dodeli prioriteta, lokalno pretrazivanje, itd. Više o heurističkim metodama i njihovoj primeni može se naći u [182].

Metaheuristike su opšte tehnike koje se koriste za razvoj heurističkih algoritama u rešavanju problema optimizacije. Termin *metaheuristika* je predložen u [55] i podrazumeva opšte (nadređene) heuristike koje imaju ulogu da usmeravaju druge heuristike pri istraživanju prostora rešenja. Neke od najpoznatijih metaheuristika su: *genetski algoritmi* (engl. *Genetic Algorithm*), *tabu pretraživanje* (engl. *Tabu Search*), *simulirano kaljenje* (engl. *Simulated Annealing*), *memetski algoritmi* (engl. *Memetic Algorithms*), *optimizacija kolonijom mrava* (engl. *Ant Colony Optimization*), *inteligencija roja* (engl. *Swarm Intelligence*), *evolutivni algoritmi* (engl. *Evolutionary Algorithm*), *veštačke neuralne mreže* (engl. *Artificial Neural Networks*), metoda promenljivih okolina (engl. *Variable Neighborhood Search*) i mnoge druge metaheurističke metode.

Veliki broj problema optimizacije su NP-teški, što ih čini veoma složenim za rešavanje egzaktnim metodama. Egzaktne metode nisu pogodne za optimizaciju mno-

gih praktičnih problema jer njihovo vreme izvršavanja značajno raste sa porastom dimenzije problema ili su egzaktne metode neefikasne sa nekog drugog stanovišta (npr. zahtevaju previše kompjuterske memorije). Kako su egzaktne metode obično pogodne samo za male ili srednje veličine instanci, u slučaju instanci velikih dimenzija neophodno je žrtvovati zahtev za dobijanje optimalnog rešenja i zadovoljiti se rešenjem dobrog kvaliteta dobijenim u relativno kratkom vremenu izvršavanja, što rezultira potrebom implementiranja heurističkih i metaheurističkih metoda. Čak iako ne garantuju optimalnost rešenja, heurističke i metaheurističke metode postaju sve zastupljenije u oblasti optimizacije, jer su zasnovane na principima koji vode proces pretraživanja do vrlo kvalitetnih rešenja bliskih optimumu. Osim toga, veoma važna karakteristika ovih metoda je efikasnost u rešavanju problema velikih dimenzija kao i jednostavnost implementacije. Metode optimizacije se implementiraju u formi algoritama, tako da se na osnovu podele samih metoda optimizacije može napraviti analogna podela optimizacionih algoritama. Ova podela je predstavljena na slici 1.2.



Slika 1.2: Podela optimizacionih algoritama

Za algoritam se kaže da je *deterministički* ako se za iste ulazne podatke uvek dobijaju iste izlazne vrednosti, bez obzira na uslove pod kojima se algoritam izvršava, Egzaktni algoritmi spadaju u ovu grupu algoritama. Kod *stohastičkih algoritama*, na izvršavanje pojedinih koraka algoritma utiče jedan ili više slučajnih elemenata, tako da ovi algoritmi ne dobijaju uvek iste izlazne rezultate za isti skup ulaznih

podataka.

Složenost algoritama

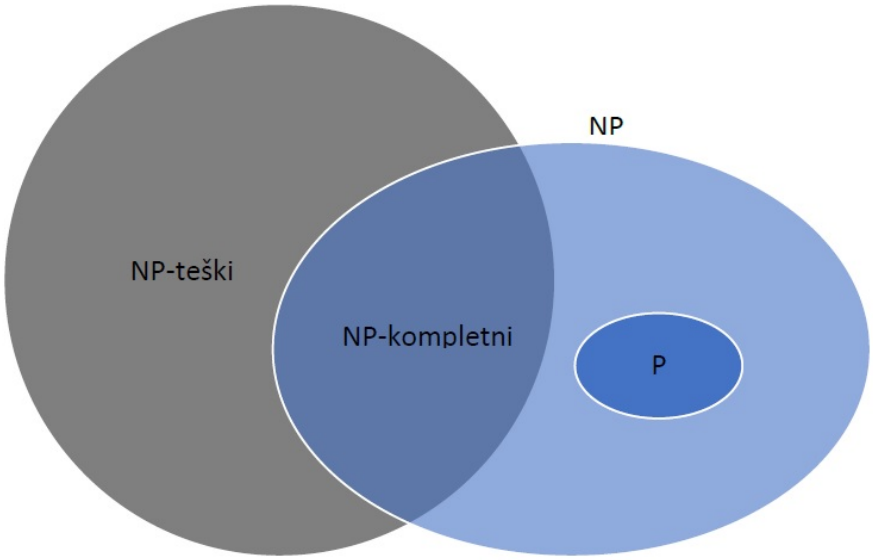
Definicija 1. Kompleksnost algoritma P je maksimalno vreme izvršavanja $t_P(n)$ za sve ulaze veličine n.

Definicija 2. Neka su f i g pozitivne funkcije argumenta n iz skupa prirodnih brojeva N. Za funkciju $f(n)$ važi $f(n) = O(g(n))$, ako postoje pozitivne konstante m i n_0 takve da $f(n) \leq m \cdot g(n)$ za svako $n \geq n_0$.

Definicija 3. Algoritam P je polinomske složenosti ako je njegova kompleksnost $t_P(n) = O(n^k)$ gde je k pozitivna konstanta. Algoritam je eksponencijalne složenosti ako je $t_P(n)$ ograničen sa $O(2^{nk})$ gde je k neka konstanta.

Uvođenjem ovih definicija omogućena je i klasifikacija problema optimizacije na osnovu njihove složenosti. Kako se za dati problem optimizacije mogu razviti različiti algoritmi koji ga rešavaju, jasno je da se pri definisanju kompleksnosti problema razmatra najefikasniji algoritam. Problem pripada *klasi* P ako postoji algoritam polinomske složenosti koji ga rešava. Da bi se uvela sledeća klasa problema i pojednostavila razmatranja, posmatraju se samo problemi odlučivanja, tj. problemi na koje se posle izvršavanja nekog algoritma može odgovoriti sa „da" ili sa „ne". *Klasu NP problema* (engl. *Nondeterministic Polynomial problems*) čine problemi kod kojih se u polinomskom vremenu može proveriti da li je dati element dopustivog skupa rešenje problema. Očigledno je da važi $P \subseteq NP$, ali jednakost još uvek nije dokazana. Problem je *NP-težak problem* ako se svaki NP problem može u polinomijalnom vremenu svesti na dati problem. *NP-kompletne probleme* čine NP problemi koji su istovremeno i NP teški. Za problem se kaže da je *NP-kompletan u jakom smislu* (engl. *strongly NP-complete*) ako ostaje NP-kompletan kada se veličina njegovih numeričkih parametara ograniči nekim polinomom. Problem je *NP-težak u jakom smislu* (engl. *strongly NP-hard*) ako se na njega može polinomijalno redukovati neki NP-kompletan problem u jakom smislu. Međusobni odnos klasa složenosti prikazan je na slici 1.3. Velika većina problema optimizacije spada u klasu NP-teških problema [85]. Detaljne informacije o složenosti problema mogu se naći u [155].

Jedno od osnovnih pitanja pri rešavanju problema kombinatorne optimizacije je da li postoji algoritam koji dati problem može da reši u polinomijalnom vremenu u odnosu na dimenziju datog problema. Vreme izračunavanja se uzima kao indikator



Slika 1.3: Klase složenosti problema

efikasnosti algoritma. Vreme izračunavanja zavisi od složenosti problema, skupa razmatranih parametara koji opisuju problem i njihove domene, dimenzije problema, procedura i metoda koje algoritam koristi, ali i od načina implementacije algoritma.

U grupu NP-teških problema [77] kombinatorne optimizacije spadaju:

- *problemi rutiranja* (engl. *routing*), kod kojih je cilj pronaći efikasne puteve transporta kroz date mreže;
- *problemi raspoređivanja* (engl. *scheduling*), obuhvataju dodelu resursa u cilju izvršavanja skupa poslova u zadatom vremenu;
- *problemi dodele* (engl. *assignment*), u kojima je neophodno grupu agenata (mašina, procesora ili drugih resursa) rasporediti na skup zadataka;
- *problemi pakovanja* (engl. *packing*), kod kojih je cilj smestiti skup objekata u jedno ili više skladišta.

Problemi optimizacije u kontejnerskom terminalu

Efikasan kontejnerski terminal može značajno da poboljša produktvnost luke i da utiče na dobru iskorišćenost infrastrukture i lučkih resursa (videti glavu 3). Zato nije iznenađujuće da optimizacija operacija kontejnerskog terminala predstavlja

čestu istraživačku temu u novijoj literaturi. U fokusu autora je veliki broj specifičnih problema, kao što su problemi dodele vezova [26], dodele kranova [110], optimizacija operacija transfera kontejnera [106], optimizacija operacija skladištenja [180], i neki drugi problemi optimizacije karakteristični za kontejnerski terminal.

Jedan od najvažnijih problema koji je potrebno rešiti u kontejnerskom terminalu je problem dodele vezova (engl. *Berth Allocation Problem*-BAP). Poznato je da ovaj problem spada u grupu NP-teških problema kombinatorne optimizacije. Pojedine varijante problema dodele vezova mogu se sagledati i kao problemi rutiranja, raspoređivanja, dodele, ali i pakovanja. U ovoj disertaciji su razmatrane statička i dinamička hibridna varijanta problema dodele vezova u kontejnerskim terminalima. Kontejnerski terminal predstavlja sistem u kojem se situacija konstantno menja i podleže nepredvidivim događajima koji zahtevaju da se preliminarno formirani planovi vezivanja vrlo često moraju menjati. Promene planova vezivanja i dobijanje novih informmacija o vezovima brodova moraju biti dostupni u najkraćem mogućem roku. Imajući u vidu da se odluke donose u realnom vremenu i zbog činjenice da se veće dimenzije problema dodele vezova ne mogu rešiti egzaktnim metodama usled nedostatka memorije u računarskim resursima ili im je potrebno neprihvatljivo mnogo procesorskog vremena da nađu optimalno rešenje, metaheuristički pristup se nametnuo kao adekvatan pristup. U ovoj disertaciji predloženo je nekoliko metaheurističkih metoda za rešavanje statičke i dinamičke varijante problema dodele vezova u kontejnerskom terminalu.

Ostatak teze je organizovan na sledeći način: U glavi 2 dat je pregled osnovnih karakteristika metaheurističkih metoda i njihova klasifikacija. Izloženi su koncepti najčešće korišćenih metaheurističkih metoda, sa posebmi osvrtom na tri metaheurističke metode: evolutivne algoritme, optimizaciju kolonijom pčela i metodu promenljivih okolina, koje su u tezi implementirane i prilagođene za rešavanje dve varijante problema dodele vezova. Glava 3 sadrži opis principa na kojima funkcioniše kontejnerski terminal, definciju problema dodele vezova u kontejnerskim terminalima, kao i definicije nekih problema koji nastaju kao posledica dodele vezova u kontejnerskom terminalu. U nastavku glave 3 data je i klasifikacija varijanti problema dodele vezova koji su najznačajniji sa teorijskog i praktičnog aspekta. Osim toga, u glavi 3 su predstavljene najčeše proučavane varijante problema dodele vezova i metaheuristički pristupi koji su najčešće korišćeni u literaturi za rešavanje ovog problema. Dat je i pregled i diskusija frekvencije i raspodele zastupljenosti pojedinih metaheuristika u zavisnosti od nekoliko parametara koji karakterišu problem dodele vezova.

U glavi 4 dat je detaljan opis matematičkog modela statičke hibridne varijante problema dodele vezova sa minimizacijom ukupnih troškova i izložena je njena matematička formulacija. Zatim je uvedena dinamička hibridna varijanta problema dodele vezova sa minimizacijom ukupnih troškova koja do sada nije razmatrana u literaturi i predložena je matematička formulacija za novouvedenu varijantu problema. Polazeći od analize složenosti BAP-a iz [113], u glavi 4 je dokazano da su obe varijante NP-teški problemi optimizacije.

U glavi 5 predstavljeni su detalji predloženih metaheurističkih metoda za rešavanje razmatranih varijanti problema dodele vezova. Navedeni su pseudokodovi i detalji implementacije svake od metoda. Nakon uvođenja osnovnih definicija i notacije slede dva odeljka koja prikazuju implementaciju dva evolutivna metaheuristička pristupa za rešavanje statičke i dinamičke varijante razmatranog problema. Dat je detaljan opis kodiranja i dekodiranja rešenja kao i specifične karakteristike evolutivnih algoritama koje su rezultat prilagođavanja posmatranom problemu. Metaheuristički pristup rešavanju razmatranih varijanti problema dodele vezova, zasnovan na inteligenciji roja opisan je u četvrtom i petom potpoglavlju. Predstavljene su dve varijante algoritma optimizacije kolonijom pčela: konstruktivna varijanta i varijanta sa popravkom kompletnog rešenja. U literaturi do sada nije primenjivana optimizacija kolonijom pčela na problem dodele vezova i slične probleme u pomorskom transportu. Implementacija sistematičnog lokalnog pretraživanja okolina dopustivog rešenja je detaljno predstavljena u poslednjem potpoglavlju glave 5 kroz četiri varijante metode promenljivih okolina: metodu promenljivog spusta, višestartnu metodu promenljivog spusta, opštu metodu promenljivih okolina i adaptivnu metodu promenljivih okolina. Korišćeno je adekvatno kodiranje, sofisticirane strukture podataka i definisane su okoline za pretraživanje u skladu sa prirodom problema.

Numerički rezultati testiranja na instancama iz realnog života poznatim iz literature, kao i na generisanim test instancama različitih dimenzija, dati su u glavi 6. Test instance problema su najpre rešene do optimalnosti, ukoliko je moguće, primenom komercijalnog CPLEX rešavača, u cilju evaluacije performasi predloženih metaheuristika u smislu kvaliteta rešenja i uštede procesorskog vremena. Prikazani su detaljni rezultati testiranja na svim klasama test instanci, koji pokazuju da metaheuristike uspešno rešavaju realne i slučajno generisane probleme. Data je i detaljna analiza uticaja tehnika popravke dopustivog rešenja na kvalitet dobijenog krajnjeg rešenja algoritma. Prikazani su i efekti empirijske kalibracije pojedinih parametara koji karakterišu optimizaciju kolonijom pčela. Glava 7 sadrži pregled rezultata di-

sertacije i glavnih naučnih doprinosa. U istoj glavi su predložene smernice za budući rad i neke zaključne napomene.

Glava 2

Metaheuristike

Metaheuristički algoritmi su postali predmet interesovanja u operacionim istraživanjima sa publikovanjem rada Kirkpatrick-a i sar. [91] iz 1983. godine, u kome je predlozeno simulirano kaljenje kao tehnika sa sposobnošću napuštanja lokalnog minimuma. Od tada su metaheurističke metode konstantno razvijane, što je omogućilo rešavanje sve većeg broja problema optimizacije, koji su ranije smatrani teškim ili čak nerešivim. Termin metaheuristika je prvi put uveo Glover 1986. godine u radu [55]. Ovaj termin potiče od dve grčke reči: *heuriskein* što se prevodi kao *pronaći*, dok prefiks *meta* označava *na višem nivou*.

Iako ne postoji precizna definicija termina metaheuristika, svi pokušaji definisanja imaju neke zajedničke elemente, tako da se može reći da su metaheuristike grupa algoritama za pretraživanje koji mogu rešiti složene probleme optimizacije koristeći skup nekoliko opštih heurističkih principa. Metaheuristike se mogu definisati i kao heuristike višeg nivoa koje su usko povezane sa stohastičkim algoritmima [114]. Stohastički algoritmi su opšta klasa algoritama i tehnika koje koriste određeni stepen slučajnosti kako bi pronašli optimalna rešenja ili rešenja koja su što bliža optimalnim rešenjima. Metaheuristike su najopštije metode i primenjuju se na veoma širok spektar problema. Neke od najpoznatijih metaheuristika su: simulirano kaljenje, tabu pretraživanje, harmonijsko pretraživanje, evolutivni algoritmi, optimizacija kolonijom pčela, optimizacija rojem čestica i druge [96, 114].

Složenost razmatranog problema optimizacije vrlo često onemogućava pretraživanje svih dopustivih rešenja u cilju pronalaženja optimalnog. Iz tog razloga, postavlja se novi zadatak: pronaći dobro rešenje u prihvatljivom vremenskom roku. Metaheuristike spadaju u grupu aproksimativnih algoritama, što znači da ne mogu garantovati kvalitet dobijenog rešenja, iako se neretko rešenje koje daje metaheu-

ristika poklapa sa optimalnim. I pored toga, široko se upotrebljavaju u rešavanju problema optimizacije, jer metaheuristika čiji su elementi prilagođeni prirodi razmatranog problema daje rešenja bliska optimalnim ili dostiže optimalna rešenja u kratkom vremenu izvršavanja.

Metaheuristike obično imaju iterativnu formu koja podrazumeva da se jedno ili više početnih rešenja poboljšavaju u smislu vrednosti funkcije cilja koja se minimizuje ili maksimizuje. Generalno, može se reći da se metaheurističke metode uklapaju u sledeći iterativni postupak:

korak 1: generisati jedno ili više početnih rešenja;

korak 2: popravljati početno rešenje (rešenja) do ispunjenja nekog kriterijuma zaustavljanja;

korak 3: najbolje rešenje iz koraka 2 vrati kao izlaz algoritma.

Može se primetiti da postupak pronalaženja dobrog rešenja nekog problema optimizacije obuhvata dva koraka: konstrukcija i popravak rešenja. U postupku konstrukcije (korak 1) gradi se jedno ili više početnih rešenja koja predstavljaju polazne tačke za proces pretrage. Metaheuristika pri konstrukciji početnog rešenja najčešće kreće od praznog rešenja u koje postepeno dodaje elemente rešenja gradeći na taj način parcijalno rešenje, i ponavlja ubacivanje novih komponenti sve dok se ne formira dopustivo rešenje problema. Postoji i drugi pristup - slučajno generisanje kompletnog rešenja nad kojim se primenjuju tehnike popravke ako rešenje nije dopustivo. U fazi popravke (korak 2), formirana rešenja se postepeno popravljaju, sve dok se ne dostigne neki od kriterijuma zaustavljanja, nakon čega se kao izlaz algoritma, vraća najbolje rešenje (korak 3). Methaheuristike najčešće koriste jedan, ili kombinaciju više kriterijuma za završetak rada algoritma: maksimalan broj iteracija, maksimalno vreme izvršavanja, maksimalan broj popravki rešenja, dosegnut određeni stepen kvaliteta rešenja, itd.

Da bi se neka metaheuristika mogla okarakterisati kao dobra, mora da zadovolji nekoliko opštih principa [69]:

- *jednostavnost*-metaheuristika treba da bude zasnovana na jednostavnim principima;
- *preciznost*-koraci metaheuristike su precizno definisani;

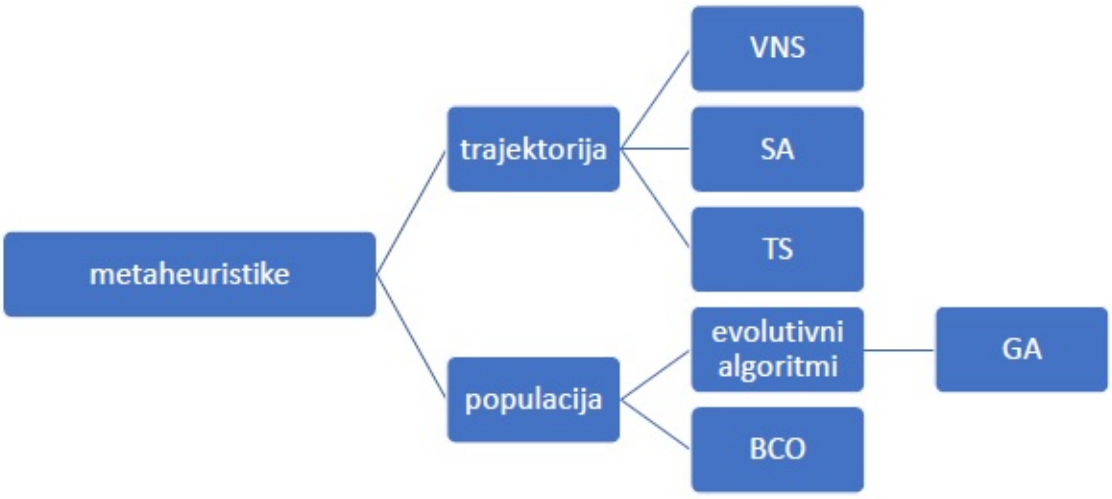
- *doslednost*-koraci metode koja rešava konkretan problem treba da slede pravila kojima je metaheuristika definisana;
- *efektivnost*-metaheuristika treba da generiše optimalna ili rešenja bliska optimumu za većinu test instanci problema;
- *efikasnost*-procesorsko vreme potrebno za pronalaženje dobrog rešenja mora biti prihvatljivo;
- *robustnost*-metaheuristika treba da pokazuje dobre performanse na raznim skupovima test instanci;
- *jasnoća*-metaheuristika treba da bude dobro opisana i jednostavna za implementaciju i upotrebu;
- *inovativnost*-dobre karakteristike metaheuristike omogućavaju modifikacije i hibridizaciju sa drugim metodama, kao i primenu na nove klase problema.

Od posebnog značaja pri dizajniranju metaheuristike je postizanje balansa između *intensifikacije* i *diversifikacije* procesa pretraživanja. Naime, neophodno je brzo identifikovati regione pretraživačkog prostora sa visokokvalitetnim rešenjima, a sa druge strane, ne trošiti previše vremena u regionima koji su ili već istraženi ili ne sadrže kvalitetna rešenja. Metaheuristika treba da ima mogućnost da generiše nova rešenja koja će generalno biti bolja od trenutnih rešenja. Osim toga, metaheuristika mora da ima mehanizme za sprečavanje završetka u lokalnom optimumu, odnosno za napuštanje lokalizovane pretrage koja ne dovodi do optimalnog rešenja. Komponenta diversifikacije ima za zadatak generisanje raznovrsnih rešenja kako bi svi delovi prostora rešenja bili obuhvaćeni pretragom, dok je cilj intensifikacije fokusiranje pretrage na lokalnom regionu koristeći informacije o poziciji dobrih rešenja u prostoru pretrage.

Metaheuristike se mogu klasifikovati na razne načine u zavisnosti od neke od karakteristika, tako da u literaturi nalazimo sledeće podele metaheurističkih metoda:

- metaheuristike inspirisane prirodom i one koje su inspirisane matematičkim konceptima;
- sa dinamičkom ili sa statičkom funkcijom cilja;
- sa jednom ili više struktura okolina koje se koriste u procesu pretrage;

- metaheuristike koje popravljaju kompletno rešenje ili konstruktivne metaheuristike;
- determinističke ili stohastičke metaheuristike;
- metaheuristike koje koriste istoriju pretrage (memoriju) ili ne;
- metode trajektorije i populacione metaheuristike.



Slika 2.1: Podela metaheuristika na osnovu broja rešenja

Jedna od najpopularnijih klasifikacija [10] deli metaheuristike na metode trajektorije (engl. *trajectory methods*) koje pokušavaju da poboljšaju jedno rešenje i na populacione metaheuristike (engl. *population-based*) koje rade nad više rešenja u prostoru pretrage i istovremeno pokušavaju da poboljšaju svako od njih [10]. Grafički prikaz ove podele, kao i neke od reprezentativnih metaheuristika iz pojedinih klasa, dati su na slici 2.1.

Metode trajektorije su dobile naziv prema činjenici da se proces pretraživanja kod ovih metoda odvija po putanji koju čini niz rešenja u pretraživačkom prostoru. Algoritam kreće od nekog početnog rešenja (inicijalnog stanja) i opisuje trajektoriju u prostoru rešenja, pri čemu sledeće rešenje može, ali i ne mora, da bude u okolini prethodnog rešenja. Za razliku od njih, populacione metaheuristike u svakoj iteraciji algoritma formiraju skup rešenja (populaciju) i na taj način istražuju prostor

pretrage. Metaheuristike zasnovane na populaciji rešenja na početku kreiraju inicijalnu populaciju a zatim se faze kreiranja populacije i faze zamene tekuće populacije smenjuju naizmenično.

Primeri metaheuristika koje su zasnovane na poboljšanju jednog rešenja su: metoda promenljivih okolina, simulirano kaljenje, tabu pretraživanje, iterativno lokalno pretraživanje, itd. dok su primeri polulacionih metaheuristika evolutivni algoritmi (uključujući genetski algoritam), optimizacija kolonijom pčela, optimizacija rojem čestica, itd. Detaljan pregled metaheuristika i njihova klasifikacija se može naći u [10, 114, 162]. U nastavku je dat prikaz nekih najčešće korišćenih metaheuristika pri rešavanju problema optimizacije.

2.1 Evolutivni algoritmi

Evolutivni algoritmi (engl. *Evolutionary algorithms*-EA) spadaju u grupu algoritama inspirisanih prirodom koji oponašaju evolutivni proces populacije jedinki u kome se jedinke tokom vremena prilagođavaju svom okruženju, a najotpornije jedinke opstaju i reprodukuju se. Evolutivni algoritmi koriste skup *jedinki* kojima su pridružena pojedinačna rešenja problema. Sve jedinke sačinjavaju *populaciju*. Veličina populacije je određena brojem jedinki i može biti konstantna ili se menjati tokom svoje evolucije. U svakoj iteraciji algoritma izvršava se niz operacija nad jedinkama trenutne populacije da bi se formirala nova populacija. Evolucija populacije jedinki u evolutivnom algoritmu se sprovodi kroz uzastopne evolutivne korake koji se nazivaju *generacije*.

U procesu *kodiranja* uspostavlja se veza između skupa rešenja problema i skupa jedinki, tako što se svako pojedinačno rešenje preslikava u definisanu strukturu jedinke poznate pod nazivom *reprezentacija jedinke*. Genetski materijal jedinke odnosno atributi koji je opisuju predstavljaju se *genima*. Reprezentacija jedinke se naziva njegovim *genetskim kodom* ili *hromozomom*. Način reprezentacije jedinke je određen informacijama koje jedinke treba da sadrže, imajući u vidu da svaka jedinka odgovara rešenju u pretraživačkom prostoru.

Funkcija koja meri kvalitet jedinke naziva se funkcija prilagođenosti (engl. *fitness function*). U procesu *selekcije* na osnovu mere kvaliteta jedinke, poznate pod nazivom *vrednost prilagođenosti* (engl. *fitness value*), jedinke se prosleđuju u sledeću generaciju. Ukoliko se EA koristi u kombinatornoj optimizaciji, funkcija cilja se obično upotrebljava kao funkcija prilagođenosti, koja svakoj jedinki dodeljuje

vrednost prilagođenosti kao meru njenog kvaliteta. Jedinke koje imaju veći kvalitet imaju i veću verovatnoću preživljavanja i veću šansu da direktno prođu u narednu generaciju ili da učestvuju u procesu strvaranja jedinki nove generacije.

Zadatak operatora varijacije (engl. *variation operators*) je kreiranje novih jedinki na osnovu starih. Operatori varijacije oslikavaju prirodni proces adaptacije jedinki i obično se primenjuju nad jednom ili dve jedinke. Ukoliko je operator varijacije unarnog tipa, nove jedinke nastaju promenom malog dela koda odabrane jedinke, dok u slučaju binarnog operatora nove jedinke nastaju razmenom informacija između jedinki odnosno nekom kombinacijom dela genetskog koda dve jedinke. Operatori varijacije najčešće imaju stohastičku prirodu: izbor jedinki nad kojima se primenjuje operator je slučajan kao i izbor dela jedinke nad kojom se operator primenuje. Primena operatora varijacije obezbeđuje raznovrsnost genetskog materijala i generalno vodi ka poboljšanju kvaliteta populacije. Postoje razni operatori varijacije kao što su inverzija, brisanje, migracija dela populacije, dok se najčešće koriste ukrštanje i mutacija. Izbor operatora varijacije direktno zavisi od reprezentacije jedinke i treba da sledi opšti princip očuvanja dobrih karakteristika jedinke nad kojom se primenjuje, ali i da ima mogućnost da proizvede nov genetski materijal za narednu generaciju jedinki.

EA pokušava da poboljša sveobuhvatni kvalitet populacije tako što je podvrgava procesu koji oponaša proces prirodne evolucije jedinki. U ovom procesu, jedinke razmenjuju informacije u cilju kreiranja novih ili u cilju transformacije postojećih rešenja. Jedinke koje u tom postupku razmenjuju informacije su *jedinke roditelji*, a novoformirane ili modifikovane jedinke čine *potomsto*. Osim preko razmene informacija između roditelja, na kreiranje potomstva može da utiče i *istorija populacije*, odnosno, u toku evolucije populacije se sakupljaju korisne informacije, koje se ne bi mogle dobiti jednostavnim posmatranjem samo trenutnog stanja u populaciji. U slučajevima kada se cela populacija menja pri prelasku iz jedne generacije u drugu, u pitanju je *generacijska zamena* (engl. *generational replacement*). Ako se samo deo populacije menja u dve uzastopne generacije, evolutivni proces prati stacionarnu politiku zamene generacija (engl. *steady state replacement*).

Proces preslikavanja jedinke u rešenje posmatranog problema naziva se *dekodiranje*. Metoda dekodiranja direktno utiče na efikasnost algoritma jer se u ovom postupku lako mogu proizvesti nekorektne jedinke, koje odgovaraju nedopustivim rešenjima. Prilikom računanja funkcije prilagođenosti, nekorektne jedinke se mogu popravljati do korektnih, može da se penalizuje njihova funkcija prilagođenosti ili

se nekorektne jedinke jednostavno brišu iz populacije.

Rezultat koji vraća EA je najbolja jedinka formirana u toku celokupnog rada algoritma. Više detalja o EA može se pronaći u [181]. Pseudokod evolutivnog algoritma sa svim karakterističnim koracima koje sadrži, prikazan je u algoritmu 1.

Algorithm 1 Evolutivni algoritam

```
procedure EA
    Pop ← GenerateInitialPopulation()
    Evaluate(Pop)
    while ¬ kriterijum zaustavljanja do
        newPop_1 ← VariationOperator(Pop)
        Evaluate(newPop_1)
        Pop ← Select(newPop_1 ⋃ Pop)
    end while
end procedure
```

Većina novijih implementacija evolutivnih algoritama potiče od četiri slična, ali nezavisno razvijena pristupa: evolutivno programiranje razvijeno u [47, 48], evolutivne strategije predložene u [144], genetsko programiranje [102] i najpoznatiji genetski algoritmi [78].

Evolutivno programiranje je originalno razvijeno kao pokušaj da se razvije veštačka inteligencija upotrebom konačnih automata koji predstavljaju apstraktne mašine sposobne da transformišu niz ulaznih simbola u niz izlaznih simbola. Evolutivno programiranje podrazumeva evolutivnu promenu stanja konačnih automata u cilju predviđanja događaja na osnovu ranijih posmatranja. Evolucija zavisi od konačnog skupa stanja automata i od konačnog skupa tranzicionih pravila poznatih pod nazivom relacije prelaza. U tom kontekstu se performanse konačnih automata mogu meriti na osnovu mogućnosti predviđanja, odnosno upoređivanjem izlaznih simbola sa sledećim ulaznim simbolima i merenjem stepena korisnosti predviđanja uz korišćenje nekih specifičnih funkcija. Ova klasa algoritama koristi stohastički operator selekcije, mutaciju kao glavni operator varijacije i princip preživljavanja u kreiranju nove populacije.

Evolutivne strategije su prvenstveno razvijene sa ciljem rešavanja komplikovanih diskretnih i kontinualnih problema parametarske optimizacije. Kod evolutivnih strategija se uglavnom u reprezentaciji jedinke koriste vektori sa realnim vrednostima i nad njima se u fazi modifikacije jedinki primenjuje mutacija sa normalnom distribucijom i rekombinacija kao esencijalni operator za istovremenu pretragu prostora rešenja i prostora parametra strategije. Operator selekcije je deterministički, a populacije jedinki roditelja i jedinki potomaka su obično različite dimenzije.

Genetsko programiranje je klasa evolutivnih algoritama u kojima strukture podataka koje se podvrgavaju adaptaciji imaju formu izvršnog kompjuterskog programa predstavljenog u vidu stabla, a izračunavanje vrednosti funkcije prilagođenosti podrazumeva izvršavanje kompjuterskog programa. Programske strukture u genetskom programiranju evoluiraju primenom operatora ukrštanja koji podrazumeva razmenu podstabala dve jedinke roditelja.

Genetski algoritmi često koriste binarnu reprezentaciju jedinki ili reprezentaciju koja ima formu niza čiji elementi su iz nekog konačnog skupa. Kod genetskih algoritama najvažniji operatori transformacije jedinki su ukrštanje i mutacija dok je selekcija obično stohastička. Detalji genetskog algoritma opisani su u sledećem odeljku.

Klasifikacija evolutivnih algoritama u kombinatornoj optimizaciji može se naći u [16].

Genetski algoritam

Genetski algoritam (engl. *Genetic Algorithm*-GA) je adaptivna metoda globalne pretrage koja koristi koncept prirodnog prilagođavanja i selektivnog razmnožavanja organizama [78]. Genetski algoritmi predstavljaju klasu evolutivnih algoritama koji se zasnivaju na evolutivnim i genetskim principima. GA transformiše populaciju jedinki tako što iterativno primenjuje genetske operatore selekcije, ukrštanja i mutacije, do ispunjenja nekog kriterijuma zaustavljanja. U cilju obezbeđivanja dobrih performansi GA, potrebno je definisati adekvatnu reprezentaciju rešenja za konkretan problem koji se razmatra, adekvatnu funkciju prilagođenosti za evaluaciju jedinki i osmisliti ogovarajuće genetske operatore. Funkcija prilagođenosti meri kvalitet jedinke i zavisi od samog problema optimizacije i reprezentacije rešenja. Pregled osnovnih i naprednih GA metoda za rešavanje različitih NP-teških problema optimizacije može se naći u [3, 4, 145, 162].

Prva faza implementacije GA algoritma podrazumeva definisanje reprezentacije jedinki i njihovo pridruživanje rešenjima problema. Reprezentacija jedinke ima strukturu podataka zvanu *hromozom* i najčešće se sastoji od niza nula i jedinica, ali postoje i druge metode kodiranja, kao što su vektori, celi brojevi ili realni brojevi [32]. Adekvatan izbor reprezentacije i dužine jedinke je uslovljen prirodom problema koji se rešava. Kodiranje treba da bude takvo da se sve validne jedinke mogu preslikati u rešenje problema, i obrnuto sva dopustiva rešenja problema mogu da se predstave definisanom strukturom jedinke.

Funkcija prilagođenosti je definisana preko funkcije cilja problema koji se rešava, i predstavlja meru kvaliteta jedinke odnosno meru kvaliteta rešenja u koje se jedinka dekodira. Na osnovu vrednosti funkcije prilagođenosti donosi se odluka da li će jedinka preživeti i ostati u populaciji ili će biti eliminisana iz sledeće generacije. Jedinke koje imaju veću vrednost funkcije prilagođenosti imaju veću verovatnoću preživljavanja. Načini računanja funkcije prilagođenosti su različiti, tako da se u literaturi mogu naći linearno skaliranje, skaliranje u jedinični interval, sigma odsecanje, itd., a najčešći pristup je direktno preuzimanje u kojem vrednost funkcije cilja postaje vrednost funkcije prilagođenosti jedinke.

Kod generisanja kodova jedinke dešava se da kod ne odgovara ni jednom dopustivom rešenju, odnosno da je jedinka nekorektna. U tim slučajevima, postoje različite strategije koje se mogu primeniti na nekorektne jedinke: startegija eliminacije u kojoj se nekorektne jedinke brišu iz populacije postavljanjem funkcije prilagođenosti na nula, strategija popravke jedinke dok ne postane korektna ili im se pomoću kaznenih funkcija smanjuje vrednost prilagođenosti.

Na početku rada, GA na slučajan način generiše populaciju jedinki koje odgovaraju rešenjima problema. Najčešći pristup je generisanje populacije korektnih jedinki nad kojima se primenjuju genetski operatori koju čuvaju korektnost. Uzastopnom primenom operatora mutacije, ukrštanja i selekcije, trenutna populacija se transformiše u novu populaciju.

U procesu selekcije, zasnovanom na izračunatim vrednostima prilagođenosti jedinki, biraju se kvalitetne jedinke za naredne faze algoritma. Operator selekcije je najčešće dizajniran tako da se bira i mali procenat loših jedinki što obezbeđuje raznovrsnost populacije i sprečava prevremenu konvergenciju algoritma. Najzastupljenije metode selekcije su: ruletska selekcija, turnirska selekcija, fino gradirana turnirska selekcija i selekcija zasnovana na rangu.

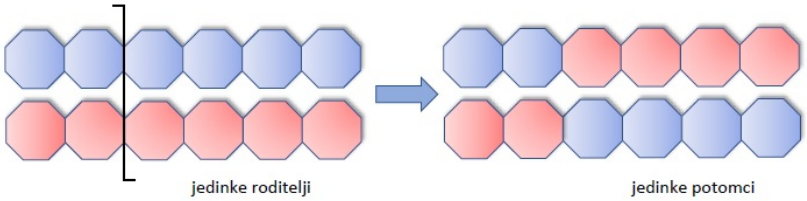
Ruletska selekcija je zasnovana na principu ruleta, pri čemu verovatnoća da će jedinka biti izabrana direktno zavisi od vrednosti njene funkcije prilagođenosti. Jedinke boljeg kvaliteta imaju manju verovatnoću da budu eliminisane iz populacije, dok jedinke koje imaju manju vrednost funkcije prilagođenosti takođe imaju šansu da se proslede u sledeću generaciju. Ovo je dobra osobina ruletske selekcije, jer i lošija rešenja mogu da nose gene koji mogu doprineti stvaranju kvalitetnijih jedinki potomaka prilikom ukrštanja. Operator turnirske selekcije bira grupe jedninki (broj jedinki u grupi je definisan veličinom turnira), zatim se jedinke u svakoj grupi upoređuju po kvalitetu i najbolja jedinka iz svake grupe se prosleđuje u sledeću

generaciju. Veličina turnira je parametar koji posredno utiče na to koje jedinke će biti odabrane. Naime, ako je veličina turnira velika, jedinke lošijeg kvaliteta imaju manje šanse da budu odabrane. U fino gradiranoj turnirskoj selekciji se koristi racionalni parametar koji označava prosečnu veličinu turnira. Kod fino gradirane turnirske selekcije veličina turnira nije jedinstveno određena već se koriste dve veličine turnira i za svaku se unapred odredi broj turnurira zadate veličine. Kao i kod turnirske selekcije na slučajan način se bira podskup jedinki koje ulaze u turnir date veličine, a najbolja jedinka turnira učestvuje u stvaranju nove generacije. Preuranjena konvergencija algoritma prevazilazi se selekcijom zasnovanom na rangu prema prilagođenosti jedinke. Kod ovog operatora selekcije izbor jedinke zavisi samo od pozicije u populaciji. Pozicija je dobijena rangiranjem jedinki u opadajući redosled vrednosti funkcije prilagođenosti.

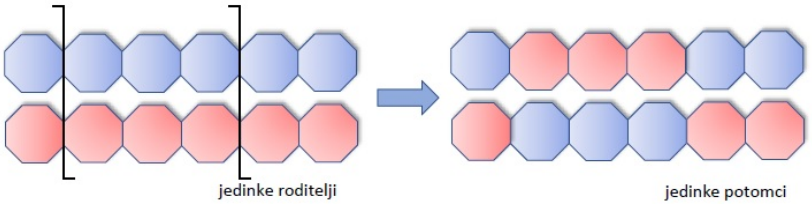
Genetski algoritam koristi operatore *rekombinacije* ili *ukrštanja* nad dve ili više jedinki i tako proizvodi novu jedinku, kao i operator mutacije ili modifikacije koji predstavlja prilagođavanje jedinke [5].

Operator ukrštanja kombinuje delove genetskog koda dve ili više jedinki roditelja i kreira jednu ili više jedinki potomaka. Uloga ovog operatora je da kombinuje dobre delove genetskog koda koji odgovaraju dobrim komponentama rešenja, u cilju dobijanja novih rešenja boljeg kvaliteta. Postoji mnogo tehnika ukrštanja a najpoznatije su jednopoziciono (engl. *one-point crossover*), dvopoziciono (engl. *two-point crossover*), višepoziciono ukrštanje (engl. *multy-point crossover*) i uniformno ukrštanje (engl. *uniform crossover*). Kod jednopozicionog ukrštanja hromozomi roditelja se presecaju na slučajno izabranoj poziciji (tačka ukrštanja), a rezultujući delovi genetskog koda roditelja se razmene. Operator dvopozicionog ukrštanja na slučajan način bira dve tačke ukrštanja i genetski kod između preseka se razmenjuje. Višepoziciono ukrštanje se realizuje na sličan način kao dvopoziciono, s tim što se bira više tačaka ukrštanja i jedinke roditelji razmenjuju delove genetskog koda između izabranih tačaka. Uniformno ukrštanje je višepoziciono ukrštanje sa $n-1$ tačaka prekida gde je n dužina jedinke. Ovaj operator ukrštanja koristi strukturu maske da bi fromirao jedinke potomke. Maska je na slučajan način generisan binaran niz dužine n. Jedinke roditelji zadržavaju vrednost gena na poziciji na kojoj maska ima vrednost 1 dok na pozicijama koje sadrže 0 razmenjuju vrednosti gena. Operatori ukrštanja su ilustrovani na slikama 2.2, 2.3 i 2.4.

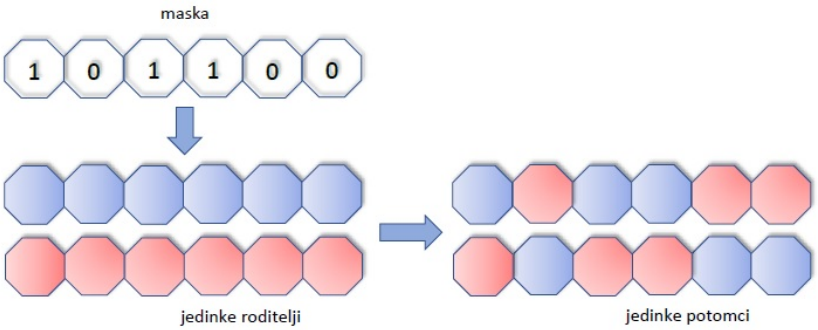
Operator mutacije se realizuje promenama malih delova genetskog koda (najčešće pojedinačnih gena) koji su slučajno izabrani (slika 2.5). Promene vrednosti



Slika 2.2: Jednopoziciono ukrštanje



Slika 2.3: Dvopoziciono ukrštanje



Slika 2.4: Uniformno ukrštanje

pojedinih gena utiču na povećavanje raznovrsnosti genetskog materijala u populaciji i omogućavaju da algoritam lakše napusti lokalni minimum. Kod binarne reprezentacije jedinki najčešće se koristi operator proste mutacije kod kojeg se ispituje gen po gen i za svaki se određuje da li će se mutirati ili ne. Da bi se ubrzao proces mutacije koristi se binomna ili normalna raspodela. Kod mutacije pomoću binomne raspodele broj mutiranih gena ima binomnu raspodelu $B(n, p)$ gde je n dužina jedinke, a p verovatnoća mutacije jednog gena. U slučajevima kada je potrebno neke delove koda jedinke mutirati sa manjom a neke druge sa većom verovatnoćom, koristi se mutacija sa normalnom raspodelom ili eksponencijalna mutacija u kojoj broj mutiranih gena eksponencijalno raste. Ako se u reprezentaciji jedinke koriste celi ili realni brojevi primenjuju se drugi tipovi mutacije, kao što su zamena gena slučajno generisanim brojem, dodavanje ili oduzimanje male vrednosti, množenje brojem bliskim jedinici,

itd.

Mutacija i ukrštanje se vrše na osnovu predefinisanih vrednosti *verovatnoće mutacije* odnosno *verovatnoće ukrštanja*. Ako je verovatnoća mutacije μ tada se slučajno odabrani gen menja sa verovatnoćom μ. Ako je verovatnoća ukrštanja υ tada će sa tom verovatnoćom jedinke roditelji razmeniti delove genetskog koda i proizvesti jedinke-potomke.

Slika 2.5: Prosta mutacija jedinke

Politika zamene generacija značajno utiče na kvalitet konačnog rešenja GA. Postoji nekoliko politika zamene: generacijska, stacionarna i elitistička. Kod generacijske zamene u svakoj generaciji menjaju se sve jedinke dok se kod stacionarne zamene deo populacije prenosi iz postojeće u narednu generaciju a ostatak jedinki se ponovo generiše. Elitistička zamena obezbeđuje da se najbolja jedinka populacije ili podskup najboljih jedinki direktno prosledi u sledeću generaciju, ukoliko već nisu odabrane u procesu selekcije.

Kod genetskog algoritma nakon generisanja početne populacije naizmenično se smenjuju faze selekcije, ukrštanja i mutacije. Ove faze formiraju iteraciju algoritma koja se ponavlja sve dok se ne ispuni neki kriterijum zaustavljanja. Najčeši kriterijumi zaustavljanja GA metode su: maksimalan broj generacija, dovoljno dobar kvalitet populacije, najbolja jedinka je ponovljena određen broj puta, dokazana je optimalnost jedinke (ako je to moguće), formirano je unapred zadato suboptimalno rešenje, itd.

Algoritamski koraci GA i njegova struktura dati su u pseudokodu predstavljenom algoritmom 2.

2.2 Optimizacija kolonijom pčela

Optimizacija kolonijom pčela (engl. *Bee Colony Optimization*-BCO) spada u grupu metaheuristika inspirisanih prirodom i lako se može modifikovati i primeniti za rešavanje brojnih problema optimizacije. BCO je prvi put opisan i predložen kao metoda optimizacije u [115] i od tada je pretrpeo nekoliko izmena koje su bile usmerene prvenstveno ka pojednostavljivanju algoritma. Osim toga, razvijene su

Algorithm 2 Genetski algoritam

```
procedure GA(popSize, μ, υ)
    Pop ← GenerateInitialPopulation(popSize)
    Evaluate(Pop)
    while ¬ kriterijum zaustavljanja do
        for i ← 1 to (popSize/2) do
            newInd ← {}
            parents ← Selection(Pop, υ)
            offspring ← Recombine(parents)
            offspring ← Mutate(offspring, μ)
            Evaluate(offspring)
            newInd ← newInd ⋃ offspring
        end for
        Pop1 ← newInd ⋃ Pop
        Pop ← Replacement(Pop1)
    end while
end procedure
```

i neke nove varijante osnovnog algoritma, kao što je BCO zasnovan na popravci kompletnog rešenja koji je poznat pod nazivom BCOi.

BCO simulira prirodni proces koji slede pčele u postupku traženja hrane. Ovaj proces obično počinje tako što pčele izviđači pretražuju neko područje sa nektarom, nakon čega se vraćaju u košnicu i prenose sakupljene informacije ostalim članovima roja o položaju, kvalitetu i količini hrane u istraženom prostoru. Ako je pčela izviđač bila uspešna u potrazi za nektarom, po povratku u košnicu izvodi ples koji ima ulogu neke vrste reklamiranja istraženog prostora i podsticanja ostalih članova košnice da slede njenu putanju u traženju hrane. Pčele koje se odluče da krenu u potragu za nektarom slede već ispitanu rutu o kojoj ih je informisala neka od pčela izviđača. Po povratku u košnicu, ove pčele mogu da preduzmu različite aktivnosti:

- da napuste prethodno odabranu putanju do hrane i da ponovo postanu neopredeljene;
- da izvedu ples i pokušaju da regrutuju još pčela iz košnice;
- da nastave sakupljanje nektara bez regrutacije ostatka roja.

U osnovi BCO algoritma nalazi se populacija agenata-veštačkih pčela koji imitiraju prirodni postupak traženja hrane i pri tome pronalaze rešenja razmatranog problema optimizacije. Da bi pronašle najbolje rešenje, veštačke pčele razmenjuju informacije i pretragu usmeravaju na delove prostora pretrage koji sadrži rešenja boljeg kvaliteta.

Pseudokod za BCO dat je u algoritmu 3. Veštačke pčele sarađuju u postupku traženja rešenja problema optimizacije. Svaka od B pčela gradi po jedno dopustivo

rešenje problema. Algoritam se satoji od dve faze: leta unapred (engl. *forward pass*) u kome se pretražuje prostor pretrage i leta unazad (engl. *backward pass*) u kome pčele razmenjuju informacije. Tokom leta unapred, pčele konstruišu parcijalno rešenje, dok se u letu unazad vrši evaluacija kvaliteta formiranih parcijalnih rešenja i razmenjuju se dobijene informacije. U toku leta unapred, pčela koristi NC konstruktivnih koraka za formiranje parcijalnog rešenja. Sa unapred definisanom verovatnoćom, pčela odlučuje da li će ostati lojalna svom rešenju, regrutovati neopredeljene pčele i sa njima nastaviti da gradi bolje parcijalno rešenje ili će odbaciti svoje trenutno rešenje, postati neopredeljena i nakon regrutacije nastaviti da istražuje prostor dopustivih rešenja. Ukoliko pčela formira parcijalno rešenje dobrog kvaliteta, ima i veću verovatnoću da će mu ostati lojalna. Svaka neopredeljena pčela sa određenom verovatnoćom bira koju će lojalnu pčelu slediti i preuzeti njeno parcijalno rešenje. Bolja rešenja imaju veću verovatnoću da će biti izabrana od strane neopredeljenih pčela. Nakon preuzimanja parcijalnog rešenja, pčela nastavlja sa konstukcijom rešenja u sledećem letu unapred i može samostalno da donosi odluke o narednim koracima. Let unapred i let unazad se smenjuju NC puta sve dok pčela ne generiše kompletno rešenje problema. Nakon toga, među svim rešenjima koje su pčele konstruisale određuje se najbolje trenutno rešenje. Ovim rešenjem se ažurira globalno najbolje rešenje ukoliko je boljeg kvaliteta. Zatim se svim pčelama dodeljuje prazno rešenje, i započinje nova iteracija algoritma. Iteracije BCO algoritma se smenjuju sve dok se ne ispuni neki kriterijum zaustavljanja: unapred definisano procesorsko vreme, broj iteracija, broj iteracija bez popravke globalno najboljeg rešenja, itd.

Verovatnoća lojalnosti i regrutovanja se obično implementira izborom pomoću ruleta (slika 2.6) koji omogućava izbor svakog rešenja, pri čemu je verovatnoća izbora proporcionalna kvalitetu rešenja. Rešenja boljeg kvaliteta imaju veću verovatnoću izbora, ali se ne garantuje da će zaista i biti odabrana.

Verovatnoća da će b-ta pčela na početku novog leta unapred biti lojalna svom generisanom parcijalnom ili kompletnom rešenju, računa se po formuli:

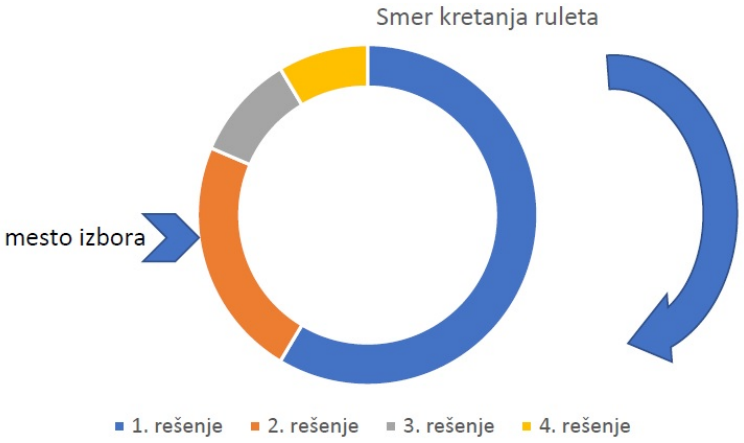
$$p_b^{u+1} = e^{-\frac{1-O_b}{u}}, \quad b = 1, 2, ..., B, \tag{2.1}$$

gde je O_b normalizovana vrednost funkcije cilja parcijalnog ili kompletnog rešenja koje je generisala b-ta pčela, a u brojač leta unapred koji može da uzima vrednosti $1, 2, \ldots, NC$.

Sa povećanjem vrednosti brojača u leta unapred, povećava se i uticaj već formiranog rešenja. To znači da se na početku pretrage pčele lakše odlučuju za nova

Algorithm 3 Optimizacija kolonijom pčela

```
procedure BCO
    Initialization(B, NC)
    while ¬ kriterijum zaustavljanja do
        for b ← 1 to B do
            solution(b) ← GenerateSolution()
        end for
        for u ← 1 to NC do
            for b ← 1 to B do
                EvaluateMove(solution(b))
                ChoseMove(solution(b))
            end for
            for b ← 1 to B do
                Loyalty(solution(b))
            end for
            for b ← 1 to B do
                if b ¬ lojalna then
                    Recruiting(solution(b))
                end if
            end for
        end for
        Update(x_best, f(x_best))
    end while
    Return(x_best, f(x_best))
end procedure
```

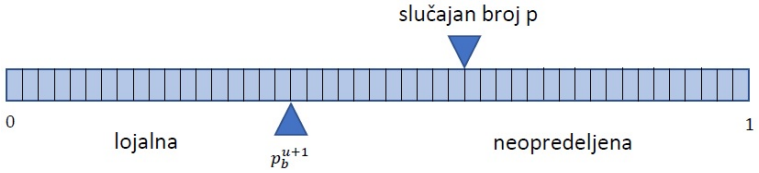


Slika 2.6: Selekcija ruletom

rešenja. Kako proces pretrage odmiče, pčele se sve više fokusiraju na poznata rešenja. Ova činjenica je izražena pozicijom parametra u u imeniocu razlomka koji

ima ulogu eksponenta u fomuli (2.1). Većim vrednostima O_b odgovaraju kvalitetnija rešenja, kao i veća verovatnoća da će pčela ostati lojalna svom rešenju.

Primenom formule (2.1) i izborom slučajnog realnog broja p iz intervala [0,1], veštačka pčela se odlučuje da li će postati neopredeljena i odbaciti svoje rešenje ili će ostati lojalna svom prethodnom rešenju. Ako je odabrani slučajni broj p manji od izračunate vrednosti verovatnoće, pčela će ostati lojalna svom rešenju. U suprotnom, ako je broj p veći od vrednosti verovatnoće p_b^{u+1}, veštačka pčela postaje nelojalna i odlučuje se koje rešenje preuzima od neke lojalne pčele (slika 2.7).



Slika 2.7: Određivanje verovatnoće i lojalnosti

Svaka neopredeljena pčela bira koju lojalnu pčelu će da prati, odnosno čije rešenje će preuzeti, uzimajući u obzir kvalitet svih rešenja lojalnih pčela. Verovatnoća da će parcijalno ili kompletno rešenje lojalne pčele b biti izabrano od strane neke neopredeljene pčele data je formulom:

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k}, \quad b = 1, 2, \ldots, R, \tag{2.2}$$

gde je sa O_k predstavljena normalizovana vrednost funkcije cilja k-tog ponuđenog rešenja, a R označava broj lojalnih pčela. Uzimajući u obzir vrednosti izračunate formulom (2.2) i vrednost slučajno generisanog realnog broja u postupku selekcije ruletom, svaka neopredeljena pčela bira jednu lojalnu pčelu koju će da sledi. U implementaciji BCO algoritma, ovo znači da će parcijalno ili kompletno rešenje lojalne pčele biti iskopirano u rešenje neopredeljene pčele.

Optimizacija kolonijom pčela sa popravkom kompletnog rešenja

Postoje dve varijante algoritma optimizacije kolonijom pčela: konstruktivna varijanta i varijanta u kojoj se popravlja kompletno rešenje. Kod konstruktivne varijante, veštačke pčele postepeno grade rešenje u nizu konstruktivnih koraka dok ne

Algorithm 4 Optimizacija kolonijom pčela sa popravkom rešenja

```
procedure BCOi
    Initialization(B, NC)
    while ¬ kriterijum zaustavljanja do
        for b ← 1 to B do
            solution(b) ← CompleteSolution()
        end for
        for u ← 1 to NC do
            for b ← 1 to B do
                Improve(solution(b))
            end for
            for b ← 1 to B do
                Loyalty(solution(b))
            end for
            for b ← 1 to B do
                if b ¬ lojalna then
                    Recruiting(solution(b))
                end if
            end for
        end for
        Update(x_best, f(x_best))
    end while
    Return(x_best, f(x_best))
end procedure
```

formiraju kompletno dopustivo rešenje problema. Za razliku od konstruktivne verzije algoritma, kod BCOi algoritma, svakoj pčeli se na početku algoritma dodeljuje kompletno rešenje koje ona dalje popravlja kroz iteracije algoritma. BCO algoritam sa popravkom kompletnog rešenja predložili su Davidović i sar. u radu [29].

Pseudokod varijante optimizacije kolonijom pčela sa popravkom kompletnog rešenja dat je u algoritmu 4. BCOi implementacija se sastoji od pet koraka:

- prvi korak je predprocesiranje u kome se ulazni podaci transformišu da bi se smanjilo vreme izračunavanja u kasnijim fazama algoritma;
- drugi korak je inicijalizacija kompletnih rešenja;
- u trećem koraku, pčele kroz NC letova unapred (u jednoj iteraciji algoritma) modifikuju trenutna rešenja. Ovim korakom se obezbeđuje da se ista rešenja različitih pčela modifikuju na različite načine, upotrebom stohastike i bez lokalne pretrage;

- četvrti korak uključuje evaluaciju rešenja i njihovo upoređivanje po kvalitetu;
- peti korak predstavlja postupak regrutovanja.

Ideja popravke kompletnog BCOi rešenja može da se implementira na različite načine što za posledicu ima da ovaj pristup rešavanju problema može dati dobre rezultate kod rešavanja teških problema optimizacije. U novijoj literaturi je BCOi metoda uspešno primenjena na rešavanje teških problema optimizacije u oblasti rutiranja vozila [132], problemima zadovoljivosti u logici [159], dizajniranju mreža [131], itd.

2.3 Metoda promenljivih okolina

Metoda promenljivih okolina (engl. *Variable Neighborhood Search*-VNS) je jednostavna ali efikasna metaheuristička metoda zasnovana na proceduri lokalnog pretraživanja [70]. VNS su prvi put predložili Mladenović i Hansen 1997. godine u radu [124], kao pogodnu metodu za rešavanje problema trgovačkog putnika. Osnovna ideja VNS-a je sistematska promena okolina u fazi spuštanja (engl. *descent phase*) u kojoj se pronalazi lokalni optimum, i u fazi razmrdavanja (engl. *shaking*) koja za cilj ima udaljavanje pretrage iz lokalnog optimuma. VNS metaheuristika se koristi za rešavanje brojnih problema kombinatorne i globalne optimizacije [70], kao što su problem trgovačkog putnika, problemi raspoređivanja, problemi teorije grafova, pakovanja, rutiranja vozila, itd. Pregled varijanti VNS metode i primena VNS algoritma na razne probleme optimizacije može se naći u radovima Hansena i sar. [67, 72].

VNS metoda se oslanja na tri jednostavna principa:

- lokalni minimum u odnosu na jednu okolinu ne mora biti i lokalni minimum u odnosu na neku drugu okolinu;
- globalni minimum je lokalni minimum u odnosu na sve okoline;
- za većinu problema, lokalni minimumi su međusobno bliski u odnosu na razne okoline.

VNS ima dve osnovne komponente: proceduru razmrdavanja (engl. *shaking*) koja menja rešenje i pomaže algoritmu da izađe iz lokalnog minimuma, i proceduru

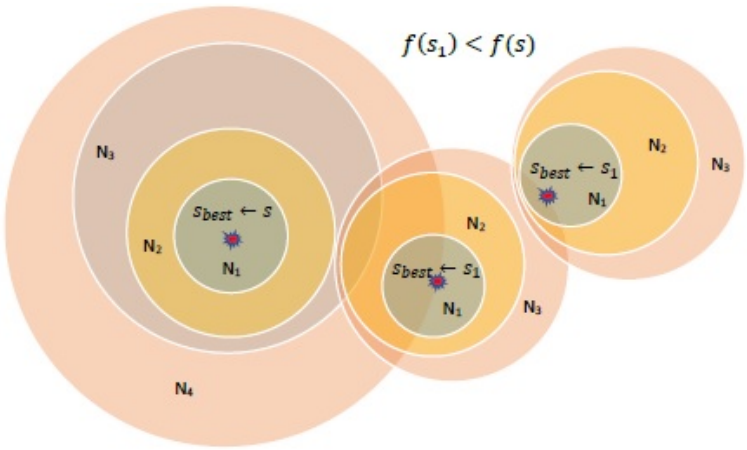
lokalne pretrage (engl. *local search*) koja, na osnovu predefinisane strukture okolina, pretražuje okolinu trenutnog rešenja. Ako se prednost daje fazi razmrdavanja pojačava se diversifikacija, dok stavljanje akcenta na drugu fazu pospešuje intensifikaciju pretrage. Ove dve faze se iterativno ponavljaju do zadovoljenja nekog od unapred definisanog kriterijuma zaustavljanja: maksimalno dozvoljeno vreme izvršavanja algoritma, broj iteracija između dve popravke rešenja, maksimalan broj iteracija algoritma, itd. Razvijene su brojne varijante osnovnog VNS algoritma koje se razlikuju u jednoj ili obe glavne komponente algoritma: metoda promenljivog spusta, redukovana metoda promenljivih okolina, opšta metoda promenljivih okolina, itd. Glavna razlika ovih varijanti VNS algoritma je u načinu pretrage okolina koja može da bude deterministička, slučajna ili da bude kombinacija determinističke i slučajne pretrage. Metoda promenljivog spusta je deterministička, redukovana metoda promenljivih okolina je stohastička dok opšta metoda promenljivih okolina koristi kombinovanu metodologiju. Stohastički pristup obezbeđuje veću diversifikaciju dok deterministička pretraga akcenat stavlja na intensifikaciji. Kod kombinovane metodologije se uočava balans intensifikacije i diversifikacije, pa obično ove metode imaju i veću šansu da nađu globalni optimum.

Osnovni parametar VNS metode je maksimalan broj okolina (k_{max}). U kombinatornoj optimizaciji, skup okolina se relativno lako može definisati odabirom nekog operatora koji definiše potez čijom primenom se formira prva okolina, a svaka sledeća se dobija iterativnom primenom istog operatora.

Definicija 4. Neka je s dopustivo rešenje problema a $N_k, k = 1, 2, \ldots, k_{max}$ skup definisanih okolina. Tada se za rešenje σ kaže da je na rastojanju k od rešenja s u odnosu na uvedenu metriku ako se σ može dobiti sa k transformacija iz rešenja s. Skup $N_k(s)$ svih rešenja σ predstavlja k-tu okolinu rešenja s.

Mnoge metaheuristike koriste u pretrazi jednu ili dve okoline. Na slici 2.8 ilustrovana je pretraga kroz više definisanih okolina. Kod pretrage kroz više okolina potrebno je dobro definisati okoline i strategije pretrage, voditi računa o veličini i redosledu okolina, smeru njihove pretrage, načinima intensifikacije i diversifikacije pretrage kao i o balansu između spusta i penjanja.

Kod heuristika lokalnog pretraživanja, polazna tačka je neko inicijalno rešenje s, definisan smer spusta i okolina rešenja $N(s)$ u okviru koje se vrši pomeranje u istom smeru do najmanje vrednosti $f(s)$. Kao alternativa potpunom pretraživanju, koje je poznato pod nazivom strategija najboljeg poboljšanja (engl. *best improve-*



Slika 2.8: Lokalno pretraživanje sa okolinama

ment), a koje može biti vremenski zahtevno, uvodi se strategiija prvog poboljšanja (engl. *first improvement*). U slučaju ove strategije, rešenja iz $N(s)$ se razmatraju sistematično, a pomeranje se vrši odmah po pronalasku boljeg rešenja. Polazeći od početnog rešenja s, LS ispituje vrednost funkcije cilja za svako rešenje iz okoline $N(s)$. Najbolje rešenje u celoj okolini čuva se kao s_{min}. Nakon obilaska okoline $N(s)$, pretraga se premešta u okolinu $N(s_{min})$. Postupak se nastavlja iterativno sve dok u okolini $N(s_{min})$ ima boljih rešenja. Često se dešava da je okolina $N(s_{min})$ velike kardinalnosti, pa se pribegava raznim metodama redukovanja pretrage okoline u cilju smanjenja utrošenog vremena i povećanja efikasnosti heuristike.

Metoda promenljivog spusta

Metoda promenljivog spusta (engl. *Variable Neighborhood Descent*-VND) je deterministička varijanta metode promenljivih okolina. Koraci VND metode ilustrovani su u pseudokodu u algoritmu 5 i to za slučaj algoritma koji rešava problem minimizacije vrednosti funkcije cilja $f(s)$. Polazeći od generisanog početnog rešenja s, VND u svakoj od definisanih okolina $N_k, k = 1, 2, \ldots, k_{max}$ vrši lokalno pretraživanje. Sa k_{max} je označen maksimalan broj okolina. Okolina $N_1(s)$ se pretražuje sve dok u njoj ima poboljšanja rešenja. Trenutno najbolje rešenje se ažurira lokalnim minimumom s_1 iz okoline $N_1(s)$, i prelazi se na pretragu okoline N_2. Ukoliko se u nekoj okolini $N_k, k = 1, 2, \ldots, k_{max}$ pronađe lokalni minimum s_1 koji je bolji od trenutno najboljeg rešenja s_{best}, prvo se ažurira s_{best} i pretraga se nastavlja u N_1 okolini novog trenutno najboljeg rešenja. Ako ne dođe do popravke trenutno najbo-

ljeg rešenja, pretraga se premešta u sledeću okolinu, povećavanjem brojača okolina k. Pretraga se zaustavlja ukoliko se trenutno najbolje rešenje ne može popraviti ni u jednoj od okolina $N_1, N_2, \ldots, N_{k_{max}}$.

Algorithm 5 Metoda promenljivog spusta

```
procedure VND
    s ← InitialSolution()
    k ← 1
    while k ≤ k_max do
        s_1 ← BestNeighbor(s, k)
        if f(s_1) < f(s) then
            s ← s_1
            k ← 1
        else
            k ← k + 1
        end if
    end while
    Return(s)
end procedure
```

Višestartna metoda promenljivog spusta

Lokalna pretraga primenjena na probleme optimizacije često ostaje zarobljena u lokanom optimumu. Uobičajen način rešavanja ovog problema je implementacija višestartne metode lokalnog pretraživanja. Višestartne metode lokalnog pretraživanja su iterativne procedure koje se sastoje od dve faze: u prvoj se generiše početno rešenje, u drugoj fazi se najčešće to rešenje popravlja nekom od heuristika zasnovanih na lokalnom pretraživanju. Svaka iteracija algoritma proizvodi rešenje koje je lokalni optimum, a najbolje rešenje iz svih iteracija je krajnji rezultat algoritma. Interakcija između dve faze treba da obezbedi ravnotežu između diverzifikacije i intenzifikacije pretrage, u cilju dobijanja visoko kvalitetnih rešenja. Jedan od načina interakcije pomenutih faza je da se početno rešenje u svakoj narednoj iteraciji generiše iz lokalnog optimuma dobijenog u prethodnoj iteraciji adekvatnim metodama perturbacije ili se početno rešenje za narednu iteraciju generiše slučajnim izborom.

Na ovim jednostavnim idejama zasniva se i višestartna metoda promenljivog spusta (engl. *Multistart Variable Neighborhood Descent*-MSVND). MSVND iterativno pretražuje prostor pretrage, tako što se VND procedura restartuje više puta iz različitih slučajno generisanih početnih rešenja i najbolji lokalni minimum se vraća

kao rešenje problema. Procedura se zaustavlja kada se ispuni neki unapred definisan kriterijum zaustavljanja. Implementacija MSVND metode je predstavljena pseudokodom prikazanim u algoritmu 6.

Algorithm 6 Višestartna metoda promenljivog spusta

```
procedure MSVND
    while ¬ kriterijum zaustavljanja do
        s ← RandomSolution()
        k ← 1
        while k ≤ k_max do
            s_1 ← BestNeighbor(s, k)
            if f(s_1) < f(s) then
                s ← s_1
                k ← 1
            else
                k ← k + 1
            end if
        end while
    end while
    Return(s)
end procedure
```

Redukovana metoda promenljivih okolina

Redukovana metoda promenljivih okolina (engl. *Reduced Variable Neighborhood Search*-RVNS) je pogodna za primenu na instancama problema optimizacije velikih dimenzija, na kojima je primena lokalnog pretraživanja vremenski zahtevna. RVNS metoda nema fazu lokalnog pretraživanja, a zasnovana je na stohastičkom izboru jednog rešenja u svakoj od unapred definisanih okolina. Struktura RVNS metode za rešavanje problema minimizacije predstavljena je algoritmom 7. Primena RVNS metode zahteva definiciju strukture okolina rešenja $N_k, k = 1, 2, \ldots, k_{max}$ koja se koristi u fazi razmrdavanja. Izbor slučajnog rešenja je implementiran u fazi razmrdavanja kroz proceduru Shake(S,k) koja za dato rešenje s generiše slučajno rešenje iz njegove k-te okoline. Preostali koraci algoritma su identični kao kod VND metode. Na slučajan način se bira rešenje iz prve okoline. Ukoliko novo rešenje ima bolju vrednost od vrednosti trenutnog rešenja pretraga se premešta u bolje rešenje ($s \leftarrow s_1$) a u suprotnom se pretraga nastavlja u sledećoj okolini. Nakon poboljšanja u nekoj od okolina ili pretrage svih okolina, ponovo se prelazi u prvu okolinu i postupak se ponavlja sve dok se ne ispuni kriterijum zaustavljanja.

Algorithm 7 Redukovana metoda promenljivih okolina

```
procedure RVNS
    s ← InitialSolution()
    while ¬ kriterijum zaustavljanja do
        k ← 1
        while k ≤ k_max do
            s_1 ← Shake(s, k)
            if f(s_1) < f(s) then
                s ← s_1
                k ← 1
            else
                k ← k + 1
            end if
        end while
    end while
    Return(s)
end procedure
```

Skup okolina $N_k, k = 1, 2, \ldots, k_{max}$ je najčešće ugnežden, odnosno svaka sledeća okolina sadrži prethodnu, odatle proizilazi da dimenzije okolina obično rastu. Posledica ove činjenice je da se temeljnije pretražuju okoline koje su na malom rastojanju od rešenja s u odnosu na one koje su na većem rastojanju.

Osnovna metoda promenljivih okolina

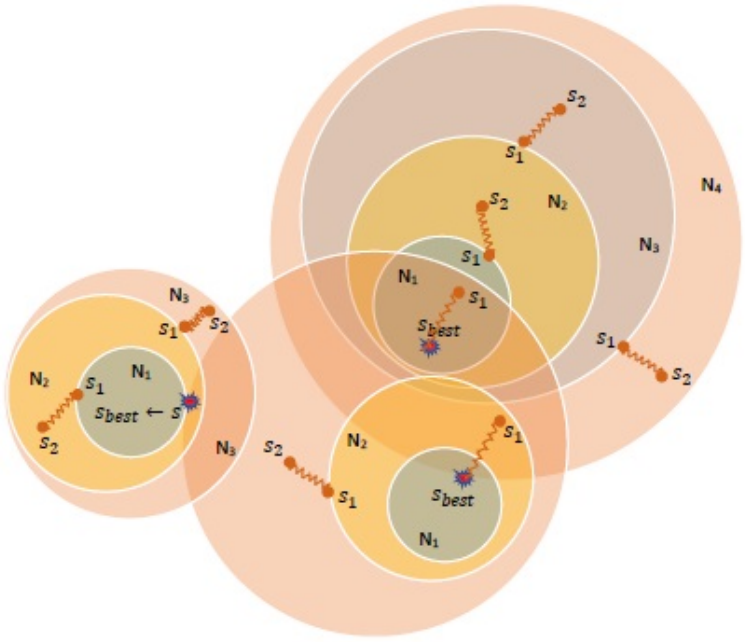
Osnovna metoda promenljivih okolina (engl. *Basic Variable Neighborhood Search*-BVNS) nastaje kombinacijom determinističke pretrage okolina i stohastičkog izbora početnog rešenja u trenutnoj okolini, nad kojim se vrši lokalna pretraga. Struktura BVNS metode za rešavanje problema mimimizacije date funkcije cilja ilustrovana je algoritmom 8. Primena faze razmrdavanja zahteva definiciju $N_k, k = 1, 2, \ldots, k_{max}$ okolina rešenja. U fazi razmrdavanja se bira slučajno rešenje iz k-te okoline trenutnog rešenja, a zatim se nad razmrdanim rešenjem primenjuje faza lokalnog pretraživanja. Kao i kod prethodno izloženih varijanti VNS algoritma, ukoliko je dobijeni lokalni optimum bolji od trenutno najboljeg rešenja, ažurira se vrednost najboljeg rešenja i pretraga se nastavlja u njegovoj okolini N_1. U suprotnom, se povećava brojač okolina, i pretraga se nastavlja u sledećoj neposećenoj okolini N_k. Opisani koraci se iterativno ponavljaju dok se ne ispuni kriterijum zaustavljanja.

Faza razmrdavanja obezbeđuje da se pretražuju različiti regioni prostora pretrage. Okoline koje definišu razmrdavanje i okoline u kojima se vrši lokalna pretraga,

Algorithm 8 Osnovna metoda promenljivih okolina

```
procedure BVNS
    s ← InitialSolution()
    while ¬ kriterijum zaustavljanja do
        k ← 1
        while k ≤ k_max do
            s_1 ← Shake(s, k)
            s_2 ← LocalSearch(s_1)
            if f(s_2) < f(s) then
                s ← s_2
                k ← 1
            else
                k ← k + 1
            end if
        end while
    end while
    Return(s)
end procedure
```

mogu ali i ne moraju biti jednake. Na slici 2.9 ilustrovana je promena okolina u BVNS metodi sa strategijom prvog poboljšanja.



Slika 2.9: Promena okolina u BVNS metodi

Opšta metoda promenljivih okolina

Osnovna razlika između osnovne i opšte metode promenljivih okolina ogleda se u fazi intenzifikacije pretrage. Nakon faze razmrdavanja, opšta metoda promenljivih okolina (engl. *General Variable Neighborhood Search*-GVNS) umesto lokalne pretrage, koristi metodu promenljivog spusta-VND. Koraci opšte metode promenljivih okolina dati su u algoritmu 9. Okoline koje se koriste kod razmrdavanja i u VND fazi mogu biti iste ili različite, ili eventualno mogu da imaju neke zajedničke elemente.

Algorithm 9 Opšta metoda promenljivih okolina

```
procedure GVNS
    s ← InitialSolution()
    while ¬ kriterijum zaustavljanja do
        k ← 1
        while k ≤ k_max do
            s_1 ← Shake(s, k)
            s_2 ← VND(s_1)
            if f(s_2) < f(s) then
                S ← s_2
                k ← 1
            else
                k ← k + 1
            end if
        end while
    end while
    Return(s)
end procedure
```

Adaptivna metoda promenljivih okolina

Kod nekih prblema optimizacije, dešava se da udaljeni regioni prostora pretrage sadrže rešenja bliska optimumu. U takvim slučajevima, neophodno je modifikovati osnovnu VNS metodu tako da se omogući ispitivanje delova prostora pretrage koji su udaljeni od trenutno najboljeg rešenja. Modifikacija se sastoji u prihvatanju rešenja bliskih trenutno najboljem rešenju, koja nisu nužno bolja od trenutno najboljeg rešenja. Dobijena varijanta VNS metode poznata je kao adaptivna metoda promenljivih okolina (engl. *Skewed Variable Neighborhood Search*-SVNS).

Algorithm 10 Adaptivna metoda promenljivih okolina

```
procedure SVNS
    s ← InitialSolution()
    while ¬ kriterijum zaustavljanja do
        k ← 1;
        s_best ← s;
        while k ≤ k_max do
            s_1 ← Shake(s, k)
            s_2 ← FirstImprovement(s_1);
            s_best ← FindBetter(s_best, s_2);
            if f(s_2) − αρ(s_2, S) < f(s) then
                s ← s_2;
                r ← 1;
            else
                r ← r + 1;
            end if
        end while
        S ← S_best;
    end while
    Return(s_best)
end procedure
```

Pseudokod ove metode je predstavljen algoritmom 10. Nakon razmrdavanja trenutnog rešenja prihvata se prvo bolje rešenje s_2 i pretraga se premešta u njegovu okolinu ukoliko je vrednost izraza $f(s_2) - \alpha\rho(s_2, S)$ manja od vrednosti funkcije cilja trenutnog rešenja, čak i pod uslovom da rešenje s_2 nije bolje od trenutnog rešenja s. Premeštanje fokusa pretrage se kontroliše izračunavanjem vrednosti izraza $f(s_2) - \alpha\rho(s_2, S)$ gde je sa $\rho(s_2, S)$ označena međusobna udaljenost rešenja s_2 i s a α je parametar metode. Procedura FindBetter(a,b) upoređuje dva rešenja a i b i bolje rešenje vraća kao izlaznu vrednost.

Parametar α se bira tako da omogući pretragu udaljenih prostora rešenja. Adekvatna vrednost α parametra se određuje eksperimentalno i zavisi od prirode problema koji se rešava. U slučajevima kada je udaljenost rešenja s_2 i s mala, male vrednosti parametra α sprečavaju česta premeštanja pretrage u rešenja bliska trenutnom rešenju s. Osim na ovaj način, vrednost parametra α može da se odredi i sofisticiranim strategijama koje uključuju procese učenja i prenošenja znanja.

Pored navedenih varijanti metode promenljivih okolina, u literaturi su poznate i druge modifikacije osnovne VNS metode:

- metoda promenljivih okolina bazirana na dekompoziciji (engl. *Variable neighborhood Decomposition Search*-VNDS);
- primalno-dualna metoda promenljivih okolina (engl. *Primal-dual VNS-PDVNS*);
- paralelna metoda promenljivih okolina (engl. *Parallel Variable Neighborhood Search*-PVNS);
- metoda promenljivih formulacija (engl. *Variable Neighborhood Formulation Space Search*-VNFSS);
- metoda programiranja promenljivih okolina (engl. *Variable Neigborhood Programming*-VNP);

i druge. Detaljan pregled varijanti metode promenljivih okolina može se pronaći u radovima [71, 124, 135, 139].

2.4 Ostale metaheurističke metode

U prthodnim potpoglavljima detaljnije su opisane metaheurističke metode koje su korišćene u disertaciji. U nastavku je dat kratak pregled ostalih metaheurističkih metoda koje se takođe često koriste za rešavanje problema optimizacije.

Tabu pretraživanje (engl. *Tabu Search*-TS) je metaheuristika koja kontroliše heurističku proceduru lokalnog pretraživanja na način da se može prevazići problem zaglavljivanja u lokalnom optimumu. Ova metaheuristika je prvi put predložena u [55], a danas se često koristi u rešavanju kombinatornih optimizacionih problema. TS se zasniva na korišćenju adaptivne memorije i pretrage tabu liste. Detalji algoritma i noviji trendovi u primeni TS-a mogu se naći u [52, 53].

Pohlepna stohastičko-adaptivna procedura pretrage (engl. *Greedy Randomized Adaptive Search Procedure*-GRASP) je metaheuristički algoritam koji se sastoji od dve faze: pohlepne stohastičko-adaptivne faze konstrukcije rešenja (engl. *greedy randomized solution constuction*) i lokalnog pretraživanja koje popravlja generisana rešenja. Ove dve faze se naizmenično ponavljaju dok se ne ispuni kriterijum zaustavljanja. GRASP je predstavljen u radu [42]. U prvoj fazi, pohlepni algoritam donosi odluke o narednom koraku u kreiranju rešenja. Algoritam uvek bira akcije koje se čine kao najbolje u tom momentu što rezultira u formiranju rešenja koja su bliska optimalnom. Detaljan opis algoritma, njegova proširenja i primene mogu se naći u [146].

Optimizacija škripećeg točka (engl. *Squeaky Wheel Optimization*-SWO) je predložena u [82] kao tehnika pretrage koja ima sposobnost rešavanja širokog spektra optimizacionih problema. SWO koristi pohlepni pristup za konstrukciju početnog rešenja. Nakon ove faze sledi postupak provere formiranih rešenja i pronalaženja potencijanlo dobrih mesta na kojima se početno rešenje i vrednost funkcije cilja mogu popraviti. Uočena mesta popravke se koriste za definisanje prioriteta i redosleda konstruktivnih koraka pohlepnog algoritma.

Kaljenje metala je prirodni proces u kome se metali prvo zagrevaju a zatim postepeno hlade kako bi promenili fizičke karakteristike i unutrašnju strukturu. Simulirano kaljenje (engl. *Simulated Annealing*-SA) koje je razvijeno u [91] simulira prirodni proces kaljenja metala tako što energiju sistema posmatra kao funkciju cilja. Algoritam sa određenom verovatnoćom menja trenutno rešenje sa slučajno generisanim rešenjem. Verovatnoća izbora novog rešenja zavisi od razlike u energijama dva rešenja i od vrednosti parametra koji opisuje temperaturu. Kod velikih vrednosti temperature prihvataju se rešenja lošijeg kvaliteta. Temperatura se smanjuje u toku izvršavanja algoritma čime se pretraga fokusira samo na rešenja boljeg kvaliteta od trenutnog. Efikasnost SA zavisi od nekoliko parametara kao što su početna temperatura, stepen hlađenja i vrednosti temperaturne razlike koja obično definiše veličinu okoline za posmatrano rešenje. Sveobuhvatni pregled SA optimizacionog algoritma dat je u [130, 160].

Optimizacija rojem čestica (engl. *Particle Swarm Optimization*-PSO) predložena je u radu [86], kao tehnika optimizacije zasnovana na ponašanju ptica u jatu, kretanju skupa čestica u prostoru. Kod PSO metode čestice odgovaraju rešenjima problema i kreću se kroz prostor pretrage, koristeći sopstveno iskustvo i iskustvo roja. Poziciju čestice karakteriše vektor položaja, vektor brzine i pravac kretanja. PSO počinje sa početnim rojem čestica koje se raspoređuju na slučajne pozicije pretraživačkog prostora i dodeljuje im se slučajni pravac kretanja, a zatim se, dok se ne ispuni neki kriterijum zaustavljanja, ponavljaju iteracije algoritma u kojima se ažuriraju vektori položaja i kretanja čestica. Detaljno istraživanje o modifikacijama i aplikacijama PSO-a može se pronaći u [184].

Optimizacija kolonijom mrava (engl. *Ant Colony Optimization*-ACO) je metaheuristička metoda predložena u [24] i zasnovana na ponašanju mrava pri njihovom pokušaju da pronađu optimalan put od mravinjaka do lokacije hrane. Veštački mrav generiše po jedno rešenje slučajnim izborom komponenti proporcionalno nivou dodeljenog feromona. Da bi se povećala slučajnost pretrage i istraživanje novih regiona

koristi se činjenica da u prirodi feromon isparava s vremenom. Zbog toga, ACO algoritam sadrži parametar koji opisuje koeficijent isparavanja, sa kojim se na kraju svake iteracije ažurira vrednost feromona svake komponente. Vrednost feromona se povećava za komponente koje odgovaraju kvalitetnim rešenjima a feromon isparava za komponente loših rešenja, čime se povećava verovatnoća izbora visokokvalitetnih rešenja u narednim iteracijama algoritma. Opis strukture algoritma i pregled primene mogu se pronaći u [35, 36].

Metaheuristika delimične optimizacije uz posebne uslove intensifikacije (engl. *Partial Optimization Metaheuristic Under Special Intensification Conditions* - POPMUSIC) zasniva se na ideji da se posmatrani problem može podeliti na manje podprobleme koji se mogu dalje rešti do lokalnog optimuma. Iz tog razloga POPMUSIC se može posmatrati kao algoritam lokalne pretrage koji se obično koristi u specifičnim problemima sa mnogo okolina. U radu [161] POPMUSIC je predložen kao pogodna metoda za rešavanje kompleksnih kombinatornih problema.

U novijoj literaturi se pojavljuju algoritmi koji ne slede paradigmu samo jedne tradicionalne metaheuristike već kombinuju komponente različitih algoritama koji često potiču iz različitih oblasti istraživanja. Ovakav pristup kombinovanju metaheurističkih metoda naziva se hibridna metaheuristika. Hibridni algoritam je algoritam koji se sastoji iz dva ili više algoritama koji se izvršavaju zajedno i sarađuju u rešavanju unapred definisanog problema tako da se nedostaci jednog algoritma eliminišu prednostima drugog [148].

Kombinovanje algoritama koji se hibridizuju ne treba da bude slučajno već da se bazira na detaljnoj analizi dobrih i loših aspekata svakog algoritma. Spora konvergencija, lako zaustavljanje u lokalnom optimumu, ili neki sličan nedostatak poništava se kombinovanjem sa koracima drugog algoritma koji može da poboljša performanse formirane strategije. Zbog toga, hibridni pristup rešavanju problema optimizacije je generalno komplikovan proces koji zahteva široka znanja i ekspertizu u raznim područjima optimizacije.

Hibridni pristupi iz literature se ukratko mogu klasifikovati kao [9]: hibridizacija metaheuristike sa drugom metaheuristikom ili heuristikom, hibridizacija metaheuristike i programiranja ograničenja, hibridizacija metaheuristike i tehnike pretrage stabla, hibridizacija metaheuristike i relaksacije problema, hibridizacija metaheuristike sa dinamičkim programiranjem, i druge.

Neki od načina realizacije hibridizacije su: hibridi niskog nivoa (funkcija jednog algoritma, obično funkcija pretrage, zamenjuje se metodom drugog algoritma),

hibridi visokog nivoa (algoritmi se kombinuju bez menjanja načina funkcionisanja pojedinačnih algoritama), globalni hibridi (svaki algoritam vrši pretragu celog prostora rešenja), parcijalni hibridi (svaki algoritam rešava podproblem dobijen dekompozicijom problema optimizacije koji se rešava), kolaborativni hibridi (algoritmi se izvršavaju sekvencijalno ili paralelno), integrativni hibridi (jedan algoritam je nadređen a drugi podređen u zavisnosti od stepena učešća u procesu pretrage), i drugi [168].

Preuranjena konvergencija je inherentna karakteristika GA. Hibridizacija evolutivnog algoritma sa metodom lokalne pretrage vodi do konstrukcije memetskog algoritma. Memetski algoritmi (engl. *Memetic Algorithm*-MA) su dizajnirani tako da prevaziđu ovu neželjenu karakteristiku. MA je predložen u [126] i primenjuje operatore koji kombinuju lokalna poboljšanja skupa početnih rešenja da bi formirali novo rešenje. U osnovi, MA pretražuje okolinu datog rešenja. Svaki MA prolaz sastoji se od četiri uzastopna koraka: odabira roditelja, uparivanja roditelja za formiranje nove generacije potomaka, lokalnog poboljšavanja potomaka i ažuriranja populacije. Detaljan pregled i opis MA algoritma, može se naći u [129].

Zbog napretka u razvoju egzaktnih metoda neki MILP modeli mogu da se reše do optimalnosti ili blisko optimumu u razumnom vremenu izračunavanja. Ova činjenica je podstakla istraživače da dizajniraju heuristike koje u pojedinim fazama koriste MILP modele ili matematičko programiranje. Ovakvom vrstom hibridizacije formirane su takozvane matematičke metaheurističke metode ili matheuristike (engl. *Matheuristics*) [11]. Bitna karakteristika ovih metoda je upotreba matematičkog modela problema koji se rešava, tako da su ove metode poznate još i pod nazivom metaheuristike bazirane na modelu (engl. *Model-based metaheuristics*).

Postoje razni načini realizacije kombinovanja metaheuristike i matematičkog programiranja. U prvom pristupu, matematičko programiranje se izvršava nakon što je metaheuristika vratila rešenje problema optimizacije, sa ciljem da se rešenje popravi do optimalnosti. U drugom slučaju koristi se obrnuti pristup, kvalitetno početno rešenje se generiše matematičkim programiranjem a zatim se nad njim primenjuje neka metaheuristička metoda. Poslednji, često korišćeni pristup je integracija matematičkog programiranja direktno u neku od faza metaheuristike sa ciljem unapređivanja procesa pretrage. Kompleksni problemi optimizacije se obično mogu razložiti na podprobleme koji se mogu rešiti nezavisno a zatim se rešenja podproblema kombinuju u dopustivo rešenje problema. Kod matheuristikog pristupa rešavanju ovakvog vida problema, jedan ili više podproblema se rešavaju tehnikama matematičkog pro-

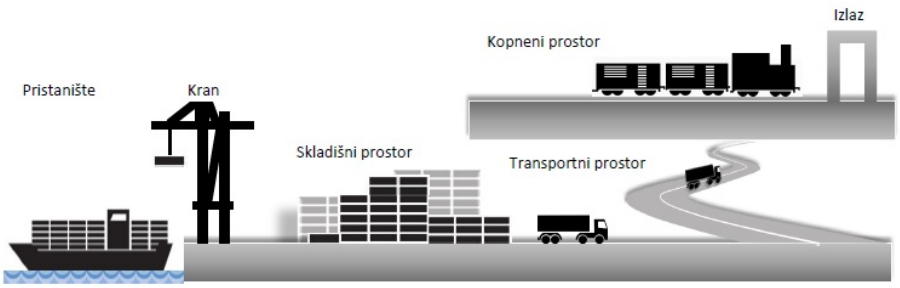
gramiranja.

Hibridne metaheuristike su uspešno primenjene pri rešavanju raznih problema optimizacije u telekomunikacijama [178], obradi slika [6], analizi strukture proteina [164], neuralnim mrežama [83], lokacijskim problemima [123], klaster analizi [45], raspoređivanju [122], itd.

Glava 3

Kontejnerski terminal

Kontejnerski terminal (engl. *container terminal*-CT) se može opisati kao otvoreni sistem sa dva spoljna pristupna područja u kojima se obavlja protok kontejnerskog tereta. Pristupna područja su *pristanište* (engl. *quayside*) u kome se teret iskrcava ili ukrcava u brodove i *kopneni prostor* ili *zaleđe* (engl. *landside*) koje predstavlja prostor eksternog transporta u kome se, obično kamionima i vozovima, kontejnerski teret dalje prenosi van luke u šire kopneno područje do destinacija korisnika. Kao spona između pristaništa i kopnenog prostora, u kontejnerskom terminalu se obično grade još i *transportni* (engl. *transport area*) i *skladišni prostor* (engl. *yard area*) [119]. Transportni prostor ima ulogu prenošenja kontejnerske robe unutar samog terminala, dok se u skladišnom prostoru privremeno smeštaju kontejneri dok ne budu upućeni na krajnje destinacije. Slika 3.1 prikazuje prostornu strukturu kontejnerskog terminala.



Slika 3.1: Prostorni prikaz kontejnerskog terminala

Kada brod stigne u pristanište i dodeli mu se vez, treba istovariti kontejnere koje je prevozio. *Kranovi* (engl. *quay cranes*-QC) premeštaju kontejnere sa broda na

specijalizovana vozila (engl. *automated guided vehicles*) koja dalje prenose kontejnere do privremenog skladišnog prostora u kome se kontejneri čuvaju određeni kraći vremenski period, složeni u veliki broj grupa u obliku traka (staza). Broj kontejnera koji će biti utovaren ili istovaren je poznat unapred, tj., pre dolaska broda u terminal. Broj kontejnera kojima CT može rukovati zavisi od broja i kapaciteta kranova kojima je terminal snabdeven. Staze u koje su smešteni kontejneri opslužuju specijalizovani kranovi (engl. *automated stacking cranes*-ASC) koji imaju ulogu da preuzmu kontejner i smeste ga na predviđeno mesto unutar postojećih traka. Jedan od najzagušenijih delova CT-a je privremeni skladišni prostor u kome se smeštaju prazni i puni kontejneri. Kako je jedan od zadataka menadžera terminala efikasna organizacija kompletnog transporta kontejnera na terminalu, oni posebno vode računa da se ovo zagušenje svede na minimum tako što skraćuju vreme zadržavanja kontejnera u privremenim skladištima. Nakon određenog perioda skladištenja, ASC kranovi ponovo prenose kontejnere do specijalizovanih vozila koji ih dalje transportuju do barži, brodova, kamiona ili vozova. Sličan postupak se sprovodi i kada se kontejneri utovaraju na kontejnerske brodove.

Problemi odlučivanja u pristaništu imaju značajan uticaj na operativne karakteristike CT-a. Produktivnost CT-a direktno zavisi od efikasnog načina planiranja operacija povezanih sa utovarom i istovarom broda: alokacija vezova, dodela kranova, raspoređivanje kranova i planiranje skladištenja kontejnera.

Alokacija vezova (engl. *berth allocation*) je kompleksan postupak raspoređivanja brodova na unapred definisane pozicije duž slobodnih vezova u pristaništu. Obično se u pristaništu u razmatranom vremenskom trenutku nalazi više brodova, tako da nekoliko parametara utiče na donošenje odluke o mestu vezivanja broda: omiljena pozicija broda (engl. *preferred berth*), koja najčeće predstavlja najbližu poziciju rezervisanom skladišnom prostoru za kontejnere koji se transportuju, međusobni raspored brodova koji treba da preuzmu kontejnere koji se samo pretovaraju ali ne i skladište u terminal, visina i dohvat raspoloživih kranova, visina i gaz broda i neki drugi faktori. Donošenje kvalitetnih odluka u procesu dodele vezova je vrlo važan proces, jer utiče na celokupan transport kontejnera kroz terminal, prvenstveno zbog toga što dobra pozicija veza svakog broda skraćuje ukupan pređeni put ostalih prevoznih sredstava u procesu protoka kontejnera kroz terminal. Aktivnosti vezane za proces dodele vezova počinju sa dolaskom broda u luku. Ukoliko su neki od vezova slobodni, menadžer terminala donosi odluku o tome koji će vez biti dodeljen. U suprotnom, tj., ako nema raspoloživih vezova, brod se stavlja u red čekanja.

Problem dodele kranova (engl. *Quay Crane Assignment Problem*-QCAP) ima za cilj dodelu određenog broja slobodnih kranova brodu kojem je prethodno dodeljen vez. Problem dodele kranova se vrlo često posmatra zajedno sa problemom dodele vezova. Veći broj dodeljenih kranova skraćuje vreme zauzetosti veza, jer se na taj način skraćuje vreme obrade broda i vez se brže oslobađa za obradu sledećeg broda. Vreme obrade broda se definiše unapred i svako prekoračenje vremena se dodatno naplaćuje (kažnjava). Nakon što je završen proces dodele kranova, pristupa se *raspoređivanju kranova* (engl. *Quay Crane Scheduling Problem*-QCSP), koje za cilj ima definisanje optimalnog procesa utovara i istovara broda na kontejnerskom nivou. U ovom procesu se uzimaju u obzir: raspored kontejnera na brodu (kako bi se izbegao negativan međusobni uticaj kranova), produktivnost svakog pojedinačnog krana, udaljenost kranova, struktura broda i drugi parametri.

Planiranje skladištenja (engl. *yard planning*) je postupak dodele skladišnog prostora pojedinačnim kontejnerima. Dodela skladišta se vrši u zavisnosti od protoka kontejnera (ponovo se ukrcavaju na brodove, prevoze se na kopno), vrste tereta koji je smešten u kontejner (opasne materije, hrana), dimenzije kontejnera (standardne dimenzije, predimenzionirani kontejneri), itd. Vrlo je važno obezbediti optimalan raspored kontejnera u skladišnom prostoru, jer je ponovno rukovanje i premeštanje (engl. *housekeeping*) već uskladištenih kontejnera sa jednog na drugo mesto unutar skladišta nepoželjno i utiče na povećanje troškova CT-a.

Kako CT predstavlja neku vrstu spone između raznih transportnih puteva, jasno je da produktivnost CT-a direktno utiče i na globalnu ekonomiju. Da bi privukli što više prevoznika kontejnerske robe, menadžeri terminala sa jedne strane imaju zadatak da ubrzaju što je moguće više operacija na terminalu i da pruže informacije o protoku kontejnera u realnom vremenu, a sa druge strane da smanje troškove efikasnim korišćenjem vezova, kontejnerskog skladišta, kranova i ljudskih resursa. Iz navedenih razloga, definišu se razni ciljevi optimizacije, kao što su: minimizacija troškova, maksimalna iskorišćenost kranova, minimizacija premeštanja kontejnera u skladišnom prostoru, maksimizacija produktivnosti obrade broda, minimizacija ukupnog vremena obrade brodova, ali i neki drugi specifični ciljevi za konkretnu realnu situaciju koji povećavaju efikasno poslovanje kontejnerskog terminala i luke.

Adekvatan broj vezova i njihova dobra iskorišćenost mogu da povećaju produktivnost CT-a. Menadžeri terminala poseduju sve informacije o brodovima koji dolaze u terminal i koriste ih da bi brodu dodelili konkretan vez na kome će biti izvršena obrada (utovar i/ili istovar kontejnera) u određenom vremenskom intervalu za dati

planirani poslovni period. Najčešće se razmatra period od jedne ili dve radne nedelje i za njega menadžeri terminala pripremaju plan vezivanja koji treba da garantuje da će svi najavljeni brodovi biti obrađeni u najkraćem vremenu i uz minimalne troškove. Da bi se ovi ciljevi ispunili potrebno je voditi računa o nekoliko faktora, kao što su: vreme, dubina veza, dužina broda, tip broda i ukupno vreme zadržavanja brodova u luci. Vremenski faktor zavisi od očekivanog vremena dolaska broda, ali i od vremena dostupnosti veza. Da bi se obezbedilo sigurno kretanje broda unutar veza, njegova dubina mora biti veća od dubine gaza broda, dok dužina veza takođe mora biti veća od same dužine broda. Osim toga, vez mora imati i adekvatnu opremu za efikasan utovar i istovar kontejnera. Tip broda, odnosno njegova veličina, utiče na broj kranova koji će mu biti dodeljeni, a broj dodeljenih kranova definiše vreme zadržavanja broda na vezu. Dostupnost vezova, vreme dolaska broda u luku, vreme početka obrade broda, kao i vreme koje je potrebno da se završi utovar i istovar kontejnera, utiču na ukupno vreme zadržavanja brodova u luci. Menadžeri luke planiraju pre samog dolaska broda u luku sve aktivnosti vezane za obradu broda da bi skratili vreme koje brod provodi na vezu, i da bi smanjili ukupno vreme koje brod provodi u luci.

Generalno, odluke vezane za planiranje dodele vezova zasnivaju se na tehničkoj specifikaciji broda, kranova i vezova, prioritetu brodova, planiranom zadržavanju brovova u terminalu, itd. Odluke su usmerene ka formiranju plana vezivanja koji će rasporediti brodove na optimalan način i pri tome definisati efikasan prostorni i vremenski raspored brodova u terminalu. Pri donošenju ovih odluka, kao smernice se uzimaju ukupno vreme koje brod provede u luci od momenta dolaska do momenta kada brod napusti terminal i svi troškovi koji bi mogli da optimizuju poslovanje luke.

Pregled literature koja razmatra probleme iz oblasti operacionih istraživanja u kontejnerskim terminalima može se naći u [38, 39, 100, 157].

3.1 Problem dodele vezova (BAP)

U radu [63] pod problemom dodele vezova smatra se problem dodele slobodnog prostora na vezu brodovima koji su u lučkom terminalu. Prema definiciji u radu [125], problem dodele vezova se sastoji od određivanja momenta vezivanja i pozicije za svaki brod u lučkom terminalu. Prema potpunijoj definiciji problem dodele vezova obuhvata formiranje plana vezivanja na raspoložive vezove za brodove koji se nalaze u lučkom terminalu, tako da je svaki brod vezan određeni vremenski period

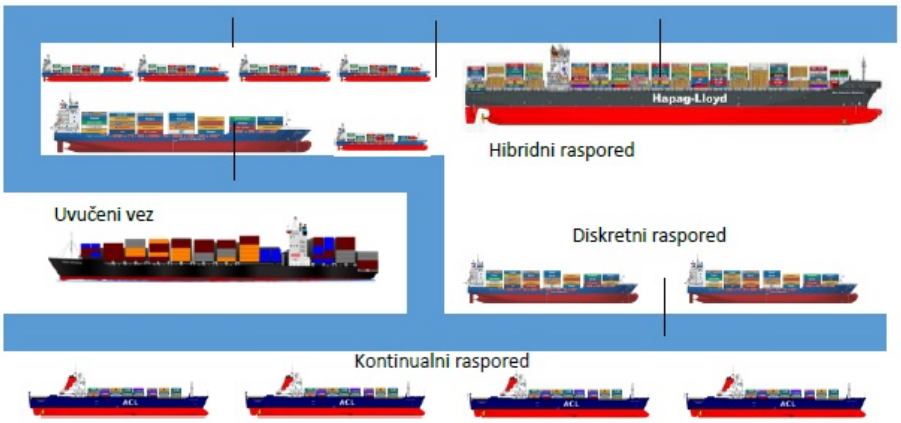
u kojem se obavljaju aktivnosti utovara ili istovara sa ciljem smanjivanja operativnih troškova. Kako je prostor pristaništa limitiran, a dnevno se u terminalu rukuje hiljadama kontejnera, efikasan način dodele vezova postaje presudan za uspešnu administraciju protoka kontejnera kroz terminal.

Kod problema dodele vezova pretpostavlja se da je zadata prostorna struktura kontejnerskog terminala zajedno sa skupom brodova koje treba opslužiti u razmatranom vremenskom intervalu (slika 3.2). Brodovi se karakterišu skupom podataka, kao što su: očekivano vreme dolaska, veličina, predviđeno vreme obrade, preferirani vez u luci, i drugim parametrima, u zavisnosti od varijante BAP-a koja se razmatra. Cilj BAP-a je da se svaki brod priveže na neki od raspoloživih vezova u određenom vremenskom trenutku, tako da se vrednost zadate funkcije cilja optimizuje. Funkcija cilja može biti definisana na razne načine: minimizacija ukupnih troškova vezivanja, minimizacija vremena čekanja i obrade brodova, minimizacija preuranjenosti i kašnjenja (engl. *earliness and tardiness*), minimizacija utroška goriva, maksimizacija profita, maksimizacija iskorišćenosti kranova, itd. Dokazano je da je BAP NP-težak problem optimizacije [113].

Pregled klasifikacije varijanti BAP-a dat je u tabeli 3.1, dok se detaljnija klasifikaciona šema BAP-a može naći u [7]. Tabela 3.1 opisuje atribute i njihove skraćenice koje se koriste za BAP klasifikacije u literaturi. Klasifikacija se definiše na osnovu četiri parametra: prostornog i vremenskog atributa, dužina obrade i merila performansi.

Na osnovu prostornog atributa, varijante BAP-a se dele na *diskretni*, *kontinualni*, *hibridni* ili *zavistan od gaza broda*. U diskretnom slučaju, pretpostavlja se da je pristanište izdeljeno na sekcije koje se nazivaju vezovi. Svaki vez može da opsluži samo jedan brod u datom vremenskom trenutku. Osim toga, i vremenska osa može biti izdeljena na diskretne jedinice, što omogućava upotrebu celobrojne aritmetike za računanje vrednosti funkcije cilja. Kod kontinualne varijante BAP-a, brod koji se obrađuje može da se smesti na bilo koju poziciju uz očuvanje zahteva da ne dolazi do preklapanja sa već vezanim brodovima. Različite kombinacije diskretnog i kontinualnog BAP-a formiraju hibridnu varijantu [7, 8]. Kod BAP-a koji zavisi od gaza broda, brodovi čiji gaz prevazilazi minimalnu propisanu dubinu vode u vezu ne mogu se proizvoljno vezivati na svaki vez, već se pored drugih restikcija u odabiru veza u obzir uzima i ovaj parametar. Navedene varijante BAP-a, kao i jedan specijalan slučaj BAP-a sa uvučenim vezom, ilustrovani su na slici 3.2.

Najzastupljenije varijante BAP-a sa stanovišta vremenskog atributa su *statički*



Slika 3.2: Varijante izgleda pristaništa

i *dinamički* BAP. U statičkoj varijanti BAP-a vremena dolaska brodova ili nisu unapred fiksirana ili ograničenja koja definišu vreme veza nisu stroga, pa i vreme veza može da varira u širem vremenskom intervalu. U prvom slučaju, pretpostavka je da brodovi čekaju u luci i da im se vez može dodeliti odmah nakon njegovog oslobađanja, dok se u drugom slučaju brodovi mogu ubrzavati ili usporavati uz određene troškove. Ako su vremena dolaska broda fiksirana i brod se ne može obraditi pre očekivanog vremena dolaska, takva varijanta BAP-a se klasifikuje kao dinamička.

Na osnovu dužine obrade, BAP se deli na četiri podkategorije: sa fiksnim vremenom obrade, vremenom obrade koje zavisi od pozicije veza, vremenom obrade koje zavisi od dodele kranova ili od operativnog rasporeda kranova. Poslednji atribut, merilo performansi, u suštini predstavlja opis funkcije cilja koja se optimizuje u posmatranom BAP-u. Vrednost funkcije cilja može da zavisi od vremena čekanja broda, dužine obrade broda, vremena završetka obrade broda, ubrzavanja brodova da stignu u terminal pre očekivanog vremena dolaska, kašnjenja u odnosu na predviđeni rok završetka obrade, dodele veza koji nije preferiran i nekih drugih činilaca.

Različitim varijantama BAP-a se mogu pridružiti neki srodni optimizacioni problemi. Tako se, na primer, kontinualna verzija BAP-a može posmatrati kao dvodimenzionalni problem pakovanja (engl. *Bin Packing Problem*) sa jednodimenzionalnim ograničenjem resursa definisanog kranovima i sa promenljivom dužinom obrade koja je određena brojem kranova. Dvodimenzionalni problem sečenja (engl. *Cutting-stock problem*) je takođe sličan kontinualnoj verziji BAP-a u kojoj je broj dodeljenih kranova fiksan i gde se uzima da je jedna dimenzija vreme a druga veličina

Tabela 3.1: Oznake za različite varijante BAP-a

Prostorni atribut		**Vremenski atribut**		**Dužina obrade**	
Skraćenica	*Opis*	*Skraćenica*	*Opis*	*Skraćenica*	*Opis*
disc	diskretni	*stat*	statički	*fix*	fiksna vremena
cont	kontinualni	*dyn*	dinamički	*pos*	zavisno od položaja
hybr	hibridni	*cycl*	ciklički	*QCAP*	QC dodela
draft	gaz broda	*stoch*	stohastički	*QCSP*	QC raspoređivanje
		due	rokovi	*stoch*	stohastički

brodova [113]. Dinamički diskretni BAP se može modelovati kao problem raspoređivanja na heterogene procesorske sisteme (engl. *Unrelated Machine Scheduling Problem*) [136]. Na osnovu ovih međusobnih povezanosti optimizacionih problema i navedenih varijanti BAP-a, može se zaključiti da su one takođe NP-teški problemi optimizacije [51]. Problem raspoređivanja na jednoprocesorske sisteme (engl. *Single Machine Scheduling Problem*) sa minimizacijom ukupnih troškova preuranjenosti [49] je specijalni slučaj BAP-a. Kako se u ovom problemu određuju samo početna vremena obavljanja poslova može se reći da je ovo jednodimenzionalna verzija BAP-a. Hibridna varijanta BAP-a može se posmatrati kao problem raspoređivanja na višeprocesorske sisteme (engl. *Multiprocessor Task Scheduling Problem*), jer u toku utovara i istovara, brodovi mogu istovremeno da zauzmu više susednih vezova. Proces utovara i istovara se može smatrati za zadatak dok su vezovi ekvivalent za procesore.

3.2 Metaheuristički pristup za BAP

U dosadašnjoj literaturi je predložen mali broj egzaktnih algoritama koji rešavaju BAP. Uglavnom se zasnivaju na korišćenju mešovitih celobrojnih formulacija problema (engl. *Mixed-integer programming*-MIP) u okviru komercijalnog rešavača CPLEX [75, 79, 170]. Kombinatrone formulacije BAP-a predložene su u [92, 93, 94, 95] i implementirane su u metodi grananja i ograničavanja i tehnikama pogleda unapred (engl. *Lookahead techniques*). Egzaktni algoritmi grananja i vrednovanja (engl. *Branch-and-price algorithms*) za rešavanje BAP-a implementirani su u [147, 174]. Bendersova kombinatorna odsecanja (engl. *Combinatorial benders' cuts*) predložena su kao metode za rešavanje BAP-a u [19]. Generalno, egzaktni algoritmi mogu da daju optimalna ili dopustiva rešenja na instancama koja imaju

5-40 brodova smeštenih na 3-30 vezova. Generisane test instance većih dimenzija su najčešće previše komplikovane za egzaktne algoritme. Čak i u slučaju instanci manjih dimenzija, može se desiti da egzaktne metode ne daju optimalna rešenja, posebno u slučajevima kada više brodova pretenduje na isti omiljeni vez i/ili imaju isto očekivano vreme dolaska. U navedenim primerima, egzaktni algoritmi mogu zahtevati previše procesorskog vremena za rešavanje problema ili vrlo često ne mogu da pronađu ni početno dopustivo rešenje.

Uzimajući u obzir ograničenja egzaktnih metoda u rešavanju BAP-a, metaheurističke metode su prirodan pristup rešavanju instanci BAP-a. U praksi je važno razviti efikasan sistem podrške odlučivanju koji će pomoći menadžeru terminala da balansira između zahteva za brzom obradom brodova i ekonomski isplativim korišćenjem dodeljenih vezova. Imajući u vidu da su troškovi vezani za korišćenje lučkih resursa i obradu kontejnerskih brodova vrlo visoki, poželjno je da se koriste što je moguće efikasnije.

U sledećim odeljcima dat je pregled literature u kojoj su implementirani metaheuristički pristupi u rešavanju BAP-a. I pored toga što je u procesu sakupljanja pregledne literature korišćeno nekoliko baza za pretraživanje (*ScienceDirect*, *SpringerLink*, *IEEE Explorer*, *Web of Science*, *GoogleScholar*, itd.), moguće je da ovaj pregled nije potpun.

Tabu pretraživanje

TS heuristika je implementirana u [25] za rešavanje dinamičkog diskretnog BAP-a. Autori su koristili model u kojem se minimizuje funkcija cilja definisana nad ukupnim vremenom obrade brodova (za svaki brod se računa razlika između momenta dolaska broda u luku i vremena kad je završena obrada posmatranog broda). Pri generisanju početnog rešenja autori koriste pohlepnu slučajnu proceduru (engl. *Random Greedy Procedure*) i proceduru obrade po redosledu dolaska (engl. *First Come-First Serve*). Inicijalno rešenje se modifikuje premeštanjem brodova sa trenutno dodeljenog veza na novoodabrani vez. Ova premeštanja definišu okolinu tekućeg rešenja. Kada se brod pomeri sa trenutnog veza njegovo ponovno vraćanje na isti vez se zabranjuje u sledećoj iteraciji algoritma postavljanjem tabu statusa na unapred definisane atribute. Istraživanje iz rada [25] je prošireno u radu [104] u kojem autori koriste skup visoko kvalitetnih (elitnih) rešenja sa idejom da se poveća efikasnost TS-a. Korišćena je i dodatna okolina zasnovana na zameni brodova (engl. *swapping*) u kojoj brodovi dodeljeni na isti vez ili na različite vezove mogu da

razmene svoje pozicije. TS heuristika generiše nova rešenja algoritmom ponovnog povezivanja puteva (engl. *Path Relinking Algorithm*) koji se iterativno ponavlja u cilju približavanja početnog rešenja skupu elitnih rešenja. Skup elitnih rešenja se u ovom postupku ažurira ukoliko se pronađe novo najbolje rešenje.

Dodela kranskih profila (engl. *QC profiles*) karakteriše taktičku podvarijantu BAP-a i predstavlja broj dostupnih kranova u svakom vremenskom segmentu obrade broda. Autori u [54] rešavaju diskretni taktički BAP sa ciljem minimizacije troškova ponovnog rukovanja i premeštanja uzrokovanog tokovima pretovara između brodova, kao i maksimizacije ukupne vrednosti odabranih kranskih profila. Problem je rešavan u dve faze: u prvoj fazi se za svaki brod identifikuju profili kranova za brod, dok se u drugoj fazi brodovima dodeljuju vezovi zasnovani na datom kranskom rasporedu. Autori u [54] su prilagodili TS heuirstiku ranije predloženu u [25] tako što je implementirana nova procedura za minimizaciju troškova ponovnog rukovanja u skladištu koji nastaju usled protoka kontejnera između brodova. U radu [109] predstavljen je TS za rešavanje verijante BAP-a koja se odnosni na veliki kontejnerski pretovarni čvor sa više terminala, sa zadatkom da se minimizira ukupan unutarterminalski i međuterminalski trošak rukovanja kontejnerima. Autori u [109] su ponudili hijerarhijsko rešenje terminalskog i pristanišnog problema dodele, a TS heuristka je integrisana u heurističku proceduru koja određuje protok kontejnera u skladišnom prostoru. Skladišni prostor se posmatra kao dvodimenzionalna mreža ograničenog kapaciteta, a TS pronalazi dobar redosled učitavanja na mrežu. Problem upravljanja smetnjama u toku dodele vezova je takođe uspešno rešen upotrebom TS metode u [183]. Kako nepredviđeni problemi mogu uticati na planiranu dodelu vezova inicijalni plan može zahtevati izmene jer je postao nedopustiv. Autori su u [183] kombinovanjem TS-a i metode ponovne dodele ovaj problem rešili lokalno na mestima pojave smetnji, najpre u malim vremenskim i prostornim intervalima, a zatim su intervali proširivani sve dok ne obuhvate ceo razmatrani horizont.

Pohlepna stohastičko-adaptivna procedura pretrage

Dve verzije GRASP-a koje minimizuju težinsko ukupno vreme protoka za dinamički kontinualni BAP su razvijene u [107]. U prvoj varijanti algoritma početno rešenje se formira po pravilima pakovanja definisanog redosledom dolaska (engl. *first-come-first-pack rule*), dok u drugoj verziji ne postoje dodatna ograničenja pri konstrukciji inicijalnog rešenja. U prvoj verziji GRASP-a iz [107], autori su implementirali dve procedure lokalnog pretraživanja: prva je zasnovana na razmeni

susednih brodova u listi, a druga se bazira na A* proceduri pretrage stabla. Druga verzija GRASP-a predložena u [107] u osnovi ima istu ideju ali bilo koja dva broda mogu razmeniti pozicije. Salido i sar. su u [152] razvili integrisani pristup zasnovan na GRASP-u koji nezavisno razmatra problem slaganja kontejnera i BAP i uključuje funkciju cilja u kojoj se minimizuje vreme čekanja brodova. U [153], Salido i sar. su predložili sistem podrške odlučivanju za kontejnerski terminal. Autori su dodatno posmatrali problem dodele kranova (QCAP) integrisan sa BAP-om i takođe predložili GRASP kao heuristiku za rešavanje ovog problema. BAP i QCAP koji minimizuju ukupno vreme čekanja brodova razmatrani su i u [150] i dizajniran je GRASP kao metoda rešavanja. Predloženi GRASP iz [150] formira početno dopustivo rešenje odabirom brodova iz ograničene liste kandidata dobijene na osnovu vrednosti pohlepne funkcije i vrednosti slučajnog parametra. Procedura lokalnog pretraživanja je kontrolisana pravilom otpuštanja (engl. *dispatching rule*) zasnovanog na redosledu brodova definisanom na osnovu vremena vezivanja broda. Okolina trenutnog rasporeda se dobija razmenom pozicija dva slučajno odabrana broda u pravilu otpuštanja.

Optimizacija škripećeg točka

Zhen i sar. su u [185] posmatrali istovremenu optimizacija BAP-a i QCAP-a za pretovarna čvorišta i predložili su SWO za rešavanje realnih primera velikih dimenzija. Kako SWO formira nova rešenja razmenom pozicija samo susednih brodova u trenutnom rešenju i premeštanjem brodova sa velikom cenom na početak liste, u radu [185] je predložena kombinacija SWO i pretrage okoline kritičnim pretresanjem (engl. *critical-shaking neighborhood search*). Ova hibridna metoda bira unapred definisani broj brodova sa najvećom vrednošću troškova i na slučajan način menja njihov prioritet kako bi se omogućio izlazak iz lokalnog optimuma. Nadovezujući se na prethodna istraživanja o integrisanom BAP-u iz [120], Meisel i Bierwith u [121] razmatraju QCAP i raspoređivanje kranova. SWO je u [121] upotrebljen u fazi dodele vezova, kao i u fazi dodele kranova jer je dao bolje rezultate od TS-a.

SWO je takođe korišćen za rešavanje dinamičkog hibridnog BAP-a u [171]. Početno rešenje za dodelu vezova dobijeno je na osnovu redosleda dolaska (engl. *first come-first served*) brodova koji se obrađuju. Na kraju svake iteracije, na osnovu trenutnog vremena obrade brodova, računa se njihov novi prioritet. Brod sa datim prioritetom se smešta na vez koji minimizuje ukupno vreme obrade i vreme čekanja brodova, ali tek kad su svim brodovima sa većim prioritetom već dodeljeni vezovi.

Algoritam premešta brodove sa dužim vremenom obrade i dužim vremenom čekanja na početak liste prioriteta, što vodi ka poboljšanju njihovog opsluživanja. Opsežni eksperimenti nad realnim podacima dobijenim iz luka rasutog tereta su pokazali da SWO metoda iz [171] pronalazi rešenja bliska optimalnim za veće instance problema.

Metoda promenljivih okolina

Diskretni BAP sa minimalnim troškovima je razmatran u [73] i rešavan varijantom metode promenljivih okolina. Preciznije, predložena je metoda poznata pod imenom metoda spusta kroz promenljive okoline (engl. *Variable Neighborhood Descent*-VND) koja koristi tri okoline. Prva okolina zasnovana je na *Or opt* algoritmu. Algoritam odabira jedan, dva ili tri broda i umeće ih između bilo koja dva broda koja se obrađuju na istom vezu. Druga okolina razmenjuje rasporede dva broda koja se opslužuju na različitim vezovima, dok treća okolina uklanja odabrani brod sa veza i pokušava da ga umetne na neku drugu raspoloživu poziciju. U fazi razmrdavanja su korišćene dve ugneždene okoline. U prvoj se za dva slučajno odabrana broda razmenjuju njihovi vezovi i vremena dolaska. U drugoj okolini se slučajno odabrani brod uklanja iz skupa obrađenih brodova na datom vezu i dodaje se skupu brodova na slučajno odabranom vezu u najpogodnijem redosledu. Takođe, u poređenju sa genetskim i memetskim algoritmom, VND iz [73] pokazuje bolje performanse na istom skupu instanci i na skoro svim instancama dostiže poznata optimalna rešenja.

Simulirano kaljenje

Kontinualni BAP u kome se minimizuju troškovi vezivanja na neoptimalne pozicije i troškovi kašnjenja proučavan je u [90]. Upotrebljeni su takozvani x i y *klasteri*, koji predstavljaju skupove brodova predstavljene pravougaonicima čije se vertikalne ili horizontalne strane dodiruju. Klasteri se nazivaju *stabilnim* ako se ne mogu pomerati duž x ili y-ose. Stabilnost je korišćena da bi se popravio kvalitet generisanog rešenja. Oliveira i sar. su u [34] rešavali dinamički diskretni BAP kombinacijom metode klasterskog pretraživanja (engl. *clustering search*) i simuliranog kaljenja, pri čemu funkcija cilja razmatranog problema minimizuje težinsku sumu vremena opsluživanja. Na svakom nivou temperature, trenutno rešenje se menja procedurom klaster pretrage. Tri različita preuređenja redosleda brodova koriste se za definisanje okoline rešenja (*reorder ships*, *reallocate ships*, *change ships*). U cilju obezbeđivanja raznovrsnosti uzastopnih generisanih rešenja, brodovi se biraju nasumično i na njih

se primenjuje jedan od tri tipa pomenutih preuređenja. Kontinualni *robustni BAP*, koji uzima u obzir neizvesnost u kašnjenju dolaska broda i vremenu obrade, razmatran je u radu [177]. Autori opisuju korisna svojstva optimalnog rešenja i koriste ih za smanjivanje prostora pretrage. Prostor pretrage je podeljen u podskupove koji su predstavljeni nizom brodova. Nad svakim od tih podskupova primenjena je tehnika grananja i ograničavanja da bi se našlo optimalno rešenje podskupa, a heuristika SA je korišćena u cilju efikasnog pretraživanja kompletnog prostora. Zhen i sar. su u [186] proučavali varijantu BAP-a uzimajući u obzir odstupanja vremena dolaska brodova i vremena rada kao faktora nesigurnosti sa zadatkom minimizacije troškova taktičkog BAP-a i očekivane vrednosti troškova oporavka. Integrisani dinamički kontinuirani BAP sa ograničenjima za dubinu vode i QCSP rešavan je u [41]. Da bi se odredio redosled vezivanja brodova, autori definišu listu prioriteta brodova, pri čemu veliki brodovi i oni sa većim očekivanim vremenom završetka imaju veći prioritet. SA se koristi za istraživanje okoline liste prioriteta. Okolina je definisana razmenom dva nasumično odabrana susedna broda u listi prioriteta. Kada se konstruiše dopustivo rešenje primjenjuju se prostorna i vremenska pomicanja brodova u cilju dobijanja kvalitetnijih rešenja.

Optimizacija rojem čestica

Ting i sar. su u [167] prvi put primenili PSO heuristiku za rešavanje BAP-a. Autori su istraživali dinamički diskretni BAP i posmatrali ga kao problem rutiranja vozila sa vremenskim prozorima. Preciznije, vezovi odgovaraju vozilima, brodovi su posmatrani kao korisnici, dok sekvenca obrade na određenom vezu odgovara ruti vozila. PSO se koristi za pretraživanje prostora rešenja, a nakon svake iteracije PSO-a primjenjuje se procedura lokalne pretrage, ali samo na najbolju pronađenu česticu zbog velike vremenske složenosti LS procedure.

Optimizacija kolonijom mrava

ACO algoritam sa više kolonija mrava je korišćen u [22] za rešavanje višekriterijumskog BAP-a koji minimizuje ukupno vreme obrade brodova i ukupno kašnjenje u odlasku brodova. Grupe mrava se koriste za traženje jednog rešenja, a svaki mrav je odgovoran za raspored brodova na jednom vezu. ACO predložen u [22] koristi elitističku strategiju u cilju intenziviranja pretrage u okolinama pronađenih najbo-

ljih rešenja. Takođe, koristi se više heterogenih kolonija mrava koje se razlikuju u feromonskoj matrici i u drugim ACO parametrima.

Evolutivni algoritmi

U literaturi su evolutivni algoritmi najčešći pristup za rešavanje raznih varijanti BAP-a, a među njima su najzastupljeniji genetski algoritmi. Višekriterijumski EA (engl. *Multi-Objective EA*-MOEA) je razvijen u [21] za rešavanje varijante višekriterijumskog BAP-a sa ciljem istovremene minimizacije tri konfliktna cilja, troškova ukupnog vremena obrade, broja konflikata i vremena čekanja. Vreme čekanja se smanjuje za svaki brod u skladu sa strategijom *raniji dolazak - ranija obrada*. MOEA koristi jedinke fiksirane dužine i na njima primenjuje operatore ukrštanja i mutacije. Šema dekodiranja se zasniva na redosledu dodele (a ne na češće korišćenom redosledu vezova) i podrazumeva da brodovi mogu da se privežu samo u isto vreme ili posle brodova koji im prethode. Autori su u [21] koristili efikasan redosled dodele u optimizovanju korišćenja prostora za vez (brod se smešta na krajnju levu dopustivu poziciju za vez sa najranijim vremenskim trenutkom raspoloživim za vez).

Cheong i sar. su u [20] implementirali višekriterijumski EA koji uključuje koncept Pareto optimalnosti, primenjen na varijantu višekriterijumskog BAP-a. Razmatrani su minimizacija vremena ukupne zauzetosti luke, ukupnog vremena čekanja i odstupanja od unapred definisanog prioritetnog rasporeda brodova. Da bi ovaj problem bio rešen, razvijen je EA koji sadrži lokalnu pretragu, hibridnu šemu dekodiranja rešenja i proceduru optimalne dodele vezova. EA iz [20] koristi hromozome fiksirane dužine jednake broju vezova, a svakom vezu se dodeljuje lista brodova koje treba da opsluži. Lokalna pretraga uključuje sortiranje brodova na nasumično odabranom vezu na osnovu njihovih prioriteta. Skup formiranih rešenja se dekodira i rangira na osnovu Pareto šeme, a loše rangirana rešenja se uklanjaju iz populacije. Cheong i sar. u [20] koriste dve različite šeme dekodiranja (redosled vezivanja i redosled dodele), i ispituju njihov uticaj na kvalitet rešenja. Operator ukrštanja nasumično bira vezove u oba roditelja i razmenjuje njihove liste brodova. U procesu razmene lista brodova pojedini brodovi mogu nestati, i stoga se moraju ubaciti na nasumično odabran vez. Verovatnoća da je brod ubačen u određeni vez je obrnuto proporcionalna vremenu opsluživanja broda na tom vezu.

Dinamički diskretni BAP sa stohastički vremenom obrade brodova i poznatom raspodelom verovatnoće je uveden u [84]. Funkcija cilja obuhvata minimizaciju rizika i ukupnog vremena obrade. Autori su u [84] predložili način računanja mere rizika

za dati raspored na vezu, koji je zasnovan na raspodeli verovatnoće. Za rešavanje razmatrane varijante BAP-a, implementiran je višepopulacioni EA sa celobrojnom reprezentacijom hromozoma i četiri operatora mutacije: ubacivanje, zamena, mešanje i inverzija (engl. *insert*, *swap*, *scramble*, *inversion*). EA je kombinovan sa Post-Pareto simulacijom zasnovanom na jednostavnom postupku Monte Carlo koja se koristi za odabir jednog nedominantnog rešenja iz skupa svih efikasnih rešenja (engl. *Pareto front*). Odabrano nedominantno rešenje predstavlja rešenje diskretnog dinamičkog BAP-a.

Golias i sar. u [56] koriste GA i koncept vremenskih prozora za rešavanje dinamičkog diskretnog BAP-a. Cilj je istovremeno smanjiti troškove kasnog odlaska brodova i maksimizirati korist od odlaska brodova pre i unutar traženog vremenskog prozora. Pretpostavka je da na vreme utiče položaj vezivanja, što odgovara realnoj situaciji u luci. U radu [165], za rešavanje dinamičkog diskretnog BAP-a predložen je GA koji je nezavisan od funkcije cilja. Autori rada [56] su predložili GA u cilju minimizacije ukupnog težinskog vremena obrade u kojem brodovi mogu imati različite prioritete obrade. GA iz [165] ne koristi ukrštanje zbog velikog broja nedopustivih jedninki koje ovaj operator generiše. Pre formiranja sledeće generacije, primenjena je interna faza optimizacije na slučajno izabrani broj nedopustivih jedinki. Algoritam grananja i ograničavanja ponovo raspoređuje brodove dodeljene svakom vezu uz minimiziranje ukupnog težinskog vremena obrade za svaki brod. Međutim, ekperimentalni rezultati su pokazali da je faza optimizacije vremenski veoma zahtevna ako je broj vezova veći od 5.

U radu [81] predložen je GA za rešavanja BAP-a sa dve funkcije cilja koje minimizuju težinsko kašnjenje u odlasku brodova i ukupno vreme obrade. Ova dva cilja su konfliktna, a GA se koristi za identifikaciju neinferiornih rešenja. GA se pokazao kao efikasniji pristup u odnosu na metod subgradijentne optimizacije, posebno u slučaju preopterećenog terminala sa čestim dolaskom brodova i dugim vremenom obrade brodova.

Višekorisnički kontejnerski terminal s uvučenim vezovima za brzo rukovanje megakontejnerskim brodovima i minimizacijom ukupnog vremena obrade proučavali su Imai i sar. u radu [80]. Pretpostavka je da se nekoliko malih brodova može istovremeno obrađivati na vezu. Autori su pokazali da uvučeni terminal opslužuje megakontejner brže od konvencionalnog terminala. Sa druge strane, ukupno vreme obrade za sve brodove je bilo duže u poređenju sa neophodnim vremenom obrade na standardnom terminalu. U predloženom GA, svaki hromozom je predstavljen

kao niz karaktera sa kratkom reprezentacijom čija dužina jedinke je jednaka broju brodova uvećanom za broj vezova minus 1. Reprezentacija hromozoma jednostavno definiše odnos između veza, broda i redosleda obrade. Umesto klasične funkcije prilagođenosti, koja je obično definisana kao recipročna vrednost funkcije cilja, autori u [80] koriste sigmoidalnu funkciju i eksperimentalno potvrđuju da GA sa tom funkcijom prilagođenosti daje bolje rezultate.

BAP koji ima za cilj minimiziranje ukupnog vremena zadržavanja broda u terminalu i poboljšanja efikasnosti rada terminala razmatran je u [65]. Varijanta GA predložena u [65] koristi Metropolis algoritam umesto operatora mutacije. Metropolis algoritam se primenjuje na svaku jedinku u svakoj GA generaciji, sa ciljem izbegavanja lokalnog optimuma i povećanja prostora za pretraživanje. Korišćenjem Metropolis algoritma, izbegnute su poteškoće oko izbora verovatnoće mutacije, što rezultira boljim ponašanjem GA pri pretraživaju prostora rešenja. Implementirano je jednopoziciono ukrštanje, koje se primenjuje na slučajno odabranu jedinku i na trenutno najbolju jedinku. Ukrštanje može proizvesti jedinke potomke koje odgovaraju nedopustivim rešenjima. U tom slučaju, vrši se korekcija novog potomstva.

U radu [2] primenjena je kombinacija GA i simulacione tehike na diskretni BAP koji razmatra slučaj luke u Sevilji, sa ciljem minimizacije ukupnog vremena obrade brodova. Predloženi hibridni metod koristi strategiju dodele *raniji dolazak-ranija obrada*. 20% jedinki u trenutnoj populaciji podleže mutaciji dok se na ostatak od 80% utiče operatorom ukrštanja. Eksperimentalni rezultati na realnim test primerima iz luke u Sevilji su pokazali da predloženi hibridni metod poboljšava performanse ove luke i smanjuje vreme obrade za 14%, dok su maksimalna vremena obrade smanjena za 21%. Chang i sar. su u [18] proučavali kombinaciju BAP-a i QCAP-a koja za cilj ima minimizaciju odstupanja od lokacije veza, ukupnih penala i potrošnje goriva za kranove. Kombinacija BAP i QCAP je rešena hibridnim paralelnim GA. Početna populacija je generisana heurističkim algoritmom, dok se GA koristi da pronađe suboptimalno rešenje za BAP i QCAP. Autori u [111] uvode dinamičku dodelu kranova u BAP-u i predlažu višekriterijumski hibridni GA pristup sa metodom kodiranja zasnovanom na prioritetima. Funkcije cilja smanjuju vreme čekanja i kašnjenja brodova, vreme rukovanja kontejnerima i kretanje kranova. GA se izvodi u četiri faze: kreiranje niza brodova prema njihovim prioritetima, dodeljivanje brodova na vezove, dodeljivanje kranova brodovima i formiranje rasporeda vezivanja i rasporeda kranova.

Hibridni višestepeni GA iz [112] koristi se za rešavanje operacionog nivoa BAP-a,

u kombinaciji sa metodom kodiranja zasnovanom na prioritetu brodova. Predloženi GA razmatra protok kontejnera između brodova i pokušava da minimizuje ukupno vreme njihovog rukovanja i vremena čekanja brodova, kao i da minimizuje kašnjenje i vreme čekanja pretovara. Algoritam počinje procedurom koja odlučuje da li treba izvršiti direktnu uslugu pretovara između parova brodova, a potom sledi GA od četiri faze predstavljen u [111].

BAP je formulisan kao višekriterijumski mešoviti celobrojni optimizacioni problem u [58] i za njegovo rešavanje je predložena varijanta GA bez operatora ukrštanja. Dužina jedinke predloženog GA je jednaka proizvodu broja vezova i broja brodova. Da bi se sačuvala raznolikost rešenja u Pareto frontu, koristi se višepopulacijski višeselekcioni GA pristup. U svakoj iteraciji algoritma, mutirana populacija se kopira u dva skupa. Prvi duplikat se koristi za odabir roditelja sledeće generacije zasnovan na optimalnom skupu Pareto Fronta. Drugi duplikat predstavlja elitistički skup koji se koristi za dobijanje poboljšanih vrednosti funkcija cilja unutar Pareto fronta. Sledeća generacija mutiranih jedinki dobija se kombinovanjem ova dva skupa.

U slučaju dinamičkog diskretnog BAP-a Golias i Haralambides u [60] istovremeno minimiziraju kašnjenje brodova i njihovo vreme čekanja i maksimiziraju profit ostvaren ranim odlaskom brodova. U cilju rešavanja ovog problema, autori su koristili GA koji je prethodno predstavljen u [61]. Primenjena GA metoda iz [61] koristi celobrojnu reprezentaciju jedinki sa dva nivoa. Prvi nivo sadrži vremena dolaska brodova, dok drugi kodira redosled opsluživanja brodova na svakom vezu. U svakoj iteraciji GA, primenjuju se sa istom verovatnoćom sva četiri operatora mutacije (insert, swap, scramble, invert), ali u kasnijim generacijama, težina se pomera sa inverzije i mešanja na umetanje i razmenu. Ovom strategijom dozvoljeni su veliki skokovi u početnim iteracijama, dok se u kasnijim fazama pretraživanja intezivira pretraga manjeg, obećavajućeg regiona pretrage.

U [151] je proučavan hijerarhijski pristup optimizacije za dinamički diskretni BAP koji minimizira nezadovoljstvo korisnika i maksimizira protok robe kroz luku. Na osnovu funkcija cilja formirana su dva nivoa hijerarhije. Na prvom nivou hijerarhije je definisan MILP sa ciljem minimizacije ukupnog vremena obrade brodova svakog korisnika. Na drugom nivou se optimizuje protok robe kroz luku, minimizacijom ukupnog vremena obrade svih brodova. Da bi rešili problem optimizacije sa konfliktnim funkcijama cilja, autori su u [151] primenili GA zasnovan na *k-tom najboljem algoritmu* za višekriterijumsku optimizaciju na prvom nivou hijerarhije.

Umesto da se u svakoj iteraciji algoritma sekvencijalno rešavaju oba nivoa hijerarhije, GA predložen u [151] rešava gornji nivo samo jednom. Dobijena GA rešenja se sortiraju prema njihovom kvalitetu i šalju na niži nivo jedan po jedan, bez ponovnog rešavanja problema višeg nivoa. Sličan pristup, zasnovan na programskom modelu u dva nivoa, je primenjen u [156] za rešavanje integrisanog problema dodele vezova i dodele kranova. GA rešava BAP kao gornji nivo, dok je QCAP rešen do optimalnosti metodom grananja i ograničavanja na nižem nivou.

U studijama [187] i [188], razmatran je dinamički diskretni BAP koji minimizira ukupno vreme čekanja brodova pri čemu se vremena dolaska i vremena obrade brodova smatraju stohastičkim parametrima koji prate normalnu distribuciju. Na osnovu karakteristika optimalnog rešenja, razvijen je GA sa redukovanim prostorom pretraživanja. Jedinka je predstavljena sa dva niza, čije su dužine određene brojem razmatranih brodova. Prvi niz sadrži informacije o dodeljenom vezu, dok drugi određuje redosled brodova na svakom vezu. Brodovi se sortiraju na osnovu vremena dolaska i primenom specijalne metode (engl. *Order Limit Number*). Ovom metodom kodiranja smanjuje se prostor pretrage. Golias i sar. u [59] koriste lambda-optimalnu heuristiku za rešavanje dinamičkog diskretnog BAP-a sa minimizaijom vremena obrade i težinskog vremena obrade. Lamda-optimalna heuristika primenjena na dinamičkom diskretnom BAP-u se koristi da garantuje lokalnu optimalnost u predefinisanoj okolini rešenja u radu [59]. Autori u [59] predlažu GA za rešavanje srednjih i velikih instanci kako bi smanjili procesorsko vreme. Vrednost funkcije prilagođenosti je obrnuto proporcionalna izračunatoj vrednosti funkcije cilja hromozoma. Kombinovanjem elitizma, polupohlepne strategije i selekcije ruletom, poboljšane su performanse GA algoritma.

Evolutivni algoritam baziran na ugneždenim petljama razvijen je u [179] s ciljem ispitivanja interakcije između BAP-a i QCAP-a. U algoritmu se koriste dve unutrašnje petlje i jedna spoljna petlja. GA je implementiran u unutrašnjim petljama, koje kao izlaz daju dopustiva BAP rešenja i odgovarajuća rešenja za QCAP, respektivno. Prenos vrednosti promenljivih koje posreduju između BAP-a i QCAP-a se realizuje preko spoljne petlje. Ta petlja vraća dobijeni izlaz kao novi ulaz za prvu unutrašnju petlju.

Robustni BAP i QCSP su istovremeno proučavani u [66]. Za razliku od pristupa u [187], autori su koristili raspodelu gustine verovatnoće da bi opisali stohastičku stranu problema. Autori u [66] koriste GA za pronalaženje robusnih rešenja razmatranih problema. Procedura zasnovana na Monte Carlo simulaciji sa uzorkovanjem

ocenjuje kvalitet svakog hromozoma. Optimizacioni dvokriterijumski model za robustni BAP je definisan u [57], u slučaju neizvesnih vremena dolaska i vremena obrade broda. U problemu razmatranom u [57] minimizuju se dve funkcije cilja: prosečno ukupno vreme za obradu broda i opseg ukupnog vremena obrade. Početna populacija je kreirana strategijama *raniji dolazak-ranija obrada sa ranim početkom* i *raniji dolazak-ranija obrada sa ranim završetkom*. Vrednost funkcije cilja svake jedinke računa se heuristikom minimalne i maksimalne pretrage. Na svakom hromozomu iz trenutnog Pareto fronta primenjuju se četiri operatora mutacije koji proizvode četiri potomka. Jednopozicioni operator ukrštanja se primenjuje na slučajno izabrane hromozome iz trenutnog Pareto fronta. Svi dobijeni potomci koriste se za formiranje novog Pareto fronta.

Robustan dinamički kontinualni BAP i robustan QCSP su proučavani u [149]. Da bi se predstavila robustnost unutar BAP-a autori koriste vremenske periode (engl. *buffer times*) koji su maksimizirani tako da apsorbuju eventualne incidente ili kvarove, dok je ukupan period obrade dolaznih brodova minimiziran.

U radu [103] je proučavan taktički BAP i predložen je GA sa pristrasnim slučajnim ključem (engl. *biased random key*) za rešavanje ove varijante BAP-a. Rešenja su predstavljena dvosegmentnim hromozomima, a dužina svakog segmenta jednaka je broju brodova. U prvom segmentu hromozoma nalazi se redosled vezivanja brodova, dok drugi deo sadrži informacije o profilima kranova koji su dodeljeni brodovima. Slučajni ključevi su integrisani u GA kako bi se rešio problem nedopustivih jedinki. Slučajni ključ je realan broj iz intervala [0,1), a svaki gen u hromozomu sastoji se od jedne vrednosti slučajnog ključa. Korišćenjem pristrasnog slučajnog ključa, GA populacija je podeljena u dva podskupa: elitni i neelitni skup. Iz svakog podskupa se bira po jedna jedinka i na njima se primenjuje operator ukrštanja koji im razmenjuje genetski materijal. Potomstvo ima veću verovatnoću nasleđivanja ključeva od svog elitnog roditelja.

Ganji i sar. su u [50] uspešno rešili kontinualni BAP primenom GA. Autori su koristili celobrojnu reprezentaciju jedinki, sa dužinom hromozoma jednakom dvostrukom broju brodova. Prva polovina hromozoma predstavlja vreme obrade dok druga polovina pokazuje lokacije vezova brodova. Jednopozicioni operator ukrštanja se primenjuje na slučajno odabranim parovma jedinki roditelja, dok broj generisanih potomaka ne postane 90% trenutne populacije. Mutacija se primenjuje na 9% od jedinki trenutne populacije.

Memetski algoritmi

Mauri i sar. [118] su rešavali kontinualni BAP memetiskim algoritmom, koji je zasnovan na kombinaciji genetskog algoritma i simuliranog kaljenja. Operator mutacije u predloženom MA se zasniva na tri procedure: zamene dva slučajno odabrana broda sa istog veza (engl. *Re-order vessel*), slučajno odabrani brod se smešta na slučajno odabranu poziciju na drugi vez (engl. *Re-allocate vessel*), dva slučajno odabrana broda sa slučajno odabranog veza razmenjuju pozicije (engl. *Swap vessels*). Jedna slučajno odabrana procedura mutacije se takođe koristi u SA sa ciljem diversifikacije lokalne pretrage. Eksperimentalni rezultati predstavljeni u [118] pokazuju da MA u kratkom vremenu izvršavanja daje rešenja koja malo odstupaju od najboljih poznatih rešenja iz literature.

U radu [108], MA je uspešno primenjen na varijantu BAP-a sa cikličnom posetom malih kontejnerskih brodova za razvoženje robe (engl. *feeder*) i dodelu skladišnog prostora za protok kontjnera između matičnog broda i malih brodova za razvoženje kontejnera. Ma predložen u [108] nastao je kombinovanjem GA i tabu pretraživanja, sa ciljem minimizacije ukupne razdaljine premeštanja svih tokova kontejnera između pristaništa i skladišta i da se minimizira jaz između najveće i najmanje količine posla koji pri tome treba obaviti. Početna populacija se generiše na slučajan način dok se svaka sledeća generacija jedinki dobija primenom genetskih operatora. TS procedura se koristi za poboljšanje potomaka generisanih GA operatorima. Sledeća generacija se formira na osnovu kvaliteta rešenja, tako što operator selekcije bira jedinke iz trenutne populacije i iz skupa jedinki koje predstavljaju potomstvo.

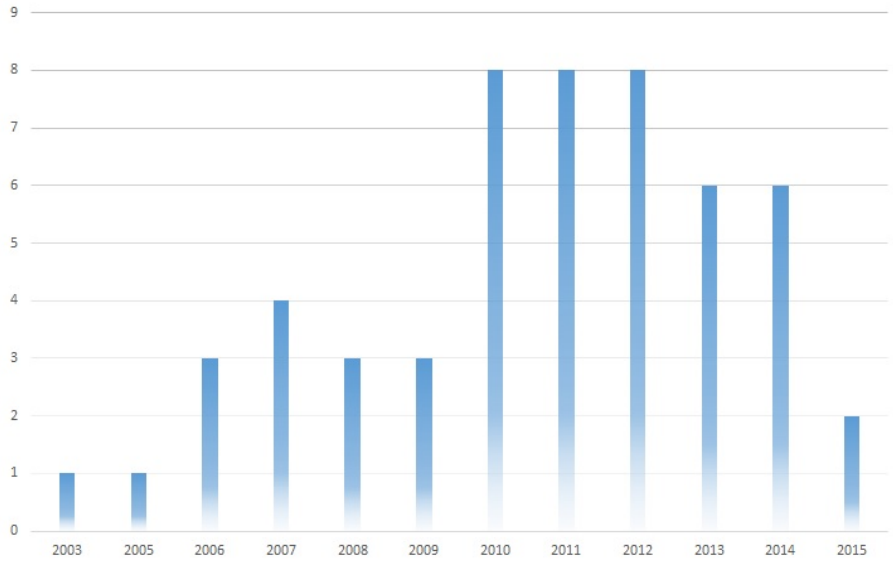
Metaheuristika delimične optimizacije uz posebne uslove intensifikacije

U slučaju dinamičkog diskretnog BAP-a, Lalla-Ruiz i Voß u [105] koriste POPMUSIC sa ciljem minimizacije ukupnog vremena obrade. Problem je modelovan kao *generalizovana disjunktna podela skupa* (engl. *Generalized Set-Partitioning Problem*) i rešen korišćenjem egzaktnog rešavača CPLEX. Pohlepna slučajna metoda je korišćena za dobijanje inicijalnog rešenja koje se zatim deli na više delova. Broj formiranih delova rešenja odgovara broju razmatranih vezova. Podproblemi su formirani na osnovu slučajnog izbora jednog dela rešenja i njegovih najbližih delova. Kada se izvrši POPMUSIC heuristika, dobijeni delovi rešenja se spajaju u jednu celinu čime se formira redukovana instanca problema koju CPLEX dalje rešava do

optimalnosti.

3.3 Pregled relevantne literature za BAP

Imajući u vidu veliki praktični značaj, BAP i njegove varijante prepoznate su kao jedna od najistaknutijih tema vezanih za pomorski transport. Odeljak 3.3 klasifikuje 50 relevantnih radova iz novije literature koji su objavljeni nakon 2003. godine. Sa slike 3.3 može se videti da se BAP konstantno nalazi u fokusu istraživača i da generalno postoji trend rasta godišnjeg broja objavljenih radova na temu BAP-a.



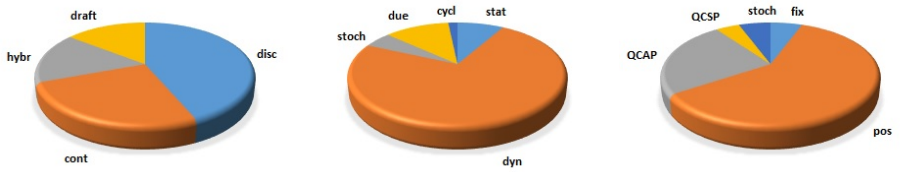
Slika 3.3: Broj godišnje objavljenih radova o BAP-u

Tabela 3.2 sumira pregledanu BAP literaturu, njenu klasifikaciju i neke od najvažnijih BAP parametara koji karakterišu modele iz pregledanih radova. U prvoj koloni tabele 3.2 date su varijante BAP-a iz novije literature koje su označene u Meiselovoj notaciji [7]. Iako ova notacija uključuje i meru performansi kao četvrti parametar, ona je izostavljena iz tabele 3.2, imajući u vidu da su funkcije cilja opisane u poglavlju 3.2. Osim toga, izostavljanjem mere performansi, mogu se bolje uočiti grupe sličnih radova. Druga kolona tabele 3.2 sadrži referencu na odgovarajući rad, a metaheuristički algoritam koji se koristi kao pristup rešavanju BAP-a je naveden u trećoj koloni. Poslednje tri kolone opisuju redom dimenzije BAP-a u smislu broja brodova, broja vezova i dužine vremenskog horizonta. Neki unosi

u tabeli 3.2 koji se odnose na broj brodova označeni su terminom *grupni dolazak* (engl. *inter arrival time*), što znači da grupa brodova stiže u jednoj jedinici vremena i njihov broj se obično definiše eksponencijalnom raspodelom. U slučaju neprekidnog BAP-a, broj vezova nije definisan, te podaci u odgovarajućoj koloni tabele 3.2 (prikazani u metrima) označavaju dužinu veza. Ako neki od BAP parametara nije jasno naznačen u odgovarajućem radu, u tabeli 3.2 taj parametar je označen *.

Iz tabele 3.2 može se videti da u dosadašnjoj literaturi postoji veliki broj različitih varijanti BAP-a. Generalno, problemi dodele vezova su teški za rešavanje. Često i metaheurističke metode prilagođene BAP-u teško pronalaze kvalitetna rešenja u prihvatljivom procesorskom vremenu. Funkcija cilja igra značajnu ulogu u definisanju složenosti BAP-a, ali i ostali parametri klasifikacije navedeni u prvoj koloni. Na primer, teško je upoređivati složenost diskretnog i hibridnog BAP-a. Diskretni BAP može imati neku simetriju, jer brodovi mogu zauzimati samo jedan vez, a samim tim se mogu lako premeštati sa jednog veza na drugi. Takođe, već raspoređene grupe brodova se mogu razmeniti između dva susedna veza, što generalno nije slučaj u hibridnoj verziji BAP-a. To je razlog raznolikosti vrednosti u kolonoma tabele 3.2 koje se odnose na broj brodova, vezova i dužine vremenskog perioda. Kako broj razmatranih brodova raste, a kapaciteti luke ostaju fiksni, gustina raspoređenih brodova takođe raste, i to može da dovede do pojave velikog broja instanci koje nemaju dopustivo rešenje. Zbog toga, pri definisanju domena skupova parametara, mora se napraviti balans između zahteva da se reše stvarne instance problema i da se postignu dobre performanse algoritma.

U literaturi se koriste različiti modeli kako bi se predstavile osobine BAP-a koje imaju praktični značaj. Postojeći modeli se mogu klasifikovati i opisati prostornim, vremenskim i atributom dužine obrade [7]. Raspodela ovih parametara na osnovu učestalosti pojavljivanja u novijim relevantnim radovima predstavljena je na slici 3.4.



Prostorni atribut Vremenski atribut Dužina obrade

Slika 3.4: Frekvencija BAP varijanti u novijoj literaturi

Što se tiče prostornog atributa, većina korištenih modela je diskretna (44%), praćena kontinualnim rasporedom (26%), a hibridni i modeli zavisni od gaza zastupljeni

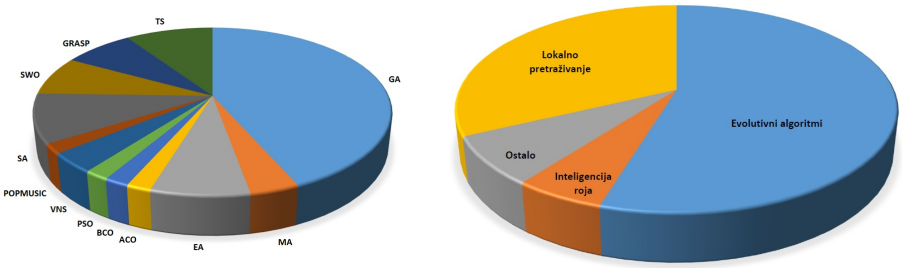
Tabela 3.2: Metaheuristički algoritmi za rešavanje BAP-a

			Dimenzije problema		
Klasifikacija problema	*Rad*	*Algoritam*	*# brodova*	*# vezova*	*Vremenski horizont*
disc | stat | pos,QCSP	[2]	GA	52	2	1 mesec
	[156]	GA	6	3	*
disc | dyn | QCAP	[111]	GA	11	4	24 h
	[112]	GA	11	4	*
	[103]	GA	10 do 50	3 do 8	56 h do 108 h
	[54]	TS	10 do 60	3 do 13	1 do 2 nedelje
disc | dyn | pos	[58]	GA	40 do 80	5 do 10	1 do 2 nedelje
	[60]	GA	grupni dolazak	5	1 nedelja
	[151]	GA	50	5 do 10	1 do 2 nedelje
	[57]	GA	grupni dolazak	4 do 5	1 nedelja
	[167]	PSO	25 do 60	5 do 13	*
	[104]	TS	25 do 60	5 do 13	600 vremenskih jedinica
	[59]	GA	40 do 80	5	1 do 2 nedelje
	[61]	GA	grupni dolazak	5	*
	[73]	VNS	10 do 200	10 do 20	*
	[165]	GA	20-25 nedeljno	5	1 do 2 nedelje
	[56]	GA	9	2	*
	[81]	GA	24	4	1 nedelja
disc | dyn | pos, stoch	[84]	EA	grupni dolazak	5	1 nedelja
disc | dyn,due | pos	[105]	POPMUSIC	40 do 55	5 do 7	600 vremenskih jedinica
	[25]	TS	25 do 35	5 do 10	*
disc | dyn, due | pos	[34]	SA	60	13	*
disc | dyn, due | f ix	[108]	MA	15 do 40	3 do 8	*
disc, draft | stoch | QCAP, stoch	[66]	GA	34 do 88	4 do 5	*
disc, draft | stoch, due | QCAP, stoch	[187]	GA	25 do 100	4	*
disc, draft | dyn | pos	[65]	GA	7	2	*
disc, draft | dyn, due | pos	[188]	GA	25, 50, 75, 100	5 do 8	*
cont | dyn | QCAP	[18]	GA	40	4	*
	[179]	GA	grupni dolazak	800 m do 1600 m	1 nedelja
	[149]	GA	5 do 20	700 m	*
	[152]	GRASP	20	*	*
	[153]	GRASP	5 do 20	*	*
	[183]	TS	26	1202 m	7 dana
cont | dyn | pos	[50]	GA	3 do 30	250 m do 3500 m	*
cont | dyn | pos,QCAP	[120]	SWO	20 do 40	1000 m	1 nedelja
	[121]	SWO	40	1000 m	168 h
cont | dyn | QCAP,QCSP	[150]	GRASP	5 do 20	700 m	*
cont | dyn | fix	[90]	SA	7 do 40	1200 m	72 h
	[177]	SA	16 do 30	1200 m	2016 vremenskih jedinica
	[107]	GRASP	5 do 200	80 m do 100 m	*
cont | stoch | stoch	[186]	SA	8 do 40	*	*
cont, draft | dyn | pos,QCAP	[41]	SA	20 do 40	1000 m	168 h
cont | cycl, due | QCAP	[185]	SWO	15 do 60	500 m do 2000 m	1 nedelja
hybr | dyn | pos	[118]	MA	60	13	*
	[109]	TS	15 do 40	3 do 4	6 do 21 vremenskih jedinica
	[80]	GA	*	2	*
hybr, draft | dyn | pos	[20]	EA	100 do 200	5 do 10	*
	[22]	ACO	100	5	*
	[171]	SWO	10 to 40	10 do 30	150 h
	[21]	EA	100	5	*

su u preostalih 30% pregledane BAP literature. Još veća razlika se manifestuje u vremenskom atributu, jer 73% radova pretpostavlja dinamički dolazak brodova, a samo 8% modela razmatra statičke varijante. U najnovijoj literaturi uvedeni su ciklički i stohastički dolazak brodova. Ciklički dolazak pretpostavlja da brodovi dolaze u luku povremeno, u fiksnim vremenskim intervalima, dok u stohastičkom slučaju

vreme dolaska zavisi od nekog stohastičkog parametra. Vreme obrade uglavnom je određeno dodeljenim položajem veza (60%). S druge strane, resursi veza bazirani na kranovima (QCAP + QCSP) utiču na vreme obrade brodova u 28% ispitivanih radova. Stohastički atribut u vremenu obrade je skoro uveden i pojavljuje se u 6% novije literature. Najjednostavniji slučaj koji uključuje fiksno vreme obrade razmatran je samo u 6% modela.

Kako je dokazano da je BAP NP-težak problem, u literaturi očekivano dominiraju algoritmi zasnovani na metaheurističkim pristupima. Leva strana slike 3.5 pokazuje metaheuristike koje se najčešće primenjuju na razne varijante BAP-a. Na desnoj strani slike 3.5, te metaheuristike su grupisane na osnovu njihove prirode. Evolutivni algoritmi dominiraju i korišćeni su u 51% BAP radova, od čega su GA primenjeni u 43% slučajeva. Tabu pretraga i simulirano kaljenje su uključeni u 18% pristupa rešavanju BAP-a. Iznenađujuće je da se inteligencija roja retko koristi, samo u 6% algoritama. Pristupi zasnovani na lokalnom pretraživanju su predloženi u 32% BAP literature.



Slika 3.5: Frekvencija metaheurističkih metoda za rešavanje BAP-a [96]

Glava 4

Opis razmatranih problema i matematičke formulacije

U današnjoj ekonomiji, ključni ciljevi su maksimizacija produktivnosti i efikasnosti, a da bi kontejnerske luke postigle ove ciljeve, potrebno je dobro upravljanje terminalima [14, 172]. Menadžeri kontejnerskog terminala (CT) imaju zadatak da maksimizuju produktivnost i minimizuju operativne troškove. Nivo konkurentnosti i dostignuća na tržištu određeni su vremenskim intervalima koje brodovi provode na čekanju u sidrištu i u luci u toku istovara ili utovara [37]. Menadžeri kontejnerskih terminala pokušavaju smanjiti troškove obrade brodova efikasnim korišćenjem ljudskih resursa, vezova, kontejnerskih skladišta, kranova i druge opreme. Među svim resursima, vezovi su se pokazali kao najvažniji a njihova dobra iskorišćenost poboljšava zadovoljstvo korisnika uslugom i povećava protok kontejnera kroz terminal, što dovodi do većih prihoda luke. Menadžeri terminala obično alociraju prostor veza intuitivnom metodom probe i greške i korišćenjem nekog grafičkog korisničkog interfejsa u računarskom sistemu.

Cilj dodele vezova je minimizacija troškova nastalih usled kašnjenja u odlasku brodova i dodatnih troškova manipulacije zbog neoptimalnih lokacija brodova na vezu. Brodski prevoznici obično obaveštavaju operatere terminala o očekivanom vremenu dolaska i traženom vremenu odlaska broda iz luke. Na osnovu tih informacija, operater terminala pokušava da ispoštuje zahteve svih brodova. Međutim, kada je učestalost dolazaka brodova visoka ili kada dolazi do neočekivanih najava brodova, dešava se da nije moguće ispuniti sve zahteve o traženim vremenima odlaska, pa odlasci nekih brodova mogu biti odloženi.

Pozicija vezivanja broda je takođe važna za donošenje odluka u formiranju plana

vezivanja za određeni period. Kontejneri koje brod treba da transportuje obično pristižu u skladište terminala nekoliko dana pre nego što brod stigne u luku. Ako je brod postavljen na lokaciju blizu mesta gde su uskladišteni kontejneri predviđeni za utovar, troškovi prevoza kontejnera unutar skladišnog prostora terminala mogu biti minimizovani [166]. Takođe, neke lokacije vezivanja mogu biti poželjnije od drugih lokacija zbog različitih faktora, kao što su: dugoročni ugovori za korišćenje određenih vezova, adekvatna dubina vode u vezu određena dubinom gaza broda, različiti nivoi talasa na različitim lokacijama veza, itd. Uopšteno govoreći, pre samog dolaska broda u terminal, operater terminala dodeljuje brodu najpovoljniju lokaciju veza, takozvani *omiljeni vez* koji obezbeđuje najmanju udaljenost veza do skladišta kontejnera predviđenih za utovar ili istovar.

Da bi se poboljšala operativna efikasnost i smanjili troškovi obrade, stvarno vreme vezivanja broda treba da bude blizu očekivanog vremena dolaska broda u terminal. Ukoliko je vreme vezivanja odloženo, neophodno je da brod uspori plovidbu pri dolasku u terminal ili da čeka na obradu u sidrištu terminala, što povećava potrošnju goriva i utiče na nastavak plana plovidbe broda. Opsluživanje broda treba da bude završeno u planiranom vremenskom intervalu, tako da brod može da isplovi iz terminala na vreme. Kašnjenja prouzrokuju dodatne troškove i smanjuju kvalitet usluge kontejnerskog terminala [88].

Smanjenje vremena koje brod provodi u luci obezbeđuje povećanje zadovoljstva brodskih kompanija kao korisnika usluga terminala [173] i vodi do bolje iskorišćenosti infrastrukture terminala [1, 62]. Drugim rečima, ovaj zahtev obezbeđuje obostranu korist i za CT i za korisnike [64, 74]. Zbog svega navedenog, minimizacija vremena koje brodovi provode u luci je često deo odgovarajućih matematičkih modela u literaturi [15, 40, 76], koji reflektuje zahtev luka [43, 87, 141].

Da bi se ispunili navedeni zahtevi koji povećavaju produktivnost CT-a, potrebno je kreirati matematički model koji sadrži sve navedene parametre i obezbezbeđuje realizuju svih procesa u predviđenom vremenu uz minimalne troškove. U ovoj disertaciji je predložen model za problem minimizacije ukupnih troškova terminala kod hibridne statičke i dinamičke dodele vezova koji uzima u obzir zahteve vezane za vreme, resurse i cenu. Detalji statičkog i dinamičkog hibridnog modela koji opisuje problem dodele vezova uz minimizaciju ukupnih troškova dati su u nastavku poglavlja.

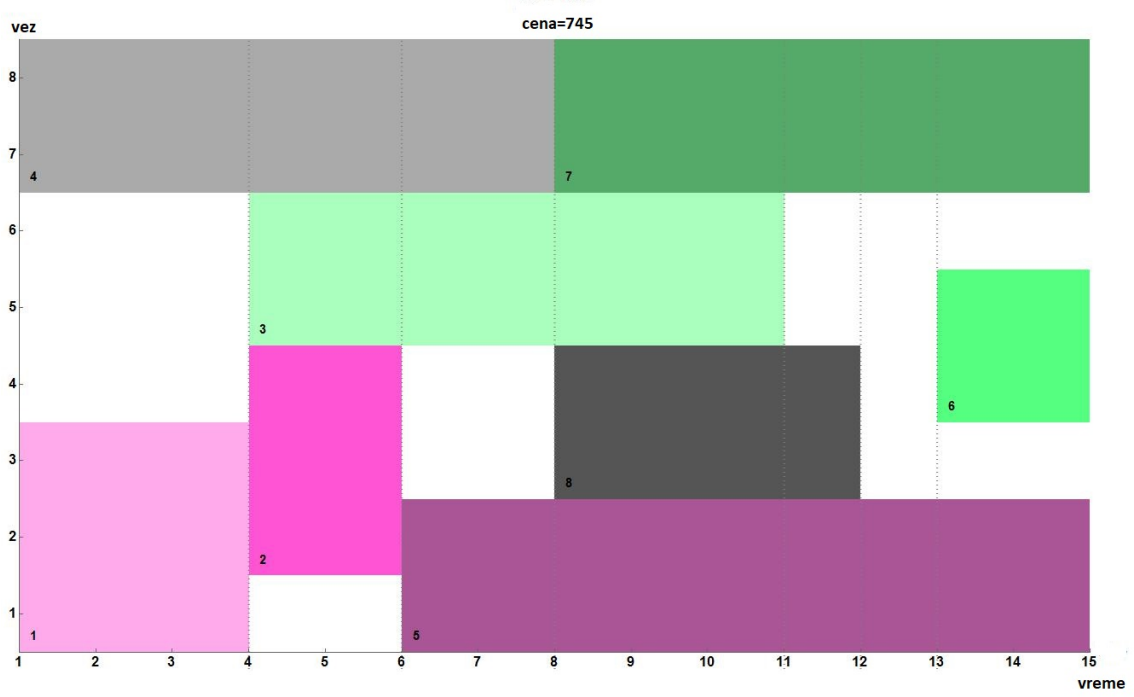
4.1 Statički hibridni problem dodele vezova sa minimizacijom troškova (MCHBAP)

Statička varijanta BAP-a podrazumeva da vremena dolaska brodova nisu striktno ograničena, odnosno, brod može da čeka u luci na obradu ili može da dođe u luku pre očekivanog vremena dolaska. Dozvoljeno je i požurivanje broda, ali u tom slučaju, luka mora da plati izvesnu kaznu brodskoj kompaniji. Hibridna varijanta BAP-a dozvoljava da brod istovremeno zauzima više susednih vezova. Rešenje statičkog hibridnog BAP-a sa minimizacijom troškova (MCHBAP) definiše pozicije vezova i vremenske trenutke kada su vezovi dodeljeni skupu brodova koje treba opslužiti u unapred definisanom periodu planiranja, tako da dodeljene pozicije i vreme dodele obezbeđuju minimizaciju ukupnih troškova vezivanja. Troškovi vezivanja se sastoje od četiri komponente: troškova pozicioniranja broda (ukoliko brod nije alociran na omiljeni vez), troškova ubrzavanja ili čekanja uzrokovanih odstupanjem od očekivanog vremena dolaska broda i troškova kašnjenja završetka obrade broda. Pretpostavka je da je vreme obrade svakog broda poznato unapred i da je to ulazni parametar modela.

MCHBAP izučavan je u radovima [27, 28, 92, 97, 98, 101]. U [28] predložena je MILP formulacija i korišćene su matheuristike, u [92] predložen je egzaktni algoritam zasnovan na kombinatornoj formulaciji koji se pokazao efikasnijim od CPLEX rešavača, ali ipak instance sa više od 50 brodova nije uspevao da reši do optimalnosti. Od metaheurističkih metoda za MCHBAP, BCO je predložen u [97], EA u [98], a VND u [27]. Sve ove metode detaljno su opisane i u [101] gde je predložen i GVNS za MCHBAP. U narednom poglavlju dati su detalji implementacije svih metaheurističkih metoda predloženih u navedenim radovima. U nastavku ovog odeljka opisani su ulazni parametri, kombinatorna i MILP formulacija problema i dokazano je da je problem NP-težak u jakom smislu.

Na slici 4.1, kontejnerski terminal je predstavljen kao pravougaonik sa dve koordinatne ose. Dimenzije pravougaonika definišu prostorna osa sa dužinom jednakom broju raspoloživih vezova u terminalu, i vremenska osa sa dužinom određenom periodom planiranja. U suštini, MCHBAP podrazumeva pakovanje manjih pravougaonika (brodova) u veći pravougaonik koji obuhvata terminal i period planiranja. Pretpostavka je da su obe koordinatne ose diskretnog tipa. Prostorna dimenzija je modelovana rednim brojevima vezova, dok je dati vremenski interval podeljen na segmente tako da se vreme vezivanja sakog broda moze prikazati kao celobrojna

koordinata. Visine malih pravouganika jednake su dužini brodova izraženih brojem susednih vezova koje brod zauzima prilikom smeštanja na vez, dok je širina pravougaonika određena dužinom obrade broda. *Referentna tačka* broda, na slici 4.1 predstavljena rednim brojem broda, je donji levi ugao pravougaonika čije koordinate odgovaraju poziciji veza i vremenskom trenutku početka obrade broda (trenutku vezivanja broda). U ovoj disertaciji se podrazumeva da brodovi imaju fiksno vreme obrade. Dodela vezova je *dopustiva* ako su svi mali pravougaonici smešteni unutar velikog pravougaonika, bez međusobnog preklapanja (slika 4.1).



Slika 4.1: Ilustracija rešenja MCHBAP-a

Uvođenjem dvodimenzionalne reprezentacije terminala, MCHBAP se svodi na dodelu rednog broja veza i vremenske koordinate sa ciljem minimizacije ukupnih troškova. MCHBAP je opisan skupom ulaznih podataka, funkcijom cilja i skupom ograničenja koja definišu dopustiva rešenja. U ovoj disertaciji nije razmatrana dodela kranova, već se pretpostavlja da je svakom vezu dodeljen jedan kran. Na osnovu notacije koju su predložili Bierwirth i Meisel u [7], MCHBAP se klasifikuje kao $hybr|stat|fix|\sum(w_1\,pos + w_2\,speed + w_3\,wait + w_4\,tard)$. Vrednost prostornog atributa je *hybr*, jer je MCHBAP model razmatran u ovoj disertaciji zasnovan na modelu koji su predložili Rashidi i Tsang u [143] i odgovara varijanti prikazanoj

na slici 3d iz [7]. Funkcija cilja u MCHBAP-u bazirana je na strukturi funkcije cilja date u [143], i predstavlja težinsku sumu četiri komponente: troškova dodele veza koji nije omiljen, troškova ubrzavanja ili čekanja u odnosu na očekivano vreme dolaska i troškova kašnjenja u odnosu na planirano vreme završetka obrade broda. Ulazni podaci za MCHBAP, grafički predstavljeni na slici 4.2, su:

l : Broj brodova koje treba opslužiti;

m : Broj raspoloživih vezova u terminalu;

T : Broj vremenskih segmenata u okviru kojih treba izvršiti dodelu vezova;

$vessels$: Skup l uređenih 9-torki koje opisuju brodove,

$$vessels = \{vessel_k | k = 1, \ldots, l\},$$

gde je

$$vessel_k = \{ETA_k, a_k, b_k, d_k, s_k, c_{1k}, c_{2k}, c_{3k}, c_{4k}\}, k = 1, \ldots, l.$$

Elementi uređene 9-torke $vessel_k$ su:

ETA_k : Očekivano vreme dolaska broda k;

a_k : Vremene potrebno za utovar/istovar broda k kada je brodu dodeljen jedan kran;

b_k : Dužina broda k, izražena brojem vezova koje zauzima;

d_k : Predviđeno vreme isplovljavanja broda k iz terminala;

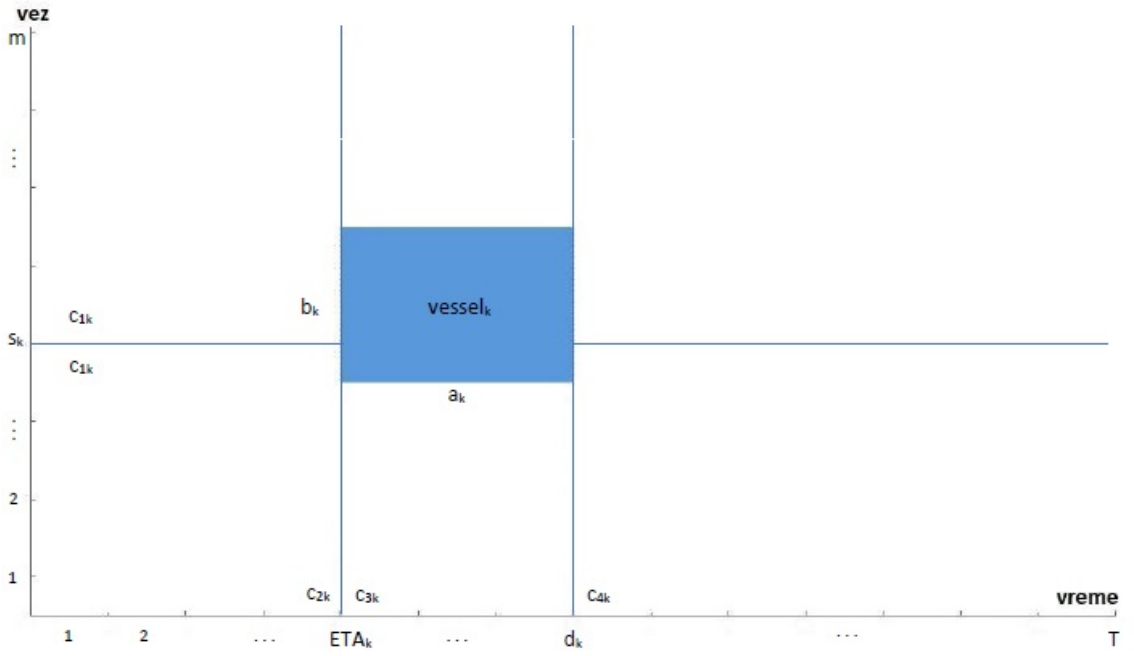
s_k : Omiljeni vez broda k;

c_{1k} : Troškovi nastali dodelom veza koji nije omiljeni brodu k;

c_{2k} : Troškovi (po jedinici) vremena koji nastaju ako brod k mora biti vezan pre očekivanog vremena dolaska ETA_k;

c_{3k} : Troškovi (po jedinici) vremena koji nastaju ako brod k mora biti vezan nakon očekivanog vremena dolaska ETA_k;

c_{4k} : Troškovi (po jedinici) vremena koji nastaju ako brod k mora ostati na vezu i nakon predviđenog vremena isplovljavanja d_k.



Slika 4.2: Ulazni parametri za MCHBAP

Dopustivo rešenje za MCHBAP se sastoji od uređenih parova (B_k, At_k), $k = 1, 2, \ldots, l$ gde B_k označava najmanji indeks veza dodeljenog brodu k, $B_k \in \{1, 2, \ldots, m\}$ a At_k najmanji indeks vremenskog segmenta obrade broda k, $At_k \in \{1, 2, \ldots, T\}$ (videti [92]). Uređeni par (B_k, At_k) odgovara referentnoj tački broda k, $k = 1, 2, \ldots, l$. Dopustivo rešenje MCHBAP-a mora da zadovolji sledeća ograničenja:

Ograničenje 1. U svakom vremenskom segmentu t, $t = 1, \ldots, T$, vez je dodeljen samo jednom brodu.

Ograničenje 2. Vez je dodeljen brodu u vremenskom intervalu definisanom vremenom dolaska broda i vremenom isplovljavanja broda iz terminala.

Cilj MCHBAP-a je minimizovati ukupne troškove kontejnerskog terminala nastale dodelom raspoloživih vezova. Precizan matematički zapis ove funkcije cilja je (videti [143]):

$$\sum_{k=1}^{l} \left(c_{1k}\sigma_k + c_{2k}(ETA_k - At_k)^+ + c_{3k}(At_k - ETA_k)^+ + c_{4k}(Dt_k - d_k)^+ \right) , \qquad (4.1)$$

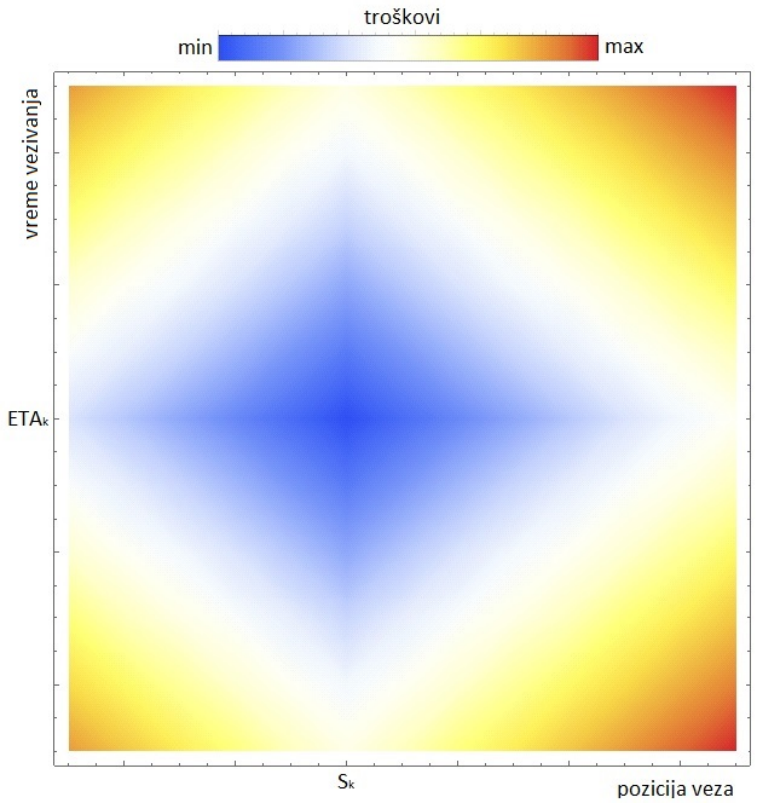
gde je

$$\sigma_k = \sum_{t=1}^{T} \sum_{i=1}^{m} \{|i - s_k| : \text{pozicija } (t, i) \text{ je dodeljena brodu } k\} ,$$

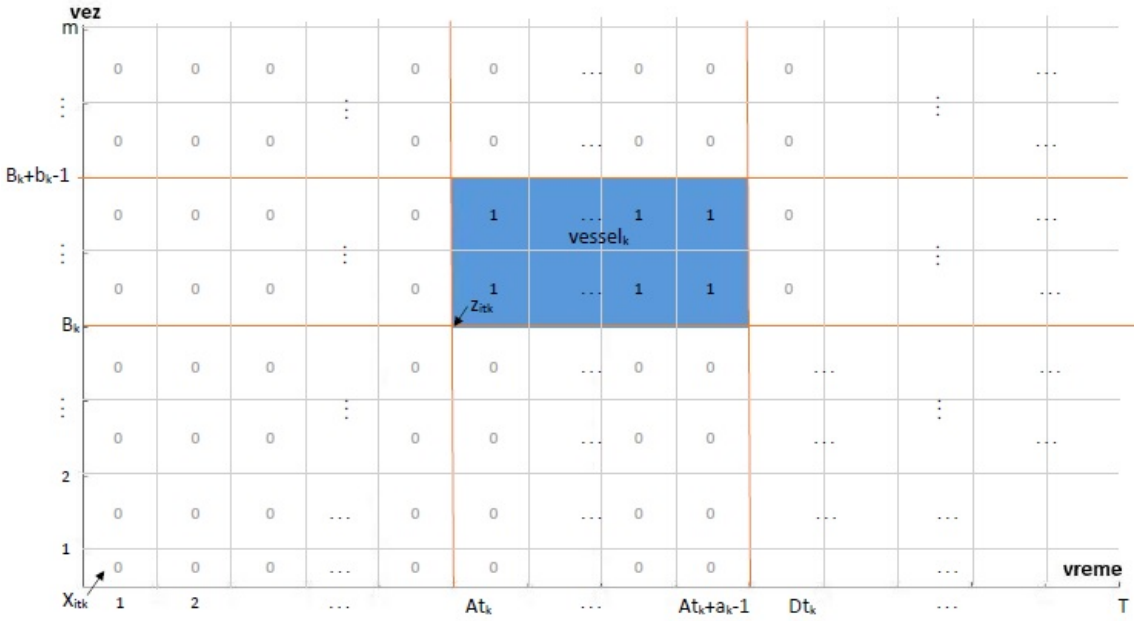
$$(a-b)^+ = \begin{cases} a-b\,, & \text{ako je } a > b\,, \\ 0\,, & \text{inače}\,, \end{cases}$$

pri čemu $Dt_k = At_k + \lceil a_k/b_k \rceil$ predstavlja vreme isplovljavanja broda k, $Dt_k \in \{1, 2, \ldots, T\}$. Ukoliko je za operaciju utovara ili istovara broda k upotrebljen samo jedan kran, vreme obrade je a_k. Međutim, ako brod k zauzima b_k vezova, a po pretpostavci, na svakom vezu se nalazi po jedan kran, vreme obrade se smanjuje b_k puta. Izraz $(a-b)^+$ utiče na vrednost funkcije cilja samo ako ima pozitivnu vrednost.

Prema definiciji funkcije cilja iz [143], σ_k je zapisana u obliku dvostruke sume i opisuje troškove terminala uzrokovane neadekvatnom dodelom vezova. Naime, brod k ima samo jedan omiljeni vez, ali zbog svojih dimenzija može zauzeti više vezova. Osim omiljenog veza, svi ostali dodeljeni vezovi uzrokuju troškove. Nedostatak odgovarajućih resursa na ovim vezovima zahteva angažovanje dodatne opreme i radne snage, što povećava trošak obrade broda. Izgled funkcije troškova za MCHBAP prikazan je na slici 4.3. Na slici je označena idealna pozicija broda sa minimalnim troškovima vezivanja, koja odgovara tački sa koordinatama (s_k, ETA_k).



Slika 4.3: Zavisnost troškova od dodeljenog veza i vremena vezivanja



Slika 4.4: Promenljive odlučivanja za MCHBAP

Kompletan matematički model za MCHBAP u obliku mešovitog celobrojnog linearnog programa (engl. *Mixed Integer Linear Programming* - MILP) dat je u [28]. MILP formulacija iz [28] koristi sledeće promenljive odlučivanja:

- binarne promenljive x_{itk}, z_{itk} i v_{tk} definisane sa:

$$x_{itk} = \begin{cases} 1, & \text{ako je vez } i \text{ u trenutku } t \text{ dodeljen brodu } k, \\ 0, & \text{inače;} \end{cases}$$

$$z_{itk} = \begin{cases} 1, & \text{ako je } (t,i) \text{ referentna tačka broda } k, \\ 0, & \text{inače;} \end{cases}$$

$$v_{tk} = \begin{cases} 1, & \text{ako se brod } k \text{ obrađuje u trenutku } t, \\ 0, & \text{inače.} \end{cases}$$

- celobrojne promenljive At_k i Dt_k koje uzimaju vrednosti iz skupa $\{1, 2, \ldots, T\}$.

Grafički prikaz promenljivih odlučivanja dat je na slici 4.4. Kako MCHBAP formulacija sadrži nelinearnosti izražene apsolutnim vrednostima, pozitivnim komponentama i uslovnim izrazima, potrebno je izvršiti neka pretprocesiranja u cilju formiranja jednostavnije MCHBAP formulacije. Troškovi alokacije za sve moguće pozicije broda su unapred izračunati i smešteni u pomoćnim matricama $E1_{tk}$, $E2_{tk}$,

$D1_{tk}$ i Zb_{ik} definisanim sa:

$$\begin{aligned} E1_{tk} &= \begin{cases} c_{2k} \cdot (ETA_k - t), & \text{ako je } ETA_k > t, \\ 0, & \text{inače;} \end{cases} \\ E2_{tk} &= \begin{cases} c_{3k} \cdot (t - ETA_k), & \text{ako je } t > ETA_k, \\ 0, & \text{inače;} \end{cases} \\ D1_{tk} &= \begin{cases} c_{4k} \cdot (t + H_k - d_k), & \text{ako je } t + H_k > d_k, \\ 0, & \text{inače;} \end{cases} \\ Zb_{ik} &= \sum_{i_1=i}^{i+b_k-1} B_{i_1 k}, \end{aligned}$$

gde je

$$B_{i_1 k} = \begin{cases} H_k \cdot c_{1k} \cdot (i_1 - s_k), & \text{ako je } i_1 \geq s_k, \\ H_k \cdot c_{1k} \cdot (s_k - i_1), & \text{inače.} \end{cases}$$

Pomoćni niz čiji su elementi $H_k = \lceil a_k / b_k \rceil$ se može jednostavno unapred izračunati.

Polazeći od formulacije prikazane u [134] i opisane šeme predprocesiranja, MILP za MCHBAP iz [28] je dat sa:

$$\min \sum_{k=1}^{l} \sum_{i=1}^{m} \sum_{t=1}^{T} z_{itk} \left(Zb_{ik} + E1_{tk} + E2_{tk} + D1_{tk} \right) \tag{4.2}$$

pri uslovima

$$\sum_{k=1}^{l} x_{itk} \leqslant 1, \ i = 1, 2, \ldots, m; \ t = 1, \ldots, T \tag{4.3}$$

$$\sum_{i=1}^{m} \sum_{t=1}^{T} z_{itk} = 1, \ k = 1, 2, \ldots, l \tag{4.4}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{m} t \cdot z_{itk} = At_k, \ k = 1, 2, \ldots, l \tag{4.5}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{m} t \cdot z_{itk} + H_k = Dt_k, \ k = 1, 2, \ldots, l \tag{4.6}$$

$$\sum_{k=1}^{l} \sum_{i=1}^{m} x_{itk} \leqslant m, \ t = 1, 2, \ldots, T \tag{4.7}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{m} x_{itk} \geqslant a_k, \ k = 1, 2, \ldots, l \tag{4.8}$$

$$\sum_{i=1}^{m} x_{itk} \leqslant M \cdot v_{tk},\ t=1,\ldots,T;\ k=1,2,\ldots,l \tag{4.9}$$

$$\sum_{i=1}^{m} x_{itk} \geqslant v_{tk},\ t=1,\ldots,T;\ k=1,2,\ldots,l \tag{4.10}$$

$$(t+1)\cdot v_{tk} \leqslant Dt_k,\ t=1,\ldots,T;\ k=1,2,\ldots,l \tag{4.11}$$

$$(t_2-t_1+1) \leqslant \sum_{t=t_1}^{t_2} v_{tk} + M\cdot(2-v_{t_1k}-v_{t_2k}),\ t_1=1,\ldots,T-1;$$
$$t_2=2,\ldots,T\ (t_1<t_2);\ \ k=1,2,\ldots,l \tag{4.12}$$

$$v_{tk} \leqslant \sum_{i=1}^{m}\sum_{t_1=1}^{t} z_{it_1k},\ t=1,\ldots,T;\ k=1,2,\ldots,l \tag{4.13}$$

$$\sum_{i=1}^{i_1-1}\sum_{t=1}^{T} x_{itk} + \sum_{i=i_1+b_k}^{m}\sum_{t=1}^{T} x_{itk} \leqslant M\left(1-\sum_{t=1}^{T} z_{i_1tk}\right),$$
$$i_1=2,\ldots,m-b_k;\ k=1,2,\ldots,l \tag{4.14}$$

$$\sum_{i=1+b_k}^{m}\sum_{t=1}^{T} x_{itk} \leqslant M\left(1-\sum_{t=1}^{T} z_{itk}\right),\ k=1,2,\ldots,l \tag{4.15}$$

$$\sum_{i=1}^{m-b_k}\sum_{t=1}^{T} x_{itk} \leqslant M\left(1-\sum_{t=1}^{T} z_{(m-b_k+1)tk}\right),\ k=1,2,\ldots,l \tag{4.16}$$

$$b_k - \sum_{i=1}^{m} x_{itk} \leqslant M\,(1-v_{tk}),\ t=1,\ldots,T;\ k=1,2,\ldots,l \tag{4.17}$$

$$x_{itk}, z_{itk}, v_{tk} \in \{0,1\}, \tag{4.18}$$

gde M predstavlja dovoljno veliku pozitivnu konstantu.

Funkcija cilja koju treba minimizovati data je jednačinom (4.2) i predstavlja težinsku sumu troškova vezivanja brodova: troškova koji zavise od udaljenosti dodeljenog veza od omiljenog veza, troškova koji nastaju zbog vezivanja pre ili kasnije od ETA i troškova koji nastaju usled kašnjenja u odnosu na planirano vreme završetka obrade broda. Prema ograničenju (4.3), vez može biti dodeljen tačno jednom brodu u datom vremenskom trenutku. Ograničenje (4.4) obezbeđuje da svaki brod može da ima samo jednu referentnu tačku. Ograničenja (4.5) i (4.6) definišu vrednosti promenljivih koje određuju vremenski interval obrade broda. Ograničenja (4.7) ne dozvoljavaju prekoračenje broja raspoloživih vezova. Vreme obrade za svaki brod je kontrolisano ograničenjem (4.8). Ograničenja (4.9) i (4.10) definišu vezu između promenljivih v_{tk} i x_{itk}, dok ograničenje (4.11) povezuje vremena isplovljavanja Dt_k sa promenljivom v_{tk}. Kako je vrednost promenljive v_{tk} jednaka 1 ako bar jedan kran

opslužuje brod k u vremenskom segmentu t, vreme isplovljavanja broda k mora biti veće ili jednako od $t+1$. Neadekvatne vrednosti v_{tk} eliminisane su ograničenjima (4.12) i (4.13). Naime, brod može biti obrađivan samo ako mu je dodeljen najmanje jedan vez u datom vremenskom segmentu. Ograničenja (4.14)-(4.17) označavaju da x_{itk} može imati vrednost 1 samo u okviru pravougaonika kojim je predstavljen brod.

Na osnovu izložene MILP formulacije (4.2)-(4.29) sledi da je složenost MCHBAP-a u odnosu na broj promenljivih $O(mlT)$. Preciznije, da bi problem bio rešen potrebno je odrediti vrednosti za $lT(2m+1)$ binarnih promenljivih i $2l$ celobrojnih promenljivih, i izračunati vrednost funkcije cilja, koja je u opštem slučaju realan broj. Složenost problema u odnosu na broj ograničenja je $O(lT^2+mT+ml)$.

MCHBAP je NP-težak problem u jakom smislu, jer se može posmatrati kao problem raspoređivanja [51, 136]. U nastavku je po prvi put u literaturi dat dokaz da je čak i najjednostavnija varijanta MCHBAP-a NP-težak problem u jakom smislu.

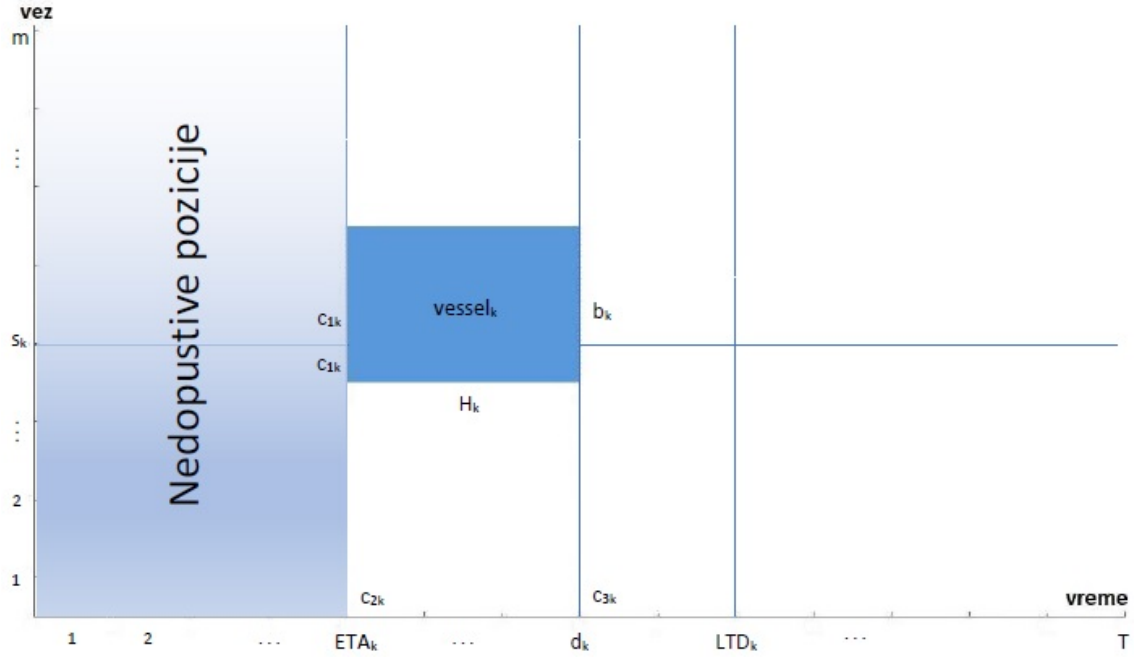
Lema 1. MCHBAP je NP-težak problem u jakom smislu. Čak je i njegov potproblem dobijen odbacivanjem zahteva za omiljenim vezom u kojem funkcija cilja predstavlja samo ukupno težinsko kašnjenje takođe NP-težak problem u jakom smislu.

Dokaz. Za dokaz ove teoreme, dovoljno je pokazati da se poznati problem raspoređivanja identičnih mašina sa vremenom pojavljivanja poslova i minimizacijom ukupnog težinskog kašnjenja (engl. *identical machine scheduling problem with release dates and minimization of total weighted tardiness*), može polinomijalno redukovati na MCHBAP. U standardnoj troparametarskoj notaciji problema raspoređivanja predloženoj u radu Gareya i Johnsona [51], problem raspoređivanja identičnih mašina klasifikovan je kao $P_m|r_j|\sum w_jT_j$. Za potrebe dokazivanja teoreme, u izloženoj MCHBAP notaciji, pogodno je indeks j preimenovati u k. Pretpostavka je da svaki brod predstavlja posao a da vez odgovara mašini. Posledično, vreme obrade broda dato sa $\lceil a_k/b_k \rceil$ odgovara vremenu izvršavanja posla p_k, dok očekivano vreme dolaska broda ETA_k odgovara vremenu pojavljivanja posla r_k. Dodatno, vreme završetka posla odgovara vremenu isplovljavanja broda d_k, dok se kašnjenje računa na standardni način, uz pretpostavku da je $w_k = c_{4k}$. Zahtev za omiljenim vezom, troškovi požurivanja i vreme čekanja su ignorisani, postavljanjem odgovarajućih troškova c_{1k}, c_{2k} i c_{3k} na vrednost nula. Poznato je da je formirani problem raspoređivanja mašina NP-težak problem u jakom smislu čak i u slučaju $m=1$ (pogledati [136]). Dakle, problem raspoređivanja identičnih mašina sa vremenom pojavljivanja poslova i mi-

nimizacijom ukupnog težinskog kašnjenja je polinomijalno sveden na MCHBAP, a kako je problem raspoređivanja NP-težak problem u jakom smislu to je i MCHBAP NP-težak problem u jakom smislu, čime je dokaz završen. □

4.2 Dinamički hibridni problem dodele vezova sa minimizacijom troškova (DMCHBAP)

Prilikom dodele vezova nije realno očekivati da su svi brodovi već stigli u luku i da čekaju na utovar ili istovar. Prema radu [89], vremena dolaska brodova u luku često podležu promenama uzrokovanim zadržavanjima u prethodno posećenim lukama, vremenskim uslovima u toku plovidbe, promenama rute plovidbe, mehaničkim problemima, itd. Zbog toga dinamički BAP bolje oslikava realnu situaciju u luci od statičke varijante [154]. Takođe, analizirajući pregledne radove iz literature koji se odnose na BAP, može se uočiti da se većina autora opredeljuje za izučavanje dinamičke varijante dodele vezova [8, 96]. U novijoj literaturi je čak sugerisano da buduća izučavanja budu fokusirana samo na dinamičku varijantu BAP-a [17, 89].



Slika 4.5: Ulazne promenljive za DMCHBAP

Dinamički MCHBAP (DMCHBAP) je prvi put predstavljen u [99]. Prilikom rešavanja DMCHBAP-a cilj je minimizovati ukupne troškove dodele vezova brodova sa

fiksnim vremenom obrade, koje treba opslužiti u određenom vremenskom intervalu. Za razliku od statičke varijante, DMHCBAP uključuje stroža ograničenja na vreme vezivanja broda. Preciznije, brod ne može biti požurivan pri plovidbi do terminala, odnosno, brod ne može biti opsluživan pre očekivanog vremena dolaska. Kao posledica ovog jakog ograničenja, dobijena je struktura troškova vezivanja koji se sastoji od tri komponente: troškovi pozicioniranja broda na vez, troškovi čekanja i troškovi kašnjenja u završetku obrade brodova. Brodovi su u DMCHBAP-u opisani skupom parametara i vrednostima troškova. Na osnovu notacije predložene u [7], DMCHBAP je klasifikovan kao $hybr|dyn|fix|\sum(w_1\,pos + w_2\,wait + w_3\,tard)$. Postojeći dinamički modeli BAP-a se od DMCHBAP-a razlikuju po jednom ili više parametara klasifikacije ili po funkciji cilja. Predloženi DMCHBAP predstavlja uopštenje dinamičkog diskretnog BAP-a koji je najzastupljenija varijanta u literaturi.

U nastavku odeljka, predložena je matematička formulacija za DMCHBAP, koja je dobijena modifikacijom modela statičkog MCHBAP-a iz [92] i njegovim prilagođavanjem dinamičkim karakteristikama DMCHBAP-a. Na slici 4.5 predstavljeni su ulazni podaci za DMCHBAP. Ulazni podaci DMCHBAP-a su slični ulaznim podacima statičkog MCHBAP-a, ali je struktura uređene 9-torke koja se odnosi na brod nešto izmenjena:

l : Ukupan broj brodova koje treba opslužiti;

m : Ukupan broj raspoloživih vezova;

T : Ukupan broj vremenskih intervala predviđenih za obradu brodova (što iznosi $T-1$ vremenskih segmenata);

$vessels$: Niz podataka koji opisuje l brodova, gde je

$$vessels = \{vessel_k : k = 1, \ldots, l\};$$

$vessel_k$: uređena 9-torka, sledeće strukture

$$vessel_k = \{ETA_k, a_k, b_k, d_k, LTD_k, s_k, c_{1k}, c_{2k}, c_{3k}\}, k = 1, \ldots, l.$$

Elementi 9-torke označeni sa ETA_k, a_k, b_k, d_k i s_k imaju isto značenje kao i kod MCHBAP-a, a preostali parametri su:

LTD_k : Najkasnije vreme isplovljavanja broda k;

c_{1k} : Troškovi koji nastaju ako brodu k ne može biti dodeljen omiljeni vez;

c_{2k} : Troškovi (po jedinici vremena), koji nastaju ako brod k ne može biti vezan u očekivanom vremenu dolaska ETA_k;

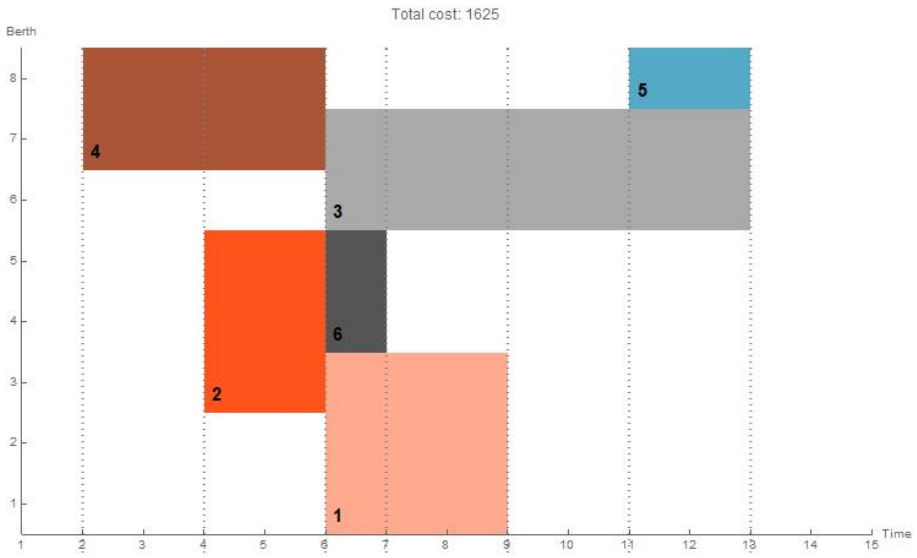
c_{3k} : Troškovi (po jedinici vremena), koji nastaju ako brod k kasni sa završetkom obrade i isplovljava nakon vremena definisanog sa d_k.

Referentne tačke brodova k su (B_k, At_k), $k = 1, 2, \ldots, l$, gde je $B_k \in \{1, 2, \ldots, m\}$ i $At_k \in \{1, 2, \ldots, T-1\}$, sa istim skupom oznaka i ograničenja kao kod MCHBAP-a. Dopustivo rešenje DMCHBAP-a se sastoji od l referentnih tačaka. Cilj DMCHBAP-a je minimizacija ukupnih troškova koje čine tri gore navedene komponente. Preciznije, funkcija cilja koju treba minimizovati ima sledeću strukturu:

$$\sum_{k=1}^{l}\left(c_{1k}\sigma_k + c_{2k}(At_k - ETA_k)^+ + c_{3k}(Dt_k - d_k)^+\right), \tag{4.19}$$

pri čemu su σ_k, $(a-b)^+$ i $Dt_k \in \{1, 2, \ldots, T\}$ definisani na isti način kao kod MCHBAP-a.

Slika 4.6 ilustruje rešenje DMCHBAP-a u prostorno-vremenskom dijagramu sa diskretnim koordinatama.



Slika 4.6: Ilustracija DMCHBAP rešenja

Matematička formulacija DMCHBAP-a, po prvi put predložena u ovoj disertaciji, koristi sledeće celobrojne promenljive odluke:

At_k : Vreme vezivanja broda k, koje može uzimati vrednosti iz skupa $\{ETA_k, \ldots, T\}$.

Dt_k : Vreme isplovljavanja broda k, koje može uzimati vrednosti iz skupa $\{At_k + 1, \ldots, LTD_k\}$.

P_k : Pozicija veza za brod k sa vrednostima iz skupa $\{1, \ldots, m\}$.

Binarne promenjive DMCHBAP modela su:
z_{ijk} : odnosi se na referentnu tačku (j, i) broda k:

$$z_{ijk} = \begin{cases} 1, & \text{ako se brod } k \text{ nalazi na vezu } i \text{ u vremenskom trenutku } j, \\ 0, & \text{inače;} \end{cases}$$

r_{k1k2} : relativna pozicija brodova $k1$ i $k2$ u odnosu na vremensku osu:

$$r_{k1k2} = \begin{cases} 1, & \text{ako je brod } k2 \text{ desno od broda } k1 \text{ u odnosu na vremensku osu,} \\ 0, & \text{inače;} \end{cases}$$

u_{k1k2} : relativna pozicija brodova $k1$ i $k2$ u odnosu na prostornu osu:

$$u_{k1k2} = \begin{cases} 1, & \text{ako je brod } k1 \text{ ispod broda } k2 \text{ u odnosu na prostornu osu,} \\ 0, & \text{inače;} \end{cases}$$

Radi dobijanja jednostavnije formulacije za DMCHBAP, neophodno je izvršiti predprocesranje, odnosno uvesti nekoliko novih parametara, čije se vrednosti lako računaju na osnovu ulaznih parametara modela:

- prvo je potrebno izračunati vrednost niza $H = (H_k)$, gde je $H_k = \lceil a_k/b_k \rceil$ i predstavlja vreme obrade broda k;
- a zatim definisati pomoćne matrice E_{kj}, D_{kj} i Zb_{ki} na sledeći način:

$$\begin{aligned} E_{kj} &= \begin{cases} c_{3k} \cdot (j - ETA_k), & \text{ako je } j > ETA_k, \\ 0, & \text{inače;} \end{cases} \\ D_{kj} &= \begin{cases} c_{4k} \cdot (j + H_k - d_k), & \text{ako je } j + H_k > d_k, \\ 0, & \text{inače;} \end{cases} \\ Zb_{ki} &= \sum_{i_1=i}^{i+b_k-1} B_{ki_1}, \end{aligned}$$

  gde je

$$B_{ki_1} = \begin{cases} H_k \cdot c_{1k} \cdot (i_1 - s_k), & \text{ako je } i_1 \geq s_k, \\ H_k \cdot c_{1k} \cdot (s_k - i_1), & \text{inače;} \end{cases}$$

- na kraju, neophodno je definisati matricu F_{ijk}:

$$F_{ijk} = Zb_{ki} + D_{kj} + E_{kj}.$$

Koristeći formulaciju dvodimenzionalnog problema pakovanja predloženog u [137, 138] i gore izloženu notaciju i promenljive odluke, DMCHBAP može biti zapisan u vidu sledećeg MILP modela:

$$\min \sum_{k=1}^{l} \sum_{i=1}^{m} \sum_{j=1}^{T} z_{ijk} F_{ijk} \tag{4.20}$$

pri uslovima

$$\sum_{i=1}^{m} \sum_{j=1}^{T} z_{ijk} = 1,\ k = 1, 2, \ldots, l \tag{4.21}$$

$$\sum_{j=1}^{T} \sum_{i=1}^{m} j \cdot z_{ijk} = At_k,\ k = 1, 2, \ldots, l \tag{4.22}$$

$$Dt_k = At_k + H_k,\ k = 1, 2, \ldots, l \tag{4.23}$$

$$\sum_{j=1}^{T} \sum_{i=1}^{m} i \cdot z_{ijk} = P_k,\ k = 1, 2, \ldots, l \tag{4.24}$$

$$r_{k1k2} + r_{k2k1} + u_{k1k2} + u_{k2k1} \geq 1,$$
$$k1 = 1, 2, \ldots, l;\ k2 = 1, 2, \ldots, l\ (k1 < k2) \tag{4.25}$$

$$At_{k1} - At_{k2} + T \cdot r_{k1k2} \leq T - H_{k1},$$
$$k1 = 1, 2, \ldots, l;\ k2 = 1, 2, \ldots, l \tag{4.26}$$

$$P_{k1} - P_{k2} + m \cdot u_{k1k2} \leq m - b_{k1},$$
$$k1 = 1, 2, \ldots, l;\ k2 = 1, 2, \ldots, l \tag{4.27}$$

$$ETA_k \leq At_k \leq T - H_k + 1,\ k = 1, 2, \ldots, l$$
$$Dt_k \leq LTD_k,\ k = 1, 2, \ldots, l \tag{4.28}$$
$$1 \leq P_k \leq m - b_k + 1,\ k = 1, 2, \ldots, l$$

$$z_{ijk}, r_{k1k2}, u_{k1k2} \in \{0, 1\}. \tag{4.29}$$

Cilj DMCHBAP-a je minimizacija funkcije cilja (4.20) koja predstavlja ukupne troškove dodele vezova. Prvi deo funkcije cilja se odnosi na trošak koji nastaje zbog vezivanja broda na vez koji nije omiljen, drugi deo definiše trošak zbog dodele veza nakon ETA, dok treći deo predstavlja trošak koji nastaje kod isplovljavanja nakon predviđenog vremena. Uloga ograničenja (4.21) je da obezbedi da svi brodovi imaju tačno jednu referentnu tačku. Vreme vezivanja brodova je definisano ograničenjem (4.22), ukupno vreme obrade ograničenjem (4.23), a pozicija vezivanja brodova ograničenjem (4.24). Zabrana međusobnog preklapanja pravougaonika koji

predstavljaju brodove definisana je ograničenjima (4.25), (4.26) i (4.27). Ograničenja (4.28), (4.28) i (4.29) obezbeđuju da postavljeni prostorni i vremenski uslovi ne budu narušeni: vezivanje broda ne može da prethodi ETA vrednosti, brod ne može da ostane na vezu nakon predviđenog vremena definisanog LTD parametrom, i ne može biti prekoračen dostupan broj vezova i predviđen broj vremenskih segmenata u periodu planiranja. Na kraju, ograničenja (4.29) definišu tip promenljivih odluke.

Slično kao u slučaju MCHBAP-a, može se pokazati da je najjednostavnija varijanta DMCHBAP-a NP-težak problem u jakom smislu, što je jedan od doprinosa ove disertacije. Preciznije, važi sledeća teorema, izložena u radu [99].

Lema 2. DMCHBAP je NP-težak problem u jakom smislu. Čak je i njegov potproblem dobijen odbacivanjem zahteva za omiljenim vezom u kojem funkcija cilja predstavlja samo ukupno težinsko kašnjenje takođe NP-težak problem u jakom smislu.

Dokaz. U prethodnom odeljku, teoremom 1 je pokazano da je MCHBAP NP-težak problem u jakom smislu uspostavljanjem analogije sa problemom raspoređivanja identičnih mašina. Za dokaz složenosti DMCHBAP-a dovoljno je pokazati da se MCHBAP može polinomijalno redukovati na DMCHBAP. Matematičke formulacije MCHBAP-a i DMCHBAP-a imaju uglavnom iste parametre, pa nije potrebno njihovo preimenovanje. Izuzetak su promenljive koje označavaju pojedine troškove. Troškovi iz DMCHBAP-a su preimenovani u troškove iz MCHBAP-a na sledeći način: trošak uzrokovan čekanjem c_{2k} postaje ekvivalentan trošku c_{3k} iz MCHBAP-a, dok trošak kašnjenja završetka poslova c_{3k} postaje ekvivalentan trošku c_{4k} u MCHBAP-u. DMCHBAP nema troškove vezivanja pre ETA_k, tako da se parametar c_{2k} u MCHBAP-u može ignorisati postavljanjem njegove vrednosti na nulu. DMCHBAP uključuje dodatni parametar LTD_k koji se ne pojavljuje u MCHBAP-u. Kako dokaz teoreme 1 koristi samo vreme isplovljavanja brodova d_k, i parametar LTD_k se takođe može ignorisati postavljanjem vrednosti na nulu. Na ovaj način je uspostavljena analogija između DMCHBAP-a i MCHBAP-a. Korišćenjem ove analogije i imajući u vidu teoremu 1, može se zaključiti da je DMCHBAP NP-težak problem u jakom smislu, jer se posredno svodi na problem raspoređivanja identičnih mašina (pogledati [136]). □

U literaturi je najzastupljenija diskretna varijanta BAP-a (slika 3.4). Potrebno je naglasiti da se MCHBAP i DMCHBAP jednostavno svode na diskretni slučaj

dodele vezova postavljanjem vrednosti promenljive b_k na 1, tako da opisane hibridne varijante ovih problema predstavljaju uopštenje diskretnog slučaja. Drugim rečima, diskretni slučajevi su jednostavnija varijanta hibridnih verzija MCHBAP-a i DMCHBAP-a.

Glava 5

Metaheuristički pristup za MCHBAP i DMCHBAP

Opšte je poznato da su BAP i sve njegove varijante vrlo kompleksni za rešavanje, čak i u slučajevima instanci problema malih dimenzija. Davidović i sar. u radu [28] navode MILP formulaciju za MCHBAP a kao metod rešavanja predlažu CPLEX komercijalni softver i tri MIP heuristike koje daju sub-optimalna rešenja. Autori su u [28] ovim pristupom uspeli da reše instance MCHBAP-a malih dimenzija koje sadrže do 20 brodova, dok su realne instance problema ostale van domašaja egzaktnog solvera. Zbog toga, adekvatan pristup rešavanju MCHBAP-a i DMCHBAP-a mora da obezbedi efikasno rešavanje problema u realnom vremenu, ali i da isprati konstantne promene koje su odraz dinamičkog procesa dodele vezova.

Kako su statički i dinamički MCHBAP NP-teški problemi u jakom smislu i imajući u vidu zahtev menadžera terminala da u procesu donošenja odluka imaju na raspolaganju prilagodljiv i efikasan softverski alat, očigledno je da su metaheuristike adekvatan pristup rešavanju ovih problema. Zbog toga je u nastavku rada predloženo nekoliko metaheurističkih metoda za rešavanje MCHBAP-a i DMCHBAP-a: evolutivni algoritmi, optimizacija kolonijom pčela i metoda promenljivih okolina, kao i neke njihove varijante.

5.1 Osnovne definicije i notacija

Sve metaheurističke metode predložene u ovoj disertaciji zasnovane su na kombinatornoj formulaciji MCHBAP-a i DMCHBAP-a. Osim toga, predložene metode koriste iste strukture podataka i obuhvataju istu fazu inicijalizacije (predprocesira-

nja). Kombinatorna formulacija, strukture podataka i predprocesirenje po prvi put su korišćeni u [92] prilikom razvoja egzaktnog kombinatornog algolitma za rešavanje MCHBAP-a, a nakon toga korišćeni su i u implementaciji metaheurističkih metoda. U fazi inicijalizacije, formira se lista uređenih trojki koje sadrže informaciju o vezu, vremenskoj koordinati i ukupnim troškovima broda pri izboru referentne tačke definisane sa prve dve koordinate uređene trojke. Liste uređenih trojki, nazvane ξ-liste, formirane su za svaki brod i sadrže sve moguće pozicije broda u datom planskom periodu. Lista od l ξ-lista, gde svaka pojedinačna ξ-lista odgovara jednom brodu, označena je sa Ψ.

U fazi inicijalizacije, ξ-liste su sortirane u neopadajućem redosledu vrednosti troškova. Uloga ξ-liste je da obezbedi efikasnu pretragu prostora rešenja. Sadržaj ξ-liste se ažurira u svakom koraku pretrage, čime se značajno redukuje njena dužina. Preciznije, svaka promena u rasporedu brodova u luci prouzrokuje promene pripadajućih ξ-listi. Ažuriranje obezbeđuje da u svakom momentu ξ-lista sadrži samo elemente koji odgovaraju preostalim dopustivim pozicijama broda. U svakom koraku pretrage, ξ-liste ostaju sortirane, što obezbeđuje jednostavno pronalaženje pozicija koje imaju manje troškove od vrednosti troškova trenutne referentne tačke broda (ukoliko takve pozicije postoje). Osim toga, struktura ξ-liste obezbeđuje i jednostavno detektovanje nedopustivih rešenja problema. Naime, ukoliko je za neki brod pripadajuća ξ-lista prazna, formirano rešenje je nedopustivo.

Važni koraci u toku izvršavanja metaheurističkih metoda su odabir broda koji treba smestiti u luku i izbor adekvatne pozicije za taj brod u prostorno-vremenskoj ravni. Ove odluke se donose stohastički, ali na osnovu prioriteta brodova i vrednosti troškova dopustivih pozicija. Prioritet broda u EA i BCO je linearna kombinacija odgovarajuće vrednosti ETA parametra, veličine pravougaonika koji predstavlja brod u dvodimenzionalnoj ravni, i izračunate prosečne vrednosti troškova za sve elemente ξ-liste koja odgovara posmatranom brodu. Koeficijenti ove linearne kombinacije su redom λ_1, λ_2 i λ_3. Njihove vrednosti su dobijene eksperimentalno uz uslov $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Na ovaj način izračunati prioritet broda koristi se u kombinaciji sa rulet selekcijom.

U BCO algoritmu, kao i u fazi inicijalizacije EA algoritma, odabir pozicije broda je takođe stohastički. Verovatnoća selekcije neke pozicije je određena ukupnim troškovima koji je karakterišu. Pozicije sa manjim troškovima imaju veću šansu da budu odabrane u procesu rulet selekcije.

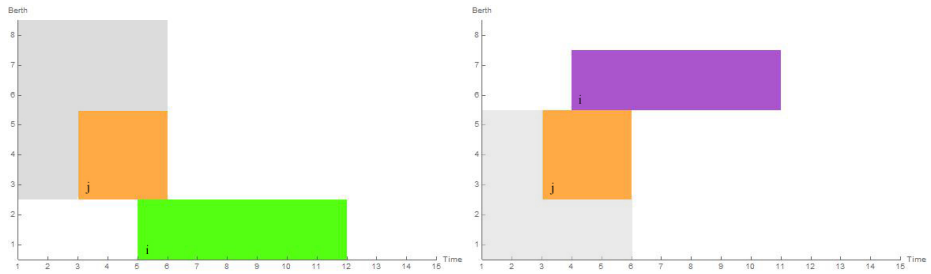
Koncept *para sekvenci*, predložen u [127], upotrebljen je za reprezentaciju rešenja

MCHBAP-a i DMCHBAP-a kod svih predloženih metoda zasnovanih na lokalnom pretraživanju. Par sekvenci čine dve permutacije H i V koje opisuju pozicije brodova u ravni. Permutacije H i V su kreirane po sledećim pravilima:

(a) ako je brod j ispred broda i u permutaciji H, tada brod j „ne vidi" brod i u pogledu „levo-gore";

(b) ako je brod j ispred broda i u permutaciji V, tada brod j „ne vidi" brod i u pogledu „levo-dole".

Permutacije H i V ilustrovane su na slici 5.1. Pogledi levo-gore i levo-dole predstavljeni su sivom bojom. Slika 5.1 prikazuje situaciju u kojoj brod j ne vidi brod i u pogledu levo-gore i levo-dole, što znači da j prethodi i u permutaciji H (levi deo slike), odnosno u permutaciji V (desni deo slike). Ilustracija na slici 5.1 ne odgovara jedinstvenom uređenom paru (H, V). Levi deo slike 5.1 prikazuje situaciju u kojoj j predhodi i u permutaciji H, ali je pozicioniran posle i u permutaciji V. Slično, na desnom delu slike 5.1, j se nalazi posle i u permutaciji H, ali prethodi i u permutaciji V. Transformacija alokacije brodova u uređeni par (H, V) je jedinstvena. Sa druge strane, par permutacija (H, V) predstavlja čitavu klasu alokacija.



Definicija H permutacije Definicija V permutacije

Slika 5.1: Ilustracija reprezentacije rešenja bazirane na paru sekvenci

Sve predložene metaheurističke metode čuvaju najbolje rešenje, vrednost ukupnih troškova najboljeg rešenja i procesorsko vreme prvog pojavljivanja najboljeg rešenja, redom u globalnim promenljivama $Solution$, $GlobalBest$ i $minT$. Osim toga, ukoliko metoda promenljivih okolina koristi fazu razmrdavanja, lokalno poboljšanje razmrdanog rešenja čuva promenljiva $LocalBest$.

5.2 Implementacija evolutivnog algoritma za MCHBAP

Predloženi EA za MCHBAP koristi celobrojno kodiranje jedinki, ruletsku i turnirsku selekciju, kao i četiri tipa operatora mutacije. Nakon primene mutacije, u svakoj generaciji su nad nekoliko odabranih jedinki primenjene dve procedure optimizacije, koje imaju za cilj poboljšanje kvaliteta odabranih jedinki. Prva procedura poboljšanja zasnovana je na promeni dodeljenih vezova, dok druga procedura dozvoljava samo perturbacije redosleda brodova na odabranom vezu. Operator ukrštanja je isključen iz razmatranja, jer proizvodi veliki broj nekorektih jedinki zbog specifičnosti MCHBAP-a. U nastavku slede detaljni opisi svih aspekata predloženog EA.

Reprezentacija jedinki

Celobrojna reprezentacija jedinki se pokazala kao najpogodnija za MCHBAP. Genetski kod (hromozom) svake jedinke čini m podlisti. Svaka podlista odgovara jednom vezu i sadrži indekse brodova čije referentne tačke pripadaju datom vezu. Brodovi su u svakoj podlisti sortirani u redosledu koji odgovara redosledu dodele pozicija na vezu.

Neka je dat primer sa $m = 8$ vezova, $l = 5$ brodova i $T = 15$ vremenskih jedinica čiji su ulazni podaci dati u tabeli 5.1. Reprezentacija jedinke kojoj odgovara dopustivo rešenje MCHBAP-a je:

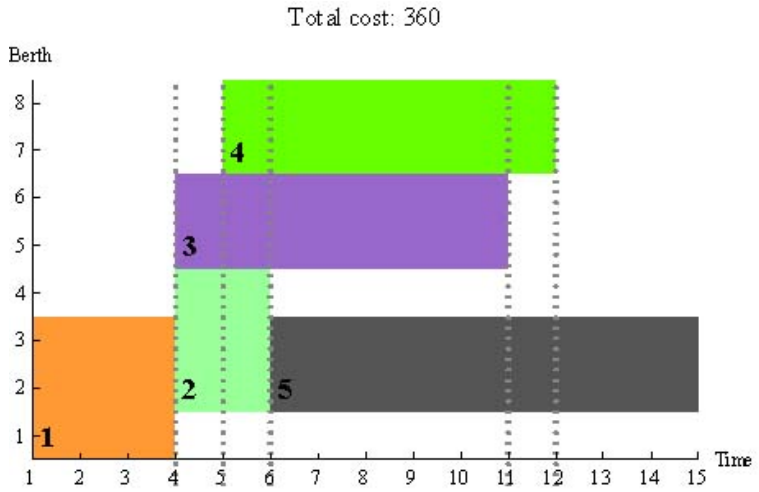
$$Individual\_1: \underbrace{\{\{1\}}_{1}, \underbrace{\{5,2\}}_{2}, \underbrace{\{\}}_{3}, \underbrace{\{\}}_{4}, \underbrace{\{3\}}_{5}, \underbrace{\{\}}_{6}, \underbrace{\{4\}}_{7}, \underbrace{\{\}\}}_{8}.$$

Tabela 5.1: Ulazni podaci za problem $m = 8$, $l = 5$ i $T = 15$

$vessel_k$	ETA_k	a_k	b_k	d_k	s_k	c_{1k}	c_{2k}	c_{3k}	c_{4k}
1	1	9	3	4	1	10	20	20	25
2	4	6	3	6	3	10	20	20	25
3	4	14	2	11	6	10	20	20	25
4	5	14	2	12	7	10	20	20	25
5	6	18	2	16	2	10	20	20	25

Indeksi brodova čije referentne tačke odgovaraju datom vezu su navedeni u zagradama, dok su indeksi vezova dati ispod zagrada. Prikazana reprezentacija pokazuje da vez 1 odgovara referentnoj tački broda 1, dok su referentne tačke brodova 5 i 2 raspoređene na vez 2. Vezovi 5 i 7 sadrže referentne tačke brodova 3 i 4 u tom redosledu. Prazne zagrade iznad vezova sa indeksima 3, 4, 6 i 8 ukazuju da su ti vezovi prazni, odnosno da im nije dodeljena ni jedna referentna tačka nekog broda.

Slika 5.2 ilustruje rešenje MCHBAP-a dobijeno dekodiranjem jedinke *Individual_1*. Ako je referentna tačka broda dodeljena vezu sa indeksom k, indeks tog broda je smešten u k-tu podlistu jedinke. Kako MCHBAP predstavlja hibridnu varijantu BAP-a, brod može da zauzima više susednih vezova.



Slika 5.2: Rešenje dobijeno dekodiranjem jedinke *Individual_1*

Slika 5.2 prikazuje situaciju u kojoj su brodovi 5 i 2 opsluženi na vezu 2, ali njihov redosled u odnosu na vremensku osu ne sledi redosled dat u reprezentaciji jedinke. Kako je brod 5 alociran pre broda 2, on je zauzeo najjeftiniju poziciju. Tek nakon smeštanja broda 5, alociran je brod 2 tako što mu je dodeljena preostala najjeftinija pozicija (koja ne mora da ima veći indeks na vremenskoj osi). Reprezentacija jedinke *Individual_1* prikazuje da je vez 3 prazan (nije mu dodeljena referentna tačka broda). Međutim, nakon dekodiranja rešenja, imajući u vidu da je rešavana hibridna verzija BAP-a, vez 3 nije ostao neiskorišćen, već je zauzet delovima brodova 1, 2 i 5. Dekodirano rešenje MCHBAP-a se sastoji od uređenih trojki (vez, vreme, troškovi) koje odgovaraju referentnim tačkama brodova u dvodimenzionalnoj ravni. Jedinka *Individual_1* i odgovarajuća dodela vezova predtavljena na slici 5.2 predstavljaju optimalno rešenje za primer sa podacima iz tabele 5.1.

Reprezentacija jedinke može da vodi do alokacije brodova koja izlazi iz okvira definisanih vremenskih i prostornih granica. U tom slučaju, jedinka se smatra nekorektnom i eliminiše se iz populacije, jer nekorektne jedinke odgovaraju nedopustivim rešenjima problema. Dužine brodova su poznate unapred kao ulazni parametri algoritma, što omogućava jednostavnu proveru da li vez može da primi brod koji još nije raspoređen u luku. Sa druge strane, dekodiranje korektnih jedinki u MCHBAP rešenje je jedinstveno.

Generisanje početne populacije

Struktura procedure INITIALIZE koja generiše početnu populaciju nEA jedinki je prikazana algoritmom 11. Ulazni parametri procedure INITIALIZE su veličina populacije (nEA) i veličina turnira koji selektuje vez za razmatrani brod. Za svaku od nEA jedinki u početnoj populaciji, procedura INITIALIZE formira odgovarajuću Ψ-listu.

Algorithm 11 Generisanje početne EA populacije

```
procedure INITIALIZE(nEA, size)
    i ← 1
    while i ≤ nEA do
        UnusedVessels ← {1, 2, ..., l}
        individual(i) ← m praznih lista
        while UnusedVessels ≠ ∅ do
            ID ← ROULETTE(UnusedVessels, selectionCriteria)
            Berths ← Odredi sve podskupove susednih vezova
                     sa dovoljno slobodnog prostora za brod ID
            berthID ← TOURNAMENTFORBERTHS(size, Berths)
            individual(i) ← Stavi brod ID na poslednju poziciju u berthID
            UnusedVessels ← UnusedVessels\{ID}
        end while
        if FEASIBLE(individual(i)) then
            i ← i + 1
        end if
    end while
end procedure
```

Struktura nove jedinke zavisi od redosleda alokacije brodova i od dodele indeksa veza svakom brodu. Redosled izbora brodova zavisi od prioriteta, koji je definisan kao težinska suma tri komponente sa koeficijentima λ_i, $i = 1, 2, 3$. Vrednosti ovih koeficijenata su određene eksperimentalno. Prioritet broda je izračunat na osnovu inicijalne ξ-liste u kojoj se sve pozicije broda smatraju dopustivim. U toku alokacije prioriteti brodova ostaju nepromenjeni.

Procedura ROULETTE koristeći pravilo rulet selekcije bira jedan po jedan brod iz skupa neobrađenih brodova. Verovatnoća izbora broda je proporcionalna prethodno

izračunatom prioritetu broda. Dužine brodova i ukupan broj vremenskih segmenata u periodu planiranja su poznati unapred. Zbog toga je jednostavno odrediti podskup susednih vezova sa dovoljno slobodnog prostora za alokaciju odabranog broda. Odabir indeksa veza za selektovani brod vrši procedura TournamentForBerths. Turnirskom selekcijom se za svaki neraspoređeni brod bira jedan vez iz skupa svih mogućih vezova sa dovoljno slobodnog prostora. Procedura smešta odabrani brod na poslednju poziciju na vezu koji mu je dodeljen. Preciznije, indeks broda je smešten na kraj podliste koja opisuje redosled alokacije na odabranom vezu.

Faze odabiranja brodova i dodele veza se smenjuju dok se ne rasporede svi indeksi brodova u odgovarajuće podliste. Nakon toga sledi postupak dekodiranja jedinke u rešenje koje sadrži stvarne pozicije brodova u prostorno-vremenskoj ravni. Nedopustive jedinke algoritam uklanja iz populacije. Postupak kreiranja novih jedinki se ponavlja sve dok se ne formira nova populacija sa nEA dopustivih jedinki. Procedura Feasible ispituje da li je moguće kreiranu jedinku dekodirati u dopustivo rešenje. Korektne jedinke se evaluiraju tako što im se računa funkcija prilagođenosti koja je jednaka odgovarajućoj vrednosti funkcije cilja.

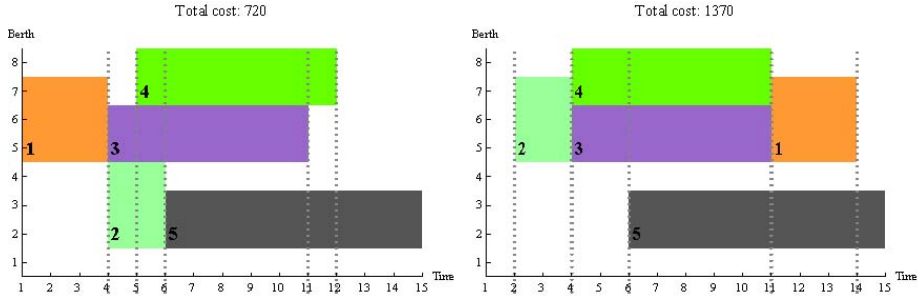
EA operatori

Predloženi evolutivni algoritam koristi koncept *elite* za formiranje nove generacije jedinki. Elitu čine jedinke visokog kvalieta koje direktno prelaze u sledeću generaciju sa ciljem očuvanja dobrog genetskog materijala. U implementiranom EA za MCHBAP, broj elitnih jedinki je $eliteNO = \frac{1}{3} \cdot nEA$ a preostalih $\frac{2}{3} \cdot nEA$ individua nove generacije odabrano je fino gradiranom turnirskom selekcijom [44] sa dve veličine turnira. Turnir veličine $size_1$ je primenjen u $40\% \cdot (\frac{2}{3}nEA)$ slučajeva, dok je ostalih $60\% \cdot (\frac{2}{3}nEA)$ jedinki odabrano turnirom veličine $size_2$, pri čemu je $size_1 < size_2$. Operator turnirske selekcije povećava šansu jedinkama lošijeg kvaliteta da budu prosleđene u sledeću generaciju. Osim toga, turnirska selekcija sprečava duplikate jedinki da uđu u sledeću EA generaciju i obezbeđuje raznovrsnost genetskog materijala. Raznovrsne jedinke pomažu EA algoritmu da prevaziđe problem završetka u lokalnom minimumu.

Pri rešavanju raznih problema optimizacije EA metaheuristikom, korišćeni su različiti tipovi genetskih operatora, u skladu sa karakteristikama konkretnog problema. Na primer, EA algoritmi iz radova [117, 158] koriste operator ukrštanja, dok su u radovima [33, 142] primenjeni samo operatori mutacije. Implementirani EA pristup za MCHBAP ne koristi operator ukrštanja jer su svi ispitani tipovi operatora

ukrštanja formirali veliki broj nekorektnih jedinki. Generalno, postoje dve strategije koje rešavaju problem pojave nekorektnih jedinki u toku rada EA: nekorektne jedinke se uklanjaju iz naredne generacije ili se popravljaju tako da postanu korektne. Strategija uklanjanja velikog broja nekorektnih jedinki koje nastaju primenom operatora ukrštanja je rezultirala u značajnom smanjenju veličine populacije. Sa druge strane, razne strategije korekcije nekorektnih jedinki su značajno povećavale vreme izvršavanja algoritma i uticale na smanjenje njegove efikasnosti. Osim toga, kvalitet najboljeg EA rešenja je bio lošiji od implementiranih varijanti algoritma koje nisu koristile operator ukrštanja. Zbog toga, EA implementacija za MCHBAP koristi samo operatore mutacije dizajnirane u skladu sa karakteristikama problema. Preciznije, implementirani EA koristi četiri tipa mutacije: *ubacivanje*, *inverziju*, *mešanje* i *zamenu*. Na svaku jedinku u populaciji, sa datom verovatnoćom μEA, primenjena su sva četiri operatora mutacije, čime su kreirana najviše četiri potomka mutirane jedinke. Svi predloženi operatori mutacije mogu da proizvedu promene u prostornim i vremenskim koordinatama brodova jedinke koja je mutirana. Na taj način, mutacija značajno utiče na strukturu jedinke i može dovesti do velikih promena u vrednostima funkcije cilja. Zadatak primenjenih operatora mutacije je da pojačaju diversifikaciju pretrage prostora rešenja i spreče algoritam da se zaustavi u lokalnom optimumu.

EA za MCHBAP koristi mutaciju ubacivanja (engl. *insert mutation*), ilustrovanu na slici 5.3. Ovaj operator na slučajan način bira dva broda iz hromozoma jedinke i pomera drugi brod iza prvog broda u hromozomu. Na slici 5.3 je predstavljena situacija u kojoj su iz hromozoma jedinke odabrani brodovi 3 i 2, a zatim je u mutiranoj jedinki brod 2 pomeren iza broda 3.



Pre mutacije:

{{},{**2**,5},{},{},{**3**,1},{},{4},{}}

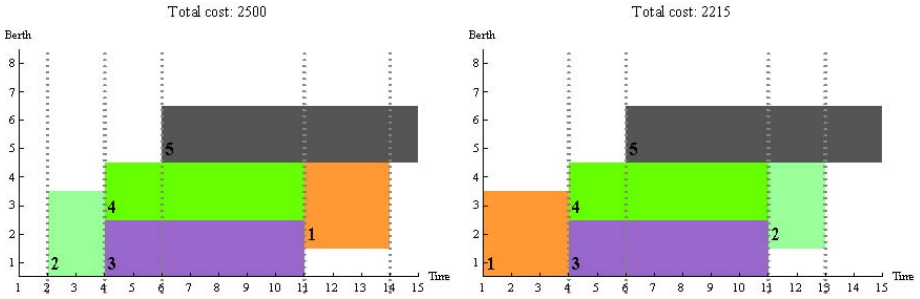
Posle mutacije:

{{},{5},{},{},{3,2,1},{},{4},{}}

Slika 5.3: Primer mutacije ubacivanja

Mutacija inverzije (engl. *inversion mutation*) na slučajan način bira dva broda i

invertuje deo hromozoma između njih, uključujući i dva odabrana broda. Slika 5.4 prikazuje efekat mutacije inverzije: odabrani su brodovi 2 i 1, i okrenut je ceo podniz između njih, uključujući i ta dva broda.



Pre mutacije:

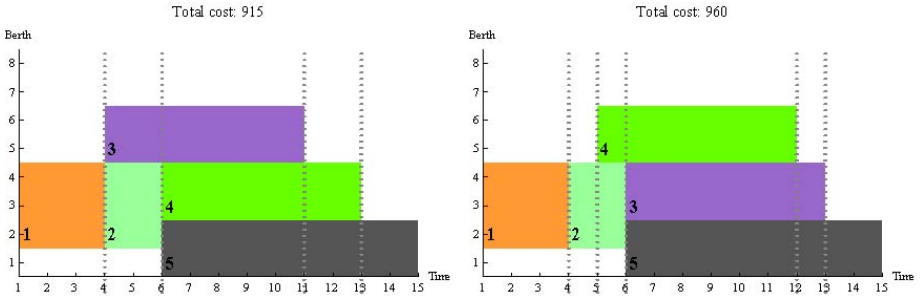
{{**2**,3},{**1**},{4},{},{5},{},{},{}}

Posle mutacije:

{{1,3},{2},{4},{},{5},{},{},{}}

Slika 5.4: Primer mutacije invezije

Mutacija zamene (engl. *swap mutation*) na slučajan način bira dva broda i zamenjuje njihove pozicije u hromozomu jedinke. Na slici 5.5 je prikazan efekat ovog tipa mutacije: odabrani su brodovi 4 i 3 i razmenjene njihove pozicije u mutiranoj jedinki.



Pre mutacije:

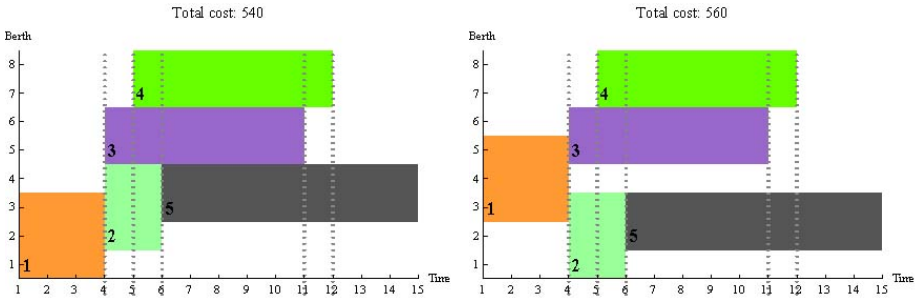
{{5},{1,2},{**4**},{},{**3**},{},{},{}}

Posle mutacije:

{{5},{1,2},{3},{},{4},{},{},{}}

Slika 5.5: Primer mutacije zamene

Mutacija mešanja (engl. *scramble mutation*) na slučajan način permutuje podskup brodova u hromozomu. Na slučajan način se iz hromozoma biraju dva broda, koji definišu početnu i krajnju tačku podskupa brodova koji se permutuje. U nekim slučajevima, mutacija mešanja može biti primenjena na ceo hromozom jer odabrani brodovi mogu da pripadaju različitim podlistama brodova. Primer primene mutacije mešanja je prikazan na slici 5.6. Na slučajan način je permutovan podniz koji čine svi brodovi između brodova 1 i 5, uključujući i ta dva broda.



Pre mutacije:

{{**1**},{2},{**5**},{},{3},{},{4},{}}

Posle mutacije:

{{2},{5},{1},{},{3},{},{4},{}}

Slika 5.6: Primer mutacije mešanja

Verovatnoće primene operatora mutacije su određene parametrom μEA i one menjaju svoje vrednosti tokom izvršavanja EA algoritma, slično strategiji predloženoj u radu [165]. Na početku EA pretrage, favorizuju se mutacije inverzije i mešanja, dok se u kasnijim fazama izvršavanja EA, akcenat stavlja na mutacije ubacivanja i razmene. Primenom ove strategije, favorizuju se značajne promene hromozoma jedinke na početku izvršavanja algoritma što odgovara premeštanju pretrage u udaljene prostore rešenja. U kasnijim fazama algoritma, podstiču se manje promene hromozoma jedinke, što znači da se fokus premešta na pretragu u manjim regionima prostora rešenja. Verovatnoća primene mutacije mešanja i inverzije na gen hromozoma tokom izvršavanja algoritma se smanjuje prema formuli

$$\mu EA = 1 - 0.9 \cdot \frac{popID}{nGen} \ , \tag{5.1}$$

dok verovatnoća mutacija ubacivanja i razmene raste po formuli

$$\mu EA = 0.9 \cdot \frac{popID}{nGen} \ , \tag{5.2}$$

gde je sa $popID$ označen redni broj trenutne EA generacije, a $nGen$ predstavlja maksimalan broj EA generacija. Obe formule su dobijene na osnovu rezultata preliminarhih eksprimenata u cilju određivanja adekvatnih vrednosti EA parametara.

Nakon faze mutacije, predloženi EA primenjuje fazu popravke pojedinih rešenja. Turnirskom selekcijom, algoritam bira podskup jedinki iz populacije i nad njima primenjuje proceduru popravke rešenja. Veličina turnira i broj selektovanih jedinki za primenu strategije popravke su ulazni parametri EA algoritma. Promenom veličine turnira se jednostavno utiče na kvalitet jedinki koje prolaze turnirsku selekciju i ulaze u fazu popravke rešenja. Veće veličine turnira daju manju šansu jedinkama

lošijeg kvaliteta da budu izabrane za fazu popravke, i obrnuto, manje dimenzije turnira favorizuju odabir jedinki lošijeg kvaliteta.

Pseudokod primenjene procedure popravke rešenja predstavljen je algoritmom 12. Procedura IMPROVE ima zadatak da u odabranoj jedinki pronađe bolji raspored brodova na svakom vezu. Procedura CHEAPESTALLOCATION za svaki brod iz grupe brodova datog veza pronalazi poziciju sa najmanjim troškovima na istom vezu. Nakon smeštanja svih brodova na odgovarajuće vezove, algoritam poziva proceduru REORDER koja sortira brodove u nerastući redosled prema vrednostima troškova u formiranoj dodeli vezova. Za svaki brod iz sortiranog niza brodova, procedura popravke proverava ξ-liste. Ako postoji pozicija sa nižim troškovima, procedura pomera razmatrani brod na tu poziciju, bez obzira da li pri tome dolazi do promene dodeljenog veza. Pozicije sa nižim troškovima su identifikovane pomoću procedure EXISTSCHEAPER. Faza popravke rešenja se završava pozivom procedure MAKEINDIVIDUAL, čiji je zadatak da formira odgovarajuće promene u reprezentaciji popravljene jedinke. Preciznije, brod koji je menjao poziciju veza u fazi popravke rešenja, postaje prvi elemenat liste koja odgovara dodeljenom vezu. Redosled brodova u svakoj grupi se menja u skladu sa formiranom alokacijom, sa ciljem da se prvo razmotre brodovi koji prouzrokuju veće troškove.

Algorithm 12 Prva faza popravke EA rešenja

```
procedure IMPROVE(ind)
    for berth ← 1, m do
        group ← ind(berth)
        for j ← 1, LENGTH(group) do
            v ← group(j)
            sol(v) ← CHEAPESTALLOCATION(v, berth)
        end for
    end for
    vessels ← REORDER(sol)
    for i ← 1, l do
        if EXISTSCHEAPER(ξ(vessels(i)), sol(vessels(i))) then
            newSol(vessels(i)) ← ξ(vessels(i), 1)
        else
            newSol(vessels(i)) ← sol(vessels(i))
        end if
    end for
    newInd ← MAKEINDIVIDUAL(newSol)
    RETURN (newInd, newSol)
end procedure
```

U drugoj fazi popravke rešenja, na svaku poboljšanu jedinku formiranu procedurom IMPROVE, primenjuje se procedura lokalne pretrage čija je struktura prikazana algoritmom 13. U ovoj fazi lokalne pretrage, procedura REARRANGE formira nov redosled brodova na odabranom vezu. Procedura CALCULATECOST računa ukupne

troškove grupe brodova. Algoritam prihvata novi redosled brodova ukoliko on vodi ka smanjenju troškova alokacije. Opisani koraci algoritma se ponavljaju za svaku grupu brodova na posmatranom vezu. Promena redosleda brodova ne može dovesti do konfliktnih situacija i proizvesti nedopustivu jedinku. Naime, brodovi mogu menjati pozicije samo ako rezultujuća alokacija ne vodi ka novim konfliktima sa brodovima na susednim vezovima.

Algorithm 13 Druga faza popravke rešenja-procedura lokalne pretrage za EA

```
procedure BerthLocalSearch(ind, sol)
    for berth ← 1, m do
        group ← ind(berth)
        groupSol ← Rearrange(group)
        if CalculateCost(groupSol) < CalculateCost(sol(group)) then
            sol(group) ← groupSol
        end if
    end for
    newInd ← MakeIndividual(sol)
    return (newInd)
end procedure
```

Posle faze popravke rešenja, algoritam računa funkcije prilagođenosti svih jedinki, i na osnovu njih formira sledeću generaciju jedinki. Opisani koraci evolutivnog algoritma se ponavljaju dok se ne ispuni neki od postavljenih kriterijuma zaustavljanja. EA koristi kombinaciju dva kriterijuma zaustavljanja: ukupno utrošeno procesorsko vreme i maksimalan broj formiranih generacija. Promenljiva $minT$ čuva vrednost procesorskog vremena u kojem je algoritam prvi put pronašao rešenje najboljeg kvaliteta. Koraci evolutivnog algoritma za MCHBAP predstavljeni su algoritmom 14.

EA za MCHBAP uključuje nekoliko parametara, kao što su: veličina populacije, veličina turnira, kriterijumi zaustavljanja, verovatnoće mutacije, broj jedinki na koje se primenjuje faza popravke, itd. Ovi parametri su eksperimentalno podešeni na adekvatne vrednosti koje obezbeđuju najbolje performanske EA algoritma.

5.3 Implementacija genetskog algoritma za DMCHBAP

Genetski algoritmi spadaju u klasu evolutivnih algoritama, i u dosadašnjoj BAP literaturi su veoma zastupljeni kao popularan pristup rešavanju različitih varijanti BAP-a. Glavni problem koji se javlja kod primene GA na BAP je formiranje velikog

Algorithm 14 EA za MCHBAP

procedure EAforMCHBAP
  Initialize(nEA)
  $popID \leftarrow 1$
  $GlobalBest \leftarrow \infty$
  while $(SessionTime \leq RunTime) \wedge (popID \leq nGen)$ **do**
    for svaka jedinka **do**
      $\mu EA \leftarrow 0.9 \cdot popID/nGen$
      $newindividual1 \leftarrow$ InsertMutation($noGenes, \mu EA$)
      $newindividual2 \leftarrow$ SwapMutation($noGenes, \mu EA$)
      $newindividual3 \leftarrow$ InversionMutation($noGenes, 1 - \mu EA$)
      $newindividual4 \leftarrow$ ScrambleMutation($noGenes, 1 - \mu EA$)
    end for
    for $noImprovements$ jedinki **do**
      $individualID \leftarrow$ TournamentForImprovement($size_1, size_2, cost$)
      $\{newindividualI, newSol\} \leftarrow$ Improve($IndividualID$)
      $newindividualI \leftarrow$ BerthLocalSearch($newIndividualI, newSol$)
    end for
    Izračunaj troškove za svaku jedinku
    if bestCost($popID$) $< GlobalBest$ **then**
      BestSolUpdate()
    end if
    $eliteNO \leftarrow nEA/3$
    Kopiraj najboljih $eliteNO$ jedinki u sledeću generaciju
    $size1NO \leftarrow$ Round($(nEA - eliteNO) \cdot 0.4$)
    Odaberi $size1NO$ jedinki turnirom veličine $size_1$
    $size2NO \leftarrow nEA - eliteNO - size1NO$
    Odaberi $size2NO$ jedinki turnirom veličine $size_2$
    $popID \leftarrow popID + 1$
  end while
end procedure

broja nekorektnih jedninki koje odgovaraju nedopustivim rešenjima. Ovaj problem se u literaturi obično prevazilazi izostavljanjem operatora ukrštanja [60, 61, 98].

Algorithm 15 cGA za DMCHBAP

procedure cGA($vessels, nGA, RunTime, nGen, nImpr$)
  $population \leftarrow$ InitGA($nGA, vessels$)
  $popID \leftarrow 1$
  while $SessionTime \leq RunTime \wedge popID \leq nGen$ **do**
    $tempPop1 \leftarrow$ Mutate($population$)
    CalculateCost($tempPop1$)
    Update($population, tempPop1$)
    $statInd \leftarrow$ CreateStatisticalIndividual()
    Update($population, statInd$)
    $tempPop2 \leftarrow$ Improve($population, nImpr$)
    CalculateCost($tempPop2$)
    Update($population, tempPop2$)
    $population \leftarrow$ Selection($population$)
    $popID \leftarrow popID + 1$
  end while
end procedure

U nastavku je dat opis kombinovanog genetskog algoritma (cGA) za dinamički MCHBAP. Pseudokod predloženog cGA dat je algoritmom 15. Broj jedinki u populaciji, označen sa nGA, je ulazni parametar algoritma. Na početku rada cGA poziva

se procedura INITGA za generisanje nGA korektnih jedinki početne populacije koje odgovaraju dopustivim DMCHBAP rešenjima. Nakon generisanja početne populacije, cGA se izvršava kroz niz iteracija dok se ne ispuni jedan od dva postavljena kriterijuma zaustavljanja definisana maksimalnim vremenom izvršavanja ($RunTime$) i maksimalnim brojem generacija ($nGen$). Svaka iteracija cGA se sastoji od faze selekcije, faze mutacije sa dva tipa operatora mutacije, i faze optimizacije u kojoj algoritam poziva dve procedure popravke rešenja i formira statističku jedinku. Pomoćne promenljive $tempPop1$ i $tempPop2$ su uvedene da bi se izbeglo višestruko računanje vrednosti funkcija cilja.

U sledećim odeljcima detaljno su izloženi svi aspekti predloženog cGA pristupa za rešavanje DMCHBAP-a.

Reprezentacija jedinki

Jedinka u cGA ima strukturu liste sa l podlisti, pri čemu svaka podlista odgovara grupi brodova koji se istovremeno razmatraju u procesu alokacije, dok podliste odgovaraju genima (delovima hromozoma). Pri alociranju brodova na vezove, algoritam razmatra grupe brodova u redosledu definisanom strukturom jedinke. Elementi podliste su indeksi brodova sortirani u redosled u kojem su smešteni na vezove. Međutim, ova reprezentacija ne garantuje da će brodovi biti obrađeni u istom redosledu u kojem su alocirani na vezove. Svaki indeks broda je element tačno jedne podliste pa je ukupan broj elemenata podlisti takođe l.

Tabela 5.2 sadrži ulazne podatke za primer sa $m = 8$ vezova, $l = 6$ brodova i $T = 15$ vremenskih jedinica. Reprezentacija jedinke kojoj odgovara optimalno rešenje ovog primera je:

$$Individual\_2{:}\{\{\}, \{3, 1\}, \{4\}, \{5\}, \{2\}, \{6\}\}.$$

U reprezentaciji jedinke $Individual\_2$, indeksi brodova su podeljeni u 6 podlisti jer je $l = 6$. Za razliku od reprezentacije rešenja korišćene u EA pristupu za MCHBAP, cGA reprezentacija rešenja za DMCHBAP ne sadrži informaciju o indeksu veza na koji će brodovi biti alocirani, već definiše grupe brodova koje algoritam istovremeno alocira i pri tome minimizuje njihove ukupne troškove alokacije. Na osnovu reprezentacije jedinke $Individual\_2$, može se videti da su brodovi 3 i 1 prvi smešteni u luku jer pripadaju prvoj nepraznoj podlisti. Nakon toga se alocira brod

Tabela 5.2: Ulazni podaci za DMCHBAP $m = 8$, $l = 5$ i $T = 15$

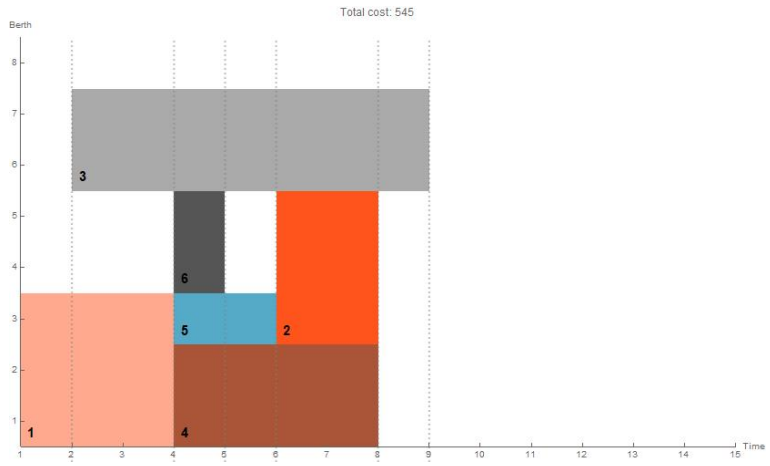
$vessel_k$	ETA_k	a_k	b_k	d_k	LTD_k	s_k	c_{1k}	c_{2k}	c_{3k}	c_{4k}
1	1	9	3	4	15	1	10	20	20	25
2	4	6	3	6	15	3	10	20	20	25
3	2	14	2	11	15	6	10	20	20	25
4	2	8	2	4	15	7	10	20	20	25
5	4	2	1	5	15	2	10	20	20	25
6	4	1	2	5	15	2	10	20	20	25

4 i to tako što se smešta na najeftiniju preostalu poziciju, a zatim se ista strategija primenjuje na brodove 2 i 6.

Slika 5.7 ilustruje optimalno rešenje

$$\{\{1,1,90\},\{3,6,150\},\{6,2,70\},\{1,4,180\},\{3,4,25\},\{4,4,30\}\}$$

DMCHBAP-a dobijeno dekodiranjem jedinke *Individual_ 2*.



Slika 5.7: Rešenje dobijeno dekodiranjem jedinke *Individual_ 2*

Sa slike 5.2 se vidi da brodovi 3 i 1 imaju referentne tačke koje odgovaraju predviđenom vremenu dolaska i omiljenoj poziciji. Ovakva alokacija je kreirana jer su brodovi 3 i 1 elementi prve neprazne podliste, tako da su imali prioritet kod smeštanja na vezove. Kako idealnu referentnu tačku broda 4 blokira brod 3, brod 4 je smešten na najjeftiniju preostalu poziciju. Sledi alokacija broda 5, kojem brod 4 blokira idealnu poziciju, pa je brod 5 smešten na nešto skuplju referentnu tačku na vezu 3 u odnosnu na njegovu idelnu poziciju. Na kraju, vezovi se dodeljuju za brodove 2 i 6, koji su takođe smešteni na preostale dopustive najjeftinije pozicije.

U većini slučajeva, reprezentacija jedinki koju koristi cGA omogućava, dekodiranje jedinke u dopustivo DMCHBAP rešenje. Kako pozicija veza nije nametnuta reprezentacijom jedinke, retko se dešava da na nekom od vezova nema dovoljno slobodnog prostora za alokaciju broda. Ukoliko se nekorektna jedinka ipak pojavi, algoritam je eliminiše iz populacije. Dekodiranje korektnih jedinki u dopustivo rešenje DMCHBAP-a je jedinstveno, dok jednom dopustivom rešenju može da odgovara više različitih korektnih jedinki.

Faza inicijalizacije

U fazi inicijalizacije, procedura InitGA formira početnu cGA populaciju sa nGA korektnih jedinki, pri čemu svaka jedinka odgovara jednom dopustivom DMCHBAP rešenju. Osim toga, u fazi inicijalizacije, algoritam kreira Ψ-listu sa odgovarajućim uređenim trojkama (vez, vreme, trošak) za sve dopustive pozicije brodova. Pri formiranju jedinki, procedura bira brodove na slučajan način ruletskom selekcijom na osnovu prioriteta definisanog kao linearna kombinacija ETA parametra, veličine pravougaonika pridruženog brodu u dvodimenzionalnoj ravni, i prosečnih troškova elemenata ξ-liste posmatranog broda. Koeficijenti linearne kombinacije su redom λ_1, λ_2, i λ_3. Pozicije alokacije brodova su takođe određene stohastički: pozicije sa manjim troškovima imaju veću šansu da budu izabrane.

Operator selekcije

Nova cGA generacija formirana je stacionarnom zamenom jedinki u populaciji, pri čemu je korišćena elitistička strategija. Preciznije, trećina najkvalitetnijih jedinki populacije (elitne jedinke) se direktno prosleđuje u sledeću generaciju. Preostalih $\frac{2}{3}nGA$ jedinki se biraju fino gradiranom turnirskom selekcijom, što obezbeđuje raznovrsnost genetskog materijala. Nekorektne jedinke eliminisane su iz populacije postavljanjem vrednosti ukupnih troškova jedinke na beskonačno. Operator selekcije ne dozvoljava duplikatima jedinki da pređu u sledeću generaciju, što doprinosi očuvanju raznovrsnosti genetskog materijala i sprečavanju preuranjene konvergencije algoritma.

Faza mutacije

Faza mutacije je implementirana procedurom Mutate, koja primenjuje na jedinke dva tipa mutacije: *promeni podlistu* i *pomeri podlistu*. Oba operatora muta-

cije primenjena su na gene jedinke sa promenljivom verovatnoćom μGA. Mutacija *promeni podlistu* na slučajan način bira brod i podlistu, a zatim odabrani brod prebacuje na poslednju poziciju u podlisti. Za razliku od prvog tipa mutacije, mutacija *pomeri podlistu* se primenjuje na slučajno odabranu podlistu. Algoritam premešta selektovanu podlistu na prvu poziciju liste, čime se obezbeđuje da elementi podliste imaju prioritet u dodeli vezova.

Verovatnoća primene mutacije *promeni podlistu* opada po formuli

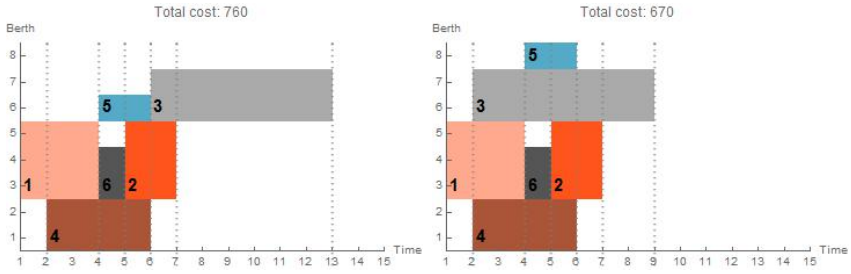
$$\mu GA = 1 - 0.9 \cdot \frac{popID}{nGen}, \tag{5.3}$$

dok verovatnoća primene mutacije *pomeri podlistu* raste na osnovu vrednosti

$$\mu GA = 0.9 \cdot \frac{popID}{nGen}. \tag{5.4}$$

Oba tipa mutacije obezbeđuju promene prostornih i vremenskih koordinata brodova i mogu značajno da utiču na promene vrednosti funkcije prilagođenosti. Pri tome, ovi operatori mutacije proizvode samo nekoliko procenata nekorektnih jedinki. Jedinke dobijene nakom faze mutacije se evaluiraju, odnosno računa im se vrednost funkcije prilagođenosti koja je jednaka funkciji cilja.

Slika 5.8 ilustruje rezultat primene operatora mutacije *promeni podlistu*. Ovaj operator smešta slučajno odabrani brod u slučajno odabranu podlistu. U navedenom primeru, brod 3 je iz druge podliste premešten u prvu podlistu koja je inicijalno bila prazna. Promena podliste omogućava da brod 3 bira poziciju nezavisno od brodova 2, 6 i 4. Osim toga, brod 3 će biti alociran pre brodova 2, 6 i 4. Nakon dodele veza brodu 3 i smeštanja brodova 2, 6 i 4 na odgovarajuće vezove, alocirani su i brodovi 1 i 5. Brodovi 1 i 5 posmatrani su istovremeno jer pripadaju istoj podlisti.

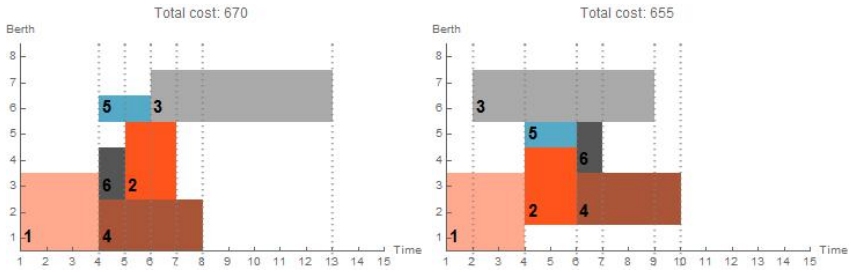


Pre mutacije: $\{\{\mathbf{\}}, \{2, 6, 4, \mathbf{3}\}, \{\}, \{\}, \{1, 5\}, \{\}\}$

Posle mutacije: $\{\{3\}, \{2, 6, 4\}, \{\}, \{\}, \{1, 5\}, \{\}\}$

Slika 5.8: Primer mutacije *promeni podlistu*

Slika 5.9 ilustruje efekat mutacije *pomeri podlistu*. U ovom primeru, podlista koja sadrži brod 3, premeštena je sa poslednje na prvu poziciju liste. Na taj način je brod 3 dobio prioritet kod dodele vezova. Nakon dodele pozicije brodu 3, alocirani je prvo brod 2 a nakon njega brod 1, jer pripadaju drugoj i trećoj podlisti. Brodovi 5 i 4 su razmotreni istovremeno kao elementi četvrte podliste i smešteni su na neke od dopustivih pozicija koje su preostale nakon alokacije brodova 3 i 1. Poslednji elemenat podliste, brod 6, alociran je na najpogodniju preostalu poziciju.



Pre mutacije:
$\{\{2\},\{1\},\{5,4\},\{\},\{6\},\mathbf{\{3\}}\}$

Posle mutacije:
$\{\{3\},\{2\},\{1\},\{5,4\},\{\},\{6\}\}$

Slika 5.9: Primer mutacije *pomeri podlistu*

Broj gena na koje su primenjena oba operatora mutacije može biti l, k ($1 \leq k \leq l$) ili 1. Na taj način su dobijene tri varijante GA algoritma, označene sa GA-l, GA-k i GA-1. Predloženi cGA nastaje komninovanjem ove tri varijante na sledeći način. U prvoj trećini od maksimalnog broja cGA iteracija (od iteracije 1 do $\frac{1}{3}nGen$) primenjuje se GA-l varijanta. U drugoj trećini generacija primenjen je GA-k, gde je k slučajan celi broj iz intervala $[1, l]$. U poslednjoj trećini ukupnog broja generacija primenjena je varijanta GA-1. Kombinovanjem ove tri varijante GA, omogućena je intenzivnija diversifikacija pretrage na početku izvršavanja cGA algoritma, dok se u kasnijim iteracijama algoritma favorizuje detaljnija pretraga okolina dobrih rešenja.

Faza optimizacije

Predloženi cGA sadrži dve dodatne faze: fazu kreiranja *statističke jedinke* i fazu poboljšanja pojedinih jedinki.

Sledeći ideje predložene u radovima [98, 165], na deo populacije, odnosno, na tačno $nImpr$ slučajno odabranih jedinki, primenjena je faza poboljšanja. Faza poboljšanja jedinke sadrži dve procedure: u prvoj je dozvoljena promena veza za oda-

brani brod, dok druga procedura izvršava lokalnu pretragu koja dozvoljava samo promene redosleda alokacije brodova na odabranom vezu.

Statistička jedinka ima ulogu prenošenja znanja formiranog prikupljanjem informacija u prethodnim generacijama cGA algoritma. Ova veštački formirana jedinka sadrži podskupove brodova smeštene na pozicije sa najvećim frekvencijama. Frekvencije za pozicije svakog broda izračunate su u odnosu na sva prethodno formirana dopustiva rešenja. Preciznije, da bi statistička jedinka *statInd* bila formirana, za svaki brod se najpre identifikuje najfrekventnija referentna tačka iz svih prethodnih cGA generacija, uključujući i trenutnu generaciju *tempPop*1. Brod je označen kao *rešen* ako je alociran na istu referentnu tačku u 85% dopustivih rešenja koja odgovaraju korektnim jedinkama u populaciji. Statistička jedinka je kreirana formiranjem liste jednoelementnih podlista na osnovu skupa *rešenih* brodova. Podliste koje odgovaraju *rešenim* brodovima postavljene su na početak statističke jedinke. Primenom ove strategije, *rešeni* brodovi imaju veći prioritet i mogućnost da u sledećoj generaciji ponovo budu smešteni na najfrekventniju referentnu tačku.

Brodovi čije najzastupljenije referentne tačke imaju frekvenciju manju od 85% označeni su kao *nerešeni* brodovi. *Nerešeni* brodovi se na osnovu međusobne konfliktnosti grupišu u podliste koje se smeštaju na kraj jedinke. Postupak alokacije podliste u koju su smešteni *nerešeni* brodovi je isti kao u fazi formiranja početne populacije.Elementi podliste sa više brodova sortirani su u nerastući redosled prosečnih troškova dodele vezova u trenutnoj populaciji. Redosled podlisti u strukturi statističke jedinke dobijen je sortiranjem u neopadajući redosled prosečne vrednosti *ETA* za brodove podliste. Statistička jedinka se uključuje kao poslednja jedinka u trenutnu populaciju, pod uslovom da jedinka sa istom reprezentacijom već ne postoji u populaciji.

Nakon kreiranja statističke jedinke, sledi faza popravke rešenja. Ulazni parametar $nImpr$ određuje broj jedinki čija rešenja algoritam popravlja u svakoj generaciji. $nImpr$ jedinki nad kojima je primenjena procedura popravke rešenja selektovane su turnirskom selekcijom male konstantne veličine [145]. Nakon selekcije jedinki za fazu popravke, cGA pokušava da optimizuje alokaciju brodova definisanu strukturom odabranih jedinki.

U reprezentaciji jedinke, brodovi su sortirani u nerastućem redosledu troškova alokacije. Taj redosled ujedno definiše redosled u kojem algoritam bira brodove u postupku popravke rešenja. Za selektovani brod, na osnovu ξ-liste, algoritam najpre određuje sve pozicije sa nižim troškovima alokacije u odnosu na poziciju do-

deljenu dekodiranjem jedinke. Nakon toga, za odabrani brod i svaku poziciju sa nižim troškovima alokacije, algoritam formira grupe konfliktnih brodova. Odabrani brod se posmatra sa prvom grupom konfliktnih brodova i, ukoliko je moguće, formira se nova dodela vezova koja smanjuje ukupne troškove posmatranog podskupa brodova. U slučajevima kada realokacija podskupa brodova nije moguća, prelazi se na sledeću formiranu grupu konfliktnih brodova. Postupak se ponavlja sve dok se za l brodova ne ispitaju sve moguće pozicije sa nižim troškovima i sve grupe konfliktnih brodova. U ovom postupku, pojedini brodovi menjaju referentne tačke, dok ostali ostaju fiksirani na prethodno dodeljenim pozicijama. Zbog promene pozicije brodova, struktura podlisti jedinke može biti promenjena. Proces popravke rešenja se završava adekvatnim transformacijama reprezentacije poboljšane jedinke. Preciznije, ako je za neku grupu brodova određen nov raspored sa manjim ukupnim troškovima, date brodove je potrebno istovremeno razmatrati u procesu alokacije i zato oni postaju elementi jedne podliste.

Na svaku poboljšanu jedinku, primenjuje se procedura lokalne pretrage. U toku lokalne pretrage, najpre se proverava da li je moguće redukovati vrednost funkcije cilja novim rasporedom brodova na nekom od vezova. Nov raspored je prihvatljiv samo ako ne proizvodi dodatne konflikte sa brodovima raspoređenim na susedne vezove. Sve poboljšane jedinke zahtevaju ponovnu evaluaciju, prilikom koje se dekodiraju, i za njih se računa funkcija prilagođenosti jednaka vrednosti funkcije cilja. Privremenim smeštanjem poboljšanih jedinki u promenljivu $tempPop2$ izbegnuto je nepotrebno računanje funkcije prilagođenosti za jedinke koje nisu menjale strukturu.

5.4 Implementacija metode optimizacije kolonijom pčela za MCHBAP

Konstruktivna varijanta BCO algoritma je prvi put primenjena na optimizacione probleme pomorskog transporta u radu [97]. Rešavanje MCHBAP-a zahteva formiranje populacije od B veštačkih pčela koje generišu dopustivo rešenje problema u NC koraka. Osim toga, predloženi konstruktivni BCO algoritam sadrži tri tehnike popravke kompletnog rešenja. U nastavku odeljka dati su detalji implementacije tehnika za popravku rešenja, kao i detalji svih ostalih faza predloženog BCO algoritma primenjenog na rešavanje MCHBAP-a.

Detalji implementacije

Jedna iteracija BCO algoritma za rešavanje MCHBAP-a sastoji se od nekoliko koraka i predstavljena je algoritmom 16. Na početku BCO iteracije, svakoj pčeli se dodeljuje prazno rešenje i kreiraju se odgovarajuće Ψ-liste. U procesu konstrukcije parcijalnog rešenja, u toku leta unapred, svaka pčela donosi sledeće dve odluke: odabira $\lceil \frac{l}{NC} \rceil$ brodova i vrši izbor dopustivih pozicija iz ξ-liste za selektovane brodove.

Algorithm 16 Iteracija BCO algoritma za MCHBAP

```
procedure ITERATION(B, NC)
    for i ← 1, B do
        BeeΨ(i) ← startΨ
        BeeSol(i) ← {}
        BeeCost(i) ← ∞
    end for
    for u ← 1, NC do
        for b ← 1, B do
            for j ← 1, ⌈l/NC⌉ do
                v ← SELECTV(vessels)
                pos ← SELECTP(BeeΨ(b))
                BeeSol(b, v) ← pos
                BeeΨ(b) ← UPDATELIST(BeeΨ(b))
            end for
        end for
        for b ← 1, B do
            BeeCost(b) ← CALCULATECOST(BeeSol(b))
        end for
        RECRUITINGPROCESS(B)
    end for
end procedure
```

Procedura SELECTV bira brodove na osnovu prioriteta definisanog linearnom kombinacijom čiji su koeficijenti λ_i, $i = 1, 2, 3$, za koje važi $\sum_{i=1}^{3} \lambda_i = 1$. Vrednosti ovih koeficijenata određeni su eksperimentalno. Novi brodovi koje procedura dodaje u parcijalno rešenje, selektovani su ruletskom selekcijom na osnovu prioriteta, procedurom SELECTV. Pozicija alokacije odabranog broda selektovana je iz skupa svih dopustivih pozicija na osnovu njihovih troškova. Pozicije koje imaju niže troškove imaju veću verovatnoću izbora. Svaka pčela izračunava verovatnoću izbora pozicije iz ξ-liste i bira jednu od dopustivih pozicija. Pozicija broda se dobija procedurom SELECTP korišćenjem slučajno generisanog broja i ruletske selekcije. Nakon fiksiranja selektovanog broda na odabranu poziciju, procedura UPDATELIST ažurira ξ-liste preostalih brodova koji još nisu alocirani na vezove.

Opisani proces selekcije broda i odabir pozicija se ponavlja $\lceil \frac{l}{NC} \rceil$ puta za svaku pčelu, čime se kompletira faza leta unapred. Vrednosti parametara NC i B su određene preliminarnim eksperimentima.

Na početku leta unazad, procedura CalculateCost računa vrednost ukupnih troškova za rešenje svake pčele. Na osnovu kvaliteta formiranog rešenja, pčela donosi odluku da li će ostati lojalna svom rešenju i regrutovati neopredeljene pčele, ili će postati neopredeljena i odbaciti svoje formirano rešenje. Pčela postaje nelojalna svom rešenju ukoliko ne može da formira dopustivo parcijalno rešenje, odnosno kada selektovani brod nema ni jednu dopustivu poziciju u ξ-listi. Ako parcijalno rešenje pčele vodi ka nedopustivoj dodeli vezova, pčela postaje neopredeljena, što je implemetirano postavljanjem vrednosti funkcije cilja pridruženog rešenja na beskonačno. Pčele koje formiraju rešenja sa nižim ukupnim troškovima sa većom verovatnoćom ostaju lojalne svom rešenju. Na početku novog leta unazad, b-ta pčela ostaje lojalna svom rešenju sa verovatnoćom

$$p_b^{u+1} = e^{-\frac{1-O_b}{u}} , \qquad b = 1, 2, ..., B , \tag{5.5}$$

gde je O_b normalizovana vrednost ukupnih troškova rešenja b-te pčele, dok u označava redni broj leta unapred ($u = 1, 2, \ldots, NC$). Vrednost O_b dobijena je po formuli:

$$O_b = \begin{cases} \frac{C_{max}-C_b}{C_{max}-C_{min}} , & \text{if } C_{max} \neq C_{min} , \\ 1 , & \text{if } C_{max} = C_{min} , \end{cases} \qquad b = 1, 2, ..., B , \tag{5.6}$$

gde su C_{min} i C_{max} minimalna i maksimalna vrednost ukupnih troškova rešenja, formiranih u letu unapred.

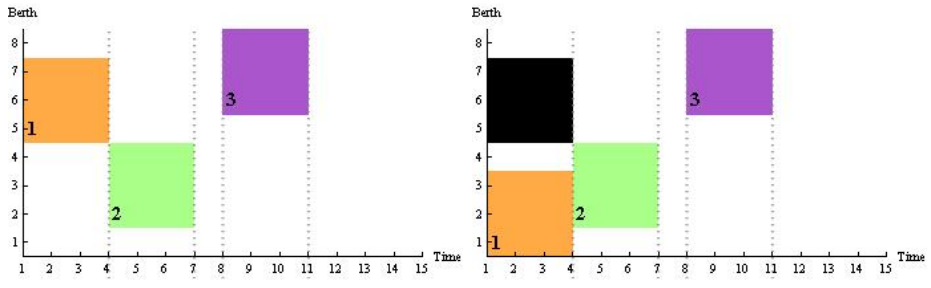
Kao što je opisano u odeljku 2.2, na osnovu verovatnoće lojalnosti i izbora slučajnog broja na intervalu [0,1] pčela odlučuje da li će ostati lojalna svom rešenju ili ne, na sledeći način: ako je vrednost slučajno izbranog broja manja od verovatnoće lojalnosti, pčela će ostati lojalna, u suprotnom postaje neopredeljena. U okviru procedure RecruitingProcess, primenom ruletske selekcije svaka neopredeljena pčela bira nov raspored brodova. Naime, u procesu regrutovanja neopredeljena pčela bira neki od rasporeda brodova lojalnih pčela i preuzima ga kao sopstveno parcijalno rešenje. Lojalne pčele sa kvalitetnijim rešenjem, tj. rešenjem kojem odgovaraju niži ukupni troškovi alokacije brodova, sa većom verovatnoćom regrutuju nelojalne pčele. Neopredeljene pčele preuzimaju rešenja lojalnih pčela ruletskom selekcijom na osnovu verovatnoća izračunatih formulom (2.2). Osim toga, neopredeljena pčela preuzima i Ψ-listu lojalne pčele. BCO algoritam nastavlja fazom leta unapred ukoliko rešenja nisu kompletna, odnosno ako neki brodovi još nisu alocirani na vezove.

Faze leta unapred i leta unazad se smenjuju NC puta sve dok pčele ne formiraju kompletna rešenja. Nakon NC koraka, neke pčele su konstruisale kompletna

(dopustiva) rešenja, dok su ostale pčele formirale neadekvatne rasporede brodova koji blokiraju sve dopustive pozicije za neraspoređene brodove. Najbolje dopustivo rešenje sa najmanjim ukupnim troškovima algoritam koristi za ažuriranje najboljeg globalnog rešenja, čime je kompletirana jedna BCO iteracija. U BCO algoritam za MCHBAP, implementirane su dodatne procedure koje imaju za cilj popravku rešenja, a koje su opisane u nastavku teksta. Algoritam izvršava BCO iteracije i procedure popravke rešenja dok se ne ispuni kriterijum zaustavljanja, koji je definisan kao maksimalno vreme izvršavanja algoritma. Globalno najbolje BCO rešenje se proglašava za rešenje MCHBAP-a.

Tehnike popravke kompletnog BCO rešenja

U cilju poboljšanja kvaliteta formiranog BCO rešenja, algoritam je proširen fazom popravke kompletnog rešenja. Na kraju svake BCO iteracije, algoritam primenjuje prvu proceduru popravke na svako kompletno rešenje. Svaka pčela koja je formirala kompletno rešenje pokušava da ga popravi koristeći svoju finalnu ξ-listu. Prolazeći kroz listu brodova, pčela traži brod sa najvećim troškom alokacije i nepraznom ξ-listom koja sadrži element sa nižim troškovima dodele veza. Ako za brod v_k, $k \in \{1, ..., l\}$ takva pozicija postoji, pčela pomera brod na prvu (najjeftiniju) poziciju u postojećoj ξ-listi. Nakon pomeranja broda na povoljniju poziciju, pčela detektuje nedopustive pozicije preostalih brodova nastale zbog nove situacije u luci. Nedopustive pozicije se brišu iz Ψ-liste. Isti postupak se primenjuje na sve brodove za koje postoje povoljnije pozicije u postojećoj Ψ-listi. Usled promene referentnih tačaka pojedinih brodova, neke od pozicija koje su bile prethodno zauzete sada su oslobođene za novu alokaciju. Međutim, u ovoj fazi se, zbog kompleksnosti postupka, slobodne pozicije ne vraćanja u Ψ-listu.



Pre popravke Posle popravke

Slika 5.10: Ilustracija prve popravke rešenja

Algoritam primenjuje proceduru prve popravke rešenja na svaki brod. Nakon izvršavanja ove procedure neka kompletna rešenja su popravljena dok neka ostaju nepromenjena. Za svako od formiranih kompletnih rešenja, algoritam računa vrednosti funkcije cilja i rešenje sa najmanjim ukupnim troškovima alokacije proglašava za najbolje globalno rešenje.

Slika 5.10 ilustruje primer alokacije brodova pre i posle primene prve popravke kompletnog rešenja. Pozicije koje nisu oslobođene za ponovnu alokaciju brodova su označene crnom bojom. Nakon pomeranja broda 1, pozicije koje je ovaj brod prethodno zauzimao ostaju neiskorišćene i pored toga što su suštinski slobodne za novu alokaciju. Kako Ψ-lista nije ažurirana ovim pozicijama, ostali brodovi oslobođene pozicije ne mogu razmotriti za potencijalnu realokaciju. Neka je za brod 3 pozicija na vezu 6 u vremenskom intervalu 3 jeftinija od njegove trenutne pozicije. Da bi brod 3 mogao zauzeti tu poziciju, potrebno je prvo ažurirati Ψ-listu sa svim oslobođenim pozicijama nastalim realokacijom broda 1.

Algorithm 17 Prva popravka rešenja

```
procedure FirstImprovement(BeeΨ, BeeSol)
    Sort(vessels)
    for i ∈ vessels do
        if ExistsSmallerCost(BeeSol(i)) then
            BeeSol ← UpdateSol(BeeSol(i), BeeΨ(i, 1))
        end if
    end for
    BestSolUpdate()
end procedure
```

Pseudokod algoritma 17 prikazuje korake prve popravke rešenja. Procedura FirstImprovement primenjena je na kompletno rešenje svake pčele. Brodovi su najpre sortirani procedurom Sort u nerastućem poretku prema visini troškova. Procedura ExistsSmallerCost vraća vrednost *True* ako lista $Bee\xi$ broda i, sadrži bar jedan element sa nižim troškom alokacije u odnosu na troškove trenutne pozicije. U slučaju da je pronađena povoljnija pozicija za posmatrani brod, procedura UpdateSol ažurira rešenje na adekvatan način. Kao poslednji korak faze prve popravke, u okviru procedure BestSolUpdate, vrednost promenljive *GlobalBest* se ažurira formiranim rešenjem *Solution*.

Procedura FirstImprovement za svaki brod v_k, $k = 1, \ldots, l$ ispituje samo prvu (najjeftiniju) poziciju iz postojeće ξ-liste. Za brod v_k i prvi elemenat ξ-liste, potrebno je proveriti da li postoji konflikt sa preostalih v_i, $i = 1, \ldots, l$, $i \neq k$ brodova. Dakle, prva popravka rešenja se izvršava kroz dve petlje najviše $l(l-1)$ puta, što znači da složenost procedure FirstImprovement iznosi $O(l^2)$.

Neka su brodovi v_k, $k \in \{k_1, \ldots, k_i\}$, $1 \leq i \leq l$, koji su inicijalno bili smešteni na pozicije p_k, prvom popravkom rešenja pomereni na pozicije n_k. Pozicije p_k su slobodne i moguće je da su neke od njih jeftinije od trenutnih pozicija pojedinih brodova. Generalno, Ψ-lista ima mnogo elemenata i često ažuriranje njenog sadržaja može biti vremenski zahtevno. U cilju očuvanja efikasnosti algoritma, Ψ-lista nije korigovana slobodnim pozicijama u okviru prve popravke rešenja. Ideja vraćanja slobodnih pozicija u Ψ-listu je implementirana u drugoj popravci rešenja. Ova procedura je primenjena samo na rešenje *Solution* sa najmanjom vrednošću troškova *GlobalBest*.

Da bi pozicije p_k iz rešenja *Solution* postale dostupne ostalim brodovima, potrebno je vratiti elemente Ψ-liste koji sadrže pozicije p_k. Zbog kompleksnosti postupka, umetanje elemenata u Ψ-listu je izbegnuto kreiranjem nove alokacije brodova. Brodovi v_k, $k \in \{k_1, \ldots, k_i\}$ alocirani su na pozicije n_k, dok su preostali brodovi v_k, $k \in \{1, \ldots, l\} \setminus \{k_1, \ldots, k_i\}$ fiksirani na pozicije p_k. Alokacija startuje od inicijalne Ψ-liste. Svaki brod v_k, $k \in \{1, \ldots, l\}$ je alociran ili na poziciju n_k ili na poziciju p_k i pri tome su redukovane ξ-liste ostalih brodova v_n, $n \neq k$. Za svaki brod v_n, procedura proverava svaki element odgovarajuće ξ-liste u odnosu na mogući konflikt sa dodeljenom pozicijom broda v_k. Konfliktne pozicije se brišu iz ξ-liste. Proces se iterativno ponavlja za sve brodove i njihove fiksirane pozicije iz globano najboljeg rešenja. Nakon postupka prečišćavanja ξ-listi uklanjanjem konfliktnih elemenata, Ψ-lista je potpuno ažurirana i pozicije brodova su zapamćene kao rešenje *Solution*. Postupak prečišćavanja Ψ-liste najčešće vodi do oslobađanja jeftinijih pozicija za neke brodove i omogućava primenu druge popravke rešenja.

U cilju smanjenja ukupnog vremena izvršavanja algoritma, druga popravka rešenja se primenjuje nakon NC iteracija. BCO algoritam poziva proceduru za drugu popravku rešenja nakon što izvrši NC iteracija, a zatim ažurira vrednosti promenljivih *Solution* i *GlobalBest*. Brodovi su sortirani u nerastući redosled troškova alokacije u odnosu na rešenje *Solution*. Algoritam razmatra promenu pozicije za prvi brod, koji ujedno predstavlja brod sa najvećim troškovima. Brod menja poziciju ako postoji dopustiva jeftinija pozicija od trenutne pozicije sačuvane u *Solution*. Nakon pomeranja broda algoritam ažurira Ψ-listu i ponovo sortira brodove. U slučaju da za posmatrani brod ne postoje jeftinije pozicije, njega je nemoguće realocirati, pa algoritam razmatra sledeći brod iz sortirane liste brodova. Algoritam ponavlja opisanu proceduru sve dok je moguće formirati rešenje sa manjim ukupnim trškovima, odnosno, dok se bar jedan brod može realocirati.

Pseudokod druge popravke rešenja dat je algoritmom 18. Procedura SecondImprovement započinje postavljanjem vrednosti Ψ-liste na inicijalnu vrednost i pozivom procedure Sort koja sortira brodove u nerastući redosled u odnosu na njihove troškove alokacije. Procedura PossibleToMove vraća indeks prvog broda iz sortirane liste brodova, za koji postoji pozicija sa manjim troškovima. Uloga procedure FixAndClean je pomeranje posmatranog broda na poziciju sa manjim troškovima i kompletno ažuriranje Ψ-liste, kao i odgovarajućeg rešenja *Solution*.

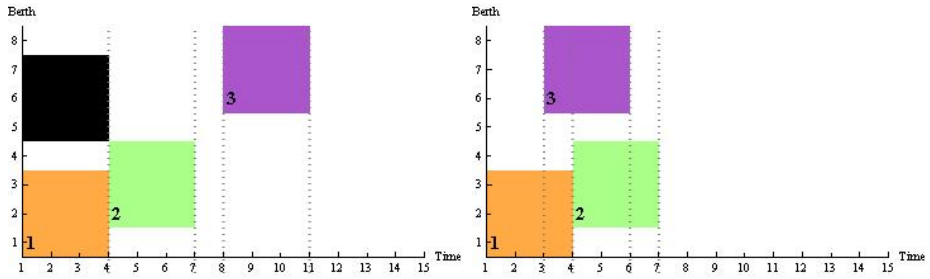
Algorithm 18 Druga popravka rešenja

```
procedure SecondImprovement( )
    repeat
        Ψ ← startΨ
        Sort(vessels)
        V_m ← PossibleToMove(vessels)
        if V_m ≠ {} then
            FixAndClean(Ψ)
        end if
    until V_m = {}
    BestSolUpdate()
end procedure
```

Iz strukture algoritma 18 može se videti da procedura SecondImprovement sadrži dodatne korake ažuriranja ξ-lista. Ovo ažuriranje zahteva dodatnih $l(l-1)$ operacija. Slično kao u slučaju procedure FirstImprovement, potrebno je izvršiti $l(l-1)$ operacija da bi se za svaki brod i svaku prvu poziciju ξ-liste proverio mogući konflikt sa preostalim brodovima. U dopustivom rešenju, svaki brod ima poziciju trenutne alokacije kojoj odgovara neki trošak. Iz ξ-liste je potrebno selektovati podskup jeftinijih pozicija i proveriti moguće konflikte. U najgorem slučaju, broj jeftinijih pozicija iznosi $mT-1$, što znači da vremenska složenost procedure SecondImprovement iznosi $O((l(l-1)+l(l-1)) \cdot (mT-1)) = O(l^2mT)$.

Primer dodele vezova pre i posle primene druge popravke rešenja ilustrovan je na slici 5.11. Nakon izvršene prve popravke i realokacije broda 1, neke pozicije (označene crnom bojom) ostaju nedostupne preostalim brodovima. Procedura druge popravke ažurira Ψ-listu i za brod 3 oslobađa jeftiniju poziciju na vezu 6 u vremenskom segmentu 3. Sada brod 3 može zauzeti tu kvalitetniju poziciju sa manjim troškovima. Ψ-lista ostaje potpuno ažurirana nakon svih realokacija.

Treća tehnika popravke rešenja primenjuje se na najbolje rešenje, pri svakom smanjenju vrednosti promenljive *GlobalBest*. Procedura treće popravke za svaki brod ispituje sve nedopustive pozicije sa manjim troškovima u odnosu na troškove pozicije koja mu je trenutno dodeljena. Pozicija je nedopustiva ukoliko je već zauzeta drugim brodom. Za brodove koji blokiraju jeftiniju poziciju, kaže se da su u *konfliktu*

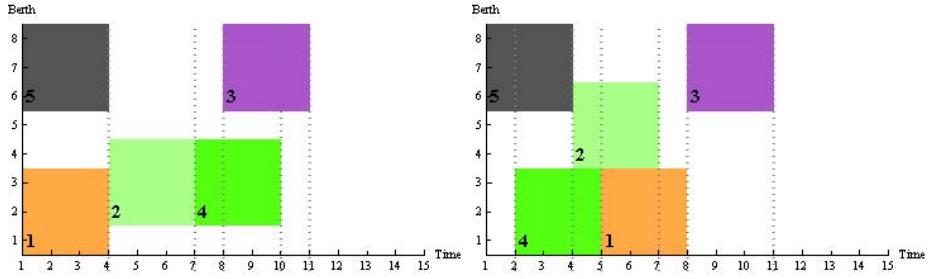


Pre popravke Posle popravke

Slika 5.11: Ilustracija druge popravke rešenja

sa posmatranim brodom. Procedura treće popravke rešenja ima zadatak da razreši konflikte, tako da realokacija konfliktnih brodova redukuje ukupne troškove.

Slika 5.12 ilustruje tehniku treće popravke rešenja. Neka je za brod 4 pozicija na vezu 1 u vremenskom intervalu 2 jeftinija od trenutno dodeljene poziciije brodu 4. Međutim, ova jeftinija pozicija je već blokirana brodom 1. Ako bi se pozicija na vezu 1 u vremenskom intervalu 2 razmotrila kao potencijalna referentna tačka broda 4, nastao bi konflikt sa brodovima 1 i 2. Zbog toga, procedura treće popravke rešenja pokušava da realocira brodove 4, 1 i 2, tako da minimizuje ukupne troškove ova tri broda. U ovom primeru, mala pomeranja brodova 1 i 2 omogućavaju pomeranje broda 4 na jeftiniju poziciju i smanjivanje ukupnih troškova ova tri broda.



Pre popravke Posle popravke

Slika 5.12: Ilustracija treće popravke rešenja

Struktura procedure treće popravke rešenja je predstavljena algoritmom 19. Slično kao i kod prve dve popravke rešenja, brodovi su najpre sortirani u nerastući redosled prema troškovima alokacije. Pojedini brodovi su smešteni na pozicije sa visokim troškovima usled postojanja konflikta sa prethodno alociranim brodovima. Ideja treće popravke rešenja je da se razmotre sve jeftinije pozicije i razreše konflikti realokacijom nekih brodova. Procedura ConflictVessels(i) za brod i i sve jefti-

nije pozicije j formira podskup konfliktnih brodova V_j. Generalno, ova pretraga ne zahteva mnogo procesorskog vremena, jer su skupovi V_j male kardinalnosti.

Za svaku poziciju j, procedura istovremeno posmatra brod i i skup V_j. Ideja je dozvoliti mala pomeranja brodova iz V_j na skuplje pozicije, tako da realokacija dovede do smanjenja ukupnih troškova. Pomeranje brodova iz V_j na nove pozicije oslobađa jeftinije pozicije za brod i. Na taj način dolazi do redukovanja ukupnih troškova za brodove iz V_j i brod i u odnosu na njihove ukupne troškove definisane rešenjem *Solution*.

Algorithm 19 Treća popravka rešenja

```
procedure ThirdImprovement( )
    repeat
        Sort(vessels)
        temp ← GlobalBest
        for i ∈ vessels do
            V ← ConflictVessels(i)
            for all V_j do
                movingVessels ← V_j ⋃{i}
                ξ(V_j) ← startΨ(V_j)
                tempAllocation(movingVessels) ← Solve(movingVessels, Ψ)
            end for
            MaxSavings(tempAllocation)
            if Solution ≠ {} then
                BestSolUpdate()
                Break()
            end if
        end for
    until temp = GlobalBest
end procedure
```

Zadatak funkcije Solve(s, Ψ) je da ispita sve jeftinije pozicije iz Ψ-liste i da odredi nove pozicije za brodove u listi s. U slučaju da je neki brod pomeren sa originalne pozicije, nova alokacija je sačuvana u promenljivoj *tempAllocation*, a u suprotnom, *tempAllocation* je prazan skup. Funkcija MaxSavings identifikuje najbolju realokaciju sa najvećom redukcijom ukupnih troškova. Ako su brodovi promenili pozicije, funkcija MaxSavings formira novo rešenje *Solution*, u suprotnom, vraća prazno rešenje. Procedura koristi vrednost promenljive *Solution* za ažuriranje vrednosti *GlobalBest*. Postupak treće popravke se ponavlja dok ima poboljšanja vrednosti *GlobalBest*.

U proceduri treće popravke rešenja, potrebno je za svaki brod v_k, $k = 1, \ldots, l$ odrediti podlistu jeftinijih pozicija od njihove trenutne pozicije. Za svaku jeftiniju poziciju broda v_k, neophodno je $l(l–1)(mT–1)$ operacija za formiranje skupa brodova koji su u konfliktu. U najgorem slučaju, broj jeftinijih pozicija je $mT-1$ i svaki brod je u konfliktu sa preostalih $l-1$ brodova. Dakle, procedura treba da odabere l

pozicija za alokaciju brodova od mogućih $mT-1$ pozicija. Broj različitih alokacija je $l!\binom{mT-1}{l}$. Konačno, vremenska složenost jednog prolaza procedure treće popravke rešenja je $O(l(l-1)(mT-1)+l!\binom{mT-1}{l}) = O(l^2mT + l!\binom{mT-1}{l})$.

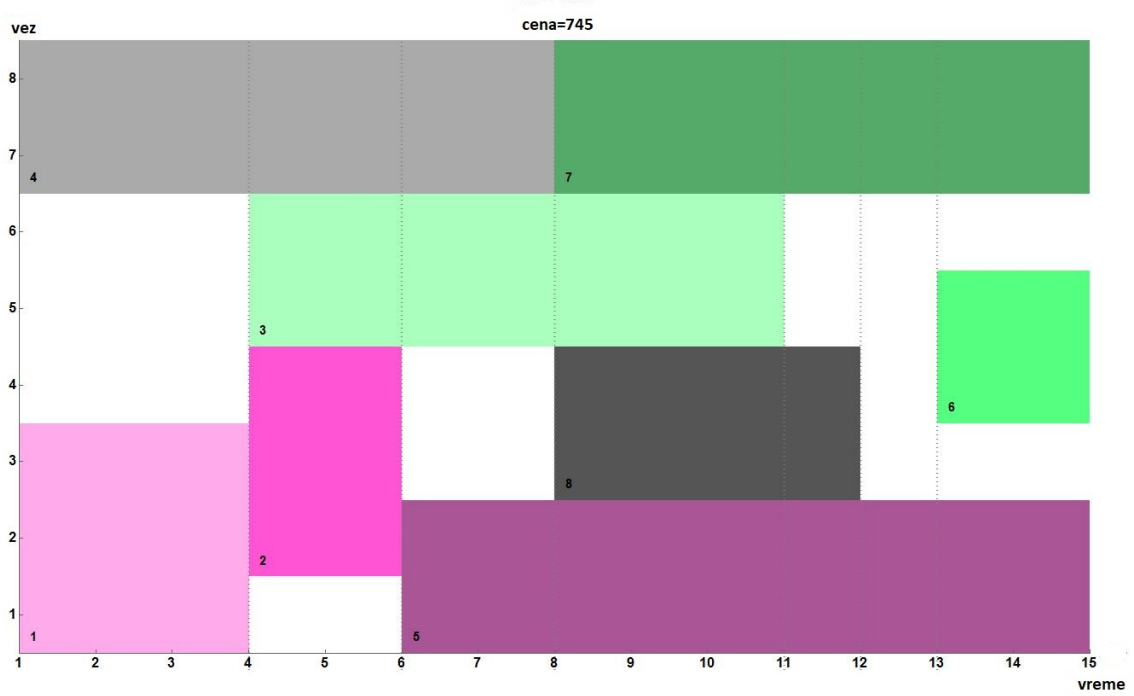
Algorithm 20 Konstruktivni BCO algoritam za MCHBAP

```
procedure BCOforMCHBAP(B, NC, RunTime)
    run ← 1
    GlobalBest ← ∞
    while SessionTime() ≤ RunTime do
        Iteration(B, NC)
        for b ← 1, B do
            FirstImprovement(BeeΨ(b), BeeSol(b))
        end for
        if Mod(run, NC) = 0 then
            SecondImprovement()
        end if
        if Changed(GlobalBest) then
            ThirdImprovement()
        end if
        run ← run + 1
    end while
end procedure
```

Struktura BCO algoritma je prikazana pseudokodom 20. Promenljiva *run* broji iteracije algoritma. Kada algoritam izvrši proceduru ITERATION, završena je jedna iteracija BCO algoritma. Algoritam nastavlja izvršavanje pozivom procedure prve popravke svih kompletnih rešenja. Drugo poboljšanje rešenja primenjeno je samo nad rešenjem kojem odgovara *GlobalBest* nakon NC iteracija. Treća popravka se koristi u cilju daljeg poboljšanja globalno najboljeg rešenja, svaki put kada se ono promeni, odnosno kada se redukuju ukupni troškovi. Posledica je da se treća popravka može primeniti nad globalno najboljim rešenjem odmah nakon prve popravke rešenja. Procedura BCOFORMCHBAP se izvršava do zadovoljenja kriterijum zaustavljanja, koji je u ovoj BCO implementaciji definisan maksimalnim vremenom izvršavanja algoritma.

Imajući u vidu da BCO metoda sadrži stohastičke elemente u procesu pretraživanja, veoma je važno garantovati da je implementirana na adekvatan način [30] pri rešavanju MCHBAP-a. Drugim rečima, potrebno je dokazati da se predloženim BCO algoritmom za MCHBAP može dostići optimalno rešenje u konačnom broju BCO koraka. U nastavku je po prvi put u literaturi dat dokaz da BCO metoda može da formira optimalno rešenje za problem dodele vezova sa verovatnoćom strogo većom od nule.

Svako dopustivo rešenje MCHBAP-a moguće je prikazati u vidu permutacije indeksa brodova, ali obrnuto ne mora da važi i neka je *sol* neko dopustivo rešenje



Slika 5.13: Ilustracija MCHBAP rešenja

MCHBAP-a predstavljeno u dvodimenzionalnoj ravni dimenzija $m \times T$. Neka je sa $v_k^{sol} = (x_k, y_k)$ označena referentna tačka broda v_k, $k = 1, \ldots, l$. Na primer, na slici 5.13, brod sa indeksom 2 je alociran u ravni na referentnu tačku $v_2^{fig1} = (2, 4)$, dok brod 6 ima referentnu tačku $v_6^{fig1} = (4, 13)$.

Definicija 5. Neka je dopustivom rešenju *sol*, dodeljena permutacija indeksa brodova, označena sa $perm(l)^{sol}$. Ova permutacije je generisana u odnosu na $v_k^{sol} = (x_k, y_k)$, $k = 1, \ldots, l$ po sledećim pravilima:

- ako dva broda v_i i v_j, $1 \leq i \leq l$, $1 \leq j \leq l$, $i \neq j$ imaju isto vreme vezivanja, tj., $y_i = y_j$, i $x_i < x_j$, tada i prethodi j u permutaciji;
- ako su dva broda v_i i v_j, $1 \leq i \leq l$, $1 \leq j \leq l$, $i \neq j$ smeštena na isti vez, tj., $x_i = x_j$ i $y_i < y_j$, tada i prethodi j u permutaciji.

Definicija 6. Permutacija je *dopustiva* ako odgovara dopustivoj alokaciji brodova.

Dopustivom rešenju prikazanom na slici 5.13 može se pridružiti permutacija $perm^{fig1}(8) = (1, 4, 2, 3, 5, 8, 7, 6)$ koja ga opisuje. Svako dopustivo rešenje može biti na jedinstven način predstavljeno permutacijom indeksa brodova. Međutim,

obrnuto ne mora da važi, jer jedna permutacija može da se dekodira u više različitih dopustivih alokacija. Osim toga, kod dopustive alokacije brodova, za date indekse brodova i i j, ne može istovremeno da važi $x_i = x_j$ i $y_i = y_j$.

Teorema 1. Predložena konstruktivna varijanta BCO algoritma za MCHBAP može da vrati kao rezultat rešenje koje se poklapa sa optimalnim sa verovatnoćom strogo većom od nule.

Dokaz. Da bi BCO algoritam dostigao optimalno rešenje *opt* za MCHBAP, potrebno je da najmanje jedna pčela generiše optimalnu permutaciju indeksa brodova $perm^{opt}(l)$ i da alocira brodove na optimalne pozicije. Pčela generiše $perm^{opt}(l)$ sa verovatnoćom

$$p(perm^{opt}(l)) = p_p^* \geq \frac{1}{l!} > 0. \tag{5.7}$$

Prva nejednakost je ispunjena jer sve permutacije nisu dopustive za dati MCHBAP.

Neka su sa b_k i $h_k = \lceil a_k/b_k \rceil$ označene redom visina i širina pravougaonika koji predstavlja brod v_k, $k = 1, \ldots, l$. U procesu konstrukcije rešenja, verovatnoća da je brod v_1 alociran na optimalnu poziciju iznosi

$$p(v_1^{opt}) = p_{a_1}^* = \frac{1}{(m - b_1)(T - h_1)} > \frac{1}{mT} > 0. \tag{5.8}$$

Pod pretpostavkom da su brodovi v_j, $1 \leq j < i$ alocirani na optimalne pozicije, uslovna verovatnoća da je brod v_i, $i = 2 \ldots, l$, alociran na optimalnu poziciju iznosi

$$p(v_i^{opt} | v_{i-1}^{opt} \ldots v_2^{opt} v_1^{opt}) = p_{a_i}^* \geq \frac{1}{mT - \sum_{j=1}^{i-1} b_j h_j} > 0. \tag{5.9}$$

Prve nejednakosti u izrazima (5.8) i (5.9) važe, jer sve pozicije u ravni ne mogu biti dopustive za brod v_i. Drugim rečima, brod v_i ne može da izlazi van dimenzija terminala datog pravougaonikom dimenzija $m \times T$.

Verovatnoća generisanja optimalnog rešenja jednaka je proizvodu svih prethodno izračunatih verovatnoća, preciznije važi

$$p^* = p_p^* \prod_{i=1}^{l} p_{a_i}^*. \tag{5.10}$$

Očigledno, p^* ima vrednost veću od nule. Time je dokaz završen. □

5.5 Implementacija verzije optimizacije kolonijom pčela sa poboljšanjem kompletnog rešenja za DMCHBAP

Za razliku od konstruktivne varijante BCO algoritma koji gradi rešenje problema u više iteracija, BCOi algoritam startuje od jednog ili više dopustivih rešenja problema nad kojim(a) primenjuje tehnike popravke. Pseudokod BCOi metode implementirane za DMCHBAP dat je algoritmom 21. Na početku svake BCOi iteracije,

Algorithm 21 BCOi za DMCHBAP

```
procedure BCOi(vessels, B, NC, RunTime)
    while SessionTime() ≤ RunTime do
        BeeSol ← InitialSolution(B, vessels)
        for i ← 1, NC do
            BeeSol ← FirstTransformation(B)
            bestBee ← SmallestCost(B)
            BeeSol ← SecondTransformation(B)
            BeeSol ← RecruitingProcess(B)
        end for
        currentBest ← SmallestCost(B)
        Update(GlobalBest)
        currentBest ← Improve(GlobalBest ∨ currentBest)
        Update(GlobalBest)
    end while
end procedure
```

procedura InitialSolution kostruiše početna rešenja za svaku pčelu. U prvom koraku, procedura svakoj pčeli dodeljuje Ψ-listu i prazno rešenje. U fazi konstrukcije početnog rešenja, pčela donosi dve odluke: selektuje jedan brod i za njega dopustivu poziciju iz ξ-liste. Slično kao u slučaju cGA algoritma, brodovi se biraju stohastički na osnovu prioriteta brodova primenom rulet selekcije. Prioritet je definisan kao linearna kombinacija ETA parametra, veličine pravougaonika koji reprezentuje brod u ravni i prosečnih troškova svih elemenata ξ-liste, sa koeficijentima λ_1, λ_2, and λ_3.

Svi elementi ξ-liste se razmatraju kao potencijalne pozicije za odabrani brod, pri čemu pozicije sa manjim troškovima imaju veću verovatnoću izbora. Pčela računa verovatnoću izbora za svaku dopustivu poziciju ξ-liste. Ruletskom selekcijom i generisanjem slučajnog broja pčela bira jednu poziciju na osnovu izračunatih verovatnoća izbora. Brod se alocira na odabranu poziciju, nakon čega pčela redukuje ξ-liste preostalih brodova eliminacijom nedopustivih pozicija. Svaka pčela ponavlja ovaj proces l puta. Pčele koje ne mogu da formiraju dopustivu alokaciju brodova preuzimaju kompletno rešenje druge pčele u uobičajenom postupku regrutacije [30]. Globalno najbolje rešenje se ažurira najboljim formiranim dopustivim rešenjem u

smislu minimalnih ukupnih troškova, čime se završava faza inicijalizacije BCOi algoritma.

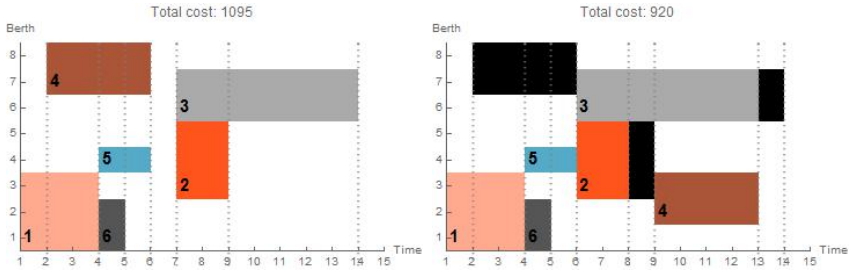
Svaka BCOi iteracija se sastoji od NC faza formiranih od leta *unapred* i leta *unazad*. Let unapred formiraju dve uzastopne modifikacije rešenja: *prva* i *druga transformacija*. Ove modifikacije su detaljno opisane u narednim odeljcima. U toku leta unazad, pčele donose odluke da ostanu lojalne svom rešenju ili da ga odbace i postanu neopredljene. Ove odluke se donose stohastički, ali na osnovu kvaliteta rešenja. Pčela čije rešenje ima niže ukupne troškove ima veću verovatnoću da ostane lojalna svom rešenju. Pčela odmah postaje neopredeljena ako druga transformacija ne popravi njeno rešenje. Neopredeljene pčele preuzimaju rešenje od lojalnih pčela. Odluka o lojalnosti i proces regrutovanja realizuju se na standardni način [30].

Na kraju BCOi iteracije, procedura SmallestCost identifikuje trenutno najbolje rešenje *currentBest* koje se dalje prosleđuje proceduri Update za ažuriranje globalno najboljeg rešenja i vrednosti promenljive *GlobalBest*. U nastavku BCOi algoritma, procedura Improve se primenjuje u cilju popravke globalno najboljeg rešenja. Postupak popravke rešenja počinje sortiranjem brodova u nerastući poredak prema visini troškova alokacije. Za svaki brod i i za svaku jeftiniju poziciju j iz inicijalne Ψ-liste, procedura Improve formira podskupove konfliktnih brodova V_j, a zatim pokušava da realocira brodove V_j i brod i tako da se minimiziraju troškovi alokacije, pri čemu je korišćena ideja treće popravke BCO rešenja. Proces popravke rešenja je deterministički i ponavlja se sve dok je moguće redukovati ukupne troškove. Zbog toga, ako je u prethodnim iteracijama algoritma globalno najbolje rešenje već popravljano, procedura Improve pokušava da popravi trenutno najbolje rešenje *currentBest*. BCOi iteracije i koraci popravke rešenja se izvršavaju dok se ne ispuni kriterijum zaustavljanja, koji je u predloženoj BCOi implementaciji definisan maksimalnim vremenom izvršavanja $RunTime$.

Prva transformacija

Prva transformacija, implementirana procedurom FirstTransformation, na osnovu postojećih ξ-lista popravlja kvalitet dopustivog rešenja svake pčele. Pčela formira skup brodova $V_b = \{v_{k_1}, v_{k_2}, ..., v_{k_n}\}$ tako što identifikuje brodove sa najmanje jednim elementom ξ-liste sa nižim troškovima u odnosu na troškove trenutne pozicije broda. Na osnovu generisanog slučajnog broja $s_b \in [1, k_n]$, pčela formira podskup od s_b slučajno odabranih brodova iz skupa V_b. Za svaki brod iz skupa s_b, pčela na slučajan način bira poziciju sa nižim troškovima iz postojeće ξ-liste, pri

čemu kvalitetnije pozicije imaju veću šansu izbora. Brodovi v_k, $k \in \{k_1, \ldots, k_n\}$, se zatim pomeraju na nove pozicije, a Ψ-lista se ažurira uklanjanjem pozicija koje bi dovele do konflikta brodova. Nakon ažuriranja Ψ-liste i formiranja nove situacije na vezovima, može se desiti da neki od brodova iz skupa s_b više ne mogu da se realociraju. Ovakvi brodovi se ignorišu i prelazi se na realokaciju sledećeg broda iz skupa s_b. U cilju obezbeđivanja dobrih performansi algoritma, u ovoj fazi se ne oslobađaju prethodno zauzete pozicije brodova. Opisani koraci procedure se ponavljaju s_b puta, pri čemu procedura pronalazi bolje rešenje problema, ili postojeće rešenje ostavlja nepromenjeno ako je skup V_b prazan. Ako je V_b prazan jedini način da pčela popravi rešenje je primena druge transformacije. Zbog toga se rešenja sa praznim skupovima V_b direktno prosleđuju proceduri druge transformacije. Nakon toga, procedura SmallestCost upoređuje kvalitet svih rešenja i pronalazi rešenje sa minimalnim troškovima. Pčela koja je uspela da generiše najbolje rešenje se proglašava za *bestBee*.



Pre transformacije: Posle transformacije:

s_b={2,4,3}

Slika 5.14: Primer *prve transformacije* rešenja

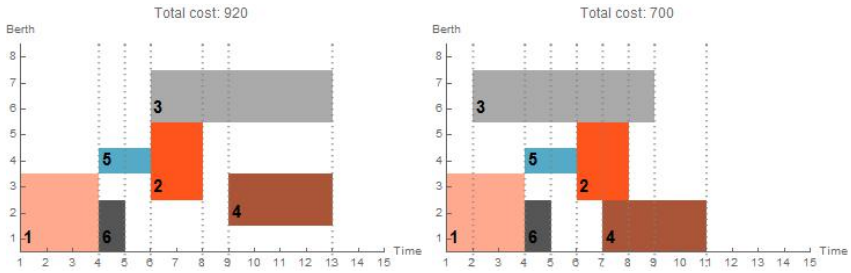
Primer alokacije pre i posle primene prve transformacije prikazan je na slici 5.14. Lista s_b sa elementima 2, 4 i 3 definiše redosled u kojem se brodovi razmatraju za realokaciju. Nakon pomeranja brodova 2, 4 i 3 na nove pozicije, ξ-liste nisu ažurirane na adekvatan način. Zbog toga, stare pozicije ovih brodova (označene crnom bojom) ostaju blokirane i ne mogu biti razmatrane za alokaciju preostalih brodova. Prvobitne pozicije brodova 2, 4 i 3 ostaju neiskorišćene, čak i u slučaju da vode do alokacije sa nižim ukupnim troškovima.

Neka su brodovi v_k, $k \in \{k_1, \ldots, k_i\}$, $1 \leq i \leq l$, pomereni sa inicijalnih pozicija p_k na nove pozicije n_k. Pozicije p_k su prazne i moguće je da sadrže jeftinije pozicije za neke druge brodove. Da bi bilo moguće ponovo iskoristiti pozicije p_k za

alokaciju preostalih brodova, potrebno je ažurirati Ψ-listu i izvršiti drugu transformaciju. Druga transformacija se izvršava samo nad najboljim rešenjem trenutnog leta unapred ili nad rešenjem koje nije moguće popraviti prvom transformacijom.

Druga transformacija

Procedura SecondTransformation počinje ažuriranjem Ψ-liste rešenja koje treba transformisati. Ažuriranjem Ψ-liste, pozicije p_k postaju dostupne za alokaciju ostalih brodova. Zbog jednostavnosti, postupak ažuriranja se svodi na novu alokaciju. Brodovi v_k, $k \in \{k_1, \ldots, k_i\}$, se alociraju na pozicije n_k, a preostali brodovi v_j, $j \in \{1, \ldots, l\} \setminus \{k_1, \ldots, k_i\}$, se fiksiraju na inicijalne pozicije p_j. Nakon alokacije svakog broda, ξ-liste se redukuju, čime se rešenje priprema za drugu transformaciju. Slično kao kod prve transformacije, procedura SecondTransformation formira skup V_b, zatim na slučajan način bira broj $s_b \in [1, |V_b|]$ i selektuje s_b slučajnih brodova. Odabrani brodovi se smeštaju na slučajno odabrane jeftinije pozicije, ako je takva realokacija moguća. Za razliku od procedure FirstTransformation, u proceduri SecondTransformation elementi Ψ-liste se ažuriraju nakon svake promene pozicije brodova.



Pre transformacije: Posle transformacije:

s_b={4,3}

Slika 5.15: Primer *druge transformacije* rešenja

Polazeći od alokacije nastale nakon primene prve transformacije (druga ilustracija na slici 5.14), nakon adekvatnog ažuriranja Ψ-liste, druga transformacija može uspešno da realocira brodove 4 i 3 na prethodno zabranjene pozicije označene crnom bojom. Situacija u luci pre i posle druge transformacije je prikazana na slici 5.15.

Slično kao i u slučaju kontruktivne verzije BCO algoritma, može se pokazati da predloženi BCOi može vratiti kao rezultat rešenje koje se poklapa sa optimalnim rešenjem sa verovatnoćom strogo većom od nule.

5.6 Metode zasnovane na lokalnom pretraživanju za MCHBAP i DMCHBAP

Motivacija za primenu više varijanti metode promenljivih okolina na rešavanje MCHBAP-a i DMCHBAP-a bio je rad Hansena i sar. [73]. Autori su u [73] primenili opštu varijantu VNS metode (GVNS) na statički diskretni BAP sa minimizacijom troškova. GVNS iz [73] koristi tri okoline za minimizaciju funkcije cilja koju čine troškovi čekanja i obrade, troškovi kašnjenja i premija za raniji završetak obrade brodova. Uz ignorisanje nekih ograničenja, MCHBAP i DMCHBAP se mogu posmatrati kao problemi dvodimenzionalnog pakovanja.

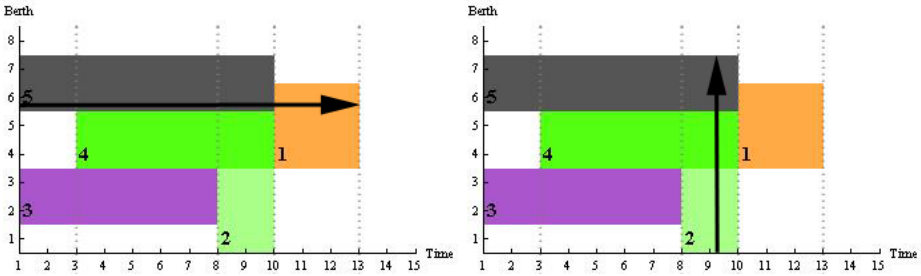
Na prvi pogled, čini se da VNS nije adekvatna metoda za rešavanje ovakvog tipa problema zbog karakterističnih koraka koji forsiraju popravku rešenja. Međutim, u radu [27] je pokazano da sofisticirane strukture podataka i definicije okolina u VNS metodi i njenim varijantama obezbeđuju dobre performanse algoritma pri rešavanju statičkog MCHBAP-a. Zbog toga su u ovoj tezi predložene četiri varijante metode promenljivih okolina za rešavanje MCHBAP-a i DMCHBAP-a: metoda promenljivog spusta, višestartna metoda promenljivog spusta, opšta metoda promenljivih okolina i adaptivna metoda promenljivih okolina. Predloženi VNS pristupi koriste adekvatnu reprezentaciju rešenja, strukture okolina i strategije pretrage prilagođene razmatranim varijantama BAP-a.

Implementacija metode promenljivog spusta za MCHBAP i DMCHBAP

Reprezentacija rešenja metode promenljivog spusta (VND) za MCHBAP i DMCHBAP ima strukturu para sekvenci koju čine dve permutacije H i V. U radu Murata i sar. [127], par (H, V) predstavlja rešenje problema projektovanja digitalnih kola visokog stepena integracije sa ciljem minimizacije utrošenog porostora. Zbog toga se postupak dekodiranja para (H, V) predložen u [127] ne može direktno primeniti na MCHBAP i DMCHBAP, već je potrebno implementirati specijalnu proceduru koja će iz klase alokacija definisanih parom (H, V) odabrati raspored brodova sa minimalnim troškovima.

Tang i sar. su u radu [163] predložili koncept najdužeg zajedničkog podniza (engl. *longest common subsequence*-LCS). LCS koncept obezbeđuje dekodiranje para permutacija (H, V) u dopustivo BAP rešenje. Par sekvenci se može pove-

zati sa usmerenim težinskim acikličnim grafom dodeljenim posmatranom problemu pakovanja. Ovi grafovi oslikavaju horizontalnu i vertikalnu vezu između elmenata koji se pakuju. Horizontalni put u pakovanju je zajednički podniz para sekvenci, i obrnuto, zajedničkom podnizu odgovara horizontalan put u pakovanju. Vertikalni put se definiše analogno, s tim što se prvi elemenat para sekvenci mora invertovati. Najduži težinski zajednički podniz u grafu, koji opisuje horizontalnu ili vertikalnu povezanost elemenata, odgovara najdužem težinskom putu u posmatranom grafu [163]. Intuitivno, LCS se može posmatrati kao maksimalna horizontalna i vertikalna dimenzija potrebna za pakovanje datih elemenata.



Ukupno vreme obrade iz (5.11) Broj zauzetih vezova iz (5.12)

Slika 5.16: Ilustracija koncepta najdužeg zajedničkog podniza

LCS koncept primenjen na MCHBAP i DMCHBAP razmatra permutacije H i V kao težinske nizove i pronalazi najduži zajednički podniz u paru težinskih permutacija. LCS se koristi u kombinaciji sa dimenzijama brodova u cilju provere dopustivosti permutacija H i V. Dopustivost u odnosu na ukupan broj raspoloživih vezova ispituje se posmatranjem broja vezova koje brodovi zauzimaju, odnosno, posmatranjem dužina brodova. Slično, širine brodova izražene vremenom obrade koriste se za proveru dopustivosti para (H, V) u odnosu na vremensku osu. Drugim rečima, par (H, V) se dekodira u dopustivu alokaciju brodova, ako su ispunjeni sledeći uslovi:

$$\sum_{v_j \in LCS(H,V)} \left\lceil \frac{a_j}{b_j} \right\rceil \leq T \,, \tag{5.11}$$

$$\sum_{v_j \in LCS(H^R,V)} b_j \leq m \,, \tag{5.12}$$

gde H^R predstavlja obrnutu H permutaciju. Par (H, V) koji ispunjava uslove (5.11)-(5.12) naziva se *dopustivi par* (H, V).

Slika 5.16 ilustruje dopustivi par $(H, V) = (\{5, 4, 3, 2, 1\}, \{3, 2, 4, 5, 1\})$ i odgovarajuće najduže zajedničke podnizove u odnosu na prostornu i vremensku osu. LCS u

odnosu na vremensku osu je $\{5, 1\}$, dok je LCS u odnosu na prostornu osu $\{2, 4, 5\}$. Na slici 5.16, horizontalnom i vertikalnom strelicom su označeni najduži zajednički podnizovi permutacija H i V.

Na permutacijama H i V se za dva broda v_i i v_j definiše binarna relacija $\lhd$, u oznaci $\lhd_H$ i $\lhd_V$, na sledeći način:

$$v_i \lhd_H v_j \Leftrightarrow v_i \text{ se nalazi ispred broda } v_j \text{ u permutaciji } H,$$
$$v_i \lhd_V v_j \Leftrightarrow v_i \text{ se nalazi ispred broda } v_j \text{ u permutaciji } V.$$

Postupak dekodiranja para sekvenci u rešenje MCHBAP-a ili DMCHBAP-a se razlikuje od postupka predloženog u [163]. Implementacija postupka dekodiranja para (H, V) u dopustivo rešenje MCHBAP-a ili DMCHBAP-a zahteva uvođenje skupa oznaka i definicija, navedenih u nastavku teksta.

Na osnovu para (H, V), za posmatrani skup brodova $\upsilon = \{v_1, v_2, ..., v_l\}$ i brod $v_j \in \upsilon$ definišu se disjunktni podskupovi L, R, A i B skupa υ:

$$L(v_j) = \{v_i \mid v_i \lhd_H v_j \wedge v_i \lhd_V v_j \wedge v_i \in \upsilon\}\ , \tag{5.13}$$

$$R(v_j) = \{v_i \mid v_j \lhd_H v_i \wedge v_j \lhd_V v_i \wedge v_i \in \upsilon\}\ , \tag{5.14}$$

$$A(v_j) = \{v_i \mid v_i \lhd_H v_j \wedge v_j \lhd_V v_i \wedge v_i \in \upsilon\}\ , \tag{5.15}$$

$$B(v_j) = \{v_i \mid v_j \lhd_H v_i \wedge v_i \lhd_V v_j \wedge v_i \in \upsilon\}\ . \tag{5.16}$$

Skup L formiraju svi brodovi koje treba alocirati levo od datog broda v_j. Brodove iz skupa R treba smestiti desno od broda v_j. Skupovi A i B predstavljaju skupove brodova čije referentne tačke treba da budu iznad, odnosno ispod, alociranog broda v_j.

U postupku dekodiranja dopustivog para (H, V), potrebno je odrediti raspored brodova definisan permutacijama H i V koji minimizira ukupne troškove. U postupku dekodiranja koriste se podskupovi R i A. U prvom koraku procedure za dekodiranje DECODE, identifikuje se brod v_k kojem odgovaraju prazni skupovi R i A i za njega se određuje najjeftinija pozicija. Brod v_k je jedinstveno određen - to je brod koji je alociran krajnje gore desno u prostorno-vremenskoj ravni. Najjeftinija pozicija za brod v_k je prva pozicija ξ-liste definisane parom (H, V). Nakon alokacije broda v_k, procedura ažurira ξ-liste ostalih brodova, a brod v_k se uklanja iz podskupova R i A. Na ovaj način, izbegnut je konflikt sa brodovima koji još nisu alocirani. U nastavku rada, procedura selektuje brod v_j koji ima prazne skupove R i A i bira najjeftiniju poziciju iz ξ-liste odabranog broda koja ne narušava redosled

definisan parom (H, V). Ta pozicija postaje referentna tačka broda v_j. Postupak se ponavlja za sve nealocirane brodove. Važno je naglasiti da svaki brod ima najmanje jednu dopustivu poziciju i da alokacija sigurno vodi do kompletnog rešenja, jer je par (H, V) dopustiv.

Algorithm 22 Dekodiranje para (H, V)

```
procedure DECODE(H, V, vessels)
    if FEASIBLE(LCS(H,V)) then
        for v_i ∈ vessels do
            {R(v_i), A(v_i)} ← EXAMINE(H, V)
        end for
        notAllocated ← vessels
        Ψ ← startΨ
        while notAllocated ≠ {} do
            v_j ← FINDEMPTY(R, A)
            positions ← CLEAR(ξ(v_j))
            Solution(v_j) ← FINDCHEAPEST(positions)
            notAllocated ← notAllocated\{v_j}
            Ψ ← UPDATELIST(Ψ)
        end while
    else
        Solution ← {}
    end if
end procedure
```

Procedura dekodiranja para (H, V) opisana je algoritmom 22. Za svaki posmatrani brod, procedura EXAMINE formira podskupove brodova R i A. Funkcija FINDEMPTY vraća indekse brodova sa praznim podlistama R i A. Procedura CLEAR obezbeđuje da se za dodelu vezova koriste samo dopustive pozicije definisane parom (H, V). Iz skupa dopustivih pozicija, funkcija FINDCHEAPEST selektuje poziciju sa minimalnim troškovima, a procedura UPDATELIST ažurira sve ξ-liste nakon alokacije brodova.

Pseudokod za VND je opisan algoritmom 24. Na početku VND algoritma procedura INITIALSOLUTION formira inicijalno rešenje. Procedura CLUSTERPREFERREDLOCATIONS kreira grupe konfliktnih brodova u odnosu na omiljeni vez, a zatim se kreirane grupe procedurom SORT uređuju u nerastućem poretku prema kardinalnosti. U formiranim grupama, brodovi su sortirani u neopadajući redosled vrednosti ETA parametra. Procedura ALLOCATE redom alocira grupe brodova i pri tome minimizira ukupne troškove alocirane grupe. Imajući u vidu da je broj brodova u grupi mali, uzimaju se u obzir sve moguće permutacije redosleda brodova, a procedura bira redosled koji formira alokaciju sa minimalnim troškovima u odnosu na dopustive pozicije iz ξ-liste. Brodovi koji nisu u konfliktu sa ostalim brodovima mogu da se smeste na prvu (ujedno i najjeftiniju) poziciju inicijalne ξ-liste. Ovaj postupak opisuje $Solution(groups(i)) \longleftarrow \xi(groups(i), 1)$ korak procedure INITIAL-

SOLUTION. Nakon smeštanja broda na odgovarajuću poziciju, procedura UPDATE briše konfliktne pozicije u Ψ-listi za sve preostale brodove. Inicijalno rešenje je formirano kad su razmotrene sve grupe i alocirani svi brodovi. Za inicijalno rešenje se kreira par permutacija (H, V) kojim se opisuje kreirana alokacija brodova. Pseudokod opisane procedure kojom se kreira inicijalno rešenje prikazan je algoritmom 23.

Algorithm 23 Konstrukcija početnog rešenja za VND

```
procedure InitialSolution(vessels)
    groups ← ClusterPreferredLocations(vessels)
    groups ← Sort(groups)
    i ← 1
    Ψ ← startΨ
    while i ≠ Length(groups) do
        if Length(groups(i)) ≠ 1 then
            Solution(groups(i)) ← Allocate(groups(i), Ψ)
        else
            Solution(groups(i)) ← Ψ(groups(i), 1)
        end if
        Ψ ← Update(Ψ)
        i ← i + 1
    end while
    return (groups)
end procedure
```

Formirano inicijalno dopustivo rešenje se kodira odgovarajućim inicijalnim permutacijama H i V. Osim toga, algoritam formira skup brodova ωS koji čine brodovi koji nisu raspoređeni na omiljene pozicije. Brodovi iz skupa ωS sortirani su u nerastući redosled troškova iz trenutno najboljeg rešenja. Prilikom svake modifikacije skupa ωS, njegovi elementi se ponovo sortiraju.

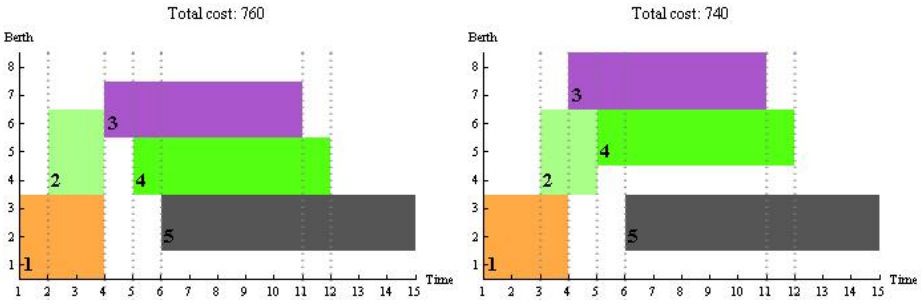
Upoređivanjem H i V permutacija dodeljenih inicijalnom i najboljem rešenju različitih test instanci, zaključeno je da brodove iz skupa ωS, treba pomeriti samo nekoliko pozicija levo ili desno u obe inicijalne permutacije H i V da bi se formirale permutacije koje odgovaraju najboljem rešenju. Koristeći ovo zapažanje, definisane su strukture okolina za VND algoritam. VND koristi tri tipa okolina primenjenih na brodove iz ωS. Brodovi se razmatraju u redosledu pojavljivanja u skupu ωS. Za zadato k, $k = 1, 2, 3, ...k_{max}$, redosled okolina je sledeći:

(i) *ChangePositionH* je prva okolina koja se formira pomeranjem odabranog broda za k pozicija levo u permutaciji H. U slučaju da promena permutacije H smanjuje troškove alokacije, formirana permutacija postaje polazna permutacija lokalne pretrage i brojač k se postavlja na vrednost 1. Ako pomeranje broda levo u permutaciji H ne popravlja rešenje, isti brod se pomera za k pozicija desno u permutaciji H. Pri transformaciji permutacije H, permutacija V ostaje nepromenjena.

Algorithm 24 VND algoritam za MCHBAP i DMCHBAP

```
procedure VND(vessels, k_max, RunTime)
    InitialSolution(vessels)
    {H, V} ← Permutations()
    ωS ← NotPreferredPosition()
    k ← 1
    while k ≤ k_max do
        i ← 1
        noImprovement ← True
        while i ≤ Length(ωS) ∧ noImprovement do
            H ← ChangePositionH(ωS(i), k)
            temp ← Decode(H, V, vessels)
            if CalculateCost(temp) < GlobalBest then
                ωS ← NotPreferredPosition()
                k ← 1
                noImprovement ← False
                BestSolUpdate()
            else
                i ← i + 1
            end if
        end while
        if noImprovement then
            i ← 1
        end if
        while i ≤ Length(ωS) ∧ noImprovement do
            V ← ChangePositionV(ωS(i), k)
            temp ← Decode(H, V, vessels)
            if CalculateCost(temp) < GlobalBest then
                ωS ← NotPreferredPosition()
                k ← 1
                noImprovement ← False
                BestSolUpdate()
            else
                i ← i + 1
            end if
        end while
        if noImprovement then
            i ← 1
        end if
        while i ≤ Length(ωS) ∧ noImprovement do
            {H, V} ← ChangePositionHV(ωS(i), k)
            temp ← Decode(H, V, vessels)
            if CalculateCost(temp) < GlobalBest then
                ωS ← NotPreferredPosition()
                k ← 1
                noImprovement ← False
                BestSolUpdate()
            else
                i ← i + 1
            end if
        end while
        if noImprovement then
            k ← k + 1
        end if
    end while
end procedure
```

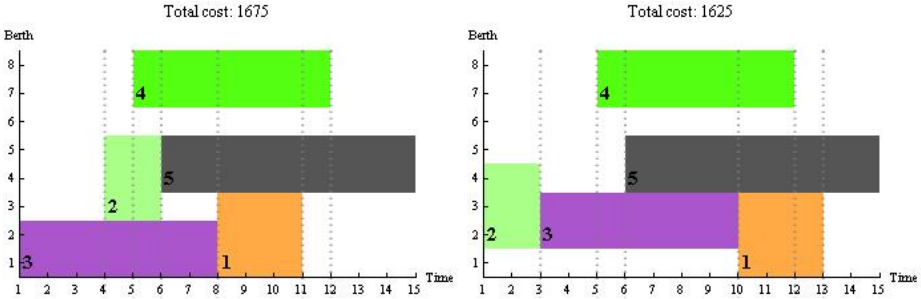
(*ii*) *ChangePositionV* je druga okolina, koja koristi analognu ideju iz definicije prve okoline. Odabrani brod se prvo pomera k pozicija levo u permutaciji



Početne permutacije: Rezultujuće permutacije:

H={**2**,3,1,4,5} V={1,2,5,4,3} H={3,2,1,4,5} V={1,2,5,4,3}

Slika 5.17: Rezultat primene okoline *ChangePositionH* na brod 2 pomeren $k = 1$ pozicija desno u permutaciji H



Početne permutacije: Rezultujuće permutacije:

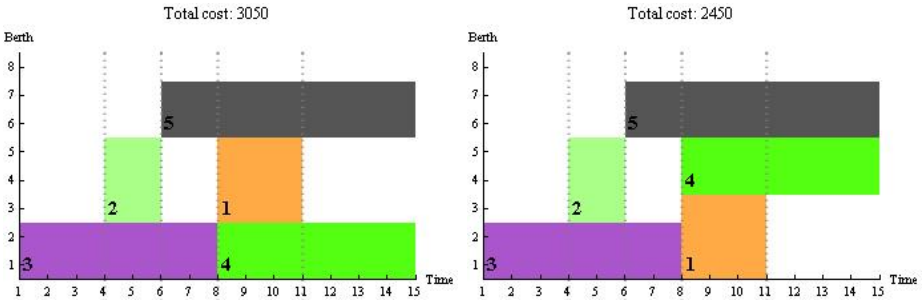
H={4,2,5,3,1} V={**3**,2,1,5,4} H={4,2,5,3,1} V={2,3,1,5,4}

Slika 5.18: Rezultat primene okoline *ChangePositionV* na brod 3 pomeren $k = 1$ pozicija desno u permutaciji V

V, a zatim i desno, ako se ne popravi rešenje. Kod popravke rešenja, pretraga se nastavlja sa resetovanom vrednošću promeljive k, uzimajući u obzir transformisanu permutaciju V. U ovoj okolini se ne menja permutacija H.

(*iii*) *ChangePositionHV* je treća okolina nastala kao kombinacija *ChangePositionH* i *ChangePositionV* okolina. U okolini *ChangePositionHV*, razmatraju se sve moguće istovremene promene permutacija H i V.

Primeri okolina *ChangePositionH*, *ChangePositionV* i *ChangePositionHV* za $k = 1$ su ilustrovani slikama 5.17-5.19. Očigledno je da parametar k_{max} može uzeti vrednosti između 1 i l–1, međutim, njegov stvarni raspon zavisi od trenutne pozicije broda.



Početne permutacije:

H={2,5,3,1,**4**} V={3,2,**4**,1,5}

Rezultujuće permutacije:

H={2,5,3,4,1} V={3,2,1,4,5}

Slika 5.19: Rezultat primene okoline *ChangePositionHV* na brod 4 pomeren $k = 1$ pozicija levo u permutaciji H i $k = 1$ pozicija desno u permutaciji V

Implementacija metode višestartnog promenljivog spusta za DMCHBAP

Prethodno opisana VND metoda je deterministička varijanta VNS-a. VND polazi od jednog dopustivog rešenja i u fazi lokalne pretrage krećući se kroz nekoliko uzastopnih okolina formira krajnje rešenje problema. Bolju divesifikaciju inicijalnog rešenja pruža metoda višestartnog promenljivog spusta (MS-VND). Na početku svake iteracije MS-VND algoritma, procedura INITIALIZE konstruiše početno rešenje. Ova procedura gradi kompletno rešenje polazeći od praznog rešenja. U postupku formiranja rešenja, procedura INITIALIZE bira brod i dopustivu poziciju iz ξ-liste za njegovu alokaciju. Kriterijum za selekciju broda je njegov prioritet definisan kao linearna kombinacija ETA parametra, veličine pravougaonika koji predstavlja brod u ravni, i prosečnih troškova svih elemenata ξ-liste selektovanog broda, čiji koeficijenti su redom λ_1, λ_2, i λ_3. Algoritam bira brod koristeći ruletsku selekciju na osnovu izračunatog prioriteta.

Algorithm 25 MSVND algoritam za DMCHBAP

```
procedure MS-VND(vessels, k_max, RunTime, λ1, λ2, λ3)
    while SessionTime() ≤ RunTime do
        Solution ← Initialize(vessels, λ1, λ2, λ3)
        (H, V) ← Permutations(Solution)
        ωS ← NotPreferredPosition(Solution)
        VND1((H, V), ωS, k_max)
    end while
end procedure
```

Prilikom selekcije pozicije broda posmatraju se sve dopustive pozicije u odnosu na vrednost troškova. U fazi inicijalizacije MS-VND algoritma, selekcija pozicije je

stohastička i bazirana je na vredosti troškova pozicije. Jeftinije pozicije imaju veću šansu da budu odabrane kroz proces ruletske selekcije. Procedura INITIALIZE računa verovatnoće izbora svih dopustivih pozicija iz ξ-liste i bira jednu poziciju ruletskom selekcijom i generisanjem slučajnog broja. Procedura fiksira posmatrani brod na odabranu poziciju i redukuje ξ-liste svih ostalih brodova. Procedura INITIALIZE ponavlja opisani postupak l puta.

Na osnovu formiranog početnog rešenja, algoritam formira par permutacija (H, V) i grupu brodova ωS. Ovi parametri predstavljaju ulazne podatke procedure VND1 koja je slična VND algoritmu opisanom u prethodnom odeljku 5.6. Glavna razlika VND i VND1 algoritma je u načinu poziva procedura PERMUTATIONS i NOTPREFERREDPOSITION. Procedura VND1 ima dodatne ulazne parametre $((H, V), \omega S)$ a procedure PERMUTATIONS i NOTPREFERREDPOSITION se pozivaju van nje. Svi koraci algoritma se ponavljaju dok se ne ispuni kriterijum zaustavljanja definisan maksimalnim vremenom izvršavanja $RunTime$. Rešenje MS-VND algoritma je najbolje rešenje dobijeno nakon višestrukih VND prolaza. Pseudokod za metodu višestartnog promenljivog spusta primenjenu na DMCHBAP dat je algoritmom 25.

Implementacija opšte metode promenljivih okolina za MCHBAP i DMCHBAP

Predložena opšta metoda promenljivih okolina (GVNS) za MCHBAP i DMCHBAP je implementirana sa ciljem poboljšanja performansi VND-a i bolje diversifikacije rešenja. Predloženi GVNS sadrži fazu razmrdavanja i koristi šest okolina u VND-u. Procedura razmrdavanja je bazirana na stohastičkim transformacijama trenutno najboljeg rešenja u cilju diversifikacije pretrage prostora rešenja. U fazi lokalne pretrage, umesto uobičajene LS procedure, implementiran je VND koji na sistematičan način pretažuje šest okolina formiranog rešenja.

Pseudokod predložene GVNS metode je predstavljen algoritmom 26. Prvi korak predložene implementacije GVNS metode je poziv procedure INITIALSOLUTION, koja je opisana algoritmom 23 i ima za zadatak formiranje početnog rešenja. Za razliku od VND-a, u nastavku GVNS algoritma potrebno je formirati grupe konfliktnih brodova. Brodovi se smeštaju u istu grupu ako su u konfliktu pri alokaciji na idealne pozicije definisane ETA parametrom i omiljenim vezom. Sve grupe konfliktnih brodova se smeštaju u strukturu liste, koja je sortirana u nerastući redosled prema broju elemenata. Posle faze formiranja inicijalnog rešenja GVNS naizmenično

Algorithm 26 GVNS algoritam za MCHBAP i DMCHBAP

```
procedure GVNS(vessels, k_max, RunTime)
    {groups} ← InitialSolution(vessels)
    {H, V} ← Permutations()
    k ← 1
    while SessionTime() ≤ RunTime do
        {tempH, tempV, tempGroups} ← Shake(vessels, groups, k)
        noImprovement ← True
        noImprovement ← SingleSwapH(noImprovement)
        noImprovement ← SingleSwapV(noImprovement)
        noImprovement ← SingleMoveH(noImprovement)
        noImprovement ← SingleMoveV(noImprovement)
        noImprovement ← DoubleSwapHV(noImprovement)
        noImprovement ← DoubleMoveHV(noImprovement)
        if noImprovement then
            k ← k + 1
            if k > k_max then
                k ← 1
            end if
        end if
    end while
end procedure
```

izvršava procedure za razmrdavanje i VND, do zadovoljenja kriterijuma zaustavljanja.

U fazi razmrdavanja novo rešenje se formira primenom dve transformacije:

(i) Prva transformacija podrazumeva slučajan izbor k grupa konfliktnih brodova koje se pomeraju na početak liste. Selekcija grupe je bazirana na izračunatom prioritetu koji je proporcionalan ukupnim troškovima grupe brodova;

(ii) U drugoj transformaciji se bira k slučajnih parova brodova na osnovu prioriteta, proporcionalnih troškovima brodova u trenutno najboljem rešenju, a zatim se selektovanim parovima brodova razmene pozicije. Pri ovoj transformaciji odabrani brodovi mogu, ali i ne moraju, da budu iz iste grupe konfliktnih brodva.

Koraci faze razmrdavanja prikazani su algoritmom 27. Procedura CalculateCost ima za cilj računanje ukupnog troška alokacije jednog broda ili grupe brodova na osnovu troškova u trenutno najboljem rešenju. Na osnovu izračunatog prioriteta, porcedura SelectGroup bira jednu grupu brodova iz liste, a zatim procedura PutAsFirst premešta tu grupu na prvu poziciju u listi. Procedura Select2Vessels selektuje dva broda na osnovu izračunatih verovatnoća izbora, a nakon toga procedura ExchangePositions razmenjuje njihove pozicije u listi grupa konfliktnih brodova.

Algorithm 27 Faza razmrdavanja u GVNS algoritmu za MCHBAP i DMCHBAP

```
procedure SHAKE(vessels, groups, k)
    done ← False
    while ¬done do
        p ← CALCULATECOST(groups)
        for i ← 1, k do
            g ← SELECTGROUP(groups, p)
            groups ← PUTASFIRST(groups, g)
        end for
        p ← CALCULATECOST(vessels)
        for i ← 1, k do
            {v1, v2} ← SELECT2VESSELS(vessels, p)
            groups ← EXCHANGEPOSITIONS(groups, v1, v2)
        end for
        ShakeSol ← ALLOCATE(groups)
        done ← FEASIBLE(ShakeSol) ∧ NEW(ShakeSol)
    end while
    {H, V} ← SHAKEPERM(ShakeSol)
    RETURN (H, V, groups)
end procedure
```

Grupe brodova se alociraju na referentne tačke po istim pravilima kao u fazi konstrukcije početnog rešenja. Formirana lista grupa brodova je ulazni parametar procedure ALLOCATE na osnovu koje procedura formira novo rešenje. Ako algoritam ustanovi da formirane grupe brodova vode do nedopustivog rešenja, na najbolje rešenje se primenjuje procedura razmrdavanja sve dok se ne generiše novo dopustivo rešenje. Postupak razmrdavanja se ponavlja i u slučaju kad je razmrdano dopustivo rešenje već razmatrano u toku izvršavanja algoritma. Procedura FEASIBLE proverava da li je formirano rešenje dopustivo ili ne, dok procedura NEW ispituje da li je rešenje ranije razmatrano u toku izvršavanja algoritma. Nakon što procedura razmrdavanja formira dopustivo rešenje, poziva se procedura SHAKEPERM sa zadatkom da formira odgovarajući par (H, V) permutacija. U nastavku GVNS algoritma, izvršava se lokalna pretraga dopustivog razmrdanog rešenja. Da bi pronašao bolje rešenje, u fazi lokalne pretrage GVNS koristi VND koji sistematično pretražuje prostor rešenja koristeći šest okolina. Ove okoline menjaju permutaciju H ili V, ili obe istovremeno, sa ciljem formiranja optimalnog para sekvenci. Za kreiranje kvalitetnih rešenja, potrebno je iskoristiti dovoljno velike okoline i adekvatan redosled transformacija permutacija. Standardne modifikacije *Swap* i *Move* upotrebljene su za transformacije permutacija H i V, u sledećem redosledu:

(i) *SingleSwapH* selektuje dva broda i razmenjuje im pozicije u permutaciji H, dok permutacija V ostaje nepromenjena;

(ii) *SingleSwapV* selektuje dva broda i razmenjuje im pozicije u permutaciji V dok permutacija H ostaje nepromenjena;

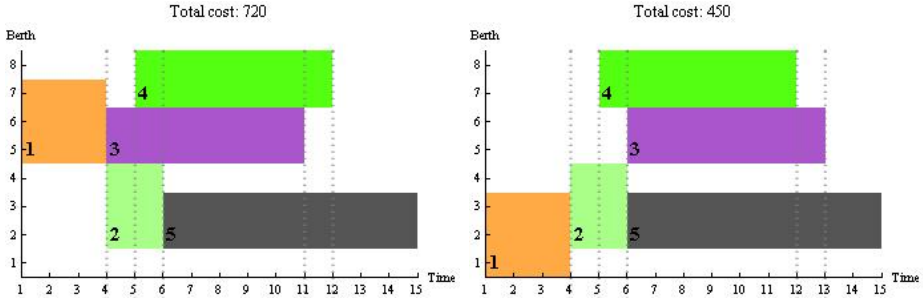
(*iii*) *SingleMoveH* selektuje dva broda v_i i v_j i pomera brod v_j iza broda v_i u permutaciji H, bez obzira na prethodni redosled brodova v_i i v_j;

(*iv*) *SingleMoveV* pomera selektovani brod v_j iza broda v_i u permutaciji V bez obzira da li je brod v_i bio ispred ili iza broda v_j u trenutnoj permutaciji;

(*v*) *DoubleSwapHV* sastoji se od jedne *SingleSwapH* i jedne *SingleSwapV* modifikacije koje nisu obavezno primenjene na istom paru brodova;

(*vi*) *DoubleMoveHV* izvršava jednu *SingleMoveH* i jednu *SingleMoveV* transformaciju koje nisu obavezno realizovane nad istim parom brodova.

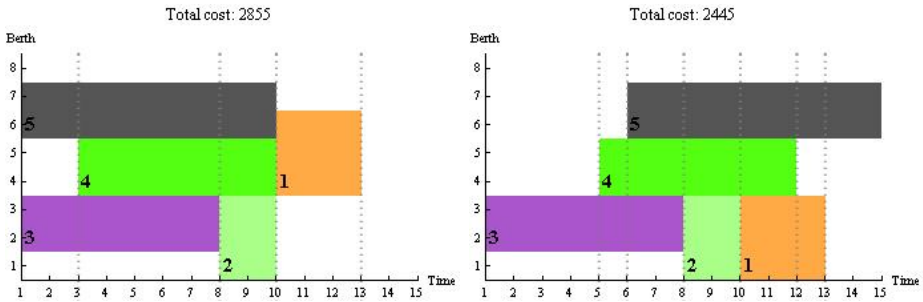
Opisane *Swap* i *Move* okoline ilustrovane su na slikama 5.20–5.25. Pseudokodovi procedura *SingleSwapH*, *SingleMoveV* i *DoubleMoveHV* dati su u tom redosledu algoritmima 28–30, dok ostale procedure imaju sa njima analogne strukture.



Početne permutacije: Rezultujuće permutacije:

H={1,4,**3**,**2**,5} V={1,2,5,3,4} H={1,4,2,3,5} V={1,2,5,3,4}

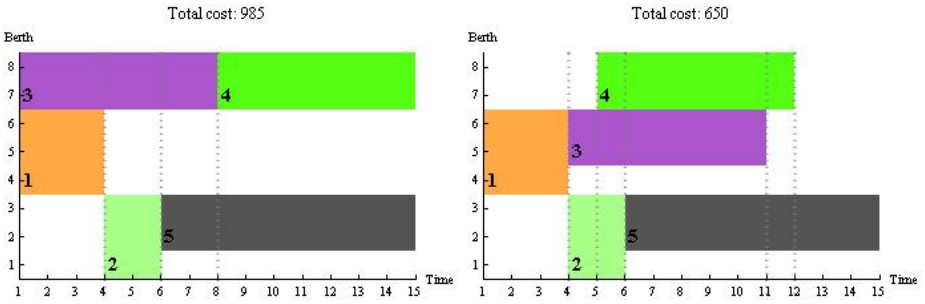
Slika 5.20: Ilustracija okoline *SingleSwapH*



Početne permutacije: Rezultujuće permutacije:

H={5,4,3,2,1} V={3,2,4,**5**,**1**} H={5,4,3,2,1} V={3,2,4,1,5}

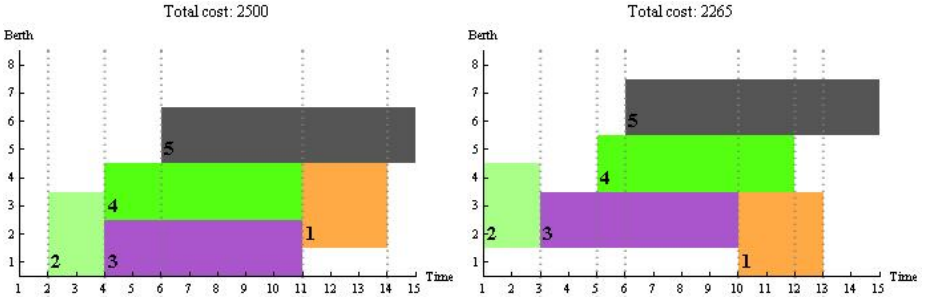
Slika 5.21: Ilustracija okoline *SingleSwapV*



Početne permutacije: Rezultujuće permutacije:

H={**3**,1,2,*textbf*4,5} V={**1**,**2**,5,3,4} H={4,1,2,3,5} V={2,1,5,3,4}

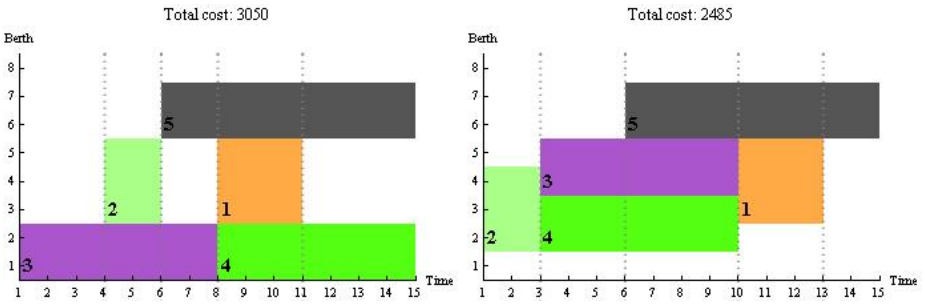
Slika 5.22: Ilustracija okoline *DoubleSwapHV*



Početne permutacije: Rezultujuće permutacije:

H={**2**,5,**4**,3,1} V={2,3,4,1,5} H={5,4,2,3,1} V={2,3,4,1,5}

Slika 5.23: Ilustracija okoline *SingleMoveH*

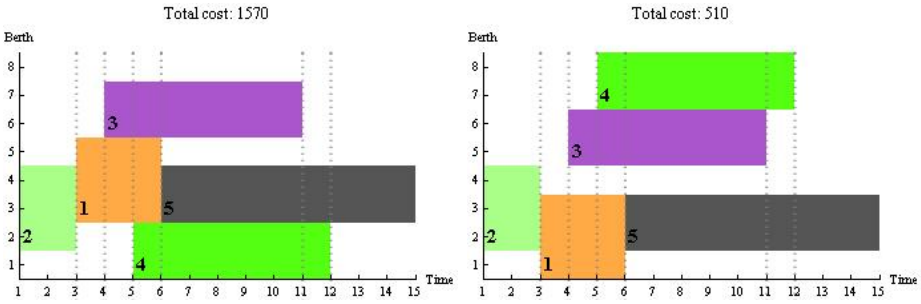


Početne permutacije: Rezultujuće permutacije:

H={2,5,3,1,4} V={**3**,2,**4**,1,5} H={2,5,3,1,4} V={2,4,3,1,5}

Slika 5.24: Ilustracija okoline *SingleMoveV*

Kao što je prikazano algoritmom 28, SINGLESWAPH poziva proceduru SWAPH. Zadatak procedure SWAPH je razmena pozicija brodova u trenutnoj permutaciji



Početne permutacije: H={**2**,3,1,5,**4**} V={2,**4**,1,5,**3**}

Rezultujuće permutacije: H={2,4,3,1,5} V={2,1,5,3,4}

Slika 5.25: Ilustracija okoline *DoubleMoveHV*

Algorithm 28 Procedura *SingleSwapH*

```
procedure SingleSwapH(noImprovement)
    ExistsTransformation ← True
    while ExistsTransformation ∧ noImprovement do
        {H1, ExistsTransformation} ← SwapH(tempH)
        temp ← Decode(H1, tempV, vessels)
        if CalculateCost(temp) < GlobalBest then
            H ← H1
            V ← tempV
            groups ← tempGroups
            k ← 1
            noImprovement ← False
            BestSolUpdate()
        end if
    end while
    return (noImprovement)
end procedure
```

Algorithm 29 Procedura *SingleMoveV*

```
procedure SingleMoveV(noImprovement)
    ExistsTransformation ← True
    while ExistsTransformation ∧ noImprovement do
        {V1, ExistsTransformation} ← MoveV(tempV)
        temp ← Decode(tempH, V1, vessels)
        if CalculateCost(temp) < GlobalBest then
            V ← V1
            H ← tempH
            groups ← tempGroups
            k ← 1
            noImprovement ← False
            BestSolUpdate()
        end if
    end while
    return (noImprovement)
end procedure
```

H smeštenoj u promenljivoj *tempH*. Osim toga, procedura SwapH u promenljivoj $ExistsTransformation$ evidentira informaciju o postojanju dopustivih modifika-

Algorithm 30 Procedura *DoubleMoveHV*

```
procedure DoubleMoveHV(noImprovement)
    ExistsTransformation ← True
    while ExistsTransformation ∧ noImprovement do
        {H1, V1, ExistsTransformation} ← MoveHV(tempH, tempV)
        temp ← Decode(H1, V1, vessels)
        if CalculateCost(temp) < GlobalBest then
            H ← H1
            V ← V1
            groups ← tempGroups
            k ← 1
            noImprovement ← False
            BestSolUpdate()
        end if
    end while
    return (noImprovement)
end procedure
```

cija razmene koje nisu primenjene na permutaciju *tempH* u prethodnim koracima algoritma. Globalne promenljive *Solution*, *GlobalBest* i *minT* se ažuriraju pri svakoj promeni najboljeg rešenja. Istovremeno, brojač k dobija vrednost 1, čime je omogućena pretraga novog najboljeg rešenja u prvoj okolini.

U predloženom GVNS algoritmu, okoline se smenjuju u redosledu u kojem su i definisane. Implementirani VND u fazi lokalne pretrage koristi strategiju prvog poboljšanja. Preciznije, nakon svakog poboljšanja trenutnog rešenja u nekoj od šest okolina, brojač okolina dobija vrednost 1 i algoritam nastavlja pretragu u okolini *SingleSwapH* novog rešenja. Vrednost parametra k_{max} je postavljena na l i definiše maksimalnu dimenziju okoline za razmrdavanje. GVNS algoritam se izvršava dok se ne ispuni kriterijum zaustavljanja, odnosno, dok se ne dostigne predefinisano vreme izvršavanja.

Implementacija adaptivne metode promenljivih okolina za DMCHBAP

Pri povećavanju broja okolina, VNS ima tendenciju degenerisanja u višestartnu heuristiku. U cilju prevazilaženja ovog problema, Hansen i Mladenović [68] su predložili novu varijantu VNS-a, adaptivnu metodu promenljivih okolina (SVNS). Motivacija za primenu SVNS metode na DMCHBAP proizašla je iz činjenice da prihvatanje lokalnog optimuma koji je neznatno lošiji od trenutno najboljeg rešenja, obezbeđuje bolje performanse algoritma pri rešavanju instanci problema koje imaju nekoliko razdvojenih i najčešće udaljenih okolina sa rešenjima bliskim optimumu [68, 70].

Algorithm 31 SVNS algoritam za DMCHBAP

```
procedure SVNS(vessels, k_max, α, RunTime)
    {Solution, groups} ← InitialSolution(vessels)
    GlobalBest ← Cost(Solution)
    while SessionTime() ≤ RunTime do
        (H, V) ← Permutations(Solution)
        k ← 1
        while k ≤ k_max do
            {H_1, V_1, groups_1} ← Shake(vessels, groups, k)
            noImpr ← True
            noImpr ← SingleSwapH(noImpr)
            noImpr ← SingleSwapV(noImpr)
            noImpr ← SingleMoveH(noImpr)
            noImpr ← SingleMoveV(noImpr)
            noImpr ← DoubleSwapHV(noImpr)
            noImpr ← DoubleMoveHV(noImpr)
            if Cost(LocalBest)−α|Cost(LocalBest) − Cost(Solution)|<Cost(Solution) then
                Solution ← LocalBest
                k ← 1
            else
                if noImpr then
                    k ← k + 1
                else
                    k ← 1
                end if
            end if
        end while
    end while
end procedure
```

Pseudokod SVNS metode primenjene na DMCHBAP dat je algoritmom 31. Struktura SVNS algoritma je slična strukturi GVNS algoritma opisanog u prethodnom odeljku. Kako SVNS dopušta prihvatanje lošijeg rešenja od trenutno najboljeg, neophodno je uvesti novu globalnu promenljivu *LocalBest*. Inicijalno rešenje je smešteno u promenljivu *Solution* koja ulazi u proceduru Shake. U fazi lokalne pretrage, rešenje dobijeno razmrdavanjem se popravlja procedurom VND koja prolazi kroz šest okolina u redosledu definisanom algoritmom 31. Na kraju lokalne pretrage, algoritam čuva dobijeni lokalni optimum u promenljivoj *LocalBest*. Razlika algoritama za GVNS i SVNS se ogleda u kriterijumu prihvatanja rezultata VND faze, odnosno lokalnog optimuma *LocalBest*. SVNS prihvata lokalni optimum *LocalBest* koji ima lošiju vrednost funkcije cilja od trenutnog rešenja *Solution*, pri čemu je stepen tolerancije prihvatanja lošijeg rešenja kontrolisan parametrom $\alpha > 0$. Preciznije, ako je vrednost izraza $Cost(LocalBest) - \alpha|Cost(LocalBest) - Cost(Solution)|$ manja od vrednosti $Cost(Solution)$, pretraga kreće od prve okoline ($k = 1$) rešenja *LocalBest*. U toku izvršavanja algoritma, potrebno je čuvati informaciju o globalno najboljem rešenju i ažurirati vrednost promenljive *GlobalBest* prilikom svake popravke globalno najboljeg rešenja.

Glava 6

Eksperimentalni rezultati

U literaturi koja se bavi problemom dodele vezova ne postoje univerzalne test instance (engl. *benchmark instances*) koje bi mogle da posluže za adekvatno testiranje predloženih algoritama za rešavanje MCHBAP-a i DMCHBAP-a. Zbog toga su u ovoj disertaciji generisane test instance na slučajan i sistematičan način. Metaheuristike razvijene za MCHBAP i DMCHBAP su testirane na nekoliko klasa test instanci čije dimenzije su predložene u radu Giallombarda i sar. [54]. U ovoj disertaciji su razmotrene samo najsloženije klase problema iz rada [54]. Ove test instance su postavljene online na adresi `http://www.mi.sanu.ac.rs/~tanjad/DMCHBAP.htm`.

Eksperimenti koji se odnose na MCHBAP izvršeni su na dva skupa instanci. Prvi skup sadrži realne instance dobijene na osnovu test primera koje su predložili Chang i sar. u [18]. Ovaj skup instanci karakteriše $l = 21$ brod, $m = 12$ vezova i horizont planiranja od $T = 54$ vremenskih jedinica. Primer predložen u [18] proširen je novim brodovima, tako da su u ovoj tezi testirane instance koje odgovaraju realnim situacijama koje sadrže između 21 i 28 brodova.

Drugi skup test instanci za MCHBAP je generisan sa $l = 35$ brodova u slučaju $m = 8$ vezova i $T = 112$ vremenskih jedinica. Ove test instance su veoma teške za rešavanje imajući u vidu njihovu dimenziiju kao i složenost problema. Za ove instance egzaktni rešavač *SEDA* zasnovan na kombinatornoj optimizaciji, nije uspeo da generiše optimalna rešenja [93] zbog neprihvatljivo velikog vremena izvršavanja. Egzaktni rešavač je testiran na računarskoj platformi sa procesorom Intel Core i7 CPU Q720 @ 1.60GHz i 6 GB RAM memorije, pod 64-bitnim operativnim sistemom Microsoft Windows 7.

U slučaju DMCHBAP-a, generisane su četiri klase test instanci, koje karakterišu sledeći parametri:

- *prva klasa instanci*: $l = 10, 15$ brodova, $m = 8$ vezova, planski period od $T = 15$ vremenskih jedinica, $l = 20$ brodova, $m = 8$ vezova, horizont planiranja od $T = 20$ vremenskih jedinica i $l = 25$ brodova, $m = 8$ vezova, vremenski horizont od $T = 25$ jedinica;
- *druga klasa instanci*: $l = 35, 40, 45$ brodova, $m = 8$ vezova, vremenski horizont od $T = 112$ jedinica;
- *treća klasa instanci*: $l = 50, 55, 60$ brodova, $m = 13$ vezova, vremenski horizont od $T = 112$ jedinica;
- *četvrta klasa instanci*: $l = 70, 80, 90, 100$ brodova, $m = 13$ vezova, i vremenski horizont od $T = 112$ jedinica.

Uobičajeni vremenski horizont planiranja od 2 nedelje podeljen je na segmente od 3 sata, tako da 112 vremenskih jedinica odgovara planskom periodu od 2 radne nedelje. Za svaku dimenziju test instanci, generisano je po 10 test primera koji se razlikuju po podacima koji se odnose na brodove.

Podaci koji karakterišu tipove brodova dati su u tabeli 6.1 a preuzeti su iz radova Meisela [119] i Bierwirtha i Meisela [7]. Test instance sadrže tri tipa brodova: *male*, *srednje* i *velike*. Za svaki tip broda, u tabeli 6.1 je dat procenat zastupljenosti u test instancama, opseg u kome se kreće vreme obrade, vrednosti troškova izražene u jedinicama od US$1000 i broj vezova koje brod zauzima. Distribucija omiljenog veza je homogena u svakom test primeru.

Tabela 6.1: Specifikacije brodova za generisane test instance

Tip broda	Zastupljenost	Vreme obrade	C1	C2	C3	C4	Broj vezova
mali	60%	1 do 3	2	3	3	9	1
srednji	30%	4 do 5	3	6	6	18	2
veliki	10%	6 do 8	4	9	9	27	3

Svi implementirani metaheuristički algoritmi za MCHBAP i DMCHBAP kodirani su u *Wolfram Mathematica v8.0* programskom jeziku. Za razliku od klasičnih programskih jezika, *Mathematica* interpretira instrukcije, što može da rezultira povećanjem vremena izvršavanja algoritma. Međutim, komparativna analiza algoritama je korektna, imajući u vidu da su svi metaheuristički algoritmi izvršavani pod istim uslovima. Svi eksprimenti su izvršeni na istoj računarskoj platformi sa procesorom

Intel Pentium 4 3.00-GHz i 512 MB RAM memorije pod 32-bitnim operativnim sistemom *Microsoft Windows XP Professional Version 2002 Service Pack 2.*

Za generisanje optimalnih rešenja instanci malih dimenzija, MILP model je izvršavan na komercijalnom CPLEX rešavaču (verzija 12.3) na istoj računarskoj platformi na kojoj su testirane i metaheuristike. Izvršna verzija CPLEX 12.3 je optimizovana za ovu platformu, što znači da je CPLEX 12.3 favorizovan u odnosu na ostale implementirane algoritme.

Stabilnost metaheuristika kao stohastičkih metoda, ispitivana je višestrukim izvršavanjem na svakoj test instanci. Na svakoj test instanci iz odgovarajućih skupova, predložene EA, cGA, BCO, BCOi, GVNS, MS-VND i SVNS metode su izvršavane po 10 puta uz vremensko ograničenje od 10 minuta, odnosno, promenljiva *RunTime* je u svim eksperimentima bila fiksirana na 600 sekundi. Izuzetak je VND metoda, koja je zbog svojih determinističkih karakteristika izvršavana samo jednom na svakoj test instanci.

Podešavanje parametara

Svaka od predloženih metaheuristika zavisi od jednog ili više parametara. Skup vrednosti ovih parametara može značajno da utiče na performanse algoritama. Stoga su, za svaku od implementiranih metaheuristika, izvršeni preliminarni testovi na odgovarajućim podskupovima test instanci u cilju određivanja adekvatne vrednosti parametara.

Tabela 6.2: Vrednosti paramatara predloženih metaheuristka

Metaheur.	Vrednosti parametara												
	k_{max}	*RunTime*	α	λ_1	λ_2	λ_3	*nGA*	*nGen*	*nImpr*	*NC*	*B*	$size_1$	$size_2$
EA	-	10 minuta	-	0.12	0.13	0.75	20	40	5	-	-	3	5
cGA	-	10 minuta	-	0.12	0.13	0.75	20	40	5	-	-	-	-
BCO	-	10 minuta	-	0.8	0	0.2	-	-	-	2	4	-	-
BCOi	-	10 minuta	-	0.8	0	0.2	-	-	-	4	4	-	-
VND	l	-	-	-	-	-	-	-	-	-	-	-	-
MS-VND	l	10 minuta	-	0.8	0	0.2	-	-	-	-	-	-	-
GVNS	l	10 minuta	-	-	-	-	-	-	-	-	-	-	-
SVNS	l	10 minuta	0.25	-	-	-	-	-	-	-	-	-	-

Tabela 6.2 prikazuje rezultate preliminarnih eksperimenata, odnosno vrednosti parametara koje vode do najboljih performansi svake pojedinačne metaheuristike. U prvoj koloni tabele 6.2 je naziv metaheuristike implementirane za MCHBAP ili

DMCHBAP. Druga kolona tabele, označena sa k_{max}, sadrži maksimalan broj okolina za VNS metode. Treća kolona sadrži vrednosti parametra $RunTime$, jednog od kriterijuma zaustavljanja, koji predstavlja maksimalno vreme izvršavanja metaheuristike. U četvrtoj koloni tabele 6.2, prikazane su vrednosti parametra α korišćenog u SVNS metodi. Vrednosti koeficijenata λ_1, λ_2 i λ_3 linearne kombinacije koja definiše prioritet brodova, dati su redom u petoj, šestoj i sedmoj koloni. Broj jedinki nGA u svakoj generaciji EA i cGA algoritma dat je u koloni osam. Maksimalan broj generacija $nGen$ evolutivnog i genetskog algoritma prikazan je u devetoj koloni, dok je u desetoj koloni dat broj jedinki $nImp$ nad kojima EA i cGA metode vrše popravku. Jedanaesta i dvanaesta kolona definiše vrednosti parametara u metodi optimizacije kolonijom pčela, broj pčela B i broj letova unapred NC. Poslednje dve kolone označene sa $size_1$ i $size_2$ prikazuju veličine turnira fino gradirane turnirske selekcije u EA algoritmu.

U nastavku teksta su dati rezultati najznačajnijih preliminarnih testova izvršenih u cilju podešavanja parametara predloženih metaheuristika.

Podešavanje parametara za SVNS metaheuristiku

Jedini parametar predloženih VNS metoda je k_{max}. Njegova vrednost je postavljena na l da bi se obezbedilo pretraživanje celog prostora rešenja. SVNS metoda ima dodatni parametar α koji omogućava kontrolisanje stepena tolerancije prihvatanja lošijeg rešenja i čiju vrednost treba odrediti eksperimentalno. U cilju poboljšanja performansi predložene SVNS metode za rešavanje DMCHBAP-a, sproveden je niz preliminarnih testova za podešavanje parametra α. Testovi su izvršeni na 12 test instanci odabranih iz druge i treće klase generisanih test instanci za DMCHBAP. Odabrane su po dve instance iz svake od grupa primera za $l \in \{35, 40, 45, 50, 55, 60\}$. Pri testiranju ispitivane vrednosti parametra $\alpha \in \{0.05, 0.10, 0.25, 0.50, 0.75, 0.90\}$, SVNS je izvršavan na svakoj instanci po deset puta sa vremenskim ograničenjem od 10 minuta.

Dobijeni rezultati su predstavljeni tabelom 6.3, koja ima sledeću strukturu. U prvoj koloni, dat je redni broj testirane instance. Test instance su sortirane u neopadajući poredak prema broju brodova l. U drugoj koloni je prikazana najbolja poznata vrednost BK funkcije cilja posmatrane instance, dobijena nekom od predloženih metaheuristika. Tabela 6.3 sadrži šest delova iste strukture, pri čemu deo odgovara jednoj testiranoj vrednosti α parametra, koja je naznačena u nazivu kolona tabele. Svaki od šest delova tabele se sastoji od četiri kolone u kojima su

Tabela 6.3: Testiranje parametra α u SVNS metodi za DMCHBAP

		$\alpha = 0.05$				$\alpha = 0.10$				$\alpha = 0.25$			
i	BK	Best	AvgCost	AvgMinT	GAP%	Best	AvgCost	AvgMinT	GAP%	Best	AvgCost	AvgMinT	GAP%
1	757	757	757.00	2.20	**0.00**	757	757.00	1.85	**0.00**	757	757.00	1.54	**0.00**
2	818	818	820.30	214.75	0.28	818	821.50	217.65	0.43	818	824.20	180.25	0.76
3	587	587	587.00	2.01	**0.00**	587	587.00	**1.52**	**0.00**	587	587.00	2.07	**0.00**
4	570	570	573.00	126.89	0.53	570	573.10	**101.33**	0.54	570	572.00	122.14	0.35
5	602	602	602.00	11.57	**0.00**	602	602.00	**8.65**	**0.00**	602	602.00	10.02	**0.00**
6	813	829	832.20	62.15	2.36	824	831.10	104.48	**2.23**	824	832.70	51.84	2.42
7	504	504	504.00	8.05	**0.00**	504	504.00	9.62	**0.00**	504	504.00	8.55	**0.00**
8	688	688	688.20	**62.53**	0.03	688	688.00	143.00	**0.00**	688	688.00	188.10	**0.00**
9	929	930	930.00	110.56	**0.11**	929	930.80	189.38	0.19	930	930.00	114.99	**0.11**
10	1045	1045	1045.00	65.92	**0.00**	1045	1045.00	99.35	**0.00**	1045	1045.00	**44.50**	**0.00**
11	822	822	822.60	98.40	0.07	822	823.20	65.18	0.15	822	822.90	**57.96**	0.11
12	863	877	879.70	73.44	1.94	877	880.60	**0.30**	2.04	874	880.20	0.55	1.99
prosek:	749.83	752.42	753.42	69.87	0.44	751.92	753.61	78.53	0.46	751.75	753.75	**65.21**	0.48
		$\alpha = 0.50$				$\alpha = 0.75$				$\alpha = 0.90$			
i	BK	Best	AvgCost	AvgMinT	GAP%	Best	AvgCost	AvgMinT	GAP%	Best	AvgCost	AvgMinT	GAP%
1	757	757	757.00	**1.39**	**0.00**	757	757.00	1.99	**0.00**	757	757.00	1.84	**0.00**
2	818	818	820.00	**179.33**	**0.24**	818	823.80	180.26	0.71	818	820.20	280.74	0.27
3	587	587	587.00	2.13	**0.00**	587	587.00	2.19	**0.00**	587	587.00	1.59	**0.00**
4	570	570	571.90	152.13	0.33	570	570.00	135.13	**0.00**	570	570.20	146.70	0.04
5	602	602	602.00	9.68	**0.00**	602	602.00	10.80	**0.00**	602	602.00	9.59	**0.00**
6	813	824	832.60	66.85	2.41	824	832.00	**51.11**	2.34	826	832.80	97.96	2.44
7	504	504	504.00	7.68	**0.00**	504	504.00	**7.62**	**0.00**	504	504.00	7.82	**0.00**
8	688	688	688.00	157.86	**0.00**	688	688.00	160.21	**0.00**	688	688.00	111.16	**0.00**
9	929	930	930.00	157.52	**0.11**	930	930.00	**105.98**	**0.11**	930	930.20	120.50	0.13
10	1045	1045	1045.00	49.13	**0.00**	1045	1045.00	62.76	**0.00**	1045	1045.00	46.60	**0.00**
11	822	822	822.60	97.88	0.07	822	822.30	86.79	**0.04**	822	822.30	85.64	**0.04**
12	863	876	880.00	134.84	1.97	877	880.50	79.86	2.03	877	879.40	130.43	**1.90**
prosek:	749.83	751.92	753.34	84.70	0.43	752.00	753.47	73.72	0.43	752.17	753.18	86.71	**0.40**

prezentovane najbolja vrednost funkcije cilja *Best* (najmanji troškovi) dobijena u deset uzastopnih puštanja SVNS algoritma sa fiksiranom vrednosti parametra α, prosečna vrednost funkcije cilja *AvgCost* za SVNS rešenja dobijena u 10 izvršavanja, prosečno vreme izvršavanja SVNS metode $AvgMinT$ i prosečno procentualno odstupanje $GAP\%$ dobijenih SVNS rešenja od najbolje poznate vrednosti funkcije cilja BK za posmatranu instancu. Sva vremena prikazana u tabeli 6.3 su izražena u sekundama. Najbolje vrednosti funkcije cilja i najmanje vrednosti $GAP\%$ su istaknute za svaku test instancu. U poslednjem redu tabele 6.3, nazvane *prosek*, prikazane su prosečne vrednosti svih rezultata iz posmatrane kolone.

Za testiranu vrednost parametra $\alpha = 0.90$, SVNS algoritam generiše rešenja koja najmanje odstupaju od najboljih poznatih rešenja. Ostale razmatrane vrednosti parametra α vode ka rešenjima koja imaju neznatno veća prosečna odstupanja od najboljih poznatih rešenja. Za vrednost $\alpha = 0.25$, implementirani algoritam za SVNS je pokazao najbolje performanse u pogledu prosečnog vremena izvršavanja algoritma, a njegova vrednost $GAP\% = 0.48$ neznatno odstupa od ostalih testiranih vrednosti. Iz tog razloga je vrednost $\alpha = 0.25$ uzeta kao najpogodnija za SVNS

algoritam za rešavanje DMCHBAP-a. U svim narednim eksperimentima sa SVNS metodom korišćena je vrednost $\alpha = 0.25$.

Podešavanje parametara za BCO metaheuristiku

Performanse BCO algoritma zavise od parametara B i NC, kao i od vrednosti koeficijenata λ_1, λ_2 i λ_3 linearne kombinacije koja definiše prioritete brodova. Adekvatne vrednosti ovih parametara treba da obezbede efikasno izvršavanje BCO algoritma.

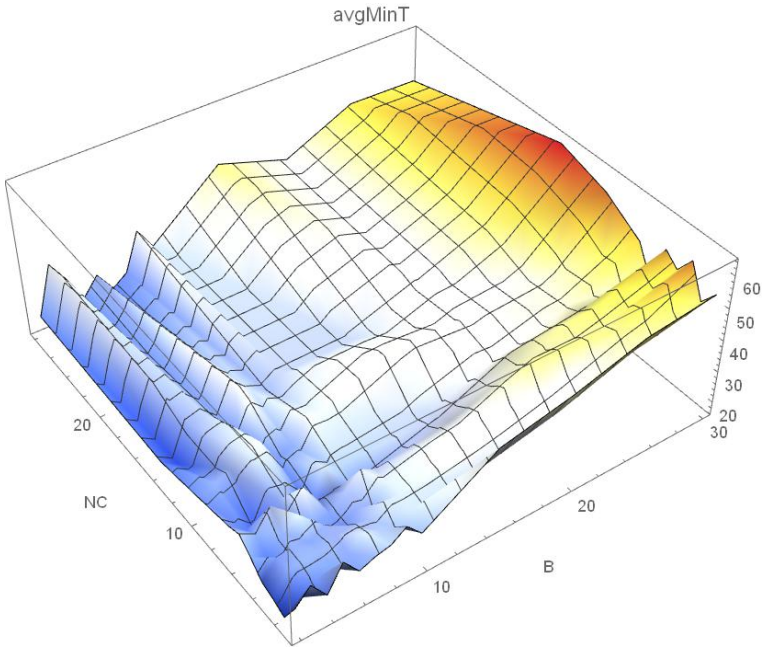
Kako su instance većih dimenzija generisane po uzoru na realne, eksperimenti sa podešavanjem parametera B, NC, λ_1, λ_2 i λ_3, su izvršeni na najvećoj realnoj instanci sa $l = 28$ brodova. Vrednost parametra $RunTime$ koji definiše maksimalno vreme izvršavanja algoritma postavljena je na 600 sekundi. Pri svakom poboljšanju vrednosti trenutno najboljeg rešenja, procesorsko vreme je sačuvano u promenljivoj $minT$. Na kraju izvršavanja algoritma, promenljiva $minT$ sadrži informaciju o procesorkom vremenu za koje je algoritam dostigao najbolje rešenje. Za svaku vrednost parametra koji se testira, algoritam je izvršen 10 puta, a nakon toga je izračunata vrednost promenljive $avgMinT$ kao prosečna vrednost sačuvanih $minT$ vrednosti iz svakog od 10 izvršavanja. U prvoj fazi testiranja, određene su vrednosti parametara B i NC, dok su sva tri λ parametra postavljena na vrednost $\frac{1}{3}$. Cilj primenjene strategije pri podešavanju parametara je da se najpre dobiju intervali pogodnih vrednosti B parametra. Na slici 6.1 prikazane su vrednosti promenljive $avgMinT$ kao funkcija parametara B i NC. Preliminarni testovi su pokazali da vrednosti parametra B imaju veliki uticaj na vrednost promenljive $avgMinT$ i da ona postaje esktremno velika kada B dobija vrednosti preko 10 (pogledati sliku 6.1). Zbog toga, u svim ostalim eksperimentima B uzima vrednost iz skupa $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ dok je $NC \in \{1, 2, 4, 7, 14, 28\}$.

Kriterijum lojalnosti izražen jednačinom (2.1) je najčešće korišćen u literaturi (videti [30]). Maksimović i Davidović su u radu [116] predložili tri dodatna kriterijuma lojalnosti, tako da su ukupno testirali i poredili 4 kriterijuma. Ovi kriterijumi lojalnosti označeni sa LC_i, $i = 1, 2, 3, 4$, su:

$$LC_1 : p_b^{1,u+1} = e^{-\frac{1-O_b}{u}} \qquad b = 1, 2, ..., B \tag{6.1}$$

$$LC_2 : p_b^{2,u+1} = e^{-\frac{1-O_b}{\sqrt{u}}} \qquad b = 1, 2, ..., B \tag{6.2}$$

$$LC_3 : p_b^{3} = e^{-(1-O_b)} \qquad b = 1, 2, ..., B \tag{6.3}$$



Slika 6.1: Prosečno procesorsko vreme generisanja optimalnog rešenja izraženo kao funkcija parametara B i NC

$$LC_4 : p_b^4 = O_b \qquad\qquad b = 1, 2, ..., B. \tag{6.4}$$

Glavna karakteristika LC_1 je da sa porastom broja letova unapred/unazad raste i verovatnoća da pčela ostane lojalna svom generisanom rešenju. Sa idejom smanjivanja zavisnosti između verovatnoće lojalnosti i broja letova unapred/unazad, u izraz za LC_2 je dodat koren u imenilac razlomka. Kriterijum LC_3 je definisan u cilju evaluacije performansi algoritma u slučaju kada verovatnoća ne zavisi od broja letova unapred/unazad, dok LC_4 definiše vrednost verovatnoće jednaku sa normalizovanoj vrednosti funkcije cilja parcijalnog/kompletnog rešenja b-te pčele.

U nastavku eksperimenata, dodatno su testirani svi predloženi kriterijumi lojalnosti, odnosno, ispitivani su sledeći parametri BCO algoritma: λ_i, $i = 1, 2, 3$, B, NC i LC_i, $i = 1, 2, 3, 4$. U novom skupu eksperimenata, promenljive λ_i, $i = 1, 2, 3$ uzimaju vrednosti iz skupa $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, uz ograničenje $\sum_{i=1}^{3} \lambda_i = 1$, za sve kombinacije vrednosti parametara λ_i. Drugim rečima, razmatrano je svih 66 uređenih trojki različitih kombinacija λ parametara koje zadovoljavaju navedene kriterijume. Parametar NC uzima vrednosti iz skupa $\{1, 2, 4, 7, 14, 28\}$, dok je B ograničeno na vrednosti iz skupa $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, jer su preliminarni testovi pokazali da veće vrednosti parametra B nisu pogodne za

problem dodele vezova. Takođe, razmotrena su sva četiri kriterijuma lojalnosti, tako da je ukupan broj različitih kombinacija parametara $14256 = 9 \cdot 6 \cdot 66 \cdot 4$. Metoda BCO ima stohastičku prirodu, i zato je svaka od 14256 kombinacija parametara testirana 5 puta da bi se povećao statistički značaj eksperimenata, odnosno, izvršeno je $14256 \cdot 5 = 71280$ testova.

Rezultati eksperimenata, izvršenih za podešavanja vrednosti parametara BCO metode, prikazani su u tabeli 6.4. Vrednost promenljive $avgMinT$ je izračunata kao prosečna vrednost pet vrednosti sačuvanih u promenljivoj $minT$. Osim toga, vrednost $Tmin$ je dobijena kao prosečna vrednost najboljih vremena sačuvanih u promenljivoj $minT$, dok je vrednost $avgT$ izračunata kao $average(avgMinT)$ za 14256 testova. Dobijene vrednosti su prikazane u drugoj koloni tabele 6.4, zajedno sa njihovim standardnim devijacijama. Preostali deo tabele 6.4 sadrži grupe od po dve kolone sa zaglavljima *parametar* i $\bar{x} \pm \sigma$, gde je *„parametar"* B, NC, λ_i; $i = 1, 2, 3$ ili LC. Kolona *„parametar"* sadrži sve razmatrane vrednosti odgovarajućeg parametra. Promenljiva $\bar{x}$ uzima vrednosti $avgT$ u gornjem delu, odnosno $Tmin$ u donjem delu tabele 6.4. Vrednost promenljive se računa na osnovu podskupa test podataka dobijenog fiksiranjem vrednosti za *„parametar"*, dok ostali parametri uzimaju sve razmatrane vrednosti. Za standardnu devijaciju je upotrebljena uobičajena oznaka σ. Na primer, za $B = 2$, vrednost prosečnog vremena $\bar{x}$ se računa kao prosek za $6 \cdot 66 \cdot 4 \cdot 5 = 7920$ sačuvanih $avgMinT$ vrednosti. U svakoj koloni tabele 6.4 su istaknute najbolje vrednosti $\bar{x} \pm \sigma$.

Na osnovu rezultata prikazanih u tabeli 6.4, može se zaključiti da kriterijumi lojalnosti LC_2 i LC_3 vode ka nešto boljim vrednostima $avgT$ i $Tmin$, u poređenju sa vrednostima dobijenim za LC_1 i LC_4, ali te razlike nisu značajne. Važno je primetiti da je u svakom od 71280 izvršavanja, BCO algoritam uspeo da generiše optimalno rešenje, odnosno, promenljiva $minT$ odgovara procesorskim vremenima za koje je BCO dostigao optimalna rešenja.

U procesu podešavanja vrednosti parametara, ako se svaki parametar tretira nezavisno od preostalih (ne uzimajući u obzir vrednosti ostalih parametara), najbolje vrednosti za sve parametre se mogu direktno preuzeti iz tabele 6.4. Međutim, vrednosti u tabeli 6.4 su slične i nije jasno da li su međusobne razlike statistički značajne. Iz tog razloga, neophodno je izvršiti dodatne analize.

Učenje stablom odlučivanja (engl. *decision tree learning*) je metoda koja se često koristi u istraživanju podataka (engl. *data mining*) i nadgledanoj klasifikaciji (engl. *supervised classification*). Učenje stablom odlučivanja je neparametraska me-

Tabela 6.4: Prosečne vrednosti merenih performansi algoritma i njihove standardne devijacije sa jednim fiksiranim parametrom

Promenljiva	$\bar{x} \pm \sigma$	B	$\bar{x} \pm \sigma$	NC	$\bar{x} \pm \sigma$	$\lambda 1$	$\bar{x} \pm \sigma$	$\lambda 2$	$\bar{x} \pm \sigma$	$\lambda 3$	$\bar{x} \pm \sigma$	LC	$\bar{x} \pm \sigma$
avgT	20.06±3.96	2	**15.74±2.77**	1	**19.60±3.81**	0.0	20.14±4.02	0.0	20.09±3.94	0.0	20.01±3.91	1	20.10±3.93
		3	16.72±2.90	2	19.81±3.88	0.1	20.12±3.85	0.1	19.97±4.02	0.1	**19.88±3.90**	2	**19.96±3.97**
		4	18.05±3.00	4	19.90±3.94	0.2	20.04±3.94	0.2	20.11±3.87	0.2	20.05±3.99	3	20.03±4.01
		5	19.08±2.81	7	20.17±3.97	0.3	20.07±4.07	0.3	20.06±4.01	0.3	20.12±3.99	4	20.14±3.93
		6	20.27±2.95	14	20.38±4.04	0.4	20.09±4.02	0.4	20.05±4.03	0.4	20.11±3.94		
		7	21.23±2.84	28	20.48±4.02	0.5	20.05±3.92	0.5	20.00±3.96	0.5	20.00±3.97		
		8	22.16±2.89			0.6	20.05±3.97	0.6	20.12±3.95	0.6	20.06±4.06		
		9	23.09±2.87			0.7	20.10±3.88	0.7	20.09±3.93	0.7	20.29±3.92		
		10	24.16±2.82			0.8	19.87±3.89	0.8	20.08±3.95	0.8	20.18±3.90		
						0.9	19.65±3.95	0.9	19.99±3.90	0.9	20.30±4.21		
						1.0	**19.62±3.89**	1.0	**19.95±3.71**	1.0	20.32±3.78		
Tmin	13.02±4.68	2	**8.71±3.77**	1	**12.55±4.59**	0.0	13.09±4.73	0.0	13.03±4.62	0.0	12.90±4.64	1	13.05±4.68
		3	9.82±3.73	2	12.80±4.62	0.1	13.13±4.67	0.1	12.96±4.73	0.1	**12.89±4.64**	2	**12.96±4.71**
		4	10.86±3.78	4	12.96±4.61	0.2	12.94±4.67	0.2	13.02±4.60	0.2	13.01±4.72	3	12.97±4.69
		5	12.03±3.80	7	13.04±4.78	0.3	13.06±4.70	0.3	13.04±4.65	0.3	13.05±4.82	4	13.10±4.64
		6	13.19±3.91	14	13.34±4.73	0.4	13.11±4.71	0.4	13.07±4.76	0.4	13.18±4.64		
		7	14.02±3.80	28	13.44±4.70	0.5	13.04±4.64	0.5	13.01±4.75	0.5	12.98±4.66		
		8	15.23±3.78			0.6	12.97±4.68	0.6	13.09±4.75	0.6	13.09±4.70		
		9	16.19±3.81			0.7	13.02±4.68	0.7	12.97±4.68	0.7	13.29±4.62		
		10	17.15±3.80			0.8	12.83±4.61	0.8	13.01±4.68	0.8	13.02±4.57		
						0.9	12.54±4.68	0.9	13.11±4.60	0.9	13.08±4.87		
						1.0	**12.53±4.55**	1.0	**12.88±4.67**	1.0	13.37±4.54		

toda koja ulazne podatke deli na podskupove koristeći niz rekurzivnih razdvajanja. Polazeći od korena, stablo se grana na osnovu unapred definisanog testa nekog atributa instance, dodeljenog svakom unutrašnjem čvoru stabla. Granama koje izlaze iz čvorova odluke odgovaraju različite vrednosti atributa, a listovima odgovaraju vrednosti funkcije cilja. Iz čvorova odluke, pretraga kreće nekom od grana, u zavisnosti od izlazne vrednosti testa. Proces se iterativno ponavlja dok se ne stigne do krajnjeg čvora, odnosno lista stabla. List sadrži izlaznu oznaku, odnosno klasu, koja je ista za sve instance koje pripadaju regionu definisanog čvorom.

Ova ideja klasifikacije se primenjuje kod regresionih stabala (engl. *regression tree*) koja se koriste u slučaju kada je očekivani izlaz klasifikacije numerička vrednost. Kod regresionog stabla, svaki list predstavlja srednju vrednost svih instanci koje su pridružene listu [13]. U ovom slučaju, instancu definiše konfiguracija parametara i odgovarajuće prosečno vreme izvršavanja metode. U stablu pretraživanja koristi se kriterijum razdvajanja zasnovan na maksimalnoj redukciji očekivane greške. Ovim kriterijumom pronalazi se najpogodnija promenljiva na osnovu koje se razdvaja podskup instanci. Greška razdvajanja se računa na osnovu standardne devijacije grana. Grananje se prekida kada je izračunata vrednost standardne devijacije instanci koje su dostigle trenutni čvor zanemarljiva u odnosu na standardnu devijaciju prvobitnog skupa instanci. Drugi kriterijum zaustavljanja grananja je kada preostane samo nekoliko instanci za klasifikaciju.

Za primenu učenja regresionim stablom u cilju podešavanja parametara BCO

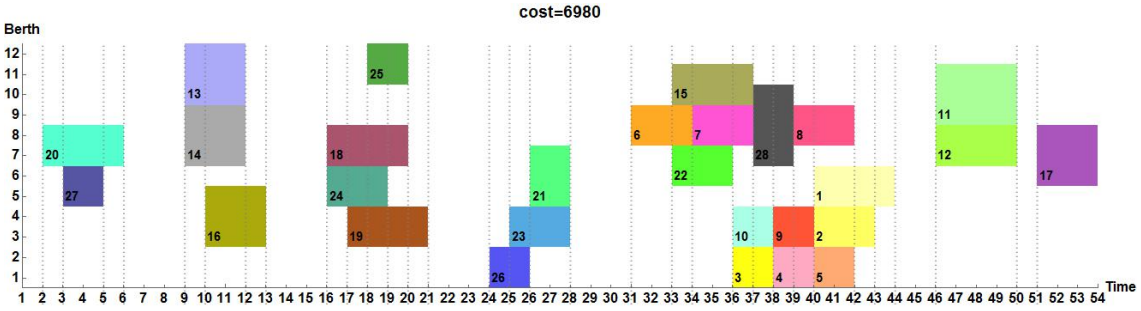
algoritma, korišćen je besplatni softver *Weka* [176] koji ima ugrađeno regresiono stablo pod nazivom $M5'$ [175]. Odnos veličine stabla i greške kontroliše se brojem instanci koje se dalje razdvajaju, a koji je u ovoj analizi postavljen na 1000. Regresiono stablo zavisne promenljive $Tmin$, koje je generisano na osnovu $M5'$ stabla ima 49 pravila, dok odgovarajuće regresiono stablo zavisne promenljive $avgT$ ima 47 čvorova. Oba generisana stabla su formirala isti skup vrednosti za parametre B, NC, λ_1, λ_2 and λ_3. Sugerisane vrednosti za navedene parametre su redom 4, 2, 0.8, 0.0 i 0.2. Ovoj konfiguraciji parametara odgovara $Tmin$ vrednost od 7.722 sekundi i $avgT = 14.707$ sekundi.

Izvršeni eksperimenti i analiza regresionog stabla su potvrdili opšteprihvaćene stavove da broj pčela u BCO algoritmu treba da bude mali [31, 116] i da parametar NC treba da ima vrednost manju od 10 [31]. Eksperimentalni rezultati su pokazali da koeficijent λ_2 nema uticaja na vreme potrebno da se generiše optimalno rešenje, odnosno da veličina pravougaonika ne utiče na prioritet broda, jer su najbolji rezultati dobijeni kada koeficijenti λ_1, λ_2 i λ_3 u linearnoj kombinaciji uzimaju redom vrednosti 0.8, 0.0 i 0.2.

Analiza uticaja tehnika popravke rešenja na performanse predloženog BCO algoritma

Za analizu uticaja predloženih tehnika popravke rešenja u BCO algoritmu, upotrebljene su realne instance iz rada Changa i sar. [18], koje karakteriše broj brodova u intervalu od 21 do 28 (pogledati sliku 6.2), 12 vezova i 54 vremenske jedinice u horizontu planiranja. U ovim testovima je korišćena kombinacija vrednosti parametara prethodno određena preliminarnim testovima za kalibraciju parametara: NC je 2 (u svakom letu unapred je dodato najviše 14 brodova u trenutno rešenje), B je 4, a koeficijenti λ_1, λ_2 i λ_3 su postavljeni redom na vrednosti 0.8, 0.0 i 0.2. Uticaj tehnika popravke rešenja na performanse algoritma je prikazan u tabeli 6.5.

U svakom od 10 izvršavanja BCO algoritma na svakoj instanci, sačuvana je vrednost funkcije cilja najboljeg rešenja. Osim toga, ukoliko neka pčela generiše rešenje koje se poklapa sa optimalnim ili se optimalno rešenje dostigne primenom neke tehnike popravke rešenja, povećava se vrednost brojača dostignutih optimuma, a istovremeno se odgovarajuće procesorsko vreme memoriše u promenljivoj $optT$. Nakon 10 izvršavanja BCO algoritma, računa se prosečna vrednost troškova i prosečna vrednost vremena potrebnih da se korstruišu rešenja koja se poklapaju sa



Slika 6.2: Optimalno rešenje realne test instance za MCHBAP sa $l = 28$, $m = 12$, $T = 54$

optimalnim. Ove vrednosti su prikazane u tabeli 6.5 u redovima označenim kao $avgCost$ i $avgOptT$. U redu $err\%$ tabele 6.5 prikazan je procenat relativne greške koji se računa po formuli $100 \cdot (avgCost - OPT)/OPT$. U poslednjem redu tabele 6.5 dat je procenat generisanih optimalnih rešenja koji odgovara verovatnoći da neka od varijanti BCO algoritma dostigne poznato optimalno rešenje. Promenljiva $opt\%$ se računa kao prosečna vrednost količnika broja generisanih optimalnih rešenja i ukupnog broja iteracija u jednom izvršavanju algoritma. Za svaku instancu su u tabeli 6.5 istaknute najbolje vrednosti $avgCost$ i $avgOptT$. U nastavku teksta su korišćene oznake I, II i III, za prvu, drugu i treću tehniku popravke rešenja.

U prvoj koloni tabele 6.5 nalazi se broj brodova koji karakteriše realnu instancu, dok je u drugoj koloni prikazana oznaka merene vrednosti: $avgCost$, $err\%$, $avgOptT$ i $opt\%$. Rezultati osnovnog BCO algoritma, bez tehnika popravke rešenja, su prikazani u trećoj koloni. Kolone 4-6 prikazuju performanse osnovnog BCO algoritma proširenog sa jednom od tehnika popravke I, II ili III. Odgovarajući rezultati dobijeni za sve kombinacije dve od tri tehnike popravke, prikazani su u kolonama 7-9. Na osnovu opisa predloženih tehnika popravke u odeljku 5.4, očigledno je da se redosled poziva procedure popravke I i II ne može menjati, kao ni redosled popravki I i III, jer se tehnika popravke I primenjuje nad kompletnim rešenjem koje je generisala pčela, dok se popravke II i III primenjuju nad globalno najboljim rešenjem. Osim toga, tehnika III mora da se primenjuje nakon tehnike II, jer nema smisla razmatrati mala pomeranja brodova u trenutnoj alokaciji, ako postoje jeftinije dopustive pozicije za neke od brodova. Tehnika III treba da se izvršava samo u slučaju kada ni za jedan brod ne postoje jeftinije pozicije u ξ-listi. Poslednja kolona prikazuje rezultate predloženog BCO algoritma, proširenog sa sve tri procedure popravke rešenja

Tabela 6.5: Uticaj tehnika popravke na kvalitet rešenja: realne test instance

l		Bez popravke	I	II	III	I+II	I+III	II+III	Sve
21	*avgCost*	37378.10	6128.24	4916.82	37353.00	4893.40	5568.28	4805.87	**4779.00**
	err%	682.13	28.23	2.88	681.61	2.39	16.52	0.56	0.00
	avgOptT	*	*	8.06	*	**3.83**	*	7.56	8.41
	opt%	0.00	0.00	28.57	0.00	29.03	0.00	50.00	100.00
22	*avgCost*	38684.00	6448.18	5122.10	38551.60	5093.12	5796.93	5007.77	**4983.00**
	err%	676.32	29.40	2.79	673.66	2.21	16.33	0.50	0.00
	avgOptT	*	*	11.56	*	12.02	*	11.19	**4.97**
	opt%	0.00	0.00	16.00	0.00	20.40	0.00	51.16	100.00
23	*avgCost*	40078.60	6828.48	5379.50	40078.60	5307.21	6075.53	5211.61	**5193.00**
	err%	671.78	31.49	3.59	671.78	2.20	17.00	0.36	0.00
	avgOptT	*	*	5.97	*	7.70	*	10.36	**4.25**
	opt%	0.00	0.00	10.00	0.00	18.75	0.00	59.09	100.00
24	*avgCost*	44004.00	7793.16	5846.10	44150.30	5779.21	6658.66	5664.85	**5643.00**
	err%	679.80	38.10	3.60	682.39	2.41	18.00	0.39	0.00
	avgOptT	*	*	11.30	*	**4.66**	*	13.59	13.52
	opt%	0.00	0.00	5.88	0.00	19.23	0.00	58.53	100.00
25	*avgCost*	46306.40	8110.76	6175.34	46147.70	6117.24	6948.15	5984.40	**5953.00**
	err%	677.87	36.25	3.74	675.20	2.76	16.72	0.53	0.00
	avgOptT	*	*	15.84	*	14.06	*	7.17	**5.30**
	opt%	0.00	0.00	4.87	0.00	12.19	0.00	22.50	100.00
26	*avgCost*	46151.50	8913.48	6535.69	46218.00	6529.44	7408.07	6324.18	**6298.00**
	err%	632.80	41.53	3.77	633.85	3.68	17.63	0.42	0.00
	avgOptT	*	*	15.44	*	14.38	*	**5.94**	18.25
	opt%	0.00	0.00	7.69	0.00	20.51	0.00	42.10	100.00
27	*avgCost*	48499.30	9545.76	6722.13	48274.80	6704.24	7816.04	6511.13	**6478.00**
	err%	648.68	47.36	3.77	645.21	3.49	20.66	0.51	0.00
	avgOptT	*	*	12.00	*	16.64	*	16.59	**8.14**
	opt%	0.00	0.00	13.15	0.00	8.11	0.00	48.38	100.00
28	*avgCost*	50863.10	10273.10	7391.71	50648.40	7295.54	8428.57	7084.62	**6980.00**
	err%	628.70	47.18	5.90	625.62	4.52	20.75	1.50	0.00
	avgOptT	*	*	*	*	*	*	23.80	**18.12**
	opt%	0.00	0.00	0.00	0.00	0.00	0.00	12.00	100.00

u redosledu opisanom u odeljku 5.4. U tabeli 6.5 je korišćena oznaka $*$ za vrednosti promenljive *avgOptT* koje se ne mogu izračunati jer u posmatranom slučaju algoritam nije pronašao optimalna rešenja, a odgovarajuća vrednost promenljive *opt%* je nula.

Na osnovu rezultata prikazanih u tabeli 6.5, može se zaključiti da BCO nije dovoljno efikasan da bi formirao rešenja razmatranih primera koja se poklapaju sa optimalnim. Prva tehnika popravke rešenja je previše jednostavna da bi vodila ka generisanju visokokvalitetnih rešenja, prvenstveno jer se primenjuje samo nad dostupnim dopustivim pozicijama brodova selektovanim ruletskom selekcijom iz ξ-liste. Treća tehnika popravke takođe ne daje dobre rezultate kad se primenjuje

samostalno. Uzrok je u činjenici da nema razloga dozvoliti perturbacije i povećanje troškova alokacije, ako se ta alokacija može popraviti pomeranjem brodova na slobodne pozicije sa manjim troškovima u ξ-listi. Kombinacija prve i treće tehnike popravke formira kvalitetnija rešenja od rešenja generisanih samo sa trećom popravkom, ali se optimum ne može dosegnuti. U pojedinim testovima, optimum je dostignut kombinacijom prve i druge tehnike popravke, ali ne i u slučaju instance sa 28 brodova. Naime, optimum se može dostići kombinacijom prve i druge tehnike popravke u slučajevima kada je razmotren pogodan redosled brodova i kada su brodovi alocirani na pozicije tako da ne blokiraju ostale brodove prilikom pomeranja na jeftinije slobodne pozicije postojeće ili ažurirane ξ-liste. Međutim, redosled brodova i povoljne pozicije nisu poznati unapred, već se mogu samo generisati na slučajan način. Druga i treća tehnika u kombinaciji sa BCO algoritmom, u nekim slučajevima generišu rešenje koje se poklapa sa optimalnim za sve testirane instance, ali generalno, rezultati dobijeni na ovaj nacin još nisu zadovoljavajućeg kvaliteta. Tek povezivanjem sve tri tehnike dobijaju se mnogo kvalitetnija rešenja posebno u pogledu smanjenja procenta prosečne greške, uz malo povećanje prosečnog vremena *avgOptT*. Očigledno, dobar pravac istraživanja pri rešavanju problema dodele vezova je razvoj efikasnih tehnika popravki rešenja za BCO algoritam.

Osim opisanih testova na realnim instancama, izvršeno je i 50 dodatnih testova nad generisanim instancama čije dimenzije odgovaraju najvećoj realnoj instanci ($l = 28$, $m = 12$ i $T = 54$) sa ciljem ocene kvaliteta tehnika popravki rešenja i da se utvrdi adekvatan redosled njihove primene. Instance su formirane na osnovu specifikacije brodova datih u tabeli 6.1.

Korak algoritma sastoji se od jedne iteracije BCO algoritma nad jednom test instancom sa odabranim tipom BCO algoritma označenim kao *BCOType* i vrednostima promenljivih B, NC, λ_1, λ_2 i λ_3 postavljenim redom na 4, 2, 0.8, 0.0 i 0.2. Tip BCO algoritma je definisan brojem i kombinacijom korišćenih tehnika popravke rešenja. Promenljiva *AlgN* broji *korake algoritma*. Na kraju svakog *koraka algoritma*, u pomoćnoj promenljivoj *bestBCO* memoriše se vrednost funkcije cilja kompletnog rešenja sa najmanje troškova. Ako je generisano optimalno rešenje, brojač *OptN* povećava vrednost za 1 a odgovarajuće procesorsko vreme kada je optimum formiran, smešta se u promenljivu *optT*. *Izvršavanje algoritma* podrazumeva uzastopne pozive *koraka algoritma* dok se ne ispuni kriterijum zaustavljanja definisan maksimalnim vremenom izvršavanja algoritma od 600 sekundi. Nakon *izvršavanja algoritma*, računa se vrednost *bestSolution* kao

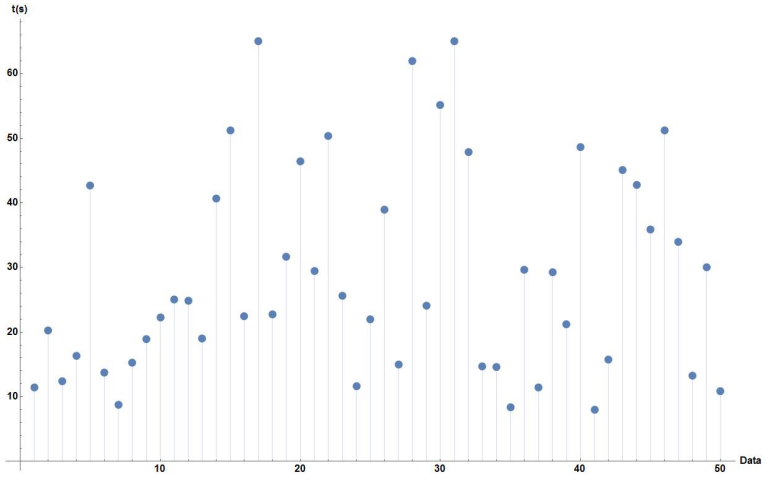
min($bestBCO$). *Eksperiment* se sastoji od deset uzastopnih *izvršavanja algoritma* za istu test instancu. Jedan *test* čine uzastopni *eksperimenti* sa fiksiranom vrednošću $BCOType$ primenjeni nad 50 generisanih test instanci. Nakon svakog *testa*, računaju se srednje vrednosti troškova i srednje vrednosti troškova najboljih rešenja kao $avgCost = average(bestBCO)$ i $avgBest = average(bestSolution)$. Osim toga, nakon svakog testa računa se i $avgOptT$ definisan kao $average(optT)$. Procenat relativne greške $(100 \cdot (avgCost - average(OPT))/average(OPT))$ prikazan je u promenljivoj $avgErr\%$. U slučajevima kada je vrednost promenljive $opt\%$ nula, odgovarajuća vrednost $avgOptT$ se ne može izračunati. Zbog toga je za takve vrednosti $avgOptT$ u tabeli 6.5 korišćena oznaka $*$. Sve opisane promenljive i njihove vrednosti prikazane su u tabeli 6.6, dok su istaknute najbolje vrednosti $avgCost$ i $avgOptT$.

Tabela 6.6: Uticaj tehnika popravke na kvalitet rešenja na skupu generisanih test instanci

BCOTypes	**avgCost**	**avgErr%**	**avgBest**	**avgOptT**	**OptN**	**AlgN**
NoImpr	4366.41	1146.69	3270.70	*	0	4049
I	679.69	94.06	531.10	570.31	1	7352
II	429.28	22.57	400.64	36.12	596	7285
III	4355.29	1143.52	3173.08	*	0	7636
I+II	425.42	21.46	400.28	82.72	663	7205
I+III	502.75	43.55	421.48	85.28	25	5134
II+III	402.46	14.91	398.84	**27.86**	575	5520
All	**350.24**	0.00	350.24	28.86	833	833

Osnovni BCO algoritam, proširen tehnikama popravke I+II+III, može da generiše optimalno rešenje u 100% svih testova. Poslednji red tabele 6.5, pokazuje da za realne primere sa 28 brodova, samo BCO tipovi algoritama sa kombinacijama II+III i I+II+III mogu da dostignu optimalna rešenja u bar jednom izvršavanju algoritma, dok su za generisane test instance uspešni i BCO tipovi algoritma sa II tehnikom popravke i kombinacijom I+II. Analizom razmatranih generisanih test primera identifikovan je razlog ove pojave. U nekim primerima postoje male grupe brodova kojima odgovaraju veliki pravougaonici i koji imaju slične inicijalne referentne tačke, dok su svi ostali brodovi uniformno raspoređeni u ravni. Ako u inicijalnoj fazi algoritam uspešno odredi dobre pozicije vezivanja za velike brodove, popravka II može da generiše optimalno rešenje. Međutim, čak i tada je procenat

uspešnosti manji od 10.5%. Tehnika popravke I i kombinacija tehnika I+III može povremeno da nađe optimum u malom broju eksperimenata, što dokazuje procenat od samo 0.5% optimalnih rešenja. Osnovni BCO, i BCO u kombinaciji sa trećom tehnikom popravke rešenja nije generisao optimalno rešenje ni u jednom testu. Na osnovu izloženih rezultata u tabeli 6.5, zaključuje se da tehnike popravke rešenja imaju značajan uticaj na efikasnost BCO algoritma i njegove performanse.



Slika 6.3: *avgOptT* za 50 generisanih test instanci

Slika 6.3 prikazuje distribuciju vrednosti promenljive *avgOptT* za 50 generisanih test instanci nad kojima je primenjen predloženi BCO algoritam proširen tehnikama popravke rešenja. Najkraće vreme je bilo potrebno za rešavanje instance 41, jer je za ovu instancu optimalno rešenje dostignuto za 7.95 sekundi u proseku. Algoritam je pokazao najlošije performanse u slučaju primera 17 i 31 kod kojih *avgOptT* iznosi 64.98 sekundi. U poslednjem redu tabele 6.6 prikazana je prosečna vrednost svih 50 vrednosti *avgOptT* predloženog BCO algoritma, koja iznosi 28.86 sekundi. Važno je primetiti da je 62% svih *avgOptT* vrednosti ispod 30 sekundi dok je čak 94% manje od 60 sekundi. Očigledno je da BCO može da nađe optimalna rešenja test instanci sličnih realnim instancama u vrlo kratkom procesorskom vremenu.

6.1 Rezultati i poređenja za MCHBAP

Statička varijanta alokacije brodova MCHBAP testirana je na dva skupa instanci: realnim i generisanim. Poređenje rezultata na realnim test primerima dato

je u tabeli 6.7. Prva kolona tabele 6.7 prikazuje broj brodova svake instance. Vrednost funkcije cilja (vrednost troškova alokacije) koja odgovara optimalnom rešenju označena je sa *Opt*, i prikazana je u drugoj koloni tabele. Optimalne vrednosti *Opt* dobijene su primenom egzaktnog rešavača iz rada [93]. Treća kolona tabele 6.7 sadrži promenljivu T koja odgovara vremenima (izraženim u sekundama) neophodnim egzaktnom rešavaču da generiše optimalno rešenje (važno je primetiti da je egzaktni rešavač testiran na mnogo boljoj platformi u odnosu na platformu na kojoj su testirane metaheuristike). Sledeće dve kolone u tabeli 6.7 sadrže rezultate dobijene primenom evolutivnog algoritma EA: prosečna vrednost troškova ($AvgC$) deset najboljih rešenja generisanih EA algoritmom i odgovarajuće prosečno minimalno procesorsko vreme ($AvgT$) za koje je EA generisao najbolja rešenja. Rezultati implementiranih cGA, BCO i GVNS metoda su prikazani na identičan način kao i za EA. Kako je VND determinističke prirode, dve kolone koje se odnose na VND sadrže najniže troškove ($Best$) iz jednog VND prolaza i odgovarajuće vreme izvršavanja VND algoritma T. Poslednji red tabele 6.7 označen sa *prosek* sadrži prosečne vrednosti predstavljenih rezultata po kolonama. Svi rezultati u tabeli 6.7 koji se odnose na vremena izvršavanja dati su u sekundama. Najbolji rezultati prikazani u tabeli 6.7 su istaknuti (boldovani).

Tabela 6.7: Eksperimentalni rezultati za MCHBAP na realnim test primerima: $m = 12$, $T = 54$

	Egzaktni rešavač		EA		cGA		BCO		VND		GVNS	
l	*Opt*	T	*AvgC*	*AvgT*	*AvgC*	*AvgT*	*AvgC*	*AvgT*	*Best*	T	*AvgC*	*AvgT*
21	4779	4.22	**4779**	14.86	**4779**	1.57	**4779**	11.43	**4779**	0.50	**4779**	**0.08**
22	4983	6.14	**4983**	21.10	**4983**	1.53	**4983**	30.82	**4983**	0.55	**4983**	**0.08**
23	5193	8.14	**5193**	24.83	**5193**	3.74	**5193**	22.51	**5193**	0.61	**5193**	**0.09**
24	5643	8.43	**5643**	31.25	**5643**	3.10	**5643**	21.95	**5643**	0.66	**5643**	**0.09**
25	5953	15.00	**5953**	36.47	**5953**	4.17	**5953**	47.45	**5953**	0.67	**5953**	**0.10**
26	6298	64.88	**6298**	33.72	**6298**	8.52	**6298**	43.93	**6298**	0.83	**6298**	**0.12**
27	6478	66.28	**6478**	41.12	**6478**	7.38	**6478**	56.46	**6478**	0.89	**6478**	**0.13**
28	6980	326.77	**6980**	67.06	**6980**	10.06	**6980**	186.87	**6980**	0.95	**6980**	**0.14**
prosek:	5788.4	62.48	**5788.4**	33.80	**5788.4**	5.01	**5788.4**	52.68	**5788.4**	0.71	**5788.4**	**0.10**

Na osnovu rezultata prikazanih u tabeli 6.7, očigledno je da su sve predložene metaheurističke metode dostigle optimalno rešenje dobijeno egzaktnim rešavačem iz [93]. Osim toga, važno je istaći da su EA, cGA, BCO i GVNS formirale optimalno rešenje u svih deset izvršavanja algoritma. Sve metaheurističke metode su nadmašile

egzaktni rešavač u odnosu na prosečno vreme izvršavanja, dok su cGA, VND i GVNS pokazali bolje performanse od egzaktnog rešavača na svakoj test instanci. Među četiri predložene metaheuristike, najkraće prosečno vreme izvršavanja na svim realnim test instancama ima GVNS. U tabeli 6.7 istaknuta su najkraća prosečna vremena izvršavanja za svaku test instancu.

Tabela 6.8 sadrži poređenja rezultata predloženih metaheurističkih metoda za MCHBAP na generisanim test primerima. Egzaktni rešavač predložen u radu [93] nije mogao da reši ove test instance do optimuma. Prva kolona tabele 6.8, označena sa i, sadrži redni broj test instance, dok se u drugoj koloni, označenoj sa BK, nalazi najmanja poznata vrednost troškova alokacije brodova za datu test instancu. Sledeće četiri kolone se odnose na rezultate EA metaheuristike. U koloni $Best$ prikazana je najbolja vrednost ukupnih troškova dobijena nakon deset izvršavanja EA algoritma. Prosečni ukupni troškovi $AvgC$ i odgovarajuće prosečno minimalno vreme izvršavanja $AvgT$ su prikazani u sledeće dve kolone. U cilju merenja kvaliteta dobijenih EA rešenja, u koloni $G\%$ prikazano je prosečno procentualno odstupanje generisanog rešenja u odnosu na najbolje poznato rešenje, dobijeno kao $100 \cdot \frac{AvgC-BK}{BK}$.

Rezultati za cGA, BCO i GVNS su u tabeli 6.8 prikazani analogno kao i u slučaju EA metode. Zbog determinističkih karakteristika VND algoritma, kolona $AvgC$ je za VND metodu izostavljena iz tabele 6.8, dok kolona T prikazuje vreme izvršavanja VND metode dobijeno u jednom prolazu. Posledično, prosečno procentualno odstupanje generisanog rešenja VND metode se računa kao $100 \cdot \frac{Best-BK}{BK}$. Da bi se u tabeli 6.8 lakše uočile metode sa najboljim performansama u odnosu na kvalitet generisanog rešenja, za svaku instancu je istaknuto najbolje poznato rešenje. Osim toga, za najbolju metodu u odnosu na vreme izvršavanja, istaknuta su i najkraća prosečna vremena izvršavanja za svaku test instancu. Poslednji red tabele 6.8 označen sa *prosek* sadrži prosečne vrednosti svih predstavljenih rezultata.

Tabela 6.8: Eksperimentalni rezultati za MCHBAP na generisanim test primerima: $l = 35$, $m = 8$, $T = 112$

		EA				cGA				BCO				VND				GVNS			
i	BK	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*T*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
1	717	**717**	717.0	104.44	0.00	**717**	717.0	152.99	0.00	718	718.0	143.75	0.14	**717**	21.16	0.00	**717**	717.0	**1.04**	0.00
2	491	**491**	491.3	369.58	0.06	**491**	491.0	22.23	0.00	**491**	491.0	54.00	0.00	493	**1.91**	0.41	**491**	491.7	49.61	0.14
3	683	**683**	683.6	280.23	0.09	**683**	683.0	51.28	0.00	**683**	683.0	51.80	0.00	**683**	22.47	0.00	**683**	683.0	**1.09**	0.00
4	554	**554**	554.0	148.58	0.00	**554**	554.0	60.83	0.00	**554**	554.0	237.64	0.00	**554**	162.91	0.00	**554**	555.8	**42.95**	0.32
5	594	**594**	594.0	63.59	0.00	**594**	594.0	39.59	0.00	**594**	594.0	40.41	0.00	**594**	121.67	0.00	**594**	594.0	**6.48**	0.00
6	486	**486**	486.0	115.40	0.00	**486**	486.0	24.04	0.00	**486**	486.0	41.42	0.00	492	**2.38**	1.23	**486**	486.0	201.44	0.00
7	543	**543**	543.0	133.57	0.00	**543**	543.0	**16.55**	0.00	**543**	543.0	34.09	0.00	**543**	203.13	0.00	**543**	543.0	21.81	0.00
8	554	**554**	554.0	351.20	0.00	**554**	554.0	34.80	0.00	**554**	554.0	52.30	0.00	**554**	3.02	0.00	**554**	554.0	**0.19**	0.00
9	531	**531**	532.4	364.74	0.26	**531**	531.0	76.11	0.00	**531**	531.0	34.84	0.00	537	**1.86**	1.13	**531**	532.2	15.73	0.23
10	486	**486**	486.3	304.19	0.06	**486**	486.0	100.47	0.00	**486**	486.0	42.31	0.00	**486**	2.00	0.00	**486**	486.0	**0.14**	0.00
11	480	**480**	480.3	82.47	0.06	**480**	480.0	12.43	0.00	**480**	480.0	190.81	0.00	**480**	1.92	0.00	**480**	480.0	**0.14**	0.00
12	573	**573**	573.3	272.73	0.05	**573**	573.0	66.39	0.00	**573**	573.0	145.77	0.00	578	**2.73**	0.87	**573**	575.3	327.26	0.40
13	520	**520**	520.0	115.42	0.00	**520**	520.0	45.84	0.00	**520**	520.0	47.23	0.00	**520**	**9.52**	0.00	**520**	520.0	94.45	0.00
14	557	**557**	557.0	116.96	0.00	**557**	557.0	57.83	0.00	**557**	557.0	59.77	0.00	569	**5.02**	2.15	**557**	560.6	179.80	0.65
15	627	**627**	631.8	266.14	0.77	**627**	627.0	32.95	0.00	**627**	627.0	124.95	0.00	**627**	6.47	0.00	**627**	627.0	**0.37**	0.00
16	479	**479**	479.0	125.14	0.00	**479**	479.0	14.02	0.00	**479**	479.0	26.28	0.00	**479**	385.70	0.00	**479**	479.0	**8.49**	0.00
17	452	**452**	452.0	135.09	0.00	**452**	452.0	31.03	0.00	**452**	452.0	20.62	0.00	**452**	71.25	0.00	**452**	452.0	**9.77**	0.00
18	595	**595**	595.0	134.57	0.00	**595**	595.0	38.77	0.00	**595**	595.0	43.64	0.00	**595**	6.22	0.00	**595**	595.0	**0.34**	0.00
19	580	**580**	580.1	159.45	0.02	**580**	580.0	90.29	0.00	**580**	580.0	**53.67**	0.00	582	104.80	0.34	**580**	584.6	349.92	0.79
20	577	**577**	577.0	309.37	0.00	**577**	577.0	**26.97**	0.00	**577**	577.0	150.28	0.00	583	149.70	1.04	**577**	581.8	38.16	0.83
21	623	**623**	623.0	131.14	0.00	**623**	623.0	**28.20**	0.00	**623**	623.0	34.84	0.00	**623**	78.16	0.00	**623**	623.0	34.33	0.00
22	495	**495**	495.3	354.55	0.06	**495**	495.0	30.26	0.00	**495**	495.0	110.23	0.00	496	**4.28**	0.20	**495**	495.8	20.87	0.16
23	459	**459**	459.0	138.90	0.00	**459**	459.0	32.27	0.00	**459**	459.0	108.55	0.00	**459**	225.39	0.00	**459**	459.0	**23.85**	0.00
24	514	**514**	514.0	139.03	0.00	**514**	514.0	148.42	0.00	**514**	514.0	40.03	0.00	**514**	121.97	0.00	**514**	514.0	**21.70**	0.00
25	613	**613**	613.0	134.45	0.00	**613**	613.0	112.82	0.00	**613**	613.0	**104.45**	0.00	645	512.64	5.22	**613**	628.2	190.54	2.48
26	477	**477**	477.0	148.93	0.00	**477**	477.0	**27.62**	0.00	**477**	477.0	35.94	0.00	**477**	108.45	0.00	**477**	477.0	40.15	0.00
27	517	**517**	517.0	144.85	0.00	**517**	517.0	18.73	0.00	**517**	517.0	48.69	0.00	**517**	2.84	0.00	**517**	517.0	**0.18**	0.00
28	517	**517**	517.0	209.62	0.00	**517**	517.0	214.09	0.00	**517**	517.0	27.19	0.00	634	560.95	22.63	**517**	517.0	**21.05**	0.00
29	464	**464**	464.0	132.01	0.00	**464**	464.0	16.74	0.00	**464**	464.0	55.52	0.00	467	**2.25**	0.65	**464**	464.3	147.64	0.06
30	592	**592**	592.0	330.95	0.00	**592**	592.0	35.27	0.00	**592**	592.0	32.02	0.00	**592**	1.81	0.00	**592**	592.0	**0.13**	0.00
31	665	**665**	665.2	229.23	0.03	**665**	665.0	**63.55**	0.00	**665**	665.0	200.38	0.00	675	196.95	1.50	**665**	672.0	178.60	1.05
32	495	**495**	495.0	183.15	0.00	**495**	495.0	**31.09**	0.00	**495**	495.0	90.05	0.00	**495**	172.83	0.00	**495**	495.0	66.54	0.00
33	481	**481**	481.0	153.04	0.00	**481**	481.0	31.88	0.00	**481**	481.0	48.23	0.00	**481**	**6.94**	0.00	**481**	481.0	14.78	0.00
34	539	**539**	539.0	231.52	0.00	**539**	539.0	26.06	0.00	**539**	539.0	58.23	0.00	**539**	160.14	0.00	**539**	539.0	**7.02**	0.00
35	528	**528**	528.0	136.32	0.00	**528**	528.0	21.52	0.00	**528**	528.0	43.98	0.00	**528**	3.50	0.00	**528**	528.0	**0.21**	0.00

Tabela 6.9: Eksperimentalni rezultati za MCHBAP na generisanim test primerima: $l = 35$, $m = 8$, $T = 112$ (nastavak)

		EA				cGA				BCO				VND			GVNS			
i	BK	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*T*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
36	522	**522**	522.6	253.14	0.11	**522**	522.0	20.76	0.00	**522**	522.0	199.94	0.00	**522**	55.52	0.00	**522**	522.0	**15.94**	0.00
37	467	**467**	467.0	318.14	0.00	**467**	467.0	17.30	0.00	**467**	467.0	342.61	0.00	**467**	1.83	0.00	**467**	467.0	**0.13**	0.00
38	479	**479**	480.4	254.55	0.29	**479**	479.0	22.32	0.00	**479**	479.0	63.61	0.00	518	**6.66**	8.14	**479**	483.5	109.43	0.94
39	534	**534**	534.0	136.48	0.00	**534**	534.0	110.91	0.00	**534**	534.0	136.67	0.00	**534**	40.56	0.00	**534**	534.0	**2.99**	0.00
40	574	**574**	575.5	114.58	0.26	**574**	574.0	33.40	0.00	**574**	574.0	213.41	0.00	**574**	**1.25**	0.00	**574**	574.0	58.36	0.00
41	554	**554**	555.5	477.55	0.27	**554**	554.0	**28.64**	0.00	**554**	554.0	60.45	0.00	563	237.05	1.62	**554**	560.7	377.57	1.21
42	448	**448**	448.0	142.48	0.00	**448**	448.0	10.00	0.00	**448**	448.0	35.55	0.00	**448**	1.77	0.00	**448**	448.0	**0.13**	0.00
43	529	**529**	534.0	590.94	0.95	**529**	529.0	**33.01**	0.00	**529**	529.0	35.47	0.00	**529**	151.89	0.00	**529**	532.9	43.75	0.74
44	553	**553**	553.0	209.99	0.00	**553**	553.0	71.17	0.00	**553**	553.0	62.02	0.00	**553**	**34.20**	0.00	**553**	554.2	108.80	0.22
45	478	**478**	478.0	134.73	0.00	**478**	478.0	21.90	0.00	**478**	478.0	72.12	0.00	**478**	1.86	0.00	**478**	478.0	**0.13**	0.00
46	575	**575**	575.0	230.33	0.00	**575**	575.0	153.35	0.00	**575**	575.0	109.73	0.00	**575**	119.75	0.00	**575**	575.1	**94.92**	0.02
47	554	**554**	558.2	386.90	0.76	**554**	554.0	**35.71**	0.00	**554**	554.0	90.06	0.00	**554**	256.53	0.00	**554**	563.3	226.79	1.68
prosek:	**538.8**	**538.8**	539.3	212.77	0.087	**538.8**	**538.8**	**50.90**	0.000	538.9	538.9	86.29	0.003	544.1	92.63	1.003	**538.8**	540.3	67.14	0.254

Iz rezultata prikazanih u tabeli 6.8, može se videti da EA, cGA i GVNS dostižu najbolja poznata rešenja za sve test instance. BCO metoda nije uspela da generiše najbolje poznato rešenje samo u slučaju prve test instance, dok VND nije bio uspešan u 14 od 47 slučajeva. Osim toga, VND ima najveće prosečno procentualno odstupanje generisanog rešenja u odnosu na najbolje poznato rešenje. Kao što se vidi iz odgovarajućih vrednosti u $G\%$ koloni, BCO je uspeo da generiše najbolje poznato rešenje u svih deset prolaza algoritma za sve generisane instance, sem u slučaju prve. Predložene EA, BCO i GVNS implementacije imaju male vrednosti prosečnog procentualnog odstupanja generisanog rešenja, ali se na osnovu poređenja vrednosti u $G\%$ koloni može zaključiti da je BCO nešto pouzdanija metoda. Isti zaključak se može izvesti i u odnosu na vrednosti prosečnih troškova $AvgC$, odnosno, BCO u proseku daje rešenja sa najnižim vrednostima funkcije cilja u odnosu na EA i GVNS. Međutim, cGA pokazuje najbolje rezultate u odnosu na prosečne troškove $AvgC$ i u odnosu na prosečno procentualno odstupanje formiranih rešenja $G\%$.

Najbolje rezultate u odnosu na prosečno minimalno procesorsko vreme pokazuje cGA, a zatim slede GVNS, BCO, VND i EA. Prosečno minimalno procesorsko vreme za cGA, GVNS, BCO, VND i EA je 50.90, 67.14, 86.29, 92.63 i 212.77 sekundi, u datom redosledu. Odnosno, može se reći da je cGA za MCHBAP 31.91% brži od GVNS metode, 69.52% brži od BCO metode, 81.99% brži od VND metode i 318.01% brži od EA algoritma.

Na osnovu prikazanih eksperimentalnih rezultata za MCHBAP, može se zaključiti da je u odnosu na vremena potrebna za generisanje najboljih rešenja cGA superiornija metoda u odnosu na ostale predložene metaheurističke metode za MCHBAP na generisanim test instancama. Predložena GVNS metoda je najbolja na realnim test instancama iz literature, a cGA je sledi sa prosečnim vremenom izvršavanja od 5.01 sekunde. Imajući u vidu da i ostale metaheurističke metode imaju malo prosečno procentualno odstupanje od najboljeg formiranog rešenja, i da su performanse cGA algoritma dobre na obe klase test instanci, cGA se može smatrati za najpogodniju metodu za rešavanje statičkog MCHBAP-a.

6.2 Rezultati i poređenja za DMCHBAP

Pretražujući postojeću literaturu o varijantama BAP-a i analizirajući pregledne radove koje se odnose na ove probleme, može se uočiti da je rad Umanga i sar. [171] jedini rad koji se bavi hibridnom varijantom dinamičkog alociranja brodova. Autori

su razmatrali hibridni BAP u lukama rasutog tereta sa ciljem minimizacije ukupnog vremena obrade brodova, primenom optimizacije škripećeg točka (SWO). Predloženi SWO algoritam iz rada [171] ne može se direktno primeniti na DMCHBAP razmatran u ovoj disertaciji, imajući u vidu da je funkcija cilja DMCHBAP-a različita od funkcije cilja u radu [171]. Osim toga i tip tereta je različit, jer DMCHBAP uključuje kontejnerski teret.

Sledeći ideje iz rada Umanga i sar. [171], razvijeno je nekoliko različitih varijanti SWO algoritma prilagođenih za rešavanje DMCHBAP-a. Sve varijante SWO algoritma polaze od dopustivog rešenja koje se dinamički menja na osnovu prioriteta brodova. Brodovi sa većim troškovima u trenutnom rešenju imaju veći prioritet, odnosno, veću verovatnoću da budu ranije izabrani u narednoj alokaciji brodova. Nakon formiranja početnog rešenja, svi brodovi se sortiraju u nerastući redosled troškova trenutne alokacije. Brodovi se redom alociraju na dopustive pozicije odabrane ruletskom selekcijom na osnovu troškova alokacije. U slučaju kada ovako opisana alokacija vodi ka nedopustivom rešenju, formira se novo slučajno generisano početno rešenje. U suprotnom, ako alokacija vodi ka kompletnom dopustivom rešenju, algoritam započinje novu iteraciju tako što ponovo sortira brodove i dodeljuje im nove prioritete. Kao i kod ostalih metaheuristika, kriterijum zaustavljanja za SWO metodu je maksimalno vreme izvršavanja od 10 minuta. SWO algoritam je implementiran u *Wolfram Mathematica v8.0* programskom jeziku i testiran je na istoj platformi kao i predložene metaheuristike. Na svakoj test instanci SWO je testiran 10 puta. U nastavku ovog odeljka, prikazani su samo rezultati najbolje SWO varijante.

Dinamička varijanta alokacije brodova, DMCHBAP, je testirana na četiri klase generisanih test instanci. U tabelama 6.10 i 6.11 prikazani su rezultati poređenja CPLEX rešavača i implementiranih metaheurističkih metoda za DMCHBAP na prvoj klasi test instanci. Prva kolona tabele 6.10, označena sa *Klasa* sadrži dimenziju instance u formi $m \times T - l$, gde m predstavlja broj vezova, T prikazuje broj vremenskih jedinica u horizontu planiranja, dok je sa l označen broj brodova. Druga kolona sadrži identifikator (indeks) svake instance u odgovarajućoj klasi. Sledeće dve kolone se odnose na rezultate CPLEX rešavača, i sadrže vrednost funkcije cilja optimalnog rešenja OPT i odgovarajuće vreme izvršavanja dato u sekundama, označeno sa $Time$. Rezultati koji se odnose na najbolju SWO varijantu su prikazani u sledeće četiri kolone. U koloni *Best* data je vrednost najmanjih ukupnih troškova dobijena nakon deset izvršavanja SWO metode, dok sledeće dve kolone sa-

drže prosečne vrednosti ukupnih troškova $AvgC$ i prosečno vreme izvršavanja $AvgT$, dobijene na osnovu deset izvršavanja. Da bi se odredio kvalitet generisanih SWO rešenja, u koloni $G\%$ je prikazano prosečno procentualno odstupanje koje se računa po formuli $100 \cdot \frac{AvgC-OPT}{OPT}$. Sledeće tri kolone tabele 6.10 sadrže rezultate koji se odnose na VND metaheuristiku. Kolona $Best$ sadrži vrednost ukupnih troškova za najbolje generisano rešenje problema, dok kolona $Time$ prikazuje procesorsko vreme (izraženo u sekundama) potrebno da VND metoda generiše najbolje rešenje. Procentualno odstupanje $G\%$ najboljeg VND rešenja od najboljeg poznatog se računa prema formuli $100 \cdot \frac{Best-OPT}{OPT}$. Rezultati za MS-VND, GVNS i SVNS u tabeli 6.10, prikazani su na isti način kao i za SWO metodu. Poslednji red tabele 6.10 predstavlja prosečne vrednosti svih predstavljenih rezultata po kolonama. Radi lakšeg uočavanja metode sa najboljim performansama u odnosu na kvalitet rešenja, u tabeli 6.10 je istaknuto najbolje poznato (optimalno) rešenje za svaku instancu. Osim toga, u tabeli 6.10 je istaknuto je i najbolje prosečno procesorsko vreme za metodu sa najboljim performansama u odnosu na vreme izvršavanja.

Na osnovu rezultata prikazanih u tabeli 6.10, može se uočiti da sve četiri metaheuristike zasnovane na lokalnom pretraživanju, za svaki test primer malih dimenzija, generišu optimalno rešenje dobijeno CPLEX rešavačem. Najbolju stabilnost pokazuje SVNS metoda sa prosečnim procentualnim odstupanjem od 0%. Drugim rečima, SVNS je dostigao optimalno rešenje u svih deset izvršavanja za svaku test instancu. Rezultati iz kolone $G\%$ ukazuju da su MS-VND i GVNS metode takođe stabilne, sa odgovarajućim prosečnim procentualnim odstupanjima od 0.03% i 0.09%. Metoda škripećeg točka je pokazala loše performanse na skupu instanci malih dimenzija. Rešenja SWO metode su daleko od optimalnih vrednosti za sve razmatrane test primere. Rešenja najbolje varijante SWO metode odstupaju od optimalne vrednosti za čak 79.68%.

U odnosu na vremena izvršavanja, GVNS metoda je najbrže generisala najbolja rešenja, a za njom slede VND, SVNS i MS-VND, dok je SWO najsporija metoda. Svih pet predloženih metaheuristika su značajno brže u odnosu na CPLEX rešavač, kojem je u proseku potrebno 2436.81 sekundi da generiše optimalno rešenje na test primerima iz prvog skupa instanci. Prosečna vremena izvršavanja predloženih VNS metaheuristika su: 15.74 s za GVNS, 19.58 s za VND, 25.44 s za SVNS i 77.34 s za MS-VND, dok je SWO metodi potrebno prosečno 241.40 s da generiše najbolje rešenje. Drugim rečima, predloženi GVNS je više od 154 puta brži od CPLEX-a, dok je 1.24, 1.62, 4.91 i 15.34 puta brži od VND, SVNS, MS-VND i SWO metode,

u tom redosledu.

Na osnovu rezultata prikazanih u tabeli 6.10, očigledno je da čak i najbolja varinata SWO metode nema zadovoljavajuće rezultate u odnosu na kvalitet DMCHBAP rešenja i u odnosu na vreme izvršavanja na instancama iz prve grupe, koje su malih dimenzija. Zbog toga je SWO isključen iz narednih eksperimenata na instancama većih dimenzija.

Tabela 6.11 sadrži poređenja rezultata CPLEX rešavača i predloženih varijanti VNS metode, na skupu najvećih primera iz prve klase instanci, koje karakterišu dimenzije $m = 8$, $T = 25$, $l = 25$. Imajući u vidu da se odluke u luci donose veoma brzo (na nivou minuta), za ovaj skup instanci je izvršavanje CPLEX-a vremenski ograničeno na 1 sat.

Prva kolona tabele 6.11 sadrži specifikaciju test primera, dok druga kolona, označena sa BK, sadrži najbolju poznatu vrednost funkcije cilja dobijenu CPLEX-om ili nekom od predloženih VNS metaheuristika. Sledeće dve kolone se odnose na CPLEX rešavač i sadrže donje i gornje ograničenje vrednosti funkcije cilja optimalnog rešenja. Ostatak tabele 6.11 prikazuje rezultate predloženih VNS metoda: VND, MS-VND, GVNS i SVNS. Rezultati ovih metaheuristika su prikazani na isti način kao u slučaju tabele 6.10. Poslednji red tabele 6.11 je *prosek* i sadrži prosečne vrednosti rezltata prikazanih u kolonama.

Rezultati prikazani u tabeli 6.11 pokazuju da su sve četiri predložene VNS metaheuristike poboljšale gornje granice dobijene CPLEX-om, osim u slučaju jedne instance kada se najbolje rešenje četiri metaheuristike poklapa sa gornjom granicom optimalnog rešenja koju je vratio CPLEX. Kao i u slučaju test instanci manjih dimenzija iz prvog skupa i za skup instanci prvog skupa sa dimenzijom 8x25-25, VND i SVNS pokazuju najbolju stabilnost, sa prosečnim procentualnim odstupanjima od 0.11% i 0.47%. Za MS-VND i GVNS metode, vrednosti za $G\%$ su redom 1.04% i 2.42%, što ukazuje da ove dve VNS metode takođe imaju dobru stabilnost. U proseku, VND se pokazao kao najbrži metod na prvom skupu instanci većih dimenzija, a zatim slede GVNS, SVNS i MS-VND. Prosečna vremena izvršavanja predloženih VNS metaheuristika su: 30.75 s za VND, 103.47 s za GVNS, 129.36 s za SVNS i 178.63 s za MS-VND. To znači da je predloženi VND 3.36, 4.21 i 5.81 puta brži od GVNS, SVNS i MS-VND.

Tabela 6.10: Rezultati dobijeni CPLEX rešavačem i SWO i VNS meteheuristikama za DMCHBAP na prvoj klasi test instanci ($m = 8$, $T = 15, 20$, $l = 10, 15, 20$)

		CPLEX		SWO				VND			MS-VND				GVNS				SVNS			
Klasa	*i*	*OPT*	*Time*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
8x15-10	1	369	40.03	465	483.9	143.52	31.14	**369**	97.31	0.00	**369**	**369**	97.25	0.00	**369**	**369**	9.86	0.00	**369**	**369**	**9.66**	0.00
	2	208	10.90	291	332.1	270.82	59.66	**208**	0.10	0.00	**208**	**208**	0.10	0.00	**208**	**208**	0.02	0.00	**208**	**208**	**0.01**	0.00
	3	188	18.19	303	332.9	171.51	77.07	**188**	13.69	0.00	**188**	**188**	13.82	0.00	**188**	**188**	**1.39**	0.00	**188**	**188**	1.41	0.00
	4	180	18.36	340	398.5	293.66	121.39	**180**	0.28	0.00	**180**	**180**	0.26	0.00	**180**	**180**	**0.03**	0.00	**180**	**180**	**0.03**	0.00
	5	225	12.48	360	396.7	227.04	76.31	**225**	0.12	0.00	**225**	**225**	0.12	0.00	**225**	**225**	**0.02**	0.00	**225**	**225**	**0.02**	0.00
8x15-15	1	360	1402.45	614	641.5	169.84	78.19	**360**	**2.95**	0.00	**360**	**360**	104.26	0.00	**360**	**360**	36.67	0.00	**360**	**360**	54.13	0.00
	2	269	486.57	511	576.5	233.45	114.31	**269**	**0.83**	0.00	**269**	**269**	28.20	0.00	**269**	**269**	12.68	0.00	**269**	**269**	12.06	0.00
	3	362	203.32	527	551	222.75	52.21	**362**	32.00	0.00	**362**	**362**	55.21	0.00	**362**	**362**	18.40	0.00	**362**	**362**	**13.85**	0.00
	4	404	547.02	624	680.1	183.04	68.34	**404**	**1.22**	0.00	**404**	**404**	26.58	0.00	**404**	**404**	21.95	0.00	**404**	**404**	37.94	0.00
	5	553	1500.74	719	790.8	227.08	43.00	**553**	5.14	0.00	**553**	**553**	3.62	0.00	**553**	**553**	11.08	0.00	**553**	**553**	**2.69**	0.00
8x20-20	1	565	4323.16	856	986.2	275.29	74.55	**565**	5.22	0.00	**565**	**565**	5.13	0.00	**565**	**565**	0.54	0.00	**565**	**565**	**0.53**	0.00
	2	416	5391.39	731	850.9	362.84	104.54	**416**	**6.81**	0.00	**416**	**416**	93.05	0.00	**416**	**416**	25.31	0.00	**416**	**416**	37.55	0.00
	3	392	4732.67	799	862.1	292.24	119.92	**392**	**23.81**	0.00	**392**	392.6	266.54	0.15	**392**	**392**	43.38	0.00	**392**	**392**	53.53	0.00
	4	509	3214.67	923	958.8	251.23	88.37	**509**	84.38	0.00	**509**	509.1	177.48	0.02	**509**	**509**	**13.38**	0.00	**509**	**509**	18.97	0.00
	5	600	14650.23	1034	1117.5	296.70	86.25	**600**	**19.84**	0.00	**600**	602	288.54	0.33	**600**	608.4	41.34	1.40	**600**	**600**	139.15	0.00
prosek:		373.33	2436.81	606.47	663.97	241.40	79.68	**373.33**	19.58	0.00	**373.33**	373.51	77.34	0.03	**373.33**	373.89	**15.74**	0.09	**373.33**	**373.33**	25.44	0.00

Tabela 6.11: Rezultati dobijeni CPLEX rešavačem i VNS meteheuristikama za DMCHBAP na prvoj klasi test instanci ($m = 8$, $T = 25$, $l = 25$)

			CPLEX		VND			MS-VND				GVNS				SVNS			
Klasa	*i*	*BK*	*LB*	*UB*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
8x25-25	1	505	395	534	**505**	49.93	0.00	**505**	**505.00**	115.64	0.00	**505**	**505.00**	**7.16**	0.00	**505**	**505.00**	45.28	0.00
	2	645	349	887	646	**43.59**	0.16	**645**	660.80	285.09	2.45	646	657.00	181.34	1.86	**645**	647.20	137.45	0.34
	3	395	346	395	**395**	12.26	0.00	**395**	**395.00**	**6.23**	0.00	**395**	**395.00**	88.41	0.00	**395**	**395.00**	27.76	0.00
	4	474	376	485	**474**	**41.25**	0.00	**474**	481.70	114.96	1.62	**474**	494.40	92.58	4.30	**474**	479.70	213.03	1.20
	5	508	318	667	510	**6.71**	0.39	**508**	513.8	371.24	1.14	**508**	538.1	147.87	5.93	**508**	**512.1**	223.28	0.81
prosek:		505.40	356.80	593.60	506.00	**30.75**	0.11	**505.40**	511.26	178.63	1.04	505.60	517.90	103.47	2.42	**505.40**	**507.80**	129.36	0.47

Tabele 6.12, 6.13 i 6.14 prikazuju rezultate testiranja na drugom, trećem i četvrtom skupu instanci. Ove tri klase test primera sadrže instance većih dimenzija koje CPLEX nije rešio do optimalnosti. Zbog toga, tabele 6.12, 6.13 i 6.14 sadrže samo poređenja rezultata predloženih metaheuristika za rešavanje DMCHBAP-a, odnosno rezultate dobijene sa BCOi, cGA, VND, MS-VND, GVNS i SVNS metodom. Prva kolona tabele 6.12 sadrži broj brodova l. Druga kolona, koja ima naziv i, prikazuje indeks posmatrane instance, dok se treća kolona sa nazivom BK, odnosi na najbolju poznatu vrednost troškova. Rezultati za BCOi, cGA, VND, MS-VND, GVNS i SVNS su prikazani na isti način kao i u tabeli 6.10. Kako optimalno rešenje za ove instance nije poznato, prosečno procentualno odstupanje $G\%$ za VND se računa kao $100 \cdot \frac{Best-BK}{BK}$, dok se $G\%$ za BCOi, cGA, MS-VND, GVNS i SVNS računa po formuli $100 \cdot \frac{AvgC-BK}{BK}$. U tabeli 6.12 su istaknuta najbolja poznata rešenja i najkraća vremena izvršavanja predloženih metoda za svaku test instancu. Tabele 6.13 i 6.14 imaju istu strukturu kao tabela 6.12, i sadrže poređenja rezultata predloženih metaheuristika pri rešavanju DMCHBAP primera iz trećeg i četvrtog skupa instanci.

Kao što se vidi iz tabele 6.12, BCOi i cGA su dobili najbolje poznato rešenje za sve primere iz druge klase instanci. Osim toga cGA je generisao najbolje poznato rešenje u svih deset prolaza za sve instance ove klase, dok je BCOi u svih deset izvršavanja algoritma uspeo da generiše najbolje rešenje za 26 od 30 test instanci. Drugim rečima, prosečno procentualno odstupanje cGA i BCOi metode od najboljeg rešenja je redom 0.00% i 0.01%. Od 30 test instanci iz drugog skupa, VND, MS-VND, GVNS i SVNS su generisali najbolje poznato rešenje u 12, 12, 15 i 22 primera. Međutim, i pored toga, prosečna procentualna odstupanja od najboljeg rešenja za ove metode su veoma mala: 3.30% za VND, 3.57% za MS-VND, 2.56% za GVNS i 1.08% za SVNS. U odnosu na prosečno minimalno vreme izvršavanja, najbolje performanse pokazuje MS-VND, a zatim slede GVNS, VND, cGA, BCOi i SVNS. Odgovarajuća prosečna minimalna vremena izvršavanja su redom 3.74, 5.37, 8.55, 33.38, 48.60 i 74.35 sekundi. To znači da je MS-VND 1.44 puta brži od GVNS metode, 2.29 puta brži od VND metode, 8.92 puta brži od cGA, 12.99 puta brži od BCOi i 19.88 puta brži od SVNS metode.

Rezultati prikazani u tabeli 6.13 na test instancama većih dimenzija iz treće grupe, pokazuju da cGA ostaje superioran metod u pogledu kvaliteta rešenja, u poređenju sa ostalih pet predloženih metoda. Predloženi kombinovani GA je generisao najbolje rešenje najmanje jednom u deset izvršavanja za svaki test primer iz

treće klase instanci. BCOi je formirao najbolje rešenje u svih deset izvršavanja u 22 od 30 primera, dok je SVNS bio uspešan u svih deset izvršavanja na 19 od 30 primera iz trećeg skupa. Predloženi cGA samo u slučaju tri od 30 test instanci ne dostiže najbolje poznato rešenje u svakom izvršavanju. U slučaju instance $l = 55$ i $i = 3$ iz trećeg skupa test primera, BCOi, MS-VND, GVNS i SVNS imaju prosečnu vrednost troškova $AvgC = 1129.00$ (što odgovara najboljem rešenju), dok je cGA formirao neznatno lošije rešenje sa prosečnom vrednošću troškova $AvgC = 1130.00$. Za instancu $l = 60$ i $i = 2$ iz istog skupa primera, SVNS je jedini pokazao bolje performanse u poređenju sa cGA u odnosu na kvalitet prosečnih troškova. U slučaju instance $l = 60$ i $i = 2$, SVNS je generisao prosečne troškove jednake najboljim troškovima $AvgC = 1167.00$ dok cGA ima neznatno veću vrednost $AvgC = 1167.10$. Treći slučaj kada predloženi cGA ne generiše u svakom izvršavanju najbolje rešenje je za instancu $l = 60$ and $i = 5$, ali u tom slučaju cGA ima najbolje prosečne troškove $AvgC = 1135.30$. Predloženi BCOi i SVNS imaju slične performanse: BCOi generiše bar jednom najbolje rešenje u 28 od 30 instanci dok SVNS generiše najbolje rešenje u 24 od 30 primera. Drugu grupu metoda sa sličnim performansama u pogledu kvaliteta rešenja čine VND, MS-VND i GVNS. VND generiše najbolje rešenje u 16 od 30 slučajeva, MS-VND u 15 od 30 primera i GVNS u 21 od 30 primera. Sve predložene metode imaju malo prosečno procentualno odstupanje od najboljeg rešenja: 0.01% za cGA, 0.04% za BCOi, 0.80% za SVNS, 1.32% za GVNS, 1.67% za VND i 1.84% za MS-VND. Najkraće vreme izvršavanja ima MS-VND sa 8.38 s, a zatim slede VND (13.84 s), GVNS (14.06 s), cGA (41.14 s), BCOi (43.35 s) i SVNS (45.49 s). Dakle, MS-VND je 1.65 puta brži od VND, 1.68 puta brži od GVNS, 4.91 puta brži od cGA, 5.17 puta brži od BCOi i 5.43 puta brži od SVNS.

Na osnovu prikazanih rezultata eksperimenata, može se uočiti da su prosečna procentualna odstupanja od najboljeg rešenja kao i vremena izvršavanja predloženih algoritama vrlo mala za sve implementirane metode, i zbog toga se sve predložene metode mogu smatrati pogodnim za rešavanje DMCHBAP-a. Međutim, u odnosu na kvalitet generisanih rešenja, na obe klase test instanci najbolje performanse pokazuje cGA, dok MS-VND generiše kvalitetna rešenja u kratkom vremenu izvršavanja.

Rezultati i poređenja predloženih metaheuristika na četvrtom skupu instanci, koje imaju najveću dimenziju, dati su u tabeli 6.14. Ove test instance su ujedno i najkomplikovanije, jer sa porastom broja brodova, raste i stepen njihove konfliktnosti. Naime, sve više brodova pretenduje na iste referentne tačke, što u velikoj

meri komplikuje efikasno alociranje brodova. Rezultati prikazani u tabeli 6.14 pokazuju da svih šest metoda zadržavaju stabilne performanse sa povećanjem dimenzije problema. Predloženi cGA u 32 od 40 testiranih instanci iz četvrte grupe generiše najbolje rešenje, a u slučaju 10 instanci generiše najbolje rešenje u svih deset izvršavanja. U proseku, najbrži metod je VND (77.75 s), međutim, ova metaheuristika ima najveće prosečno procentualno odstupanje od najboljeg rešenja (4.80%) u odnosu na ostale metode. Međutim, i ostalih pet predloženih metoda takođe imaju mala vremena izvršavanja, bez značajnih razlika. Vremena izvršavanja za BCOi, cGA, MS-VND, GVNS i SVNS, se nalaze u intervalu između 111.30 s i 138.63 s. U proseku, VND je 1.43 puta brži od cGA, 1.50 puta brži od GVNS, 1.59 puta brži od SVNS, 1.70 puta brži od BCOi i 1.78 puta brži od MS-VND, na instancama iz četvrte klase. Predložene populacione metode, cGA i BCOi, pokazuju najbolje performanse u pogledu stabilnosti, jer su prosečna procentualna odstupanja od najboljeg rešenja 1.63% i 1.88%. Međutim, i implementirane VNS metode imaju vrlo mala odstupanja od najboljeg rešenja (sve vrednosti G% su ispod 5%).

Rezultati prikazani u tabeli 6.14 pokazuju da sve predložene metode za rešavanje DMCHBAP-a, ostaju stabilne i pokazuju dobre performanse kad se primene na teške test instance sa velikim brojem alociranih brodova. Ovi rezultati potvrđuju da sve predložene metaheuristike predstavljaju pogodne metode za rešavanje DMCHBAP-a. Sve metode daju rešenja sa malim odstupanjem od najboljih vrednosti, dok očekivano vreme izvršavanja na velikim instancama ostaje u okviru poželjnih, malih vrednosti.

Tabela 6.12: Rezultati dobijeni BCOi, cGA i VNS meteheuristikama za DMCHBAP na drugoj klasi test instanci ($m = 8$, $T = 112$, $l = 35, 40, 45$)

			BCOi				cGA				VND			MS-VND				GVNS				SVNS			
l	i	BK	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
35	1	527	**527**	**527.00**	49.80	0.00	**527**	**527.00**	27.97	0.00	**527**	8.25	0.00	**527**	**527.00**	**3.53**	0.00	**527**	**527.00**	11.23	0.00	**527**	**527.00**	5.41	0.00
	2	711	**711**	**711.00**	0.21	0.00	**711**	**711.00**	6.47	0.00	**711**	0.66	0.00	**711**	**711.00**	0.27	0.00	**711**	**711.00**	0.15	0.00	**711**	**711.00**	**0.08**	0.00
	3	973	**973**	**973.00**	0.19	0.00	**973**	**973.00**	5.58	0.00	**973**	0.63	0.00	**973**	**973.00**	0.25	0.00	**973**	**973.00**	0.15	0.00	**973**	**973.00**	**0.08**	0.00
	4	566	**566**	**566.00**	0.35	0.00	**566**	**566.00**	3.84	0.00	**566**	0.58	0.00	**566**	**566.00**	0.24	0.00	**566**	**566.00**	0.15	0.00	**566**	**566.00**	**0.08**	0.00
	5	536	**536**	**536.00**	7.01	0.00	**536**	**536.00**	10.58	0.00	**536**	6.97	0.00	**536**	**536.00**	2.99	0.00	**536**	**536.00**	0.79	0.00	**536**	**536.00**	**0.38**	0.00
	6	528	**528**	**528.00**	0.18	0.00	**528**	**528.00**	3.95	0.00	**528**	0.58	0.00	**528**	**528.00**	0.26	0.00	**528**	**528.00**	0.15	0.00	**528**	**528.00**	**0.08**	0.00
	7	757	**757**	**757.00**	17.23	0.00	**757**	**757.00**	7.19	0.00	787	3.83	3.96	787	787.00	1.62	3.96	784	786.70	**0.48**	3.92	**757**	773.40	169.04	2.17
	8	567	**567**	**567.00**	12.77	0.00	**567**	**567.00**	6.73	0.00	583	0.63	2.82	598	598.00	0.26	5.47	583	583.00	**0.16**	2.82	**567**	572.60	81.77	0.99
	9	957	**957**	**957.00**	0.19	0.00	**957**	**957.00**	7.69	0.00	**957**	0.69	0.00	**957**	**957.00**	0.28	0.00	**957**	**957.00**	0.16	0.00	**957**	**957.00**	**0.08**	0.00
	10	818	**818**	**818.00**	12.54	0.00	**818**	**818.00**	5.61	0.00	831	1.34	1.59	831	831.00	0.54	1.59	**818**	829.50	**0.20**	1.41	**818**	821.30	200.55	0.40
40	1	540	**540**	**540.00**	66.28	0.00	**540**	**540.00**	**14.87**	0.00	**540**	213.19	0.00	**540**	**540.00**	93.62	0.00	**540**	540.30	105.09	0.06	**540**	**540.00**	48.33	0.00
	2	577	**577**	**577.00**	0.49	0.00	**577**	**577.00**	7.03	0.00	**577**	0.8	0.00	**577**	**577.00**	0.34	0.00	**577**	**577.00**	0.18	0.00	**577**	**577.00**	**0.09**	0.00
	3	587	**587**	**587.00**	7.15	0.00	**587**	**587.00**	5.16	0.00	762	0.73	29.81	762	762.00	**0.31**	29.81	**587**	681.50	37.10	16.10	**587**	594.10	69.11	1.21
	4	874	**874**	**874.00**	36.23	0.00	**874**	**874.00**	10.96	0.00	920	0.53	5.26	920	920.00	**0.25**	5.26	888	910.90	0.26	4.22	**874**	875.40	159.17	0.16
	5	649	**649**	**649.00**	0.41	0.00	**649**	**649.00**	7.54	0.00	**649**	0.69	0.00	**649**	**649.00**	0.30	0.00	**649**	**649.00**	0.18	0.00	**649**	**649.00**	**0.10**	0.00
	6	852	**852**	**852.00**	26.51	0.00	**852**	**852.00**	37.53	0.00	915	0.84	7.39	915	915.00	0.40	7.39	873	903.20	**0.25**	6.01	**852**	862.60	88.32	1.24
	7	570	**570**	**570.00**	80.35	0.00	**570**	**570.00**	8.92	0.00	594	1.47	4.21	594	594.00	0.67	4.21	576	587.80	**0.50**	3.12	**570**	574.20	67.91	0.74
	8	655	**655**	**655.00**	0.29	0.00	**655**	**655.00**	11.10	0.00	**655**	0.84	0.00	**655**	**655.00**	0.38	0.00	**655**	**655.00**	0.21	0.00	**655**	**655.00**	**0.11**	0.00
	9	651	**651**	651.60	31.07	0.09	**651**	**651.00**	121.48	0.00	723	0.73	11.06	723	723.00	0.33	11.06	699	717.40	**0.22**	10.20	660	680.40	232.42	4.52
	10	951	**951**	**951.00**	74.77	0.00	**951**	**951.00**	34.01	0.00	967	1.02	1.68	967	967.00	0.46	1.68	967	967.00	0.22	1.68	967	967.00	**0.12**	1.68
45	1	693	**693**	**693.00**	0.37	0.00	**693**	**693.00**	16.56	0.00	**693**	1.17	0.00	**693**	**693.00**	0.52	0.00	**693**	**693.00**	0.22	0.00	**693**	**693.00**	**0.11**	0.00
	2	826	**826**	826.60	131.66	0.07	**826**	**826.00**	129.87	0.00	844	1.03	2.18	844	844.00	0.45	2.18	844	844.00	0.26	2.18	844	844.00	**0.12**	2.18
	3	848	**848**	**848.00**	187.27	0.00	**848**	**848.00**	13.76	0.00	857	1.2	1.06	857	857.00	0.54	1.06	**848**	854.30	**0.35**	0.74	**848**	848.90	176.12	0.11
	4	879	**879**	880.20	91.28	0.14	**879**	**879.00**	42.33	0.00	885	1.16	0.68	922	922.00	0.49	4.89	885	885.00	**0.29**	0.68	879	882.00	139.15	0.34
	5	602	**602**	**602.00**	30.36	0.00	**602**	**602.00**	16.91	0.00	656	0.94	8.97	656	656.00	**0.42**	8.97	623	636.90	0.63	5.80	**602**	611.30	236.51	1.54
	6	1123	**1123**	**1123.00**	13.06	0.00	**1123**	**1123.00**	301.78	0.00	1212	0.8	7.93	1212	1212.00	0.36	7.93	1212	1212.00	**0.24**	7.93	**1123**	1203.10	99.02	7.13
	7	759	**759**	**759.00**	102.05	0.00	**759**	**759.00**	40.38	0.00	786	1.06	3.56	786	786.00	0.45	3.56	770	782.60	**0.35**	3.11	761	768.80	248.18	1.29
	8	1044	**1044**	1044.10	262.29	0.01	**1044**	**1044.00**	50.83	0.00	1045	1.63	0.10	1045	1045.00	0.69	0.10	1045	1045.00	0.33	0.10	1045	1045.00	**0.16**	0.10
	9	813	**813**	**813.00**	111.50	0.00	**813**	**813.00**	15.41	0.00	834	1.34	2.58	834	834.00	0.56	2.58	834	834.00	**0.30**	2.58	824	832.80	75.42	2.44
	10	942	**942**	**942.00**	104.04	0.00	**942**	**942.00**	29.49	0.00	981	1.17	4.14	992	992.00	0.55	5.31	981	981.00	**0.30**	4.14	956	981.90	132.63	4.24
prosek:		745.83	**745.83**	745.92	48.60	0.01	**745.83**	**745.83**	33.38	0.00	769.80	8.55	3.30	771.90	771.90	**3.74**	3.57	757.63	765.14	5.37	2.56	748.20	755.03	74.35	1.08

Tabela 6.13: Rezultati dobijeni BCOi, cGA i VNS meteheuristikama za DMCHBAP na trećoj klasi test instanci ($m = 13$, $T = 112$, $l = 50, 55, 60$)

			BCOi				cGA				VND			MS-VND				GVNS				SVNS			
l	i	BK	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
50	1	575	**575**	**575.00**	17.53	0.00	**575**	**575.00**	**9.86**	0.00	**575**	44.63	0.00	**575**	**575.00**	48.02	0.00	**575**	**575.00**	76.30	0.00	**575**	**575.00**	26.63	0.00
	2	677	**677**	**677.00**	20.69	0.00	**677**	**677.00**	18.64	0.00	**677**	**14.70**	0.00	**677**	**677.00**	16.85	0.00	**677**	**677.00**	33.54	0.00	**677**	**677.00**	17.03	0.00
	3	798	**798**	**798.00**	0.90	0.00	**798**	**798.00**	8.27	0.00	**798**	1.31	0.00	**798**	**798.00**	1.26	0.00	**798**	**798.00**	0.28	0.00	**798**	**798.00**	**0.15**	0.00
	4	532	**532**	**532.00**	0.46	0.00	**532**	**532.00**	7.19	0.00	**532**	1.42	0.00	**532**	**532.00**	1.35	0.00	**532**	**532.00**	0.28	0.00	**532**	**532.00**	**0.15**	0.00
	5	893	**893**	**893.00**	1.06	0.00	**893**	**893.00**	6.93	0.00	**893**	1.45	0.00	**893**	**893.00**	1.50	0.00	**893**	**893.00**	0.33	0.00	**893**	**893.00**	**0.17**	0.00
	6	504	**504**	**504.00**	22.05	0.00	**504**	**504.00**	11.01	0.00	530	**1.50**	5.16	530	530.00	1.59	5.16	**504**	511.00	25.15	1.39	**504**	**504.00**	198.51	0.00
	7	823	**823**	**823.00**	0.98	0.00	**823**	**823.00**	27.23	0.00	**823**	1.38	0.00	**823**	**823.00**	1.54	0.00	**823**	**823.00**	0.31	0.00	**823**	**823.00**	**0.16**	0.00
	8	688	**688**	**688.00**	48.81	0.00	**688**	**688.00**	5.09	0.00	689	1.36	0.15	689	689.00	1.47	0.15	**688**	688.90	**0.34**	0.13	**688**	**688.00**	79.20	0.00
	9	1003	1005	1006.90	126.75	0.39	**1003**	**1003.40**	330.27	0.04	1095	5.58	9.17	1095	1095.00	5.64	9.17	1009	1073.00	**1.09**	6.98	1005	1024.60	98.76	2.15
	10	926	**926**	**926.00**	0.95	0.00	**926**	**926.00**	42.24	0.00	**926**	1.34	0.00	**926**	**926.00**	1.35	0.00	**926**	**926.00**	**0.32**	0.00	**926**	926.90	28.64	0.10
55	1	634	**634**	**634.00**	0.48	0.00	**634**	**634.00**	6.40	0.00	**634**	1.56	0.00	**634**	**634.00**	1.58	0.00	**634**	**634.00**	0.30	0.00	**634**	**634.00**	**0.16**	0.00
	2	911	**911**	912.50	143.53	0.16	**911**	**911.00**	17.25	0.00	917	1.61	0.66	917	917.00	**1.60**	0.66	**911**	915.30	53.21	0.47	**911**	**911.00**	69.64	0.00
	3	1129	**1129**	**1129.00**	4.25	0.00	**1129**	1130.00	23.31	0.09	**1129**	8.13	0.00	**1129**	**1129.00**	8.19	0.00	**1129**	**1129.00**	1.10	0.00	**1129**	**1129.00**	**0.46**	0.00
	4	1033	**1033**	**1033.00**	0.60	0.00	**1033**	**1033.00**	25.57	0.00	**1033**	1.72	0.00	**1033**	**1033.00**	1.72	0.00	**1033**	**1033.00**	0.43	0.00	**1033**	**1033.00**	**0.18**	0.00
	5	1025	**1025**	**1025.00**	1.46	0.00	**1025**	**1025.00**	19.70	0.00	**1025**	2.17	0.00	**1025**	**1025.00**	2.09	0.00	**1025**	**1025.00**	0.50	0.00	**1025**	**1025.00**	**0.21**	0.00
	6	741	**741**	**741.00**	0.57	0.00	**741**	**741.00**	15.73	0.00	**741**	3.17	0.00	**741**	**741.00**	3.12	0.00	**741**	**741.00**	0.61	0.00	**741**	**741.00**	**0.26**	0.00
	7	881	**881**	**881.00**	0.46	0.00	**881**	**881.00**	13.73	0.00	**881**	1.75	0.00	**881**	**881.00**	1.69	0.00	**881**	**881.00**	0.45	0.00	**881**	**881.00**	**0.19**	0.00
	8	1017	**1017**	**1017.00**	26.20	0.00	**1017**	**1017.00**	19.99	0.00	1073	1.70	5.51	1073	1073.00	1.64	5.51	1073	1073.00	**0.45**	5.51	1047	1070.40	32.01	5.25
	9	929	**929**	**929.00**	15.98	0.00	**929**	**929.00**	32.14	0.00	944	3.06	1.61	944	944.00	2.97	1.61	935	941.90	**0.69**	1.39	935	939.90	73.58	1.17
	10	1045	**1045**	**1045.00**	61.81	0.00	**1045**	**1045.00**	18.15	0.00	1106	1.84	5.84	1106	1106.00	1.83	5.84	1106	1106.00	**0.44**	5.84	1050	1085.10	103.70	3.84
60	1	991	**991**	992.77	367.84	0.18	**991**	**991.00**	**21.34**	0.00	1003	253.69	1.21	1003	1003.00	119.42	1.21	**991**	**991.00**	115.15	0.00	**991**	**991.00**	57.62	0.00
	2	1167	1168	1168.00	13.69	0.09	**1167**	1167.10	18.23	0.01	1168	33.08	0.09	1168	1168.00	**13.25**	0.09	**1167**	1187.90	105.33	1.79	**1167**	**1167.00**	67.09	0.00
	3	783	**783**	**783.00**	0.63	0.00	**783**	**783.00**	12.52	0.00	**783**	2.42	0.00	**783**	**783.00**	0.91	0.00	**783**	**783.00**	0.42	0.00	**783**	**783.00**	**0.24**	0.00
	4	822	**822**	822.20	15.63	0.02	**822**	**822.00**	21.75	0.00	825	4.58	0.36	825	825.00	1.98	0.36	825	825.00	**0.67**	0.36	**822**	824.70	38.63	0.33
	5	1134	**1134**	1137.10	242.56	0.27	**1134**	**1135.30**	303.33	0.11	1166	4.70	2.82	1166	1166.00	2.02	2.82	1166	1166.00	0.69	2.82	1166	1166.00	**0.35**	2.82
	6	1143	**1143**	**1143.00**	2.24	0.00	**1143**	**1143.00**	87.60	0.00	**1143**	2.06	0.00	1201	1201.00	0.90	5.07	**1143**	**1143.00**	**0.47**	0.00	**1143**	1189.40	37.98	4.06
	7	732	**732**	**732.00**	0.71	0.00	**732**	**732.00**	23.38	0.00	**732**	2.09	0.00	**732**	**732.00**	0.95	0.00	**732**	**732.00**	0.50	0.00	**732**	**732.00**	**0.24**	0.00
	8	863	**863**	863.70	116.52	0.08	**863**	**863.00**	44.91	0.00	881	2.27	2.09	881	881.00	0.97	2.09	876	880.00	**0.71**	1.97	875	878.70	259.06	1.82
	9	978	**978**	**978.00**	22.16	0.00	**978**	**978.00**	16.41	0.00	1126	2.91	15.13	1126	1126.00	1.28	15.13	978	1081.50	**0.84**	10.58	**978**	999.20	148.42	2.17
	10	1311	**1311**	1311.10	23.06	0.01	**1311**	**1311.00**	26.10	0.00	1314	6.11	0.23	1314	1314.00	2.64	0.23	1314	1314.00	**0.89**	0.23	**1311**	1313.40	25.34	0.18
prosek:		889.60	889.70	890.01	43.35	0.04	**889.60**	**889.69**	41.14	0.01	905.40	13.84	1.67	907.33	907.33	**8.38**	1.84	895.60	902.65	14.06	1.32	892.50	897.84	45.49	0.80

Tabela 6.14: Rezultati dobijeni BCOi, cGA i VNS meteheuristikama za DMCHBAP na četvrtoj klasi test instanci ($m = 13$, $T = 112$, $l = 70, 80, 90, 100$)

			BCOi				cGA				VND			MS-VND				GVNS				SVNS			
l	*i*	*BK*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
70	1	652	670	670.00	151.97	2.76	**652**	**652.00**	**84.24**	0.00	670	308.26	2.76	**652**	658.90	418.89	1.06	668	674.20	280.49	3.40	664	672.40	483.81	3.13
	2	1284	1290	1290.00	8.55	0.47	**1284**	**1284.00**	54.05	0.00	1290	8.94	0.47	**1284**	1288.80	85.96	0.37	1290	1290.00	**0.95**	0.47	1290	1290.00	1.05	0.47
	3	806	**806**	**806.00**	100.75	0.00	**806**	**806.00**	102.36	0.00	808	17.44	0.25	808	808.00	17.88	0.25	808	808.00	**1.76**	0.25	808	808.00	1.97	0.25
	4	905	907	907.00	186.21	0.22	**905**	**905.00**	68.00	0.00	943	10.03	4.20	905	937.90	104.17	3.64	906	932.90	**1.07**	3.08	**905**	914.90	191.48	1.09
	5	848	**848**	**848.00**	5.73	0.00	**848**	**848.00**	60.15	0.00	**848**	9.53	0.00	**848**	**848.00**	9.87	0.00	**848**	**848.00**	**0.94**	0.00	**848**	**848.00**	1.10	0.00
	6	1014	1116	1143.20	118.89	12.74	**1014**	**1082.80**	42.96	6.79	1161	**9.44**	14.50	1138	1158.70	64.80	14.27	1088	1133.50	268.69	11.79	1115	1153.00	146.23	13.71
	7	930	**930**	**930.00**	8.00	0.00	**930**	**930.00**	76.54	0.00	**930**	12.49	0.00	**930**	**930.00**	12.94	0.00	**930**	**930.00**	**1.31**	0.00	**930**	**930.00**	1.46	0.00
	8	1073	**1073**	**1073.00**	196.48	0.00	**1073**	1079.60	69.71	0.62	1142	**14.34**	6.43	**1073**	1088.40	585.77	1.44	1081	1129.80	83.14	5.29	**1073**	1078.60	280.10	0.52
	9	1488	**1488**	**1488.00**	239.62	0.00	**1488**	1529.20	128.06	2.77	**1488**	796.38	0.00	**1488**	**1488.00**	208.97	0.00	**1488**	**1488.00**	82.29	0.00	**1488**	**1488.00**	**82.20**	0.00
	10	841	**841**	842.60	196.22	0.19	**841**	**841.00**	66.58	0.00	861	**12.47**	2.38	**841**	844.00	378.09	0.36	**841**	851.80	362.36	1.28	**841**	852.80	153.01	1.40
80	1	698	699	699.00	**3.66**	0.14	**698**	**698.00**	29.42	0.00	**698**	132.17	0.00	**698**	698.70	172.03	0.10	**698**	698.10	180.78	0.01	**698**	698.20	164.34	0.03
	2	775	**775**	**776.40**	148.33	0.18	**775**	777.00	84.42	0.26	782	**12.20**	0.90	**775**	779.70	127.29	0.61	**775**	780.00	159.63	0.65	779	779.90	228.20	0.63
	3	1026	**1026**	**1026.00**	17.23	0.00	1030	1035.00	91.03	0.88	**1026**	26.95	0.00	**1026**	**1026.00**	25.01	0.00	**1026**	**1026.00**	**2.70**	0.00	**1026**	**1026.00**	2.75	0.00
	4	1700	1714	**1726.80**	272.56	1.58	**1700**	1741.60	99.73	2.45	1778	15.03	4.59	1778	1778.00	16.04	4.59	1778	1778.00	**1.78**	4.59	1751	1774.60	197.96	4.39
	5	1238	**1238**	**1238.00**	4.70	0.00	**1238**	1238.40	93.36	0.03	**1238**	16.15	0.00	**1238**	**1238.00**	15.46	0.00	**1238**	**1238.00**	1.72	0.00	**1238**	**1238.00**	**1.71**	0.00
	6	872	882	882.00	196.98	1.15	**872**	**873.20**	75.72	0.14	888	**17.51**	1.83	883	887.50	95.95	1.78	879	887.10	43.71	1.73	878	878.00	43.22	0.69
	7	1075	1076	1076.00	9.61	0.09	1076	1083.60	130.95	0.80	1076	17.28	0.09	**1075**	**1075.90**	124.86	0.08	1076	1076.00	**1.61**	0.09	1076	1076.00	**1.61**	0.09
	8	1086	1131	**1131.00**	268.21	4.14	**1086**	1133.20	175.02	4.35	1141	13.79	5.06	1141	1141.00	13.28	5.06	1141	1141.00	**1.44**	5.06	1141	1141.00	**1.44**	5.06
	9	815	820	828.60	245.66	1.67	**815**	**817.40**	65.05	0.29	863	16.27	5.89	817	850.30	255.10	4.33	863	863.00	1.76	5.89	863	863.00	**1.75**	5.89
	10	765	**765**	**765.00**	10.84	0.00	**765**	**765.00**	103.17	0.00	**765**	17.10	0.00	**765**	**765.00**	15.98	0.00	**765**	**765.00**	**1.69**	0.00	**765**	**765.00**	**1.69**	0.00
90	1	1246	1292	1315.20	233.23	5.55	**1246**	1284.20	70.95	3.07	1408	**22.29**	13.00	1250	1346.20	314.04	8.04	1248	**1282.20**	385.92	2.91	1253	1278.70	347.53	2.62
	2	1056	1065	**1065.00**	123.50	0.85	1081	1093.80	70.44	3.58	1072	16.65	1.52	1072	1072.00	15.95	1.52	**1056**	1070.40	69.91	1.36	**1056**	1069.20	171.40	1.25
	3	1014	1044	1044.00	112.95	2.96	**1014**	**1022.00**	119.40	0.79	1084	**17.30**	6.90	1077	1083.30	118.53	6.83	1017	1047.10	274.55	3.26	1017	1025.70	347.21	1.15
	4	1593	1660	1709.20	111.07	7.29	**1593**	**1641.00**	110.85	3.01	1906	13.25	19.65	1906	1906.00	12.56	19.65	1783	1881.70	**1.26**	18.12	1906	1906.00	1.47	19.65
	5	1548	1551	**1551.00**	147.39	0.19	**1548**	1563.20	40.55	0.98	1634	43.21	5.56	1634	1634.00	38.94	5.56	1634	1634.00	**4.66**	5.56	1570	1621.20	192.27	4.73
	6	1518	1523	**1525.40**	277.25	0.49	**1518**	1527.00	155.45	0.59	1535	16.58	1.12	1535	1535.00	15.47	1.12	1535	1535.00	1.91	1.12	1535	1535.00	**1.75**	1.12
	7	1930	1964	**1964.00**	230.74	1.76	**1930**	2005.20	189.69	3.90	2042	**70.94**	5.80	1931	2023.70	583.92	4.85	1942	1996.70	383.17	3.46	1942	1981.70	304.55	2.68
	8	982	992	992.00	99.40	1.02	**982**	**987.60**	233.15	0.57	988	194.46	0.61	998	998.00	**16.00**	1.63	991	995.60	141.56	1.38	993	996.00	170.25	1.43
	9	1078	1174	1174.00	214.69	8.91	**1078**	**1102.00**	130.42	2.23	1286	24.41	19.29	1286	1286.00	**24.03**	19.29	1094	1214.40	539.30	12.65	1111	1235.50	324.48	14.61
	10	1482	**1482**	1567.80	120.79	5.79	1492	**1549.60**	116.79	4.56	1911	15.48	28.95	1911	1911.00	15.06	28.95	1760	1876.50	3.38	26.62	1711	1871.10	**3.02**	26.26
100	1	1534	1535	**1535.00**	144.71	0.07	1535	1593.00	**80.40**	3.85	1587	161.04	3.46	**1534**	1572.30	345.39	1.15	1536	1551.70	358.63	1.15	1536	1572.00	451.42	2.48
	2	1195	1199	1199.00	156.97	0.33	**1195**	**1195.00**	127.89	0.00	1253	26.30	4.85	1253	1253.00	24.90	4.85	**1195**	1195.90	327.84	0.08	1237	1251.40	**3.36**	4.72
	3	1282	1366	1366.00	151.33	6.55	**1282**	**1302.00**	164.81	1.56	1416	**16.66**	10.45	1343	1408.70	208.42	9.88	1302	1315.50	314.50	2.61	1289	1396.30	278.56	8.92
	4	1387	**1387**	**1387.00**	52.07	0.00	1396	1404.40	189.74	1.25	**1387**	92.83	0.00	**1387**	**1387.00**	80.41	0.00	**1387**	**1387.00**	9.27	0.00	**1387**	**1387.00**	**8.20**	0.00
	5	1786	1788	**1791.60**	141.81	0.31	**1786**	1808.00	152.64	1.23	1797	108.37	0.62	1797	1797.00	98.08	0.62	1797	1797.00	11.32	0.62	1797	1797.00	**9.91**	0.62
	6	1783	**1783**	**1783.00**	109.37	0.00	**1783**	1813.30	175.13	1.70	1831	14.33	2.69	1831	1831.00	**13.40**	2.69	1808	1824.70	32.66	2.34	1800	1825.60	52.40	2.39
	7	1233	1257	1257.00	127.53	1.95	**1233**	**1234.00**	198.34	0.08	1259	26.09	2.11	1235	1256.30	349.91	1.89	1259	1259.00	3.08	2.11	1259	1259.00	**2.73**	2.11
	8	1157	**1157**	**1157.00**	104.42	0.00	1182	1185.20	124.47	2.44	1161	659.30	0.35	1163	1169.30	144.68	1.06	1170	1170.00	**2.14**	1.12	1168	1169.80	2.49	1.11
	9	1530	1573	1573.00	183.34	2.81	1540	1563.00	162.92	2.16	1754	**21.96**	14.64	**1530**	1730.90	318.63	13.13	1533	1558.10	306.24	1.84	**1530**	**1548.00**	264.49	1.18
	10	1418	1436	1436.00	42.37	1.27	**1418**	**1426.00**	137.30	0.56	1436	64.82	1.27	1436	1436.00	58.34	1.27	1436	1436.00	**7.05**	1.27	1436	1436.00	12.87	1.27
prosek:		1191.1	1208.1	**1213.47**	131.88	1.88	**1193.2**	1210.49	111.30	1.63	1253.6	**77.75**	4.80	1231.8	1248.19	138.63	4.30	1216.8	1234.17	116.45	3.33	1217.8	1235.94	123.44	3.44

Glava 7

Zaključak

Problem dodele vezova (BAP) je jedan od najzastupljeniji problema u literaturi koja se bavi operacionim istraživanjima u pomorskom transportu. Poslednjih decenija, to dokazuje konstantan rast broja relevantnih radova koji se bave alokacijom brodova. Efikasno funkcionisanje kontejnerskog terminala je od posebnog značaja i za luku i za brodske kompanije koje zahtevaju brzu i kvalitetnu uslugu. Neophodno je da terminal pruža usluge koje omogućavaju kratko vreme zadržavanja brodova u luci, ali istovremeno i usluge koje redukuju operativne troškove terminala. To je razlog zbog kojeg menadžeri terminala teže ka razvijanju mehanizama koji povećavaju produktivnost terminala. Brzina generisanja kvalitetnih rešenja je od najvećeg značaja za dizajniranje efikasnih i pouzdanih sistema podrške donošenju odluka u kontejnerskom terminalu.

Efikasna dodela vezova mora da ispoštuje nekoliko važnih zahteva: stvarno vreme vezivanja broda treba da bude blizu očekivanog vremena dolaska broda u terminal, vremenski interval koji brod provodi na čekanju u sidrištu i dužina vremena istovara ili utovara broda moraju se svesti na minimum, potrebno je maksimalno smanjiti troškove nastale usled kašnjenja u odlasku broda i dodatne troškove manipulacije zbog neoptimalnih lokacija brodova na vezu. Da bi se ispunili navedeni zahtevi koji povećavaju produktivnost terminala i pri tome uzimaju u obzir zahteve vezane za vreme, resurse i cenu, u ovoj disertaciji je razmatran problem minimizacije ukupnih troškova terminala kod hibridne statičke i dinamičke dodele vezova. Preciznije, razmatrane su statička (MCHBAP) i dinamička (DMCHBAP) varijanta hibridne alokacije brodova sa minimizacijom ukupnih troškova alokacije, koje su od velikog značaja u pomorskom transportu.

Za obe varijante problema predloženi su matematički modeli koji sadrže sve na-

vedene parametre i obezbezbeđuju realizaciju svih procesa u predviđenom vremenu uz minimalne troškove. Statička varijanta BAP-a podrazumeva da vremena dolaska brodova nisu striktno ograničena: brod može da čeka u luci na opslužiavnje ili može da dođe u luku pre očekivanog vremena dolaska, tj. dozvoljeno je i požurivanje broda. U modelu za MCHBAP, troškovi vezivanja se sastoje od četiri komponente: troškova pozicioniranja broda (ukoliko brod nije alociran na omiljeni vez), troškova ubrzavanja i čekanja uzrokovanih odstupanjem od očekivanog vremena dolaska broda i troškova kašnjenja završetka obrade broda. Dinamički BAP bolje oslikava situaciju u luci od statičke varijante, jer nije realno očekivati da su svi brodovi već stigli u luku i da čekaju na opsluživanje. DMHCBAP uključuje stroža ograničenja na vreme vezivanja broda: brod ne može biti požurivan u dolasku, odnosno, brod ne može biti opsluživan pre očekivanog vremena dolaska. Zbog toga je u predloženom modelu za DMCHBAP dobijena struktura troškova vezivanja koji se sastoji od tri komponente: troškovi pozicioniranja broda na vez, troškovi čekanja i troškovi kašnjenja u završetku obrade brodova. U ovoj disertaciji je dokazano da MCHBAP i DMCHBAP spadaju u klasu NP-teških problema u jakom smislu.

Imajući u vidu ograničenja egzaktnih metoda pri rešavanju komplikovanih varijanti MCHBAP-a i DMCHBAP-a, aproksimativne tehnike rešavanja problema se nameću kao adekvatan pristup. U ovoj disertaciji su predložene i implementirane efikasne metode za dobijanje kvalitetnih rešenja MCHBAP-a i DMCHBAP-a u kratkom vremenu izvršavanja: evolutivni algoritam (EA), kombinovani genetski algoritam (cGA), optimizacija kolonijom pčela (BCO), optimizacija kolonijom pčela sa popravkom kompletnog rešenja (BCOi), metoda promenljivog spusta (VND), višestartna metoda promenljivog spusta (MS-VND), opšta metoda promenljivih okolina (GVNS) i adaptivna metoda promenljivih okolina (SVNS).

Statička varijanta razmatranog BAP-a je u ovoj disertaciji rešavana primenom EA, BCO, cGA, VND i GVNS metode. Za DMCHBAP je razvijeno šest metoda, BCOi, cGA, VND, MS-VND, GVNS i SVNS. Metaheuristike primenjene na MCHBAP i DMCHBAP koriste sofisticiranu strukturu liste, koja predstavlja osnovu njihovih implementacija. Za svaki brod formirana je lista uređenih trojki koje sadrže informaciju o vezu, vremenskoj koordinati i ukupnim troškovima broda. Implementirane metaheuristike konstantno ažuriraju ove liste i na taj način obezbeđuju da u svakom momentu lista sadrži samo elemente koji odgovaraju preostalim dopustivim pozicijama broda. Osim toga, liste su sortirane tako da obezbeđuju jednostavno pronalaženje pozicija koje imaju manje troškove od vrednosti troškova trenutne re-

ferentne tačke broda i jednostavno detektovanje nedopustivih rešenja problema.

Sve predložene metaheuristike za MCHBAP su poređene međusobno, kao i sa egzaktnim rešavačem, na dva skupa test instanci: realnim test primerima poznatim iz literature i na skupu generisanih test instanci. Sve metaheurističke metode su dostigle optimalno rešenje dobijeno egzaktnim rešavačem na realnim primerima. Osim toga, EA, cGA, BCO i GVNS su dostigle optimalno rešenje u svakom izvršavanju algoritma. Sve metaheurističke metode su nadmašile egzaktni rešavač u odnosu na prosečno vreme izvršavanja, dok su cGA, VND i GVNS imale kraće prosečno vreme izvršavanja u odnosu na egzaktni rešavač na svakoj test instanci. Među četiri predložene metaheuristike za MCHBAP, najkraće prosečno vreme izvršavanja na svim realnim test instancama ima GVNS. Na generisanim test instancama MCHBAP-a, cGA pokazuje najbolje performanse u odnosu na druge metaheurističke metode u pogledu vremena potrebnih za generisanje najboljih rešenja. Imajući u vidu da i ostale metaheurističke metode imaju malo prosečno procentualno odstupanje od najboljeg formiranog rešenja, i da su performanse cGA algoritma dobre na obe klase test instanci, cGA se može smatrati za najpogodniju metodu za rešavanje MCHBAP-a.

Efikasnost predloženih metaheuristika pri rešavanju DMCHBAP-a, međusobna poređenja i poređenje sa egzaktnim rešavačem CPLEX, su ispitivani testiranjem na četiri klase generisanih test instanci. Eksperimentalni rezultati su pokazali da sve četiri VNS metaheuristike, za svaki test primer malih dimenzija, dostižu optimalno rešenje dobijeno CPLEX rešavačem. Najbolju stabilnost pokazuje SVNS metoda, dok je u odnosu na vremena izvršavanja GVNS metoda imala najbolje rezultate. Sve predložene metaheuristike su značajno brže u odnosu na CPLEX rešavač i pokazuju dobru stabilnost sa malim odstupanjima od optimuma. Na test instancama većih dimenzija, cGA je superioran u odnosu na ostale predložene metaheurističke metode u smislu kvaliteta generisanih rešenja. Prosečna procentualna odstupanja najboljih rešenja predloženih metaheuristika od najboljih poznatih rešenja, kao i odgovarajuća prosečna vremena izvršavanja su veoma mala. Iz tog razloga se sve predložene metode mogu smatrati pogodnim za rešavanje DMCHBAP-a. Ipak, u pogledu kvaliteta generisanih rešenja, na svim klasama test instanci najbolje performanse pokazuje cGA. Eksperimentalni rezultati pokazuju da BCOi, cGA, VND, MS-VND, GVNS i SVNS metode prilagođene za rešavanje DMCHBAP-a, ostaju stabilne i zadržavaju dobre performanse i pri rešavanju teških test instanci sa velikim brojem alociranih brodova. Osim toga, očekivano vreme izvršavanja na velikim instancama ostaje u

okviru poželjnih, malih vrednosti. Dobijeni rezultati potvrđuju da sve predložene metaheuristike predstavljaju adekvatan prisup rešavanju DMCHBAP-a.

Naučni doprinos ove disertacije je višestruk. Preciznije, glavni naučni rezultati teze su:

- sistematičan pregled naučne literature koja se bavi problemom dodele vezova u kontejnerskim terminalima;
- uvođenje nove varijante problema dinamičke hibridne dodele vezova u kontejnerskom terminalu, koja za cilj ima minimizaciju ukupnih troškova nastalih zbog smeštanja broda na vez koji nije označen kao omiljen, čekanja broda u odnosu na očekivano vreme dolaska i kašnjenja u odnosu na planirano vreme završetka obrade;
- formulisanje novog matematičkog modela mešovitog celobrojnog linearnog programiranja za dinamičku hibridnu varijantu dodele vezova u kontejnerskom terminalu sa minimizacijom ukupnih troškova;
- analiza složenosti statičke i dinamičke varijante problema i prilagođeni dokazi da su obe varijante dodele vezova jako (engl. *strongly*) NP-teški problemi čak i u pojednostavljenoj verziji u kojoj se ignorišu zahtevi za omiljenim lokacijama veza a funkcija cilja minimizira samo ukupno težinsko kašnjenje brodova;
- dokaz da BCO metoda može da formira optimalno rešenje za problem dodele vezova sa verovatnoćom strogo većom od nule;
- dizajniranje i implementacija metaheurističkih algoritama iz klase evolutivnih algoritama, inteligencije roja i promenljivih okolina za rešavanje statičke i dinamičke varijante dodele vezova. Predloženi algoritmi su prilagođeni specifičnim karakteristikama razmatranih DMCHBAP i MCHBAP varijanti dodele vezova ali se istovremeno lako mogu dalje transformisati i primeniti kao metaheurističke tehnike za rešavanje sličnih problema optimizacije u kontejnerskom terminalu;
- implementirana softverska rešenja mogu biti primenjena kao nezavisni alati u lučkim terminalima u cilju formiranja planova dodele vezova ili mogu biti integrisana u kompleksniji sistem za podršku donošenju odluka koji će davati sveobuhvatne informacije o kontejnerskom terminalu u realnom vremenu odziva.

Rezultati prikazani u ovoj disertaciji predstavljaju doprinos oblastima kombinatorne optimizacije, operacionih istraživanja, metaheurističkih metoda i izučavanju problema dodele vezova u kontejnerskim terminalima.

Zbog raznovrsnosti i velikog broja već razvijenih BAP modela, otvorene su razne mogućnosti daljeg istraživanja BAP-a. Većina postojećih modela primenjenih na BAP razmatra determinističke parametre, i pored toga što su stohastičke vrednosti i modeli koji uključuju neki stepen neizvesnosti realniji. U budućim istraživanjima trebalo bi razmotriti i stohastičke parametre uključene u metode rešavanja MCHBAP-a i DMCHBAP-a, kao i njihove robustne varijante.

I pored toga što je ovo istraživanje pokazalo da je metaheuristički pristup pogodan za rešavanje problema dodele vezova, egzaktne metode (posebno one koje se baziraju na kombinatornoj optimizaciji i sofisticiranim tehnikama optimizacije) predstavljaju dobar osnov za buduća istraživanja. Kontejnerski terminal je dinamičan sistem, a alokacija vezova (kao jedan od njegovih najbitnijih segmenata), ima direktan uticaj na ostale faze utovara/istovara kontejnera. To prouzrokuje da i mala odstupanja od optimalnih alokacija brodova mogu značajno da utiču na povećanje transportnih troškova i da negativno utiču na ekonomiju luke. Zbog toga razvoj egzaktnih metoda i njihovo kombinovanje sa metaheuristikama i dalje treba da ima značajan udeo u istraživanjima.

Performanse metaheurističkih algoritama primenjenih na MCHBAP ili DMCHBAP instance velikih dimenzija mogu se unaprediti upotrebom hibridnih tehnika rešavanja, (posebno hibridnih metaheuristika) kao adekvatnijeg pristupa rešavanju minimizacije troškova u BAP-u. Takođe, hibridizacija predoženih metaheuristika sa egzaktnim algoritmima može poboljšati kvalitet rešenja MCHBAP-a i DMCHBAP-a. Jedna od mogućih strategija hibridizacije je primena metaheuristika za dobijanje dobrih početnih rešenja za egzaktne metode. Sa druge strane, egzaktne metode mogu da se upotrebe kao alati za rešavanje delova kompleksnih MCHBAP-a ili DMCHBAP-a, posebno u slučajevima kada je moguće problem razdvojiti na manje podprobleme.

Implementacije pristupa rešavanju MCHBAP-a i DMCHBAP-a se na jednostavan način mogu paralelizovati, posebno kod diskretnih varijanti ovih problema, kada vezovi nezavisno jedan od drugog opslužuju brodove. Takođe, mnoge instance MCHBAP-a i DMCHBAP-a imaju karakteristike koje omogućavaju razdvajanje skupa brodova na klastere. U tim slučajevima, formirane grupe brodova se mogu alocirati konkurentno. Posmatrajući MCHBAP i DMCHBAP na taj način, i uzima-

jući u obzir njihove opisane karakteristike, očigledno je da paralelizacija predloženih implementacija predstavlja dobar pravac daljeg istraživanja.

Zbog svega navedenog, postojeće metaheurističke algoritme predložene u ovoj disertaciji u budućim istraživanjima trebalo bi nadograditi tehnikama koje podrazumevaju paralelizaciju, hibridizaciju i elemente robustne optimizacije. Perspektivan pristup daljem unapređenju predloženih metaheurističkih metoda je integracija sa postojećim ekzaktnim metodama rešavanja ovih problema, formiranjem hibridnih algoritama ili matheuristika.

Bibliografija

[1] E. Ahmed, T. Zayed, and S. Alkass. Improving productivity of yard trucks in port container terminal using computer simulation. In *Proceedings of the International Symposium on Automation and Robotics in Construction, ISARC*, volume 31, pages 1–8, Sydney Australia, 2014. Vilnius Gediminas Technical University, Department of Construction Economics & Property.

[2] C. Arango, P. Cortés, Je. Muñuzuri, and L. Onieva. Berth allocation planning in seville inland port by simulation and optimisation. *Advanced Engineering Informatics*, 25(3):452–461, 2011.

[3] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*. Taylor & Francis Group, LLC, CRC Press, New York, 2000.

[4] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 2: Advanced Algorithms and Operators*. IOP Publishing Ltd., Bristol, 2000.

[5] T. Bäck, D.B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., Bristol, 1997.

[6] S. M. Bhandarkar and H. Zhang. Image segmentation using evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 3(1):1–21, 1999.

[7] C. Bierwirth and F. Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202:615–627, 2010.

[8] C. Bierwirth and F. Meisel. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 244(3):675–689, 2015.

[9] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[10] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[11] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Röhler. Matheuristics: Optimization, simulation and control. In *International Workshop on Hybrid Metaheuristics*, pages 171–177, Udine, Italy, 2009. Springer.

[12] F. Box. A heuristic technique for assigning frequencies to mobile radio nets. *IEEE Transactions on Vehicular Technology*, 27(2):57–64, 1978.

[13] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, Boca Raton, Florida, 1984.

[14] V.N. Çaglar. Sustainable container terminal operations: Challenges and enhancements. *Karadeniz Arastirmalari*, 49:1–16, 2016.

[15] V. Caldeirinha, J. A. Felício, and A. Dionísio. The container terminal characteristics and customer's satisfaction. Technical report, University of Evora, CEFAGE-UE (Portugal), 2013.

[16] P. Calégari, G. Coray, A. Hertz, D. Kobler, and P. Kuonen. A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics*, 5(2):145–158, 1999.

[17] H. J. Carlo, I. F. A. Vis, and K. Jan Roodbergen. Seaside operations in container terminals: literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal*, 27(2-3):224–262, 2015.

[18] D. Chang, Z. Jiang, W. Yan, and J. He. Integrating berth allocation and quay crane assignments. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):975–990, 2010.

[19] J. H. Chen, D.-H. Lee, and J. X. Cao. A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):266–275, 2012.

[20] C. Y. Cheong, K. C. Tan, D. K. Liu, and C. J. Lin. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research*, 180(1):63–103, 2010.

[21] C.Y. Cheong, CJ Lin, K.C. Tan, and DK Liu. A multi-objective evolutionary algorithm for berth allocation in a container port. In *Proceedings of the CEC 2007, IEEE Congress on Evolutionary Computation*, pages 927–934, Singapore, Singapore, 2007. IEEE.

[22] C.Y. Cheong and K.C. Tan. A multi-objective multi-colony ant algorithm for solving the berth allocation problem. In Liu Y., Sun A., Loh H.T., Lu W.F., and Lim E.P., editors, *Advances of Computational Intelligence in Industrial Systems*, pages 333–350. Springer, 2008.

[23] E. K. P. Chong and S. H. Zak. *An introduction to optimization*. John Wiley & Sons, New York, 2013.

[24] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In Varela F. J. and Bourgine P., editors, *Proceedings of the First European Conference on Artificial Life*, volume 142, pages 134–142, Paris, France, 1991. Elsevier Publishing.

[25] J.-F. Cordeau, G. Laporte, P. Legato, and L. Moccia. Models and tabu search heuristics for the berth-allocation problem. *Transportation science*, 39(4):526–538, 2005.

[26] J. Dai, W. Lin, R. Moorthy, and C.-P. Teo. Berth allocation planning optimization in container terminals. In Tang C.S., Teo C.P., and Wei K.K., editors, *Supply chain analysis. International Series In Operations Research & Mana*, volume 119, pages 69–104. Springer, Boston, MA, 2008.

[27] T. Davidović, N. Kovač, and Z. Stanimirović. VNS-based approach to minimum cost hybrid berth allocation problem. In *Proceedings of the XLII International Symposium on Operations Research, SYMOPIS 2015*, pages 237–240, Silver Lake, Serbia, 2015.

[28] T. Davidović, J. Lazić, N. Mladenović, S. Kordić, N. Kovač, and B. Dragović. Mip-heuristics for minimum cost berth allocation problem. In *Proceedings of the International Conference on Traffic and Transport Engineering, ICTTE 2012*, pages 21–28, Belgrade, Serbia, 2012.

[29] T. Davidović, D. Ramljak, M. Šelmić, and D. Teodorović. Bee colony optimization for the p-center problem. *Computers & Operations Research*, 38(10):1367–1376, 2011.

[30] T. Davidović, D. Teodorović, and M. Šelmić. Bee colony optimization Part I: The algorithm overview. *Yugoslav Journal of Operational Research*, 25(1):33–56, 2015.

[31] T. Davidović, M. Šelmić, D. Teodorović, and D. Ramljak. Bee colony optimization for scheduling independent tasks to identical processors. *Journal of Heuristics*, 18(4):549–569, 2012.

[32] L. Davis. *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York, 1991.

[33] I. De Falco, A. Della Cioppa, and E. Tarantino. Mutation-based genetic algorithm: performance evaluation. *Applied Soft Computing*, 1(4):285–299, 2002.

[34] R. M. de Oliveira, G. R. Mauri, and L. A. N. Lorena. Clustering search for the berth allocation problem. *Expert Systems with Applications*, 39(5):5499–5505, 2012.

[35] M. Dorigo and T. Stützle. Ant colony optimization: overview and recent advances. *Techreport, IRIDIA, Universite Libre de Bruxelles*, 2009, (online `http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2009-013r001.pdf`).

[36] M. Dorigo and T. Stützle. Ant colony optimization: Overview and recent advances. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 227–263. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[37] DP World. Annual report and accounts: Performance+opportunity, May 2014, (online `http://web.dpworld.com/wp-content/uploads/2014/05/22294_DP_World_RA14_Web_v2.pdf`), [Accessed: 2017-08-17].

[38] B. Dragović, N. Kovač, and M. Škurić. Container port planning and advanced modeling techniques. In *Proceedings of the Third International Forum of Shipping, Port and Airport, IFSPA 2010*, pages 375–387, China, Chengdu, Sichuan, 2010.

[39] B. Dragović, D.K. Ryoo, N. Kovač, and M. Škurić. Literature review of advanced modeling techniques for container terminal planning. In *Proceedings of the 10th International Conference Research and Development in Mechanical Industry, RaDMI 2010*, pages 489–497, Donji Milanovac, Serbia, 2010.

[40] M. A. Dulebenets. *Models and solution algorithms for improving operations in marine transportation*. PhD thesis, The University of Memphis, 2015.

[41] M.H. Elwany, I. Ali, and Y. Abouelseoud. A heuristics-based solution to the continuous berth allocation and crane assignment problem. *Alexandria Engineering Journal*, 52(4):671–677, 2013.

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

[43] Fiji Ports. Fiji ports corporation limited annual report, December 2013, `http://www.fijiports.com.fj/wp-content/uploads/2016/09/FPCL-Annual-Report-2013-Final.pdf`, [Accessed: 2017-08-17].

[44] V. Filipovic. Fine-grained tournament selection operator in genetic algorithms. *Computers and Artificial Intelligence*, 22(2):143–161, 2003.

[45] B. B. Firouzi, M. S. Sadeghi, and T. Niknam. A new hybrid algorithm based on PSO, SA, and k-means for cluster analysis. *International journal of innovative computing, information and control*, 6(7):3177–3192, 2010.

[46] C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3–38, 2009.

[47] L. J. Fogel. Toward inductive inference automata. In *Proceedings of IFIP Congress 62, Information Processing 1962*, pages 395–400, Munich, Germany, 1962.

[48] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution*. John Wiley & Sons, Oxford, England, 1966.

[49] T. D. Fry, G. K. Leong, and T. R. Rakes. Single machine scheduling: A comparison of two solution procedures. *Omega*, 15(4):277–282, 1987.

[50] S.R.S. Ganji, A. Babazadeh, and N. Arabshahi. Analysis of the continuous berth allocation problem in container ports using a genetic algorithm. *Journal of Marine Science and Technology*, 15(4):408–416, 2010.

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

[52] M. Gendreau. An introduction to tabu search. In Glover F. and Kochenberger G.A., editors, *Handbook of metaheuristics. International Series in Operations Research & Management Science*, volume 57, pages 37–54. Springer, Boston, MA, 2003.

[53] M. Gendreau and J-Y. Potvin. Tabu search. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 41–59. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[54] G. Giallombardo, L. Moccia, M. Salani, and I. Vacca. Modeling and solving the tactical berth allocation problem. *Transportation Research Part B: Methodological*, 44(2):232–245, 2010.

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

[56] M. Golias, M. Boile, and S. Theofanis. The berth allocation problem: a formulation reflecting time window service deadlines. In *Proceedings of the 48th Transportation Research Forum Annual Meeting*, pages 527–547, Boston, Massachusetts, 2006. Transportation Research Forum.

[57] M. Golias, I. Portal, D. Konur, E. Kaisar, and G. Kolomvos. Robust berth scheduling at marine container terminals via hierarchical optimization. *Computers & Operations Research*, 41:412–422, 2014.

[58] M.M. Golias, M. Boile, and S. Theofanis. Berth scheduling by customer service differentiation: A multi-objective approach. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):878–892, 2009.

[59] M.M. Golias, M. Boile, and S. Theofanis. A lamda-optimal based heuristic for the berth scheduling problem. *Transportation Research Part C: Emerging Technologies*, 18(5):794–806, 2010.

[60] M.M. Golias and H.E. Haralambides. Berth scheduling with variable cost functions. *Maritime Economics and Logistics*, 13(2):174–189, 2011.

[61] M.M. Golias, G.K. Saharidis, M. Boile, S. Theofanis, and M.G. Ierapetritou. The berth allocation problem: Optimizing vessel arrival time. *Maritime Economics and Logistics*, 11(4):358–377, 2009.

[62] Government of India. Reducing dwell time of cargo at ports, report of the inter-ministerial group. *Planning Commission, Government of India*, 2007, `http://planningcommission.gov.in/sectors/ppp_report/3.Reports%20of%20Committiees%20&%20Task%20force/Railways/20.Reducing-Dwell-Time-of-Cargo-at-Ports.pdf`. [Accessed: 2017-08-17].

[63] Y. Guan and R. K. Cheung. The berth allocation problem: models and solution methods. *OR Spectrum*, 26(1):75–92, 2004.

[64] A. Gudelj, M. Krčum, and E. Twrdy. Models and methods for operations in port container terminals. *PROMET-Traffic&Transportation*, 22(1):43–51, 2010.

[65] M. Han, P. Li, and J. Sun. The algorithm for berth scheduling problem by the hybrid optimization strategy GASA. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision,* ICARCV'06, pages 1–4, Singapore, Singapore, 2006. IEEE.

[66] X.-l. Han, Z.-q. Lu, L.-f. Xi, et al. A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *European Journal of Operational Research*, 207(3):1327–1340, 2010.

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

[68] P. Hansen and N. Mladenović. Developments of the variable neighborhood search. In C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Springer, Boston, MA, 2002.

[69] P. Hansen and N. Mladenović. Variable neighbourhood search. In F. Glover and G. Kochenagen, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, Dordrecht, 2003.

[70] P. Hansen, N. Mladenović, J. Brimberg, and J. A. Moreno Pérez. Variable neighborhood search. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 61–86. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[71] P. Hansen, N. Mladenović, and J. A. M. Pérez. Variable neighbourhood search: methods and applications. *4OR*, 6(4):319–360, 2008.

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

[73] P. Hansen, C. Oğuz, and N. Mladenović. Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research*, 191(3):636–649, 2008.

[74] H. E. Haralambides. Competition, excess capacity, and the pricing of port infrastructure. *International Journal of Maritime Economics*, 4(4):323–347, 2002.

[75] M.P.M. Hendriks, D. Armbruster, M. Laumanns, E. Lefeber, and J.T. Udding. Strategic allocation of cyclically calling vessels for multi-terminal container operators. *Flexible Services and Manufacturing Journal*, 24(3):248–273, 2012.

[76] L. Henesey. *Enhancing container terminal performance: a multi agent systems approach.* PhD thesis, Blekinge Institute of Technology, 2004.

[77] D. S. Hochba. Approximation algorithms for np-hard problems. *ACM Sigact News*, 28(2):40–52, 1997.

[78] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, Cambridge, USA, 1975.

[79] Q.-M. Hu, Z.-H. Hu, and Y. Du. Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. *Computers & Industrial Engineering*, 70:1–10, 2014.

[80] A. Imai, E. Nishimura, M. Hattori, and S. Papadimitriou. Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research*, 179(2):579–593, 2007.

[81] A. Imai, J.-T. Zhang, E. Nishimura, and S. Papadimitriou. The berth allocation problem with service time and delay time objectives. *Maritime Economics & Logistics*, 9(4):269–290, 2007.

[82] D.E. Joslin and D.P. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research*, pages 353–373, 1999.

[83] C.-F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):997–1006, 2004.

[84] J. Karafa, M.M. Golias, S. Ivey, G.K.D. Saharidis, and N. Leonardos. The berth allocation problem with stochastic vessel handling times. *The International Journal of Advanced Manufacturing Technology*, 65(1-4):473–484, 2013.

[85] R. M. Karp. Reducibility among combinatorial problems. In Miller R.E., Thatcher J.W., and Bohlinger J.D., editors, *Complexity of computer computations. The IBM Research Symposia Series*, pages 85–103. Springer, Boston, MA, 1972.

[86] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, 1995.

[87] N. Khalid. Measuring the performance of malaysian container ports. `http://www.mima.gov.my/mima/wp-content/uploads/Port%20performance%20FINAL%20(Nov12).pdf`, December 2012. [Accessed: 2017-08-17].

[88] M. Kiani, S. Bonsall, J. Wang, and A. Wall. A break-even model for evaluating the cost of container ships waiting times and berth unproductive times in automated quayside operations. *WMU Journal of Maritime Affairs*, 5(2):153–179, 2006.

[89] K. H. Kim and H. Lee. Container terminal operation: current trends and future challenges. In *Lee CY., Meng Q. (eds) Handbook of Ocean Container Transport Logistics. International Series in Operations Research & Management Science*, volume 220, pages 43–73. Springer, Cham, 2015.

[90] K. H. Kim and K. C. Moon. Berth scheduling by simulated annealing. *Transportation Research Part B: Methodological*, 37(6):541–560, 2003.

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

[92] S. Kordić, T. Davidović, N. Kovač, and B. Dragović. Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. *Applied Mathematical Modelling*, 40(21-22):8952–8973, 2016.

[93] S. Kordić, B. Dragović, T. Davidović, and N. Kovač. A combinatorial algorithm for berth allocation problem in container port. In *Proceedings of the 2012 International Association of Maritime Economists Conference, IAME 2012*, Taipei, 2012.

[94] S. Kordić, N. Kovač, and T. Davidović. Divide and conquer approach to discrete berth allocation problem. In *Proceedings of the XIII Balkan Conference on Operational Research*, pages 307–316, BALCOR, 2015, Constanta, Romania, 2015.

[95] S. Kordić, N. Kovač, and Ž. Pekić. An analysis of estimation and rearrange heuristic for sedimentation algorithm for solving berth allocation problem in container port. In *Proceedings of the 4th International Maritime Science Conference, IMSC 2012*, pages 71–79, Split, Croatia, 2012.

[96] Nataša Kovač. Metaheuristic approaches for the berth allocation problem. *Yugoslav Journal of Operations Research*, 27(3):265–289, 2017.

[97] N. Kovač. Bee colony optimization algorithm for the minimum cost berth allocation problem. In *Proceedings of the XI Balkan Conference on Operational Research*, pages 245–254, BALCOR, 2013, Beograd–Zlatibor, 2013.

[98] N. Kovač, T. Davidović, and Z. Stanimirović. Evolutionary algorithm for the minimum cost hybrid berth allocation problem. In *Proceedings of the 6th IEEE International Conference on Information, Intelligence, Systems and Applications (IISA2015)*, Corfu, 2015.

[99] N. Kovač, T. Davidović, and Z. Stanimirović. Variable neighborhood search methods for the dynamic minimum cost hybrid berth allocation problem. *(accepted for publication)*, 2017.

[100] N. Kovač, B. Dragović, and S. Kordić. Short literature survey of berth allocation problem. In *Proceedings of the XIX International Conference on Material Handling, Constructions and Logistics, MHCL 2009*, pages 339–342, University of Belgrade, Faculty of Mechanical Engineering, 2009.

[101] N. Kovač, Z. Stanimirović, and T. Davidović. Metaheuristic approaches for the minimum cost hybrid berth allocation problem. In C. Konstantopoulos and G. Pantziou, editors, *Modelling, Computing and Data Handling Methodologies for Maritime Transport*, pages 1–47. Springer, New York Dordrecht, Heidelberg London, 2017.

[102] J. R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA, 1992.

[103] E. Lalla-Ruiz, J.L. González-Velarde, B. Melián-Batista, and J.M. Moreno-Vega. Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 22:60–76, 2014.

[104] E. Lalla-Ruiz, B. Melián-Batista, and J.M. Moreno-Vega. Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. *Engineering Applications of Artificial Intelligence*, 25(6):1132–1141, 2012.

[105] E.A. Lalla-Ruiz and S. Voß. Towards a matheuristic approach for the berth allocation problem. In Pardalos P., Resende M., Vogiatzis C., and Walteros J., editors, *Learning and Intelligent Optimization. LION 2014.*, volume 8426, pages 218–222. Springer, Gainesville, USA, 2014.

[106] D.-H. Lee, Z. Cao, and Q. Meng. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics*, 107(1):115–124, 2007.

[107] D.-H. Lee, J.H. Chen, and J.X. Cao. The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):1017–1029, 2010.

[108] D.-H. Lee and J.G. Jin. Feeder vessel management at container transshipment terminals. *Transportation Research Part E: Logistics and Transportation Review*, 49(1):201–216, 2013.

[109] D.-H. Lee, J.G. Jin, and J.H. Chen. Terminal and yard allocation problem for a container transshipment hub with multiple terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(2):516–528, 2012.

[110] D.-H. Lee, H. Q. Wang, and L. Miao. Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):124–135, 2008.

[111] C. Liang, J. Guo, and Y. Yang. Multi-objective hybrid genetic algorithm for quay crane dynamic assignment in berth allocation planning. *Journal of Intelligent Manufacturing*, 22(3):471–479, 2011.

[112] C. Liang, H. Hwang, and M. Gen. A berth allocation planning problem with direct transshipment consideration. *Journal of Intelligent Manufacturing*, 23(6):2207–2214, 2012.

[113] A. Lim. The berth planning problem. *Operations Research Letters*, 22(2):105–110, 1998.

[114] S. Luke. *Essentials of metaheuristics.* Lulu Raleigh, 2009, (online `http://www.cs.put.poznan.pl/mkomosinski/materialy/optymalizacja/extras/Essentials.pdf`).

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

[116] P. Maksimović and T. Davidović. Parameter calibration in the bee colony optimization algorithm. In *Proceedings of the XI Balcan Conference on Operational Research*, pages 263–272, BALCOR 2013, Beograd-Zlatibor, Serbia, 2013.

[117] M. Marić, Z. Stanimirović, and P. Stanojević. An efficient memetic algorithm for the uncapacitated single allocation hub location problem. *Soft Computing*, 17(3):445–466, 2013.

[118] G.R. Mauri, L.N. de Andrade, and L.A.N. Lorena. A memetic algorithm for a continuous case of the berth allocation problem. In *Proceedings of the Inter-*

national Conference on Evolutionary Computation Theory and Applications-Volume 1: ECTA, (IJCCI 2011).

[119] F. Meisel. *Seaside operations planning in container terminals*. Springer, Berlin Heidelberg, 2009.

[120] F. Meisel and C. Bierwirth. Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45:196–209, 2009.

[121] F. Meisel and C. Bierwirth. A framework for integrated berth allocation and crane operations planning in seaport container terminals. *Transportation Science*, 47(2):131–147, 2013.

[122] R. M'Hallah. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers & Operations Research*, 34(10):3126–3142, 2007.

[123] S. Mišković and Z. Stanimirović. Hybrid metaheuristic method for solving a multi-period emergency service location problem. *Information Technology And Control*, 45(3):321–337, 2016.

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

[125] K.C. Moon. *A mathematical model and a heuristic algorithm for berth planning*. PhD thesis, Logistics Systems Lab., Industrial Engineering, Pusan National University, 2000.

[126] P. Moscato and M. Norman. A competitive and cooperative approach to complex combinatorial search. *Caltech Concurrent Computation Program, Report C3P-790*, 1989, (online `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.776&rep=rep1&type=pdf`).

[127] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524, 1996.

[128] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, New York, 1988.

[129] F. Neri, C. Cotta, and P. Moscato. *Handbook of memetic algorithms*, volume 379. Springer-Verlag, Berlin Heidelberg, 2012.

[130] A. G. Nikolaev and S. H. Jacobson. Simulated annealing. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 1–39. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[131] M. Nikolić and D. Teodorović. Transit network design by bee colony optimization. *Expert Systems with Applications*, 40(15):5945–5955, 2013.

[132] M. Nikolić, D. Teodorović, and M. Šelmić. Solving the vehicle routing problem with time windows by bee colony optimization metaheuristic. In *Proceedings of the 1st Logistics International Conference*, pages 44–48, Belgrade, Serbia, 2013.

[133] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc. Upper Saddle River, USA, 1998.

[134] Y. M. Park and K. H. Kim. A scheduling method for berth and quay cranes. *OR Spectrum*, 25(1):1–23, 2003.

[135] J. A. M. Pérez, P. Hansen, and N. Mladenović. Parallel variable neighborhood search. In Alba E., editor, *Parallel metaheuristics: a new class of algorithms*, pages 247–266. Wiley, New York, 2004.

[136] M. L. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer-Verlag, New York, 2016.

[137] D. Pisinger and M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167, 2005.

[138] D. Pisinger and M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007.

[139] F. Plastria, N. Mladenović, and D. Urošević. Variable neighborhood formulation space search for circle packing. In *Proceedings of the 18th Mini Euro Conference VNS*, pages 137–141, Tenerife, Spain, 2005.

[140] G. Polya. *How to solve it: A new aspect of mathematical method*. Princeton University Press, Princeton, New Jersey, 1945.

[141] Port of Rotterdam. Annual report 2010: World-class in action! `https://www.portofrotterdam.com/sites/default/files/annual-report-2010-%20PortofRotterdam-Authority.pdf`, December 2010. [Accessed: 2017-08-17].

[142] C. Prodhon. A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210:204–212, 2011.

[143] H. Rashidi and E. P. K. Tsang. Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling*, 37(6):3601–3634, 2013.

[144] I. Rechenberg. *Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1970.

[145] C. R. Reeves. Genetic algorithms. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 109–139. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[146] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau and J-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 283–319. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[147] T. Robenek, N. Umang, M. Bierlaire, and S. Ropke. A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research*, 235(2):399–411, 2014.

[148] F. J. Rodriguez, C. Garcia-Martinez, and M. Lozano. Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test. *IEEE Transactions on Evolutionary Computation*, 16(6):787–800, 2012.

[149] M. Rodriguez-Molins, L. Ingolotti, F. Barber, M.A. Salido, M.R. Sierra, and J. Puente. A genetic algorithm for robust berth allocation and quay crane assignment. *Progress in Artificial Intelligence*, 2(4):177–192, 2014.

[150] M. Rodriguez-Molins, M.A. Salido, and F. Barber. A GRASP-based metaheuristic for the berth allocation problem and the quay crane assignment problem by managing vessel cargo holds. *Applied Intelligence*, 40(2):273–290, 2014.

[151] G.K.D. Saharidis, M.M. Golias, M. Boile, S. Theofanis, and M.G. Ierapetritou. The berth scheduling problem with customer differentiation: a new methodological approach based on hierarchical optimization. *The International Journal of Advanced Manufacturing Technology*, 46(1-4):377–393, 2010.

[152] M.A. Salido, M. Rodriguez-Molins, and F. Barber. Integrated intelligent techniques for remarshaling and berthing in maritime terminals. *Advanced Engineering Informatics*, 25(3):435–451, 2011.

[153] M.A. Salido, M. Rodriguez-Molins, and F. Barber. A decision support system for managing combinatorial problems in container terminals. *Knowledge-Based Systems*, 29:63–74, 2012.

[154] A. Simrin and A. Diabat. The dynamic berth allocation problem: A linearized formulation. *RAIRO-Operations Research*, 49(3):473–494, 2015.

[155] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology, Boston, 2006.

[156] L. Song, T. Cherrett, and W. Guan. Study on berth planning problem in a container seaport: Using an integrated programming approach. *Computers & Industrial Engineering*, 62(1):119–128, 2012.

[157] R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2008.

[158] Z. Stanimirović, M. Marić, S. Božović, and P. Stanojević. An efficient evolutionary algorithm for locating long-term care facilities. *Information Technology and Control*, 41(1):77–89, 2012.

[159] T. Stojanović, T. Davidović, and Z. Ognjanović. Bee colony optimization for the satisfiability problem in probabilistic logic. *Applied Soft Computing*, 31:339–347, 2015.

[160] B. Suman and P. Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, pages 1143–1160, 2006.

[161] É.D. Taillard and S. Voß. POPMUSIC—partial optimization metaheuristic under special intensification conditions. In *Essays and Surveys in Metaheu-*

ristics. Operations Research/Computer Science Interfaces Series, volume 15, pages 613–629. Springer, Boston, MA, 2002.

[162] E.-G. Talbi. *Metaheuristics: From Design to Implementation.* John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[163] X. Tang, R. Tian, and D.F. Wong. Fast evaluation of sequence pair in block placement by longest common subsequence computation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(12):1406–1413, 2001.

[164] A.-A. Tantar, N. Melab, and E.-G. Talbi. A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. *Soft Computing*, 12(12):1185, 2008.

[165] S. Theofanis, M. Boile, and M. Golias. An optimization based genetic algorithm heuristic for the berth allocation problem. In *Proceedings of the CEC 2007, IEEE Congress on Evolutionary Computation*, pages 4439–4445, Singapore, Singapore, 2007. IEEE.

[166] S. Theofanis, M. Boile, and M. Golias. Container terminal berth planning: critical review of research approaches and practical challenges. *Transportation Research Record: Journal of the Transportation Research Board*, 2100:22–28, 2009.

[167] C.-J. Ting, K.-C. Wu, and H. Chou. Particle swarm optimization algorithm for the berth allocation problem. *Expert Systems with Applications*, 41(4):1543–1550, 2014.

[168] T.O. Ting, X.-S. Yang, S. Cheng, and K. Huang. Hybrid metaheuristic algorithms: past, present, and future. In *Recent advances in swarm intelligence and evolutionary computation*, pages 71–83. Springer, 2015.

[169] A. Torn and A. Zilinskas. *Global optimization.* Springer-Verlag, New York, 1989.

[170] N. Umang, M. Bierlaire, and I. Vacca. The berth allocation problem in bulk ports. In *Proceedings of the 11th Swiss Transport Research Conference*, number EPFL-CONF-167446, Monte Verità, Switzerland, 2011.

[171] N. Umang, M. Bierlaire, and I. Vacca. Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review*, 54:14–31, 2013.

[172] UNCTAD. Review of maritime transport. `http://unctad.org/en/docs/rmt2005_en.pdf`, 2005. [Accessed: 2017-08-17].

[173] UNCTAD. Port performance: Linking performance indicators to strategic objectives. `http://unctad.org/en/PublicationsLibrary/dtlkdb2016d1_en.pdf`, 2016. [Accessed: 2017-08-17].

[174] I. Vacca, M. Salani, and M. Bierlaire. An exact algorithm for the integrated planning of berth allocation and quay crane assignment. *Transportation Science*, 47(2):148–161, 2013.

[175] Y. Wang and I. H. Witten. Inducing model trees for continuous classes. In M. Someren and G. Widmer, editors, *Proceedings of the 9th European Conference on Machine Learning Poster Papers*, pages 128–137, Prague, Czech Republic, 1997. Springer-Verlag Berlin Heidelberg.

[176] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, USA, 2005.

[177] Y. Xu, Q. Chen, and X. Quan. Robust berth scheduling with uncertain vessel delay and handling time. *Annals of Operations Research*, 192(1):123–140, 2012.

[178] Y. Xu and R. Qu. Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods. *Journal of the Operational Research Society*, 62(2):313–325, 2011.

[179] C. Yang, X. Wang, and Z. Li. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering*, 63(1):243–253, 2012.

[180] J. H. Yang and K. H. Kim. A grouped storage method for minimizing relocations in block stacking systems. *Journal of Intelligent Manufacturing*, 17(4):453–463, 2006.

[181] X. Yu and M. Gen. *Introduction to evolutionary algorithms*. Springer-Verlag, London, 2010.

[182] S. H. Zanakis, J. R. Evans, and A. A. Vazacopoulos. Heuristic methods and applications: a categorized survey. *European Journal of Operational Research*, 43(1):88–110, 1989.

[183] Q. Zeng, X. Hu, W. Wang, and Y. Fang. Disruption management model and its algorithms for berth allocation problem in container terminals. *International Journal of Innovative Computing, Information and Control*, 7(5):2130–2142, 2011.

[184] Y. Zhang, S. Wang, and G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015:1, 2015.

[185] L. Zhen, E.P. Chew, and L.H. Lee. An integrated model for berth template and yard template planning in transshipment hubs. *Transportation Science*, 45(4):483–504, 2011.

[186] L. Zhen, L.H. Lee, and E.P. Chew. A decision model for berth allocation under uncertainty. *European Journal of Operational Research*, 212(1):54–68, 2011.

[187] P. Zhou and H. Kang. Study on berth and quay-crane allocation under stochastic environments in container terminal. *Systems Engineering-Theory and Practice*, 28(1):161–169, 2008.

[188] P. Zhou, H. Kang, and L. Lin. A dynamic berth allocation model based on stochastic consideration. In *Proceedings of the Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006.*, volume 2, pages 7297–7301, Dalian, China, 2006.

Biografija autora

Nataša Kovač je rođena 04. decembra 1971. godine u Apatinu gde je završila osnovnu i srednju školu kao đak generacije. Prirodno-matematički fakultet u Novom Sadu, smer Diplomirani informatičar, upisala je 1990. godine. Diplomirala je 1996. godine, sa prosečnom ocenom 8.69 iz položenih ispita i ocenom 10 na diplomskom radu. Iste godine je na Prirodno-matematičkom fakultetu u Novom Sadu upisala magistarske studije, smer Diskretna matematika, i položila je sve ispite sa prosečnom ocenom 10. Magistarski rad na temu „Grafovi u Delphi-u" odbranila je 14. septembra 1999. godine sa ocenom 10.

Za postignut odličan uspeh iz svih predmeta u toku osnovnog i srednjeg školovanja dodeljena joj je Vukova diploma. U toku osnovnog, srednjeg školovanja i studija osvojila je više nagrada na takmičenjima iz oblasti prirodno-matematičkih nauka. U periodu od septembra 1996. do juna 1997. godine bila je zaposlena na Elektrotehničkom i Mašinskom fakultetu u Novom Sadu kao stručni saradnik. Od januara 2000. do oktobra 2002. godine bila je zaposlena u Srednjoj školi Kotor, kao profesor na predmetima Računarstvo i informatika i Matematika i položila je stručni ispit iz oblasti informatike. Od oktobra 2002. do juna 2016. bila je angažovana na Pomorskom fakultetu u Kotoru kao stručni saradnik a zatim i kao asistent. Na istom fakultetu od 31. 01. 2008. do 2011. obavljala je funkciju rukovodioca studijskog programa Brodomašinsvo. Od septembra 2016. do decembra 2016. radila je na Fakultetu Informacionih tehnologija Univerziteta Mediteran u Podgorici. Od januara 2017. zaposlena je kao predavač na Fakultetu primenjenih nauka Univerziteta Donja Gorica u Podgorici na predmetima geometrija i analitička geometrija.

Прилог 1.

Изјава о ауторству

Потписани-а Наташа Ковач

број индекса _____

Изјављујем

да је докторска дисертација под насловом

Метахеуристички приступ решавању једне класе оптимизационих проблема у транспорту

_____

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

Потпис докторанда

У Београду, _____

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Наташа Ковач

Број индекса _____

Студијски програм Математика

Наслов рада Метахеуристички приступ решавању једне класе оптимизационих проблема у транспорту

Ментор др Зорица Станимировић

Потписани/а Наташа Ковач

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

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

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

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

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

Метахеуристички приступ решавању једне класе оптимизационих проблема у транспорту

која је моје ауторско дело.

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

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

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

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

Потпис докторанда

У Београду, _____

_____