Usporedba algoritama za rješavanje problema dostave paketa pomoću flote vozila

Hucaljuk, Arijan

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:834754

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij računarstva

USPOREDBA ALGORITAMA ZA RJEŠAVANJE PROBLEMA DOSTAVE PAKETA POMOĆU FLOTE VOZILA

Diplomski rad

Arijan Hucaljuk

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 12.05.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Arijan Hucaljuk
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 740 R, 14.10.2014.
OIB studenta:	47519947336
Mentor:	Doc.dr.sc. Ivan Aleksi
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Tomislav Matić
Član Povjerenstva:	Dr.sc. Ivan Vidović
Naslov diplomskog rada:	Usporedba algoritama za rješavanje problema dostave paketa pomoću flote vozila
Znanstvena grana rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Temu rezervirao: Arijan Hucaljuk U ovom diplomskom radu potrebno je izraditi aplikaciju za prikaz i izračun problema dostave paketa pomoću flote vozila. Potrebno je međusobno usporediti eksperimentalne rezultate metoda: Best First Search, Ant colony, Genetic algorithm, Simulated Anealing i Branch and Bound.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	12.05.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 07.06.2017.

Ime i prezime studenta:	Arijan Hucaljuk
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 740 R, 14.10.2014.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Usporedba algoritama za rješavanje problema dostave paketa pomoću flote vozila**

izrađen pod vodstvom mentora Doc.dr.sc. Ivan Aleksi

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Arijan Hucaljuk, OIB: 47519947336, student/ica na studiju: Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **diplomski rad**:

Usporedba algoritama za rješavanje problema dostave paketa pomoću flote vozila

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 07.06.2017.

_____

potpis

SADRŽAJ

1. UVOD 1
1.1. Zadatak 1
2. RAČUNALNA APLIKACIJA ZA ANALIZU ALGORITAMA 2
2.1. Programi korišteni za izradu aplikacije 2
2.2. Izvedba programa 4
3. HEURISTIČKI ALGORITMI 9
3.1. Pohlepni algoritam 9
3.2. Algoritam kolonije mrava 11
3.3. Genetski algoritam 15
3.4. Simulirano kaljenje 19
3.5. Branch and bound metoda 21
3.6. Analiza rezultata 24
4. ZAKLJUČAK 26
LITERATURA 27
SAŽETAK 28
ABSTRACT 29
ŽIVOTOPIS 30
PRILOZI 31

1. UVOD

Cilj ovog diplomskog rada je izrada aplikacije za organizaciju dostave paketa primjenom različitih algoritama optimizacije puta. Problem prijevoza paketa zasniva se na dostavljanju velikog broja paketa na različite destinacije koristeći zadani broj zrakoplova. Potrebno je za svaki zrakoplov pronaći redoslijed letova kojim će se dostaviti svi paketi, uz što manju prijeđenu udaljenost. Ovaj problem je u skupini NP-kompleksnih problema, što znači da za takve probleme još uvijek ne postoje efikasni algoritmi za pronalaženje najboljeg rješenja. Izrada diplomskog rada odvijat će se kroz četiri stupnja: definiranje problema, pronalazak odgovarajućih algoritama za dobivanje rješenja, izrada aplikacije i analiza rezultata.

1.1. Zadatak

Zadatak je izraditi navedenu aplikaciju za optimiranje putanje dostavljanja paketa. Optimiranje putanje leta provodi se prema najkraćem putu te je potrebno vizualizirati putanju.

2. RAČUNALNA APLIKACIJA ZA ANALIZU ALGORITAMA

2.1. Programi korišteni za izradu aplikacije

2.1.1. Eclipse razvojno okruženje

Eclipse [1] je nastao kao projekt IBM Kanada. Razvijen je kao nasljednik VisualAge porodice alata. U studenom 2001., Eclipse je nastavljen razvijati kao program otvorenog koda.

Eclipse je višejezična integrirana razvojna okolina napisana u Javi i koristi se za razvoj aplikacija. Razvojno okruženje primarno za Java programere, ali nije ograničeno samo na Java programski jezik. Korisnicima je omogućeno instaliranje raznih dodataka, čime se povećava funkcionalnost razvojnog okruženja.

Od 2006. izdaje se svake godine nova inačica Eclipse platforme (tablica 2.1.) te one uključuje i prethodne Eclipse projekte. U pravilu se predstavljaju svake četvrte srijede u lipnju.

Tablica 2.1. Izdane verzije Eclipse platforme

Naziv platforme	***Datum izdavanja***	***Verzija platforme***	***Projekti***
Austin	21.6.2004.	3.0	
/	28.6.2005.	3.1	
Callisto	30.6.2006.	3.2	Callisto projekti
Europa	29.6.2007.	3.3	Europa projekti
Ganymede	25.6.2008.	3.4	Ganymede projekti
Galileo	24.6.2009.	3.5	Galileo projekti
Helios	23.6.2010.	3.6	Helios projekti
Indigo	22.6.2011.	3.7	Indigo projekti
Juno	27.6.2012.	3.8 i 4.2	Juno projekti
Kepler	26.6.2013.	4.3	Kepler projekti
Luna	25.6.2014.	4.4	Luna projekti
Mars	24.6.2015	4.5	Mars projekti
Neon	22.6.2016.	4.6	Neon projekti
Oxygen	Lipanj 2017 (planirano)	4.7	Oxygen projekti

2.1.2. Java programski jezik

Java [2] je objektno-orijentirani programski jezik koji su razvili James Gosling, Patrick Naughton i drugi inženjeri u tvrtki Sun Microsystems. Razvoj je počeo 1991. kao dio projekta Green, a objavljen je u studenom 1995. godine.

Tvrtka Sun polaže prava na ime Java, ali samo okruženje je moguće bez plaćanja skinuti sa Sunovih internet poslužitelja. U odnosu na većinu tadašnjih programskih jezika, Java programi nisu vezani za operativni sustav te ih je moguće izvoditi na svim operativnim sustavima za koje postoji JVM (Java Virtual Machine).

Java (tablica 2.2.) je jedan od najkorištenijih programskih jezika. Broj korisnika kreće se od 7 do preko 10 milijuna. Iako inspirirana jezikom C, Java korisnicima pruža bolji stupanj sigurnosti i pouzdanosti.

Prilikom stvaranja Java programskog jezika osnivači su postavili 5 primarnih ciljeva:

1. jednostavnost, objektna-orijentiranost i pristupačnost
2. robusnost i sigurnost
3. arhitekturna neutralnost i prenosivost
4. visoke performanse
5. interpretiranje, višenitnost i dinamičnost

Tablica 2.2. Izdane verzije Java programskog jezika

Verzija Java programskog jezika	***Datum izdavanja***
JDK 1.0	21.1.1996.
JDK 1.1	19.2.1997.
J2SE 1.2	8.12.1998.
J2SE 1.3	5.8.2000.
J2SE 1.4	6.2.2002.
J2SE 5.0	30.9.2004
Java SE 6	11.12.2006.
Java SE 7	28.7.2011.
Java SE 8	18.3.2014.

Java programski jezik ima automatsko skupljanje trenutno nekorištene memorije. Programer određuje kad će objekt biti stvoren, a Javino izvršno okruženje je odgovorno za oslobađanje memorije kad se objekti više ne koriste. Jedna od ideja iza Java automatskog upravljanja memorijom je da programer bude oslobođen ručnog upravljanja memorijom. Skupljanje memorije može nastupiti bilo kada, idealno kad je program u stanju mirovanja.

Što se tiče sintakse, Java sintaksa je napisana na temelju C++. Za razliku od C++, Java je napravljena isključivo kao objektno-orijentirani programski jezik. Cijeli kod je napisan unutra klase i sve se gleda kao objekt.

U odnosu na C++, Java ne podržava preopterećenje operatora i višestruko nasljeđivanje klasa. To pojednostavljuje programski jezik i pomaže pri sprječavanju potencijalnih pogrešaka.

Zbog svojih iznimnih svojstava, koristi se za razvoj programa na mobilnim uređajima i kod financijskih kompanija. Za programiranje Googleovog sustava Android, Java je korištena kao osnovni jezik.

2.2. Izvedba programa

2.2.1. Programski model

Osnovni programski model sastoji se od 3 klase:

- Location
- Package
- Plane

Klasa *Location* sadrži tri atributa (zemljopisnu širinu, zemljopisnu dužinu i ime). U klasi se nalaze metode za postavljanje vrijednosti atributima i metoda koja provjerava dva objekta, uspoređujući njihova imena i koordinate te vraća rezultat. To će se kasnije koristiti pri vizualiziranju putanje, jesu li početna i zadnja lokacija identične i treba li povući crtu prema odredištu.

Klasa *Package* sadrži dva atributa (ID paketa i težinu) i nasljeđuje objekt iz klase *Location*. U klasi se uspoređuju težine paketa između dva objekta ili ID objekta ako su težine jednake. Tijekom izvođenja programa pozivaju se metode iz klase *Package* u svrhu sortiranja paketa od najmanjeg prema najvećem i prema ID-u uzlazno.

Klasa *Plane* sadrži dva atributa (ID zrakoplova i kapacitet). Sastoji se od metoda za postavljanje vrijednosti atributima i metode koja uspoređuje kapacitete dva objekta ili njihove ID-ove. U klasi

se pomoću metoda sortiraju zrakoplovi s najmanjim kapacitetom prema najvećem, ili ako su kapaciteti jednaki, prema pripadajućem ID-u uzlazno.

2.2.2. Učitavanje podataka

Podatci za optimizaciju se učitavaju iz datoteke.

Kod zapisa u datoteke podatci za svaku klasu su zapisani u zasebne datoteke. Prednost učitavanja podataka iz datoteka je jednostavna mogućnost izmjene podataka kao i nadopune.

Programska podrška za učitavanje podataka sastoji se od dvije klase:

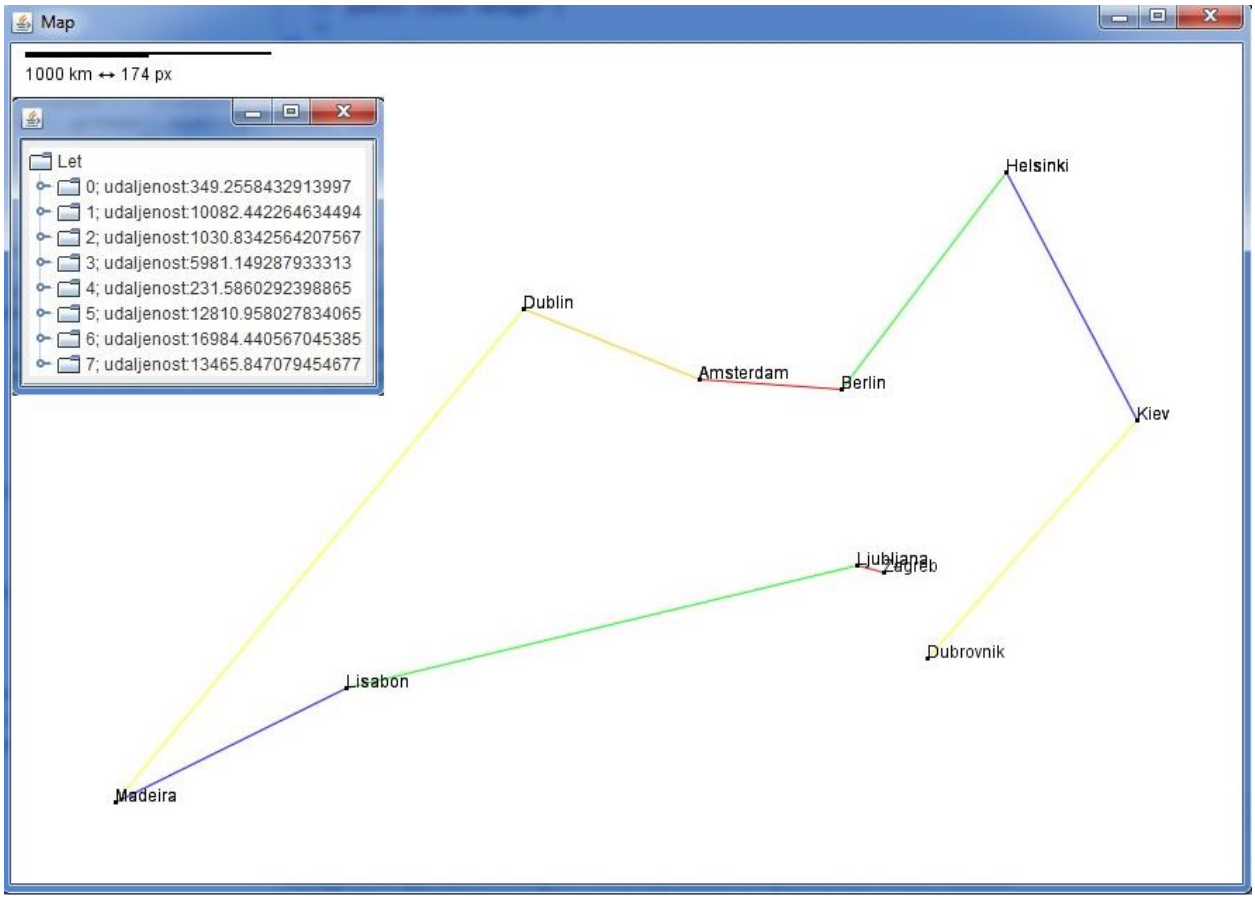
- Loader
- FileLoader

Apstraktna klasa *Loader* ne sadrži nikakvu implementaciju za čitanje podatka, već definira svojstva i metode klase koja ju nasljeđuje (*FileLoader*) te omogućava implementaciju drugog načina učitavanja podataka (npr. iz baze podataka). Sastoji se od metoda za učitavanje zrakoplova, destinacija i paketa iz kolekcija i apstraktnih metoda load i close koje obavljaju funkciju u potklasama.

Klasa *FileLoader* učitava parametre iz datoteke. Klasa implementira *load* metodu koja učitava sve podatke iz datoteka. Učitavanje podataka za svaki model (zrakoplov, lokacija, paket) ostvareno je u zasebnim metodama (*loadPackages*, *loadLocation* i *loadPlanes*). Nakon što se dobije pristup sadržaju datoteke, u *while* petlji se iterira po svim redovima datoteke. Ako red počinje sa znakom „#“ on se preskače i kreće se učitavanje novog reda. Datoteke sadrže podatke koji su definirani u programskom modelu. Metoda *getNextLine* provjerava pri čitanju iz kolekcija, radi li se o nedostupnom redu za učitanje ili ako je došlo do pogreške javlja pogrešku korisniku.

2.2.3. Vizualizacija

Vizualizacija podataka omogućava lakše predočavanje i razumijevanje dobivenih rješenja. Općenito ljudi puno lakše shvaćaju bit podatka kroz vizualni efekt u odnosu na običan tekst ili skup brojeva.

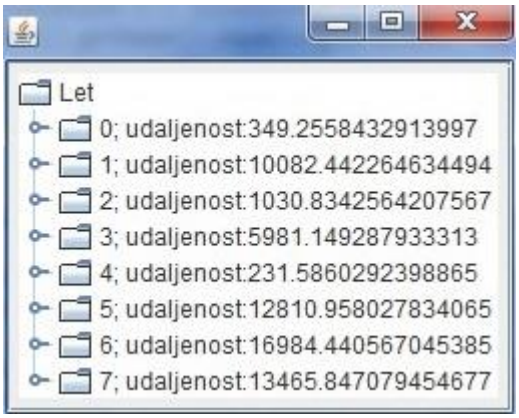


Slika 2.1. Usporedba prikaza rezultata vizualno i brojčano

Prikazana vizualizacija (slika 2.1.) pruža zadovoljavajući uvid i shvaćanje rezultata, uz jednostavno korištenje i neznatno opterećenje na memoriju.

Rezultati vizualizacije bi se mogli realnije prikazati uz nadograđivanje programa i povlačenje karte svijeta s Google Eartha te potom plotanje linija letova na tu istu mapu.

U novo kreiranom prozoru stvara se mapa Let koja sadrži sve letove koji su obavljeni za dostavljanje svih paketa na destinacije.



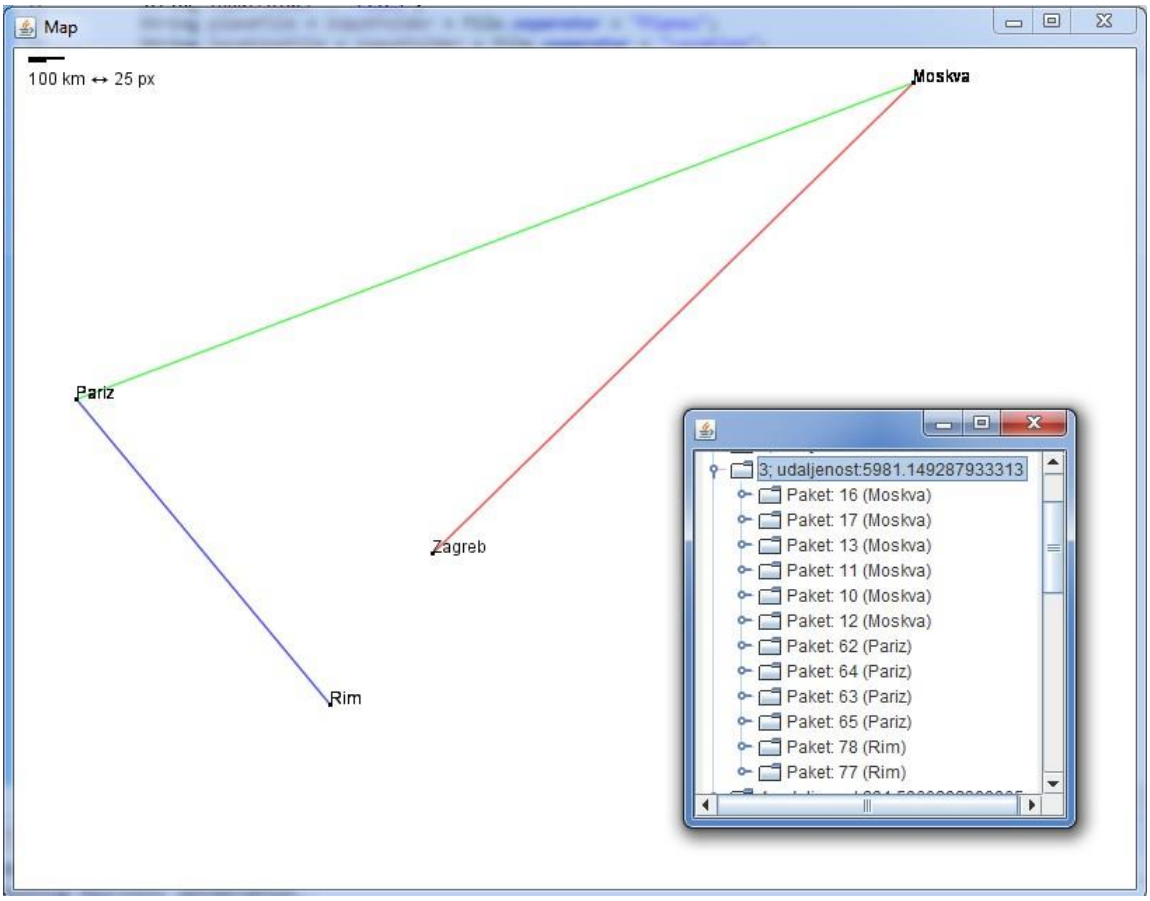
Slika 2.2. Primjer prozora sa svim letovima

Za svaki ostvareni let moguće je prikazati informacije (slika 2.2. i slika 2.3.) o prijeđenom putu i paketima (broj paketa, destinacija, težina).



Slika 2.3. Primjer sadržaja podataka leta

Iscrtavanje putanja (slika 2.4.) za odabrani let iz mape Let se obavlja tako da se odredi početna točka i dohvate se koordinate, na mapi se iscrta početna točka, zatim se dohvaća odredišna točka za paket i iscrtava se na mapi. Povlači se linija između dvije točke i trenutno odredišna točka postaje početna te se dohvaća nova odredišna točka. Boje letova se nasumično biraju.



Slika 2.4. Prikaz iscrtavanja dostave paketa na određenom letu

3. HEURISTIČKI ALGORITMI

Rješavanju NP-teških/kompletnih problema može se pristupiti na 2 načina:

- pristup koji daje optimalno rješenje
- pristup koji daje dovoljno dobro rješenje

Prvi pristup se oslanja na ispitivanju svih mogućih rješenja problema. U praksi ovaj pristup za realne zadatke nije izvediv zbog jako velikog skupa mogućih rješenja.

Drugi pristup [3] se oslanja na neku funkciju koja usmjerava pretragu rješenja. Time se skup mogućih rješenja smanjuje na razinu koju je moguće pretražiti u nekom razumnom vremenskom intervalu.

Konačni cilj optimizacije jest pronaći rješenje u kojem se svi paketi dostave na zadanu lokaciju, a da se pri tome prijeđe najkraći ukupni put. Pri pokretanju algoritama, radi dobivanja boljih početnih rješenja, svi paketi se grupiraju prema lokacijama.

3.1. Pohlepni algoritam

Heuristička metoda implementirana u ovom radu je Pohlepni algoritam [4]. Heuristika ili umijeće otkrivanja je grana proučavanja koja pripada logici, filozofiji i psihologiji. Heuristički rezultat ne pronalazi idealno rješenje, već daje rezultat koji je dovoljno dobar odnosno prihvatljiv. Za dobivanje idealnog rješenja pri kompleksnim problemima najčešće treba jako puno vremena što korisnicima nije prihvatljivo, pa je u većini slučajeva nužno prihvatiti heurističko rješenje.

Heuristički algoritmi nastali su eksperimentiranjem u svrhu dobivanja zadovoljavajućeg rezultata. Koriste se za rješavanje problema s eksponencijalnim i faktorijalnim složenostima, a da pritom daju zadovoljavajuće rezultate.

Heuristički algoritmi mogu se podijeliti na:

- Konstruktivne heuristike
- Hibridne metode
- Poboljšavajuće heuristike

Pohlepni algoritam koji spada pod konstruktivne heuristike u svakoj iteraciji pridjeljuje se vrijednost varijable odluke koja najviše pridonosi funkciji cilja, odnosno kvaliteti plana dostave.

Prednosti pohlepnog algoritma su:

- Jednostavnost implementacije
- Brzina

Nedostatci pohlepnog algoritma su:

- Lokalni pogled (lokalna optimalnost ne podrazumijeva globalnu optimalnost)
- Kvaliteta rješenja ovisi o tipu problema
- Algoritam se mora izvesti do kraja da bi se dobilo cjelovito rješenje

3.1.1. Implementacija na problemu dostave

Prilikom rješavanja problema dostave paketa potrebno je donijeti dva odabira:

- odabir zrakoplova
- odabir paketa koji će se prenijeti zrakoplovom

Odabir paketa koji će se ukrcati u zrakoplov indirektno određuje lokacije koje zrakoplov mora obići te samim time direktno i utječe na kvalitetu rješenja.

Pri implementaciji Pohlepnog algoritma u ovom radu korištene su heuristike prilikom oba odabira.

Heuristika za odabir zrakoplova definirana je raspoloživim kapacitetom zrakoplova. Zrakoplovi s većim kapacitetom prvi će se koristiti prilikom dodjele paketa.

Heuristika za odabir paketa oslanja se na lokacije paketa. Paketi se grupiraju po lokacijama te se potom iterira po paketima lokacije koja je najbliža trenutnoj lokaciji zrakoplova. Prije iteriranja po paketima s odabrane lokacije koristi se funkcija koja daje procjenu koliko bi dodavanje paketa s odabrane lokacije pridonijelo kvaliteti plana dostave odnosno rješenja problema.

Funkcija je definirana na sljedeći način:

$$Ocjena_Isplativosti = \text{udaljenost} * \text{kazna_Po_Km} - \text{broj_Dostavljenih_Paketa} * \text{nagrada_Za_Dostavu}$$

Što je ocjena niža, to je rješenje bolje. Kako bi se odlučilo hoće li se paketi s trenutne lokacije postaviti u zrakoplov ili ne, uveden je prag koji se može kontrolirati pomoću parametra. Ako je ocjena niža od definiranog praga, paketi će se postaviti u zrakoplov. U suprotnom provjerava se ocjena za iduću lokaciju u listi.

Nakon što je donesena odluka da se paketi s trenutne lokacije postave u zrakoplov, u zrakoplov se postavljaju svi paketi dok se ne napuni kapacitet zrakoplova. Tu postoje određeni rubni slučajevi;

- zrakoplov potpuno pun, preostalo još paketa na lokaciji
- zrakoplov nije pun, preostali paketi su pojedinačno veći nego raspoloživi kapacitet

U prvom slučaju preostali paketi se vraćaju u listu svih paketa te se ostavljaju za neku iduću iteraciju algoritama, odnosno postavit će se u neki drugi zrakoplov.

Drugi slučaj je kompliciraniji jer za neku drugu lokaciji može postojati dosta malih paketa koji bi stali u preostali raspoloživi kapacitet zrakoplova. Odluka hoće li se postaviti ti paketi u zrakoplov donosi se opet na temelju ocjene za lokaciju, ali ovaj put udaljenost se računa u odnosu na zadnju lokaciju koju bi zrakoplov trebao posjetiti.

Algoritam se izvodi sve dok se ne raspodijele svi paketi.

3.2. Algoritam kolonije mrava

Mravi su jednostavna bića koja svojim ponašanjem i kretanjem u prirodi zadivljuju ljude koji ih proučavaju. Promatranjem mravi, pokazano je da oni uvijek pronalaze najkraći put od staništa do hrane. Kako su po svojoj osnovi slijepi, njihovi rezultati su još fascinantniji. Mravi na putu od hrane to staništa, ostavljaju kemijski trag (feromon). Osjetom feromona odlučuju kojim putem će krenuti, a odluka se donosi na osnovi jakosti feromonskog traga koji osjećaju.

U početku mravi nasumično odabiru put kojim će se kretati, ali kako vrijeme prolazi i više mravi prođe putem, onaj put koji je kraći će imati veću koncentraciju feromona, dok na dužem putu feromoni brže isparavaju i na kraju će se dogoditi da svi mravi idu istim putem.

Eksperimentom je potvrđeno da se mravi oslanjaju na feromonski trag, tako da je stvoren umjetni duži put do hrane i nakon što je postavljen dovoljno veliki feromonski trag, dodan je i kraći put, ali većina mravi se nastavila kretati dužim putem zbog stvorenog jakog feromonskog traga.

Količina feromona koju mrav doda na neki put je 0, ako nije prošao tim putem. Ako je mrav prošao tim putem, ostavlja neki određeni iznos traga ili obrnuto proporcionalan s duljinom puta koji je prošao.

Po svojem načinu rada algoritam spada u kategoriju evolucijskih algoritama. Ne koristi tipične rutine genetskih algoritama, već se zasniva na kolektivnom znanju i dijeljenju informacija među mnogim jedinkama.

Pregled rada algoritma

1. Konstrukcija rješenja
2. Ažuriranje feromonskih tragova

Konstrukcija rješenja

Mrav će se pomaknuti iz čvora *i* u *j* s vjerojatnošću

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})} \tag{3-1}$$

gdje je:

$\tau_{i,j}$ – jačina feromona na čvoru i,j

α – kontrolira utjecaj feromonskog trag $\tau_{i,j}$

$\eta_{i,j}$ – heuristička informacija koja govori koliko je dobro otići iz grada *i* u *j*

β – parametar koji kontrolira utjecaj heurističke informacije

Ažuriranje feromona

Jačina feromona se ažurira u skladu sa sljedećom jednadžbom

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{3-2}$$

gdje je:

ρ – brzina isparavanje (od 0 do 1)

$\Delta\tau_{i,j}$ – količina deponiranog feromonskog traga

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k}, & ako\ se\ mrav\ pomakne\ iz\ čvora\ i\ u\ j \\ 0, & inače \end{cases}$$

gdje je: L_k duljina pronađenog puta.

3.2.1. Implementacija na problemu dostave

Algoritam kolonije mrava [5] na problemu dostave paketa implementiran je na sljedeći način. U početku se svi paketi grupiraju prema lokacijama. Matricom feromona (slika 3.1.) predstavljene su vrijednosti koje svaki put između lokacija ima (veća vrijednost = veća vjerojatnost prolaska tim putem). Pretpostavimo da je posjeta svakog grada inicijalno 1.

	ZG	SB	BUK	DU	LJU
ZG		1	1	1	1
SB	1		1	1	1
BUK	1	1		1	1
DU	1	1	1		1
LJU	1	1	1	1	

Slika 3.1. Inicijalno postavljena matrica feromona

Gradnja rješenja kreće tako da se lokacije određene za dostavu postave u kolekciju koju će mrav obilaziti na isti način u svakoj iteraciji petlje. Za sve ne raspodijeljene pakete se dohvaća udaljenost u odnosu na trenutnu lokaciju zrakoplova te se provjerava hoće li se paketi s odabrane lokacije staviti u zrakoplov. Vjerojatnost prihvaćanja lokacije određena je sljedećim formulama.

$$Rezultat = \text{Broj_Neraspodijeljenih_Paketa_S_Lokacije} * \boldsymbol{PV} - \text{udaljenost} \tag{3-3}$$

$$Ukupni\ rezultat = \boldsymbol{SC} * \text{Rezultat} * \boldsymbol{PC} * \text{Vrijednost traga} \tag{3-4}$$

$$Prag = \frac{\text{Ukupni rezultat}}{\sum Ukupni\ rezultat} \tag{3-5}$$

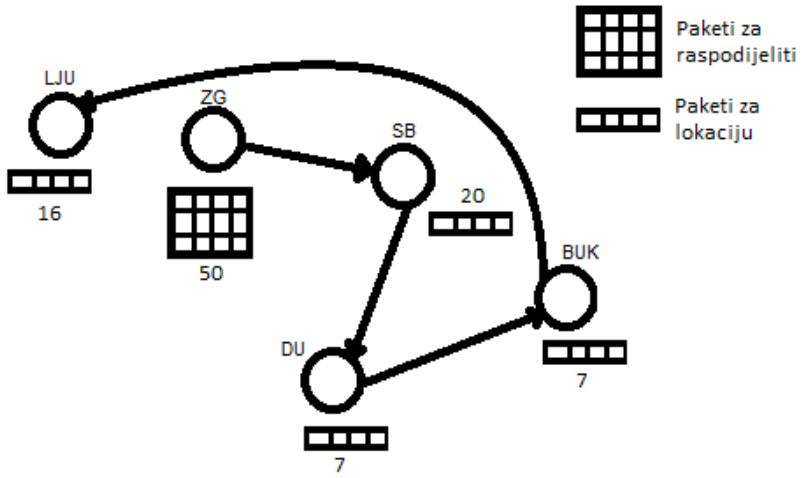
$$Lokacija\ se\ prihvaća\ ako\ je\ \text{Nasumično generirani broj} < \text{Prag}$$

gdje je:

- PV – vrijednost pojedinog paketa
- SC – koeficijent kojim se kontrolira utjecaj heuristike
- PC – koeficijent kojim se kontrolira utjecaj feromona

Ako se posjeta lokacije prihvati, u zrakoplov se stavljaju svi paketi s lokacije pri čemu prednost imaju paketi s manjom masom (maksimizira se broj paketa). Kad se zrakoplov napuni paketima s lokacije udaljenost sljedeće lokacije računa se u odnosu na posljednje posjećenu. Proces odluke o

prihvaćanju paketa s novih lokacija se odvija sve dok u zrakoplovu postoji prostor za stavljanje novih paketa. U slučaju da zrakoplov nije u mogućnosti uzeti sve pakete s lokacije, preostali paketi bit će stavljeni na neki drugi let.



Slika 3.2. Primjer rješenja jednog mrava za 4 lokacije

Nakon što svaki mrav izgradi svoje rješenje (slika 3.2.), njihova rješenja se usporede i uzme se najbolje rješenje (najmanja udaljenost). Na osnovu najboljeg rješenja osvježava se matrica feromona(slika 3.3.). Prvo se svi putevi ishlape za vrijednost ishlapljivanja i zatim se nanose novi feromoni na puteve koji su posjećeni. Ovisnost koliko će se novih feromona nanijeti na put je određena formulom (3-6).

$$Novi\ feromoni = \frac{10}{\text{Udaljenost između lokacija}} \tag{3-6}$$

Primjer izračuna količine feromona koja će dodati na put Zagreb – Slavonski Brod

$$Novi\ feromoni = \frac{10}{174.22}$$

$$Novi\ feromoni = 0.057$$

	ZG	SB	BUK	DU	LJU
ZG		0.95+0.057	0.95	0.95	0.95
SB	0.95		0.95	0.95+0.035	0.95
BUK	0.95	0.95		0.95	0.95+0.011
DU	0.95	0.95	0.95+0.015		0.95
LJU	0.95	0.95	0.95	0.95	

Slika 3.3. Dodavanje nove količine feromona na posjećene puteve za rješenje sa slike 3.2.

3.3. Genetski algoritam

Genetski algoritam [6] inspiriran je prirodnom selekcijom i ima primjenu u problemima koji zahtijevaju optimizaciju. Algoritam spada u grupu evolucijskih algoritama.

Pri radu algoritma stvara više od potrebnih jedinki te dolazi do borbe za opstanak u populaciji. Jedinke s boljim svojstvima imaju veću šansu preživljavanja i dobivaju priliku dalje se razmnožavati. Razmnožavanjem jedinke (djeca) uvelike nasljeđuju svojstva roditelja, no postoji određeno odstupanje u potomcima koje nastaje procesom križanja i mutacije kako bi se izbjeglo zaglavljivanje u lokalnom minimumu.

Algoritam u osnovi radi po principu da se stvori populacija jedinki gdje svaka jedinka predstavlja moguće rješenje. Dobrotom se mjeri kvaliteta rješenja jedinki, tako da što je veća dobrota jedinke, vrijednost funkcije će biti veća i ta jedinka predstavlja bolje rješenje.

Prilikom stvaranja potomaka proces koji se obavlja sadržava:

- Selekciju
- Križanje
- Mutaciju

Operatorom selekcije biraju se jedinke koje će u sljedećoj iteraciji postati roditelji pomoću kojih će se stvoriti jedinke koje će biti članovi nove populacije.

Operatorom križanja roditelji stvaraju djecu, koja će u konačnici imati izmijenjena svojstva u odnosu na roditelje, izgradit će se drugačije rješenje koje može biti bolje od roditelja.

Nad potomcima djeluje operator mutacije i iteracija petlje algoritma završava operatorom zamjene u kojem potomci ulaze u populaciju rješenja.

Selekcija

Jedinke iz populacije su odabrane da budu roditelji koji će se križati za stvaranje potomaka. Samo najbolje jedinke preživljavaju i stvaraju nove potomke.

Križanje

Križanjem svojstava dvije jedinke (roditelji) dobivaju se jedna ili dvije nove jedinke (djeca).

Mutacija

Mutacijom potomka mijenjaju se mali dijelovi rješenja, svaki dio rješenja ima jednaku vjerojatnost mutiranja.

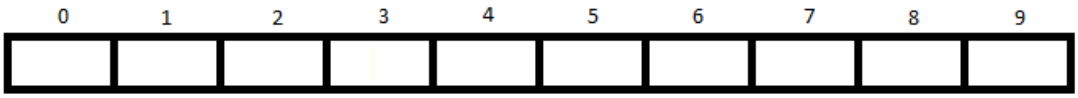
Pregled rada algoritma

1. Stvori se populacija od n jedinki
2. Procjenjuje se dobrota f(x) svake jedinke u populaciji
3. Stvaranje nove populacije
    a. Selekcija – odaberi dvije jedinke koje predstavljaju roditelje prema dobroti njihove funkcije
    b. Križanje – križanjem roditelja stvori potomke.
    c. Mutacija – operatorom mutacije promijeni svojstva potomka
    d. Prihvaćanje – dodaj potomke u novu populaciju
4. Zamjena – za daljnji rad algoritma koristi novu populaciju
5. Test – ako je krajnji uvjet zadovoljen, zaustavi rad algoritma i vrati najbolje rješenje iz trenutne populacije
6. Petlja – ponavljaj od drugog koraka

3.3.1. Implementacija na problemu dostave

Implementacija genetskog algoritma na problemu dostave izvedena je na sljedeći način.

Populacija nad kojom će se kasnije raditi potomci inicijalno je stvorena s 20 članova. Algoritam se izvodi određeni broj iteracija dok se ne ispuni uvjet zaustavljanja. Rješenja su prezentirana poljem bitova. Za primjer problema dostave 10 paketa, svi paketi koje je potrebno dostaviti stavljaju se u polje od 10 elemenata (slika 3.4.), uz prethodno grupiranje po lokacijama.

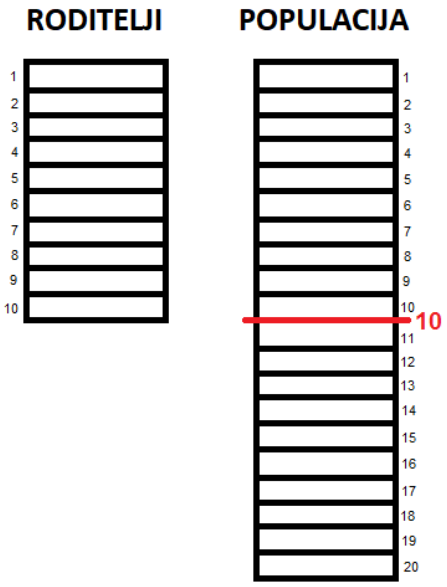


Slika 3.4. Polje s paketima

Svaki zrakoplov stvara svoje polje od 10 elemenata kojim je predstavljeno rješenje. Ako je paket ubačen u zrakoplov, na njegovo mjesto u polju stavlja se vrijednost „1“ (uzet) i proces prolaska kroz polje se izvodi sve dok u zrakoplovu ima prostora za staviti nove pakete. Nakon što je

zrakoplov napunjen uzima se novi i cijeli proces se ponavlja ispočetka sve dok se svi paketi ne uzmu za dostavu.

Inicijalno izgrađena rješenja ulaze u listu „Populacija“ (slika 3.5.) te se iz nje povlači 10 najboljih rješenja koja se sele u listu „Roditelji“. Rješenja iz liste „Roditelji“ se koriste za stvaranje potomaka.

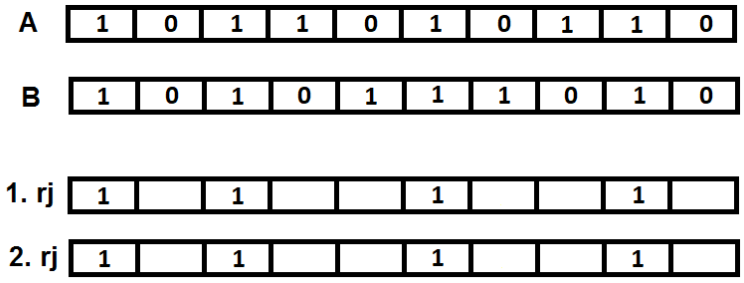


Slika 3.5. Dvije liste u kojima se rješenja drže

Rezultat križanja roditelja je 10 novih rješenja (djeca). Nova rješenja ulaze u listu „Populacija“ i u sljedećoj iteraciji se opet uzimaju 10 najboljih rješenja koja će predstavljati roditelje.

Prema kvaliteti rješenja 10 najboljih jedinki iz populacije postaju roditelji i nad njima se vrši proces stvaranja djece pomoću križanja i mutacije dva roditelja.

Križanje se provodi u dva koraka. U prvom koraku križanja (slika 3.6.) gledaju se planovi dostave za identične zrakoplove. Svi paketi koji se za određeni zrakoplov nalaze u oba roditelja, oni će biti stavljeni i u dijete.

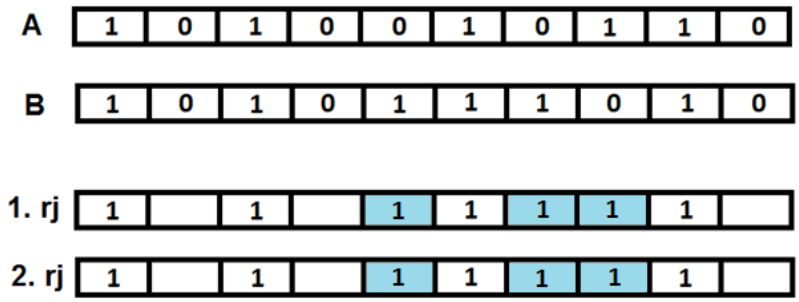


Slika 3.6. Prvi dio križanja

U drugom koraku potrebno je voditi računa o ispunjenju dva uvjeta:

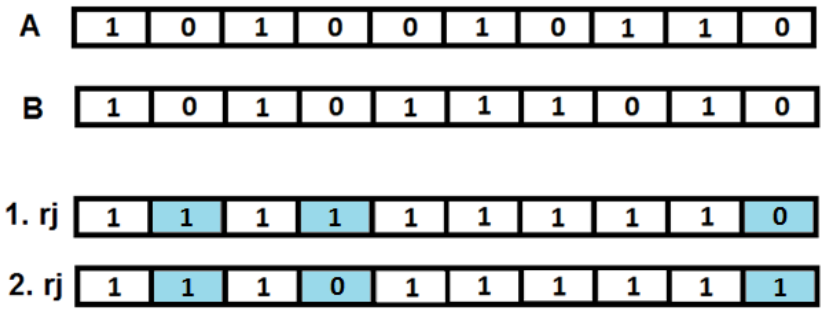
- Ubacivanjem novog paketa kapacitet zrakoplova nije premašen
- Paket je samo jednom dostavljen (postoji mogućnost da mutacijom roditelja na nekom drugu letu paket već bude dostavljen)

U drugom koraku (slika 3.7.) gleda se je li paket stavljen na let jednog od roditelja te se provjera ispunjenost uvjeta, ako je sve u redu paket će se staviti u novo rješenje.



Slika 3.7. Drugi dio križanja

Zasad su dva nova rješenja identična. U zadnjoj fazi provodi se mutacija (slika 3.8.) samo nad paketima koji nisu dostavljeni na letovima kod oba roditelja.



Slika 3.8. Rješenje nakon provedene mutacije nad roditeljima

Ako nakon mutacije u novom rješenju ostanu ne raspodijeljeni paketi, oni će se postaviti u novo-inicijalizirani zrakoplov.

3.4. Simulirano kaljenje

Algoritam [7] koji je motiviran procesom kaljenja metala, tehnikom grijanja i kontroliranog hlađenja materijala kako bi se postigla veća čvrstoća materijala, odnosno veća kvaliteta materijala. Simulirano kaljenje je metoda optimizacije u kojoj se temperatura sporo smanjuje. Pri visokoj temperaturi omogućuje se prihvaćanje velikog broja novih rješenja te kako se proces hladi vjerojatnost prihvaćanja lošijeg rješenja se smanjuje što omogućava algoritmu da se fokusira na područje koje je blizu optimalnog rješenja.

Po svojim svojstvima algoritam je dobar za izbjegavanje lokalnog minimuma. Bolja rješenja se automatski prihvaćaju, a lošija s nekom vjerojatnošću. Lošija rješenja od trenutnog se prihvaćaju iz razloga da se izbjegne zaglavljivanje u lokalnom minimumu.

Prihvaćanje rješenja:

- Prvo se provjeri je li bolje od trenutnog rješenja
    - Ako je, onda se automatski prihvaća
- Ako nije
    - Provjeri se prvo koliko je lošije od trenutnog rješenja
    - Koliko je trenutno velika temperaturna varijabla sustava

Vjerojatnost prihvaćanja

$$P = e^{-\frac{\text{Cijena_Susjednog_Rješenja} - \text{Cijena_Trenutnog_Rješenja}}{\text{Trenutna_Temperatura}}} \tag{3-7}$$

$$\textit{Rješenje se prihvaća, ako je } \text{Nasumično generirani broj} < P$$

Što je manja razlika između cijene susjednog i trenutnog rješenja i što je veća temperatura, veća je vjerojatnost prihvaćanja tog rješenja.

Pri inicijalizaciji temperature, u svrhu pronalaženja boljeg rješenja, početno se postavlja visoka temperatura kako bi se mogao napraviti bilo koji pomak u odnosu na trenutno rješenje. To mu omogućava da se pretražuje veći prostor prije nego što se ohladi i fokusira na jedno područje u pronalasku dovoljno dobrog rješenja

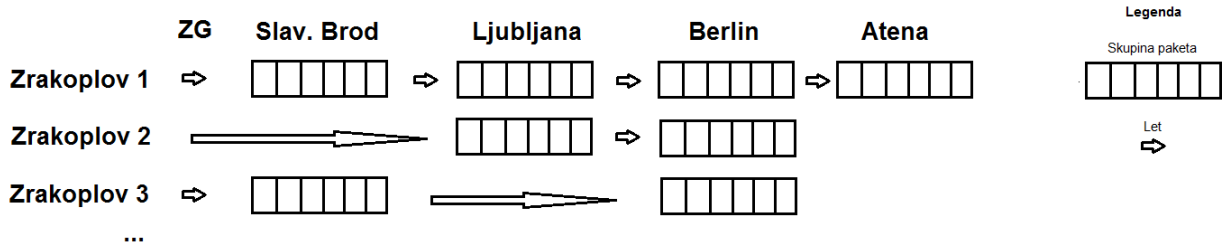
Pregled rada algoritma

1. postavi se inicijalno temperatura i stvori se inicijalno rješenje
2. petlja sve dok se uvjet za prekidanje rada ne zadovolji:
    a. sustav se ohladio
    b. dovoljno dobro rješenje
3. dohvatiti susjedno rješenje koje se malo razlikuje od trenutnog
4. odluka hoće li se ono prihvatiti kao novo rješenje
5. smanjiti temperaturu i nastaviti petlju

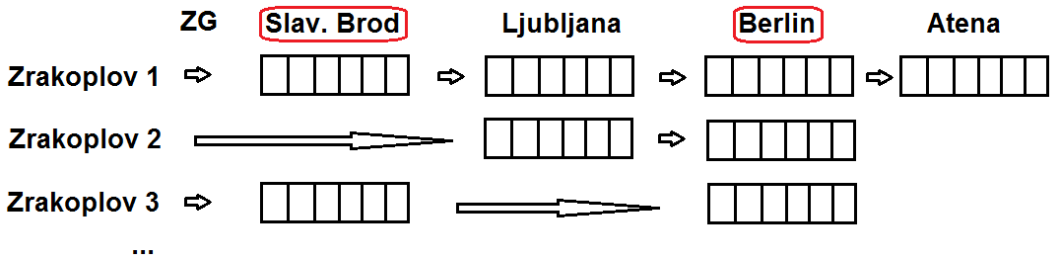
3.4.1. Implementacija na problemu dostave

Pronalazak optimalnog puta potpomognutog simuliranim kaljenje izveden je na sljedeći način. Kao inicijalno rješenje postavlja se rezultat koji Pohlepni algoritam pruža.

Gradnja novog rješenja se provodi tako da se iz prethodno prihvaćenog rješenja uzmu dvije lokacije i njihove pozicije u rješenju se zamijene. U sljedećem primjeru (slika 3.9.) prikazana je gradnja rješenja za četiri lokacije na koje je potrebno raspodijeliti pakete.



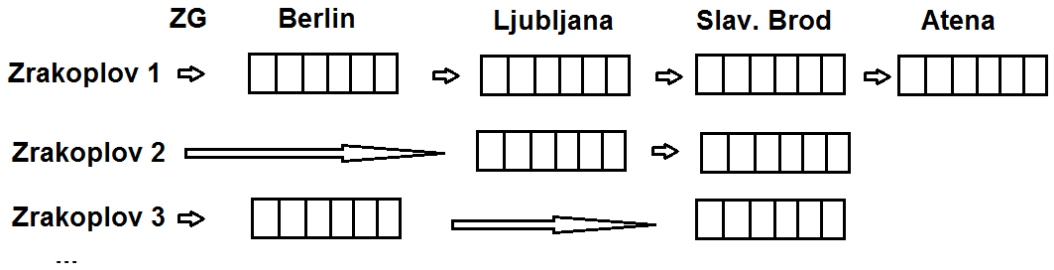
Slika 3.9. Primjer rješenja i planovi letova za zrakoplove



Slika 3.10. Odabir dvije lokacije za zamjenu

Nakon što su odabrane dvije lokacije (slika 3.10.) kojima će se zamijeniti mjesta, svi paketi se izvade iz svih zrakoplova te se potom ponovno ukrcavaju uvažavajući zamijenjeni redoslijed

lokacije (slika 3.11.). Ako se paketi s neke lokacije zbog nedostatka kapaciteta ne mogu staviti u zrakoplov, oni će biti preraspodijeljeni na neki novi let.



Slika 3.11. Novo rješenje nakon zamjene mjesta dviju lokacija

Kvaliteta novog rješenja se potom uspoređuje s prethodno prihvaćenim rješenjem i ako je bolja, automatski se prihvaća i postavlja kao najbolje, ako nije onda se prema funkciji vjerojatnosti odlučuje hoće li se rješenje prihvatiti i koristiti u gradnji novog rješenja.

$$P = e^{-\frac{\text{Ukupna udaljenost novog rj.} - \text{Ukupna udaljenost prethodno prihvaćenog rj.}}{\text{Trenutna temperatura sustava}}} \tag{3-8}$$

$$\textit{Rješenje se prihvaća, ako je } \text{Nasumično generirani broj} < P$$

Vjerojatnost prihvaćanja novog rješenja je veća što je manja razlika u udaljenosti između novog rješenja i prethodno prihvaćenog rješenja te što je temperatura sustava veća. Temperatura sustava je incijalizirana na 10000 jedinica vrijednosti i u svakoj iteraciji petlje se smanjuje za 1%.

Izbjegavanje lokalnog minimuma može se izvesti tako da ako se dogodi da 100 iteracija petlje prođe bez pronalaska boljeg rješenja, proces se pokreće ponovo i kao zadnje prihvaćeno rješenje se postavlja dosad najbolje pronađeno rješenje.

3.5. Branch and bound metoda

Algoritam [9] za pronalaženje optimalnih rješenja u diskretnoj i kombinatoričkoj optimizaciji zasnovan na implicitnoj enumeraciji svih rješenja. Prvi put algoritam su predstavili A. H. Land i A. G. Doig 1960. godine. Sastoji se od davanja velikog skupa mogućih rješenja koja su predstavljena u obliku stabla. Svako rješenje je predstavljeno u dubinu do konačne razine.

Algoritam prilikom svog rada obavlja dva procesa. Proces dijeljenja kojim se skup rješenja dijeli dva ili više sličnih skupova. Taj proces se naziva grananje kojim se definira struktura stabla čiji

čvorovi su podskupovi rješenja. Drugi proces je izračunavanje gornje i donje granice datog podskupa rješenja. Taj proces se naziva spajanje.

Provjeravanje svake grane do krajnje razine je nemoguće izvesti u realnom vremenu. Kako bi to vrijeme pretraživanja skratili, grane se režu po principu za dijelove stabla za koje znamo da sigurno ne sadrže bolje rješenje od trenutnog.

Glavna ideja algoritma je usporedba donje granice jednog čvora i gornje granice drugog čvora, ako je donja granica veća od gornje onda se taj čvor može izbaciti iz mogućih rješenja, odnosno ta grana se reže iz stabla.

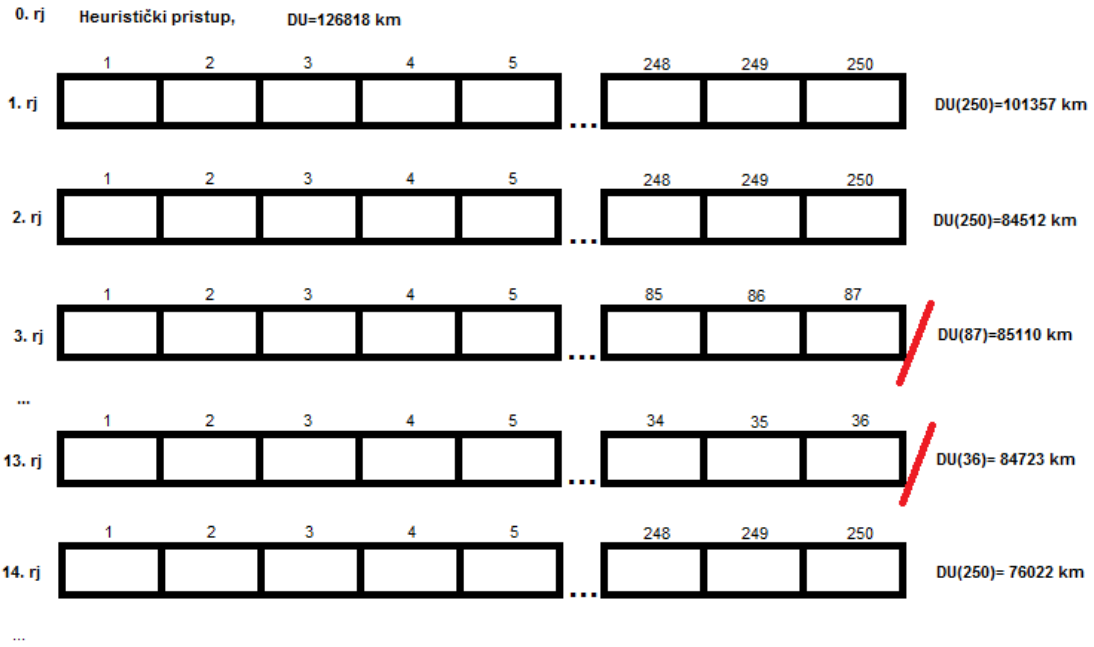
Rad algoritma se prekida kad se kandidat iz skupa rješenja smanji na jedan element, ili kada se gornja i donja granica iz skupa rješenja izjednače. Kandidat iz skupa rješenja predstavlja minimum funkcije i dobiveno rješenje.

Algoritam ne garantira kratko vrijeme izvođenja, ono zavisi od učinkovitosti rezanja grana. U najgorem slučaju za dobivanje optimalnog rješenja potrebno je cijelo stablo izračunati, dok u najboljem slučaju samo jedan put do krajnje razine, a ostale grane odrezati. Potpomognut heuristikama moguće je na temelju raspona između donje i gornje granice te kad je taj raspon manji od definiranog praga da se zaustavi daljnje grananje. Koristimo ih kad se ne zahtijeva potpuno optimalno rješenje, već dovoljno dobro za određeni tip zadatka. Taj način znatno smanjuje broj iteracija petlje, vrijeme obrade zadatka i zahtjeve na računalnu opremu.

3.5.1. Implementacija na problemu dostave

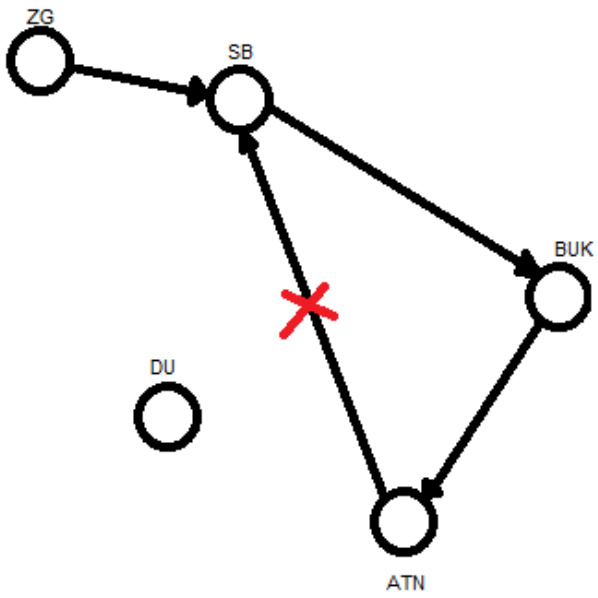
Kao inicijalno rješenje postavljeno je rješenje koje daje Pohlepni algoritam. Branch and bound je verzija egzaktne metode koja omogućuje podrezivanje stabla gradnje rješenja kad se postavljeni uvjeti ne zadovolje. Prilikom gradnje novih rješenja vođeno je računa o ispunjenju dva kriterija:

- Prije uzimanja novog paketa koji će se staviti u zrakoplov, ukupno prijeđena udaljenost rješenja ne prelazi ukupnu udaljenost dosad najbolje pronađenog rješenja (slika 3.12.)
- Svi paketi s iste lokacije koji su stavljeni u zrakoplov trebaju se dostaviti prilikom jednog posjeta gradu (slika 3.13)



Slika 3.12. Gradnja rješenja i prekidanje gradnje u trenutku prelaska ukupne udaljenosti najboljeg rješenja.

Konstantnom provjerom udaljenosti osigurano je prekidanje daljnje gradnje rješenja u trenutku kad novo uzeti paket proizvede rješenje koje je veće od najboljeg, jer svaki sljedeći paket će dodatno povećavati ukupnu udaljenost koja je već postala veća od trenutno najboljeg rješenja i samim time novo rješenje je lošije.



Slika 3.13. Posjećivanje istog grada dva puta

Rješenje koje posjećuje isti grad dva puta također se zaustavlja, jer posjećivanje istog grada dva puta nije efikasno i logično. Sve pakete s iste lokacije na letu potrebno je dostaviti istovremeno, bez ponovnog vraćanja na istu lokaciju.

U onom trenutku kad novo rješenje bude veće od gornje granice, ono se prekida i prelazi se na gradnju novog. Što omogućava obustavljanje gradnje rješenja u ranoj fazi, ako zasigurno znamo da će ono biti lošije od rješenja kojeg trenutno imamo.

3.6. Analiza rezultata

Implementirani pristupi rješavanja optimizacijskog problema usporedit će se po dva kriterija:

- brzina izvođenja
- kvaliteta rješenja (ukupno prijeđeni broj kilometara)

Kako bi se stekao bolji dojam kako se algoritmi ponašaju s veličinom početnog problema testirat će se izvođenje za sljedeći broj paketa za dostavu: 10, 50, 100, 250, 500, 1000. Za potrebe testiranja koristit će se zrakoplovi kapaciteta 2000 jedinica. Veličina paketa nasumično se odabire pri čemu maksimalna moguća vrijednost iznosi 250 jedinica. Lokacije koje se koriste kao destinacije za dostavljanje paketa su nasumično odabrane po teritoriju Europe i za potrebe testiranja dostava se izvršava na 20 različitih destinacija.

3.6.1. Rezultati

Rezultati testiranja vremena izvođenja (tablica 3.1.) i kvalitete rješenja (tablica 3.2.) prikazani su u sljedećim tablicama.

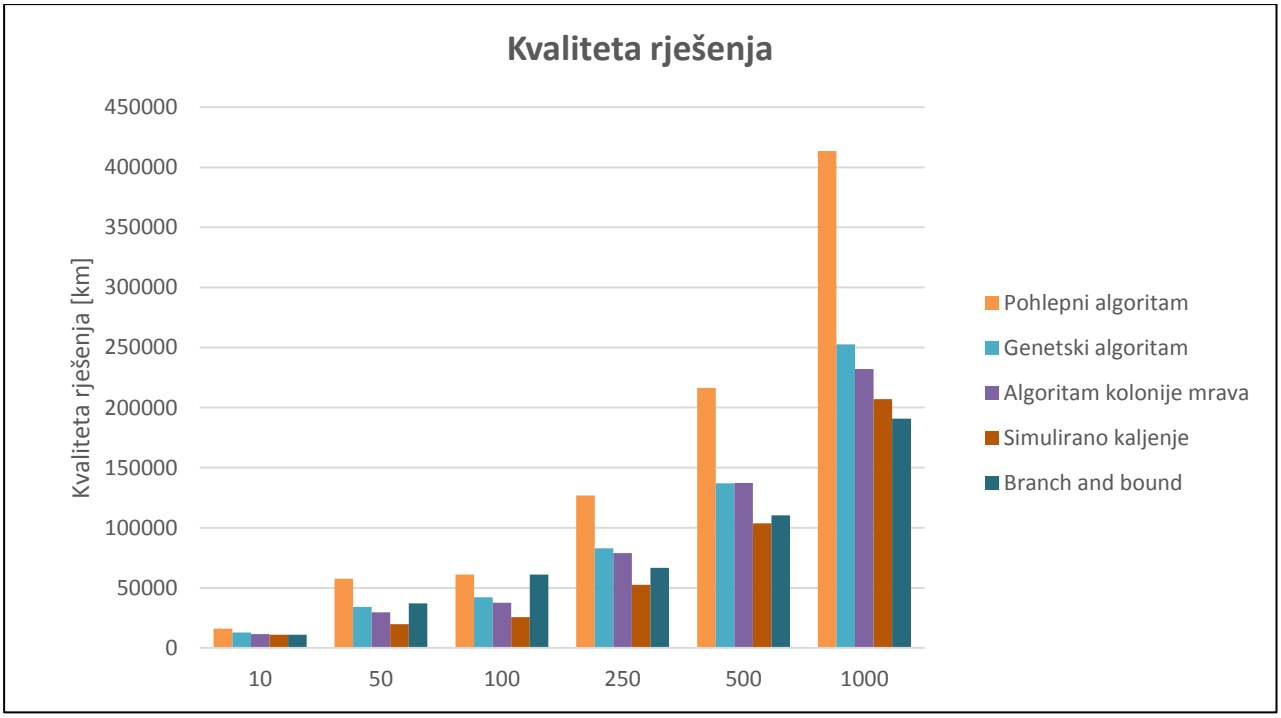
Tablica 3.1. Vremena izvođenja algoritama do zadovoljavanja uvjeta zaustavljanja

	Pohlepni algoritam	***Genetski algoritam***	***Algoritam kolonije mrava***	***Simulirano kaljenje***	***Branch and bound***
Paketi	**Vrijeme [ms]**	**Vrijeme [ms]**	**Vrijeme [ms]**	**Vrijeme [ms]**	**Vrijeme [ms]**
10	7	368	6667	316	438
50	13	3066	37437	2563	1000
100	19	11432	40884	3764	1000
250	31	56688	72050	6994	1000
500	32	294778	288029	12705	1000
1000	62	1857956	393514	24985	1000

Tablica 3.2. Usporedba rezultata testiranja prema kvaliteti izvršenja

	Pohlepni algoritam	***Genetski algoritam***	***Algoritam kolonije mrava***	***Simulirano kaljenje***	***Branch and bound***
Paketi	**Kvaliteta [km]**	**Kvaliteta [km]**	**Kvaliteta [km]**	**Kvaliteta [km]**	**Kvaliteta [km]**
10	15962	12860	11434	10871	10871
50	57474	34101	29650	19684	37074
100	60936	42268	37591	25611	60936
250	126818	83018	78909	52576	66663
500	216334	137051	137329	103605	110459
1000	413589	252580	232071	207125	190898

Kako bi se razlika u rezultatima mogla bolje uočiti, kvaliteta rješenja je prikazana na sljedećim grafom (slika 3.14.).



Slika 3.14. Usporedba kvalitete rješenja (manje je bolje)

Vremena izvođenja algoritama nije moguće pravovaljano uspoređivati, jer ona isključivo ovise o uvjetima zaustavljanja algoritma.

Analiza kvalitete pronađenih rješenja daje zanimljive rezultate. Pohlepni algoritam daje najlošije rezultate od svih algoritama korištenih prilikom testiranja. Genetski algoritam i Algoritam kolonije mrava tijekom cijelog testiranja pokazuju isti trend kvalitete rješenja uz nešto bolji rezultat od strane algoritma kolonije mrava. Za 10 paketa Simulirano kaljenje i Branch and bound uspijevaju pronaći optimalno rješenje. Simulirano kaljenje daje najbolje rezultate u svim testiranjima osim za 1000 paketa, gdje Branch and bound pronađe 8% bolje rješenje.

4. ZAKLJUČAK

U ovom diplomskom radu izrađen je program za rješavanje problema dostave paketa zračnim putem. Prije same izrade bilo je potrebno pronaći adekvatne algoritme čija rješenja će se uspoređivati. Iz rezultata dobivenih testiranjem jasno je vidljivo da Pohlepni algoritam daje najlošije rezultate uz zanemarivo vrijeme izvođenja. U sustavima gdje je najvažnija brza reakcija, Pohlepni algoritam uz ovakve rezultate može imati primjenu. Ostale algoritme prema vremenu izvođenja nije moguće uspoređivati zato što njihova vremena isključivo ovise o uvjetima zaustavljanja. Prema kvaliteti rješenja Genetski algoritam i Algoritam kolonije mrava daju slične rezultate koji su lošiji od Simuliranog kaljenja i Branch and bound algoritma. Ako uzmemo u obzir i vrijeme izvođenja koje im je bilo potrebno za stvaranje takvih rješenja, Simulirano kaljenje i Branch and bound algoritmi se nameću kao glavna dva kandidata za primjenu u rješavanju problema dostave paketa zračnim putem.

LITERATURA

[1] Eclipse (software), http://en.wikipedia.org/wiki/Eclipse_(software), travanj 2017

[2] Head First Java, Kathy Sierra/Bert Bates, travanj 2017

[3] Heuristic (computer science), http://en.wikipedia.org/wiki/Heuristic_(computer_science), travanj 2017

[4] Heurističke metode optimizacije: Pohlepni Algoritmi, predavanja s FER-a, travanj 2017

[5] Ant colony optimization, http://www.scholarpedia.org/article/Ant_colony_optimization, travanj 2017

[6] Genetic Algorithms, https://www.tutorialspoint.com/genetic_algorithms/index.htm, travanj 2017.

[7] Single-Commodity Vehicle Routing Problem with Pickup and Delivery Service, travanj 2017

[8] Branch and Bound Algorithms, http://www.imada.sdu.dk/Employees/jbj/heuristikker/TSPtext.pdf, travanj 2017

SAŽETAK

Naslov: Usporedba algoritama za rješavanje problema dostave paketa pomoću flote vozila

Diplomski rad sastoji se od predstavljanja problema pronalaska optimalne putanje za dostavu paketa zračnim putem i dolaska do rješenja istog. Razvoj izrade rada odvijao se kroz četiri stupnja: definiranje problema, pronalaska odgovarajućih algoritama za dobivanje rješenja, razvoja i implementacija algoritama i analize rezultata. Programsko rješenje napravljeno je u programskoj okolini Eclipse u programskom jeziku Java. Informacije o paketima učitavaju se iz datoteke, paketi se sortiraju prema destinacijama i veličini paketa. Kapacitet svakog zrakoplova je ograničen. Optimiranje se izvodilo pomoću pet različitih algoritama te se potom uspoređivala kvaliteta rješenja koju pruža pojedini algoritam. Kao dva najbolja algoritma za optimizaciju pokazali su se Simulirano kaljenje i Branch and bound. Za ≤500 paketa, Simulirano kaljenje daje najbolja rješenja, a za sljedeći veći broj paketa koji je testiran (1000 paketa) Branch and bound je pružio najkvalitetnije rješenje. Dobiveno rješenje se potom vizualizira na mapi i iscrtava se plan dostave.

Ključne riječi: Optimiranje, Algoritmi, Vizualizacija, Kvaliteta rješenja, Analiza rezultata

ABSTRACT

Title: Algorithms comparsion for solving delivery problems with vehicle fleet

The final paper is composed of introducing the problem of optimal package delivery with airplanes to destinations and reaching its solution. Process for reaching the solution has been made in four steps: defining the problem, finding adequate optimization algorithm, development and implementation of the algorithm and analysis of results. Programming solution is written on Eclipse platform in the Java computer language. Package information is loaded from the file, packages are sorted according to destinations and package size. Airplanes capacity is limited. Optimization has been performed with five different algorithms, and then quality of result between algorithms has been compared. Two of the best performing optimization algorithms are Simulated annealing and Branch and bound. For ≤500 packages, Simulated annealing gives best solution, for a next bigger group of packages used in tests (1000 packages) Branch and bound provided best solution. A given solution is then visualized and delivery plan is plotted on the map.

Key words: Optimization, Algorithms, Visualization, Quality of results, Analysis of results

ŽIVOTOPIS

Arijan Hucaljuk rođen je 12. prosinca 1992. godine u Slavonskom Brodu. Po završetku osnovne škole „Hugo Badalić“ u Slavonskom Brodu, upisuje 2007. godine srednju Tehničku školu u Slavonskom Brodu, smjer Elektrotehničar te maturira 2011.g.. Od 2011. boravi u Osijeku gdje je upisao preddiplomski studij računarstva na Elektrotehničkom fakultetu kojeg redovno završava te stječe titulu sveučilišni prvostupnik inženjer računarstva. Nastavio je studij na istom fakultetu 2014.g. upisom na diplomski studij, smjer procesno računarstvo. Sve ispite je položio u roku i sada se nalazi pred obranom diplomskog rada.

PRILOZI

Kompletan izvorni kod nalazi se na CD-u priloženom uz ovaj rad.