Izrada programskog sučelja za rješavanje problema vremenski ovisnog usmjeravanja vozila

Ribić, Filip

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

FILIP RIBIĆ

IZRADA PROGRAMSKOG SUČELJA ZA RJEŠAVANJE PROBLEMA VREMENSKI OVISNOG USMJERAVANJA VOZILA

ZAVRŠNI RAD

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

ZAVRŠNI RAD

IZRADA PROGRAMSKOG SUČELJA ZA RJEŠAVANJE PROBLEMA VREMENSKI OVISNOG USMJERAVANJA VOZILA

DEVELOPMENT OF A PROGRAM INTERFACE FOR TIME DEPENDENT VEHICLE ROUTING PROBLEM

Mentor: Tomislav Erdelić, mag. ing. el. techn. inf.

Student: Filip Ribić

JMBAG: 0135243353

Zagreb, 2020.

Izrada programskog sučelja za rješavanje problema vremenski ovisnog usmjeravanja vozila

Sažetak:

Problem usmjeravanja vozila problem je vezan za područje inteligentnih transportnih sustava i logistike. Rješenje problema predstavlja skup ruta vozila za dostavu robe/dobara svim korisnicima u problemu. Matematički se problem modulira pomoću teorije grafova. Problem usmjeravanja vozila uobičajeno se prikazuje u statičkim uvjetima gdje su vremena putovanja između korisnika konstantne, vremenski nepromjenjive. U stvarnome svijetu brzina na prometnici vremenski je promijenjiva veličina te se i vremena putovanja između korisnika mijenjaju ovisno o trenutku polaska. Ovaj rad prikazuje računalno rješenje problema vremenski ovisnog usmjeravanja vozila izrađenog pomoću razvojnog okruženja Microsoft Visual Studio i programskog jezika C#. Vrijeme trajanja cijelokupne dostave diskretizirano je u nekoliko vremenskih intervala kako bi se prikazala stvarna slika prometa. Ovisno o trenutku polaska od korisnika, vozilo će se do sljedećeg korisnika kretati brzinom specifičnom za vremenski interval unutar kojega se vozilo kreće. Ovaj problem može se primjeniti na stvarni svijet na temelju sljedeće pretpostavke: „vrijeme obavljanja posla i prijeđeni put vozila biti će različiti ukoliko se do korisnika dođe u 08:00 sati ujutro ili u 18:00 sati popodne“. Ovisnost o vremenu prikazana je na ukupnom prijeđenom putu i ukupnom vremenu putovanja potrebnom da bi se posjetili svi korisnici. Da bi se što bolje prikazala promjena varijabli ukupnog prijeđenog puta i vremena, programsko riješenje testirano je se na više primjera različitih vremenskih intervala s različitim pripadajućim brzinama. Rješenje je prikazano na dvije vrste testnih problema s manjim i većim brojem korisnika.

Završni rezultat rada prikazuje uštedu u vremenu i udaljenosti ovisno o brzini, te se rezultati rada mogu primijeniti u mnogim današnjim prometnim problemima opskrbe korisnika, prijevoza robe te smještaja električnih i benzinskih pumpi. Primjenom rezultata u praksi može se doći do smanjenja transportnih troškova.

Ključne riječi: Problem usmjeravanja vozila, vremenski ovisno usmjeravanja vozila, heuristički algoritmi, lokalna pretraga, Visual Studio, C#

Development of a program interface for time dependent vehicle routing problem

Abstract:

Vehicle routing problem is related to the field of intelligent transport systems and logistics. The solution of the problem represents a set of vehicle routes to deliver goods to customers in the problem. The matematical interpretation of the problem is modeled by graph theory. Vehicle routing problem usually consideres static conditions where travel times between the customers are time invariant. In the real world vehicle speed changes through time, and the travel times between customers vary, depending on the departure time. This work presents a computer solution of time dependent vehicle routing problem created with use of Microsoft Visual Studio development environment and the C# programming language. The delivery time is discretizied in several time intervals in order to represent real traffic conditions. Depending on the time interval in which the vehicle leaves the customer it will move to the next customer at a speed specific to that time interval. This problem can be applied to the real world based on the following assumption: „the time of work and distance traveled by the vehicle will be different if the user is reached at 08:00 in the morning or at 18:00 in the afternoon“. The time dependency is reflected in the total traveled distance and the time required to visit all customers. To better represent the change of total distance traveled and total travel time variables, the software solution will be tested on several examples with different time intervals and different corresponding speeds. The solution is shown on two type test problems with smaller and larger number of users.

The final result of the work shows savings in time and distance depending on the speed. The results of the paper can be applied in many of today's traffic problems of customer supply, transport of goods and placement of electric and gas stations. Applying the results in practice can reduce transport costs.

Key words: Vehicle routing problem, time dependent vehicle routing problem, total distance traveled, total time, Visual Studio, C#

Sadržaj

1. Uvod

Prijevoz ljudi i tereta kompleksan je postupak u kojem je najbitnije napraviti uštedu u vremenu i udaljenosti. Ključno pitanje u putovanju između više korisnika je kojim rutama se kretati, te koliko iznose vrijeme putovanja i ukupna prijeđena udaljenost. Vozila koja se koriste u takvim procesima često su ograničena teretnim kapacitetom: volumenom ili masom. Problemi navedeni u radu temlje se na dostavi robe do korisnika.

Skraćenje vremena putovanja i vremena obavljanja posla pozitivno utječu na uštedu goriva i smanjenje troškova u logističkim procesima dostave robe i dobara. Bolje rješenje problema dostave često rezultira većom uštedom, tj. značajnim smanjenjem transportnih troškova. Za uspješan pronalazak najboljeg rješenja problema dostave koriste se računalni algoritmi sposobni rješavati velike probleme u što kraćem vremenu.

U ovom radu razmatra se problem vremenski ovisnog usmjeravanja vozila (eng. „Time Dependent Vehicle Routing Problem“, TD-VRP), kao podvrsta problema usmjeravanja vozila s ograničenim kapacitetom (eng. „Capacitated Vehicle Routing Problem“, CVRP) uz korištenje vremenski ovisnog putovanja cestovnim segmentima, [7]. Vrijeme jednog radnog dana diskretizirano je u različite vremenske intervale s ciljem uzimanja u obzir prometnih zagušenja koja se pojavljuju u tzv, „vršnim satima“. TD-VRP usmjerava vozni park uzimajući u obzir promjenjivo vrijeme putovanja cestovnom mrežom. U periodima zagušenja brzine su snižene kako bi se što više predočilo stvarno stanje prometnog sustava dok su u ostalim intervalima brzine viših vrijednosti, bliže brzini slobodnog toka.

Klasičan VRP ima za cilj odrediti rute za flotu vozila iz jednog ili više skladišta uz uvijet da se moraju posjetiti svi korisnici pri čemu svakog korisnika može posjetiti samo jedno vozilo. Pritom se razmatraju statični uvijeti na prometnicama, bez promjena brzine. Problem pripada području Inteligentnih transportnih sustava (eng. „Intelligent Transport Systems“, ITS) i logistike. Ključno pitanje na koje VRP odgovara je: „Koji je optimalni skup ruta potreban za flotu vozila da bi se obavila dostava robe do danog skupa korisnika?“. Problem su prvi predstavili George Danzig i John Ramser 1959. godine i dobiveno rješenje primijenili na problem dostave goriva benzinskim postajama. U tu svrhu George Danzig i John Ramser predložili su prvi matematički formuliran algoritam za rješavanje ovoga problema. Problem se izvorno nazvao problem otpremanja kamiona, [1]. Važnost problema usmjeravanja vozila može se uočiti u tome da se 76,5% svog svjetskog prometa izvršava cestovnim vozilima, [2].

Poznato pojednostavljenje VRP problema koji se javlja u području prometa i transporta je problem trgovačkog putnika (eng. „Travelling Salesman Problem“). Trgovački putnik ima za cilj obići određen broj zadanih gradova točno jednom i na kraju se vratiti u početni grad. Isti put između dva grada ne smije se proći više od jednom, [3].

CVRP je podrvsta VRP problema kod kojega se u obzir uzima teretni kapacitet vozila. Svako vozilo ima svoj maseni ili volumni kapacitet koji se ne smije prekoračiti. Ukoliko u problemu

dođe do prekoračenja kapaciteta vozila, vozilo se vraća u skladište i umjesto njega iz skladišta kreće novo vozilo koje posjećuje neposjećene korisnike.

Još jedna od podvrsta VRP problema je problem usmjeravanja vozila s vremenskim prozorima (eng. „Vehicle Routing Problem with Time Windows“, VRPTW). Kod ovoga problema uz ograničenje kapaciteta dodani su i vremenski prozori korisnika. Korisnici su definirani donjom i gornjom granicom vremena početka posluživanja korisnika. Vozilo mora doći do korisnika unutar navedenih granica. Unutar testnih problema koji se učitavaju u ovome radu zadani su i vremenski prozori korisnika, ali se oni neće razmatrati u ovome radu.

Vodeće svjetske kompanije poput Amazona ili FedEx – a i hrvatske transportne kompanije poput Hrvatske pošte rješenje navedenih problema mogu primjeniti na poslovanju svojih voznih parkova. Dobar primjer je Hrvatska pošta koja u svom voznom parku sadrži preko 3 700 vozila (uključujući teretna vozila, mopede i bicikle), [5].

Danas se sve više VRP problema primjenjuje na usmjeravanje električnih vozila (eng. „Electric Vehicle Routing Problem“, E-VRP). Električna vozila spadaju u najčišće načine transporta. Ograničenje i problem predstavlja njihov domet. Postavlja se pitanje: „Da li je domet električnog vozila dovoljan za vozačeve potrebe?“ Česta potreba za punjenjem vozila povećava radne troškove i teže se određuje ostvarivanje pravovremene dostave. Rješenjem problema smanjuje se put koji prolaze električna vozila, a samim time i njihovo punjenje, [6].

U radu se koriste heurističke metode rješavanja problema koje obuhvaćaju konstruiranje početnog rješenja i lokalnu pretragu. Heurističke metode mogu se opisati kao tehnika rješavanja problema bazirana na iskustvu. Koriste se da ubrzaju proces pronalaženja dovoljno dobrog rješenja u situacijama kada provođenje detaljnog istraživanja nije praktično. Obuhvaćaju korištenje zadanih pravila, nagađanja i intuicije. Heuristika poboljšanja ili lokalno pretraživanje (eng. „Local Search“, LS) istražuje susjedstvo početnog rješenja, tražeći bolje rješenje. Susjedstvo se istražuje zamjenom korisnika unutar iste ili različitih ruta vozila na temelju složenih intra i inter operatora susjedstva. Lokalna pretraga se zaustavlja kada se više ne može naći poboljšanje za trenutno rješenje koje se tada naziva lokalni optimum, [7].

U drugom poglavlju opisan je TD-VRP problem, navedene su vrste grafova te su opisani srodni problemi i matematička formulacija problema.

Treće poglavlje opisuje postupak učitavanja podataka testnih problema, postupak dobivanja početnog rješenja, navodi i objašnjava korištene matematičke izraze, dijelove programa te prikazuje i objašnjava početno rješenje za primjer s 25 i 100 korisnika. Nakon toga opisuje se postupak lokalne pretrage i njezinih intra i inter operatora s prikazanim primjerima. Na kraju poglavlja prikazano je i objašnjeno dobivanje završnog rješenja slučajnim izbacivanjem i ubacivanjem korisnika između vozila.

Četvrto poglavlje prikazuje i objašnjava utjecaj vremenski ovisnog putovanja na dobiveno rješenje. Dana su dva primjera s različitim vrijednostima vremena koje objašnjavaju i prikazuju utjecaj vremena i brzina na ishod rezultata.

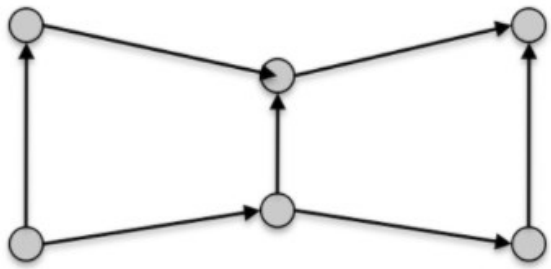
U petom poglavlju dan je zaključak rada.

2. Opis problema i korisnika

Rezultat ovoga rada daje približno rješenje TD-VRP problema jer je u stvarnome svijetu teško predvidjeti određene segmente koji utječu na rješenje. Zagušenja na prometnicama ne javljaju se na svakoj prometnici u isto vrijeme dana i ne traju jednaki vremenski period. Zbog toga rješenja ne mogu biti potpuno točna.

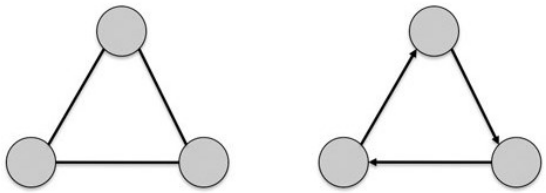
2.1. Grafovi

U prometu se zbog lakšeg razumijevanja prometna mreža prikazuje pomoću grafa. Korisnici se predstavljaju kao vrhovi, a cestovni segmenti kao bridovi grafa. Svakom bridu možemo pridružiti realan broj, što znači da je graf proširen težinskom funkcijom. U slučaju kada graf predstavlja mrežu cesta, težinska funkcija je duljina svakog puta između čvorova grafa. Takav graf zove se težinski graf. Graf G matematička je struktura koja se koristi za opisivanje relacija između dvaju objekata iz određene kolekcije. U ovom kontekstu graf se odnosi na neprazni skup vrhova i kolekciju bridova koji povezuju par vrhova. Skup vrhova obično se označava s $V(G)$, a skup bridova s $E(G)$. Bridovi mogu biti usmjereni ili ne, ovisno o primjeru. Graf kojemu se svi bridovi usmjereni zove se usmjereni graf, u suprotnom se zove neusmjereni graf. U pravom grafu, za koji se inicijalno pretpostavlja da je neusmjeren, linija od točke u do točke v identificira se s linijom od točke v do točke u. U digrafu (skraćeno za usmjereni graf), ta dva smjera smatraju se različitim lukovima, odnosno bridovima. Ako graf G nije usmjeren, tada su dva vrha pridružena jednom bridu međusobno ravnopravna. U usmjerenom grafu brid može biti usmjeren od jednog vrha prema drugome. Slike 1 i 2 primjeri su usmjerenih grafova, [8].



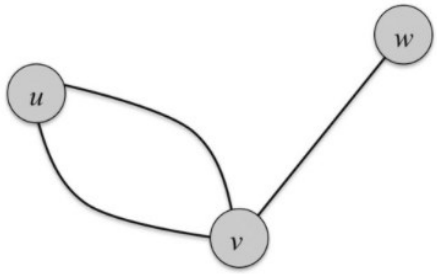
Slika 1: Usmjeren graf sa šest vrhova u sedam bridova, [8]

Ako se slika 1 zamisli kao prometna mreža, kretanje vozila prema korisnicima moguće je samo u smjeru strelica na bridovima.



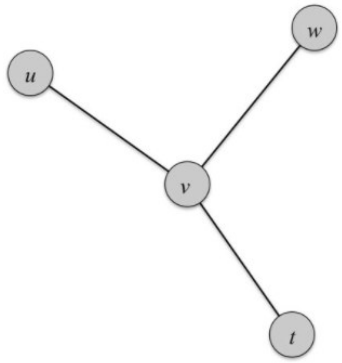
Slika 2: Neusmjeren i usmjeren graf, [8]

Brid koji počinje i završava u istom vrhu zove se petlja. Brid je višestruk ako postoji drugi brid kojemu odgovoaraju isti vrhovi (kao početni i krajnji vrh), [8].



Slika 3: Višestruki brid između vrhova u i v, [8]

Graf se naziva jednostavnim ako je neusmjeren, nema petlji i između bilo koja dva vrha nema više od jednog brida. U jednostavnom grafu svaki se brid može identificirati s parom različitih vrhova. Brid povezuje dva vrha. Ta dva vrha nazivaju se incidentnima tom bridu, odnosno brid je incidentan tim dvama vrhovima. Stupanj vrha v u grafu G je broj bridova koji su incidentni s v, pri čemu se petlje broje dva puta, [8].



Slika 4: Stupanj vrha v je 3, [8]

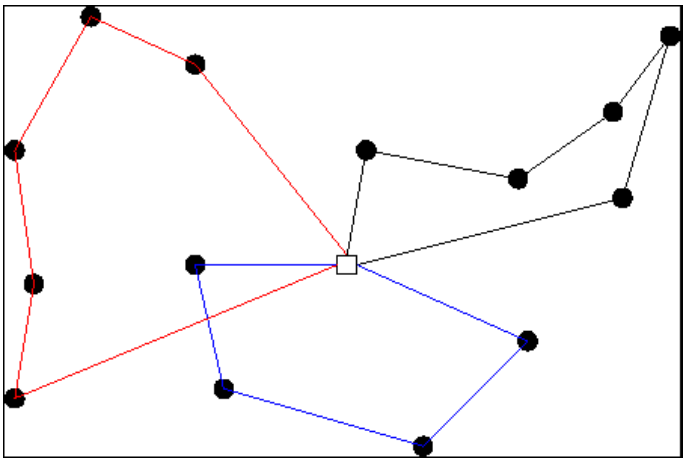
Slika 4 prikazuje da su bridovi vu, vt, vw incidentni vrhu v. Prema tome stupanj vrha v je 3, [8].

2.2 Problem kineskog poštara

Početkom 60-tih godina dvadesetog stoljeća kineski matematičar M.Guan postavlja i proučava problem kineskog poštara. To je primjer u kojem se pokušava naći šetnja kojom se prolazi kroz svaki brid na grafu samo jednom, učinivši to na najkraći mogući način, koristeći se usmjerenim ili neusmjerenim grafom. Za bolje razumijevanje zamišlja se poštar koji hoda ulicama (u ovom slučaju po grafu) i želi uručiti poštu u svaku kuću (vrhovi našeg grafa) u najkraćem vremenu i tada se vratiti u poštu (polaznu točku). Poštar nastoji uštediti vrijeme, trud i novac izvršujući svoj zadatak tako da upotrijebi najkraću rutu.

Problem trgovačkog putnika na prvi je pogled vrlo sličan problemu kineskog poštara. Bavi se slučajem u kojem se traži šetnja u usmjerenom ili neusmjerenom grafu tako da se prođe svaki vrh u grafu barem jednom i vrati se u početni vrh na najkraći mogući način. Može se zamisliti da su pri tome zadane i udaljenosti između vrhova, pa se traži i da je ukupna prijeđena udaljenost najmanja, [8].

2.3. Problem usmjerevanja vozila



Slika 5: Problem usmjeravanja vozila VRP, [1]

Slika 5 prikazuje objašnjenje problema usmjeravanja vozila, VRP. Bijeli kvadrat na slici prikazuje skladište iz kojega vozila kreću i u koje se vraćaju sva vozila. Crne točke na slici prikazuju korisnike opisane svojim atributima. Svaki korisnik ima svoju x i y koordinatu pomoću koje su smješteni u koordinatni sustav. Na prikazanom primjeru mogu se primjetiti tri rute vozila: crvena, plava i crna. Svaka ruta je zasebno vozilo i obavlja posluživanje samo svojih korisnika.

Slika 5 objašnjava i problem usmjeravanja vozila s ograničenim kapacitetom. Da postoji samo jedno vozilo s neograničenim kapacitetom na slici bi bila prikazana jedna ruta s jednom bojom. Iz prikazane slike vidljiva su različita tri vozila. Nakon što pojedino vozilo nakon određenog broja korisnika dosegne svoj maksimum kapacita, npr. popuni/isprazni svoj teretni prostor, vraća se u skladište. Uvjet je da svakog korisnika posluži samo jedno vozilo.

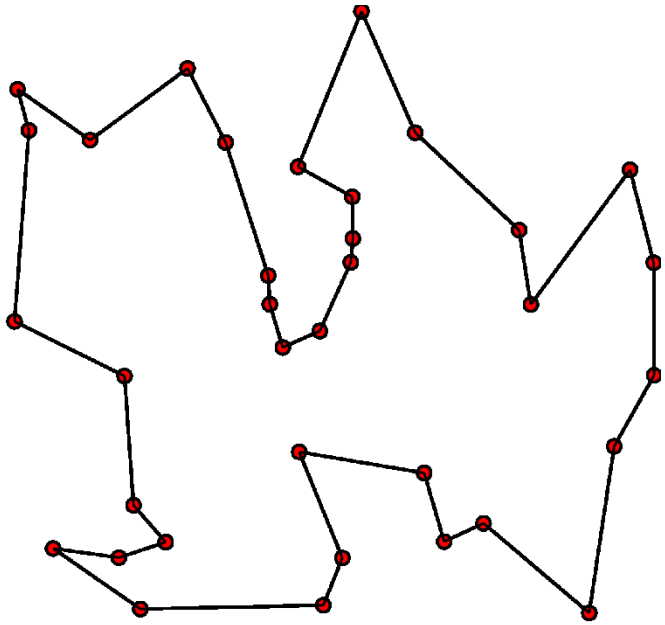
2.3.1. Matematička formulacija CVRP-a

Potpuni neusmjereni graf G može se zapisati kao $G = (V, E)$. $E = \{(i,j) : i,j \in V, i < j\}$ predstavlja skup bridova koji povezuju čvorove, a $V = \{0, \dots, n\}$ predstavlja skup čvorova. Korisnik je predstavljen čvorom $i \in V \setminus \{0\}$, negativnom potražnjom q_i, a skladište s čvorom indeksa 0. Svaki luk $e \in E$ ima pridruženu težinu c_e ili c_{ij}. Problem se može predstaviti tako da se za flotu veličine m vozila s jednakim kapacitetima Q treba odrediti m ruta minimizirajući ukupne troškove putovanja uz uvjete da svaka ruta počinje i završava u skladištu, svaki korisnik bude u točno jednoj ruti i da ukupna potražnja svih korisnika u ruti ne prelazi kapacitet vozila. Problem se može prikazati kao problem trgovačkog putnika u slučaju da je $m = 1$, što znači da postoji samo jedno vozilo za posjetiti sve korisnike. Minimalan broj vozila potreban za rješavanje CVRP problema može se odrediti izrazom (1).

$$m = \left[\frac{\sum_{i=1}^{n} q_i}{Q} \right] \tag{1}$$

2.4. Problem trgovačkog putnika

VRP generalno dobro opisuje poznati problem trgovačkog putnika (eng. „Traveling Salesman Problem“, TSP). Problem trgovačkog putnika logistički je problem iz svakodnevnog života i rješava se u teoriji grafova. Bavi se slučajem u kojem se traži šetnja u usmjerenom ili neusmjerenom grafu tako da se prođe svaki vrh u grafu barem jednom i da se vrati u početni vrh na najkraći mogući način. Može se zamisliti da su pri tome zadane i udaljenosti između vrhova, pa se traži i da je ukupna prijeđena udaljenost najmanja, [4].



Slika 6: Problem trgovačkog putnika, [4]

Slika 6 prikazuje problem trgovačkog putnika. Crvene točke na slici prikazuju gradove tj. korisnike koje trgovac mora obići. Svakog korisnika mora obići samo jednom i vratiti se u početnog korisnika koristeći najkraći mogući put,[4].

2.5. Vremenski i brzinski segment

U ovaj rad uz već navedene probleme implementirano je vremenski ovisno putovanje mrežom. Vremenski horizont dostave (najčešće jedan radni dan) diskretiziran je u različite vremenske intervale. Svaki interval ima dodijeljenu svoju brzinu. To znači da ako se do određenog korisnika dođe u određenom vremenskom intervalu, vozilo će do korisnika voziti brzinom zadanom u tom vremenskom intervalu, čime se direktno utječe na vrijeme putovanja do korisnika. Promjenama vremenskih intervala i povećanjem brzine vozila utječe se na varijable vremena obavljanja posla i ukupnog prijeđenog puta, [9]. Primjer: za vremenski interval t_1 vrijedi brzina v_1 i to vrijedi za prva četiri korisnika, za vremenski interval t_2 vrijedi brzina v_2 i ona vrijedi za sljedećih 7 korisnika itd.

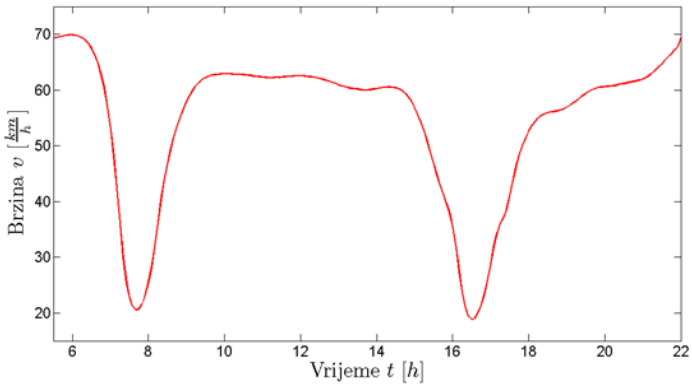
- $[0, 0.2l_0)$; $[0.2l_0, 0.4l_0)$; $[0.4l_0, 0.6l_0)$; $[0.6l_0, 0.8l_0)$; and $[0.8l_0, l_0]$.

and the corresponding travel speeds are:

- TD1a = [1.00, 1.60, 1.05, 1.60, 1.00],
- TD2a = [1.00, 2.00, 1.50, 2.00, 1.00],
- TD3a = [1.00, 2.50, 1.75, 2.50, 1.00].

Slika 7: Prikaz vremenskih intervala i povezanih brzina, [9]

Slika 7 prikazuje primjer vremenskih intervala s pripadnim brzinama vozila. Ako se do korisnika dođe u vremenskom intervalu unutar $[0\,,0.2l_0]$ vozilo će se gibati brzinom 1.00. Ako se do korisnika dođe unutar vremenskog intervala $[0.2\,,0.4l_0]$ vozilo će se gibati brzinom od 1.60. Utjecaj brzine i vremena dolaska do pojedinog korisnika na konačno rješenje zadatka biti će prikazano u sljedećim cjelinama.



Slika 8: Primjer pada brzina na Jadranskom mostu

Slika 8 prikazuje pad brzine u vršnim satima u danu na Jadranskom mostu. Brzinu koju prikazuje slika može se podijeliti u 5 dnevnih dijelova. Ujutro prije jutarnjeg vršnog sata koji se javlja malo prije 08:00 sati, brzina gibanja vozila iznosi oko 70 km/h što je otprilike brzina slobodnog toka. Oko 08:00 sati ujutro dolazi do jutarnjeg vršnog sata i brzina gibanja naglo opada na otprilike 20 km/h. Nakon prolaska jutarnjeg vršnog sata brzina se vraća na brzinu slobodnog toka i drži je do sljedećeg vršnog sata koji se pojavljaje nakon 16:00 sati popodne. Nakon njegova prolaska brzina se ponovno vraća na brzinu slobodnog toka. Upravo iz toga razloga u ovaj rad su uvedeni vremenski intervali podijeljeni na 5 intervala: prije jutarnjeg vršnog sata, za vrijeme jutarnjeg vršnog sata, između vršnih sati, za vrijeme popodnevnog vršnog sata i nakon popodnevnog vršnog sata. Unosom vremenskih intervala sa smanjenjem brzine tijekom vršnih sati i vremenskih intervala s normalnom brzinom tijekom vršnih sati program pokazuje razliku u ušteđenom vremenu i prijeđenoj udaljenosti. Vremenski intervali u ovaj rad se unose s ciljem da bi se moduliralo zagušenje na cestama. Ako tijekom gibanja vozila po ruti dođe do promjene vremenskog intervala u kojemu se nalazi korisnik kojeg obilazi vozilo, mijenja se i brzina gibanja vozila u brzinu specifičnu za taj vremenski interval. U radu se nisu koristile mjerne jedinice zato što se radi s euklidskim udaljenostima. Euklidska udaljenost je najkraća udaljenost između dvije točke u prostoru, [10].

3. Postupak rješavanja problema

U ovom poglavlju opisani su postupci učitavnja testnih problema, kreiranja početnog rješenja i lokalne pretrage. Svi primjeri potkrepljeni su s kôdom koji je napisan u C# programskom jeziku unutar programskog alata Visual Studio od strane Microsofta, [11].

3.1. Opis i učitavanje podataka i korisnika

Za testiranje razvijene aplikacije korišteni su testni problemi koje je razvio Solomon za VRP problem s vremenskim prozorima (VRP-TW), [12].

Svaki korisnik uključujući i skladište opisan je atributima. Atributi obuhvaćaju IDkorisnika, x koordinatu, y koordinatu, potražnju korisnika, vrijeme početka i kraja vremenskog prozora i vrijeme obavljanja posla kod pojedinog korisnika. Opis korisnika i njihovih atributa prikazan je slici 9.

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90

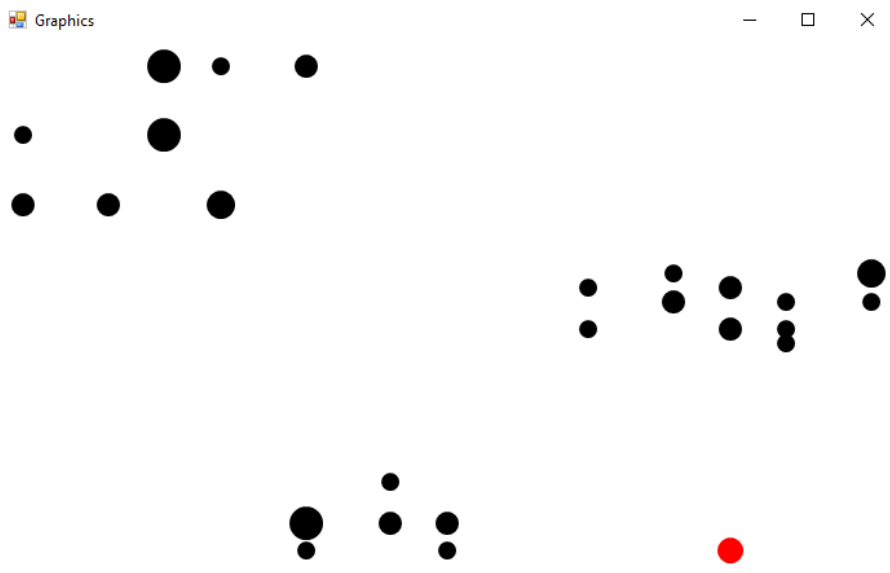
Slika 9: Prikaz datoteke s prvih pet učitanih korisnika

Svaki korisnik je spremljen unutar klase *Customer* koja se koristi kroz cijeli program. Klasa je jezična struktura koja služi za definiranje vlastitih tipova podataka. U klasu Customer sprema se svaki korisnik iz datoteke. Klasa sadrži varijable (atribute) _costNo, _xcoord, _ycoord, _demand, _readyTime, _dueDate, _serviceTime, intervalIndex i arrivalTime. Izgled klase Customer prikazan je na slici 10. U varijablu _costNo sprema se redni broj kojim je označen svaki korisnik. Varijable _xcoord i _ycoord spremaju x i y koordinatu u koordinatnom sustavu, karakterističnu za svakoga korisnika. Varijabla _Demand sprema potražnju svakoga korisnika, _readyTime i _dueDate granice početka i kraja unutar kojih korisnik mora biti posjećen i _serviceTime koji sprema vrijeme potrebno za obavljanje dostave kod pojedinog korisnika. Izgled klase Customer prikazan je slikom 10.

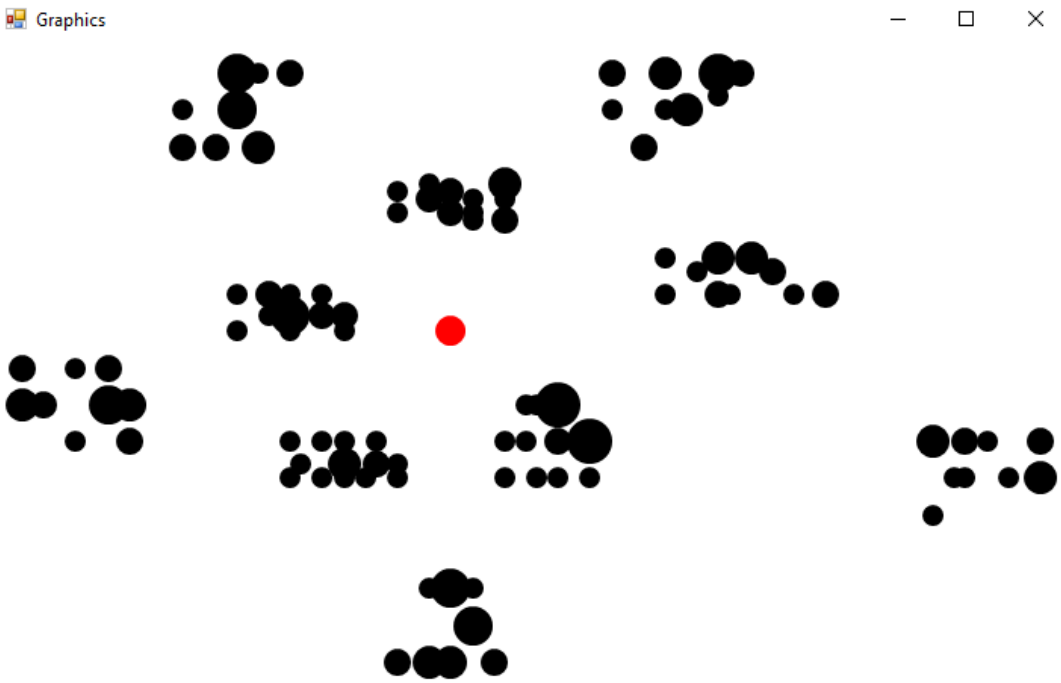
```
public class Customer
{
    public int _costNo;
    public double _xcoord;
    public double _ycoord;
    public double _demand;
    public double _readyTime;
    public double _dueDate;
    public double _serviceTime;
    public int intervalIndex;
    public double arrivalTime;

    1 reference
    public Customer(int costNo, double xcoord, double ycoord, double demand, double readyTime, double dueDate, double serviceTime)
    {
        _costNo = costNo;
        _xcoord = xcoord;
        _ycoord = ycoord;
        _demand = demand;
        _readyTime = readyTime;
        _dueDate = dueDate;
        _serviceTime = serviceTime;
    }
}
```

Slika 10: Prikaz klase Customer

Program će prikazivati rezultate za dva seta problema različitih veličina: 25 i 100 korisnika. Korištene su instance problema koje imaju klastrirane grupe korisnika što se vidi na učitanom primjeru sa 100 korisnika.



Slika 11: Primjer učitanih 25 korisnika



Slika 12: Primjer učitanih 100 korisnika

Slike 11 i 12 prikazuju izgled i položaj korisnika iz učitanih problema. Korisnici su prikazani kao crne točke dok je skladište prikazano kao crvena točka. Na slikama se vidi razlika u veličinama korisnika. Ovisno o tome koliku potražnju ima pojedini korisnik tolika je veličina njegove točke. Što je veća potražnja korisnika to je veći njegov prikaz.

3.2. Konstrukcija početnog rješenja

Nakon učitavanja podataka počinje se s izradom početnog rješenja. Program sadrži dva polja u kojima su upisani podaci o vremenskim intervalima i pripadajućim brzinama. Prvo polje sadži zapis brojeva koji predstavljaju vremenske intervale, a drugo polje brojeve koji predstavljuju brzine kretanja vozila u odgovarajućim intervalima. Brzina kretanja vozila od korisnika do korisnika ovisi o vremenu polaska od korisnika, bolje rečeno o trenutku odlaska od korisnika prije. Svaki korisnik opisan je svojom x i y koordinatom pa se pomoću matematičkog izraza 2 za izračun euklidske udaljenosti između dvije točke u kartezijevom kooridinatnom sustavu može dobiti njihova međusobna udaljenost.

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (2)$$

Dijeljenjem udaljenosti između dvoje korisnika s brzinom kojom se giba vozilo od prvoga do drugoga korisnika dobiva se vrijeme koje je bilo potrebno za prolazak tog puta.

$$t = \frac{s}{v} \tag{3}$$

Izraz za dobivanje vremena putovanja prikazana je izrazom 3, odnosno razmatra se jednoliko gibanje između dvaju korisnika.

U programu je stvorena vremenska matrica matrixTime dimenzija i, j i k. Dimenzije i i j jednake su broju korisnika u problemu, dok je dimenzija k jednaka broju diskretiziranih vremenskih intervala. Znači, za svaki par korisnika i i j sprema se pet vremena putovanja u kojima je vozilo obavilo put. Metoda koja sadrži formulu za dobivanje vremena prikazana je slikom 13, dok je punjenje matrice vrijednostima pozivom metode TimeSpeed prikazano na slici 14.

```
public static double TimeSpeed (Customer c1, Customer c2, double speed)
{
    double deltaX = c1._xcoord - c2._xcoord;
    double deltaY = c1._ycoord - c2._ycoord;

    return Math.Sqrt(deltaX * deltaX + deltaY * deltaY) / speed;
}
```

Slika 13: Prikaz metode TimeSpeed

Slika 13 prikazuje metodu TimeSpeed kojoj se predaju dva korisnika i brzina. Metoda pomoću x i y koordinate predanog korisnika i brzine računa vrijeme pomoću izraza 3.

```
for(int i  = 0; i < p.customers.Count; i++)
{
    for (int j = 0; j < p.customers.Count; j++)
    {
        for (int k = 0; k < p.maxTime.Length; k++)
        {
            p.matrixTime[i, j, k] = TimeSpeed(p.customers[i], p.customers[j], speeds[k]);
        }

    }
}
```

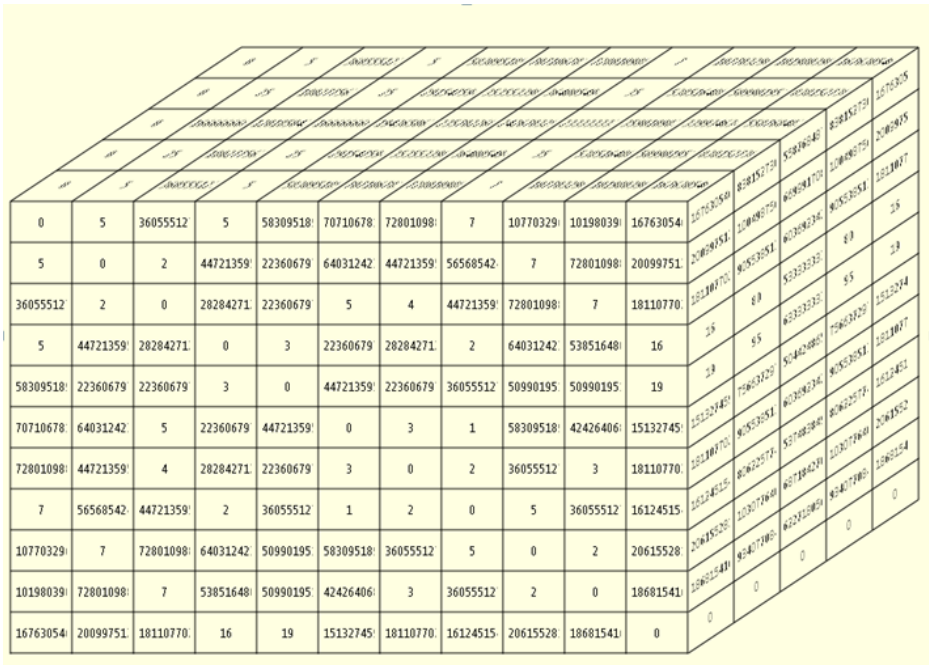
Slika 14: Prikaz punjenja matrice matrixTime

Slika 14 prikazuje punjenje matrice matrixTime s vrijednostima. Program pomoću for petlji prolazi kroz sve korisnike i sve moguće brzine između dva korisnika, odnosno sve moguće intervale.

matrixTime	{double[101, 101, 5]}
[0, 0, 0]	0
[0, 0, 1]	0
[0, 0, 2]	0
[0, 0, 3]	0
[0, 0, 4]	0
[0, 1, 0]	18.681541692269406
[0, 1, 1]	93.407708461347028
[0, 1, 2]	62.271805640898023
[0, 1, 3]	93.407708461347028
[0, 1, 4]	18.681541692269406
[0, 2, 0]	20.615528128088304
[0, 2, 1]	103.07764064044152
[0, 2, 2]	68.718427093627682
[0, 2, 3]	103.07764064044152
[0, 2, 4]	20.615528128088304
[0, 3, 0]	16.1245154965971
[0, 3, 1]	80.622577482985491
[0, 3, 2]	53.748384988656994
[0, 3, 3]	80.622577482985491
[0, 3, 4]	16.1245154965971

Slika 15: Prikaz matrice matrixTime

Slika 15 prikazuje izgled matrice *matrixTime* koja za svaka dva korisnika sadrži pet mogućih vremenskih trajanja za vrijeme kojih se dođe od jednog do drugog korisnika uz odgovarajuću brzinu.



Slika 16: 3D prikaz matrice matrixTime

Slika 16 prikazuje 3D prikaz matrice matrixTime napravljene unutar Visual Studia 2015 pomoću alata Array Visualizer. Unutar matrice x os pruža se desno, y os prema gore, a z os prema dubini.

Nakon stvaranja i popunjavanja matrice vremena svi korisnici bez skladišta kopirani su u listu neposjećenih korisnika. U klasi naziva *Vehicle* koja se brine za svako vozilo u programu napravljena je lista u koju se spremaju svi posjećeni korisnici. Znači da svako vozilo ima svoju listu posjećenih korisnika. U klasi Vehicle se osim liste posjećenih korisnika za svako vozilo nalaze i atributi kojima je opisano svako vozilo, a to su: broj vozila, maksimalni kapacitet, lokacija na kojoj se vozilo trenutno nalazi, trenutni kapacitet vozila i udaljenost koju je prešlo vozilo. Kreira se novo vozilo i u njegovu listu posjećenih korisnika dodaje se nova instanca skladišta. Svako vozilo ima jednak teretni kapacitet Q koji ne smije biti prekoračen te je polaskom iz skladišta trenutni slobodni kapacitet iznosa 0.

Postupak za kreiranje početnog rješenja prolazi kroz cijelu listu neposjećenih korisnika. Od trenutno zadnjeg dodanog korisnika u rutu vozila (u početku je to skladište) traži prvog neposjećenog korisnika do kojega ima najmanje vrijeme dolaska, a da ga može dodati u vozilo, odnosno da navedeni korisnik ne prekorači teretni kapacitet vozila. Nakon dolaska kod svakoga korisnika kapacitet vozila povećava se za njegovu potražnju.Ukoliko se uspije dodati korisnika u rutu vozila miče ga se iz liste neposjećenih korisnika i sprema se u listu posjećenih korisnika od trenutnoga vozila. Nakon što se provjere svi korisnici, te niti jedan korisnik nije bio ubačen u vozilo, vozilo se vraća u skladište. Skladište se tada ponovno dodaje na kraj liste posjećenih korisnika trenutnoga vozila. Stvara se novo vozilo i postupak se ponavlja dok nisu posjećeni svi korisnici iz liste neposjećenih korisnika. Koraci za dobivanje početnog rješenja prikazani su u tablici 1.

Algoritam za dobivanje početnog rješenja		
Ulaz	:	Podaci iz učitane datoteke, korisnici
Izlaz	:	Vozila svako sa svojom listom posjećenih korisnika
Korak 1	:	Dodavanje svih korisnika u listu neposjećenih korisnika
Korak 2	:	Dodavanje skladišta u listu posjećenih korisnika odgovarajućeg vozila
Korak 3	:	Pronalazak prvog neposjećenog korisnika sa najmanjim vremenom dolaska
Korak 4	:	Provjera za prekoračenje kapaciteta
Korak 5	:	Ako kapacitet nije prekoračen: radi se korak 5 i 6
Korak 6	:	Dodavanje korisnika u listu posjećenih korisnika vozila i micanje iz liste neposjećenih vozila
Korak 7	:	Povećanje kapaciteta vozila za potražnju korisnika
Korak 8	:	Ako je kapacitet prekoračen: radi se korak 9
Korak 9	:	Vozilo se vraća u skladište i kreće novo vozilo sa novom listom posjećenih korisnika
Korak 10	:	Dodavanje skladišta u listu posjećenih korisnika
Korak 11	:	Postupak ponavljati od koraka 3 dok se ne posjete svi korisnici

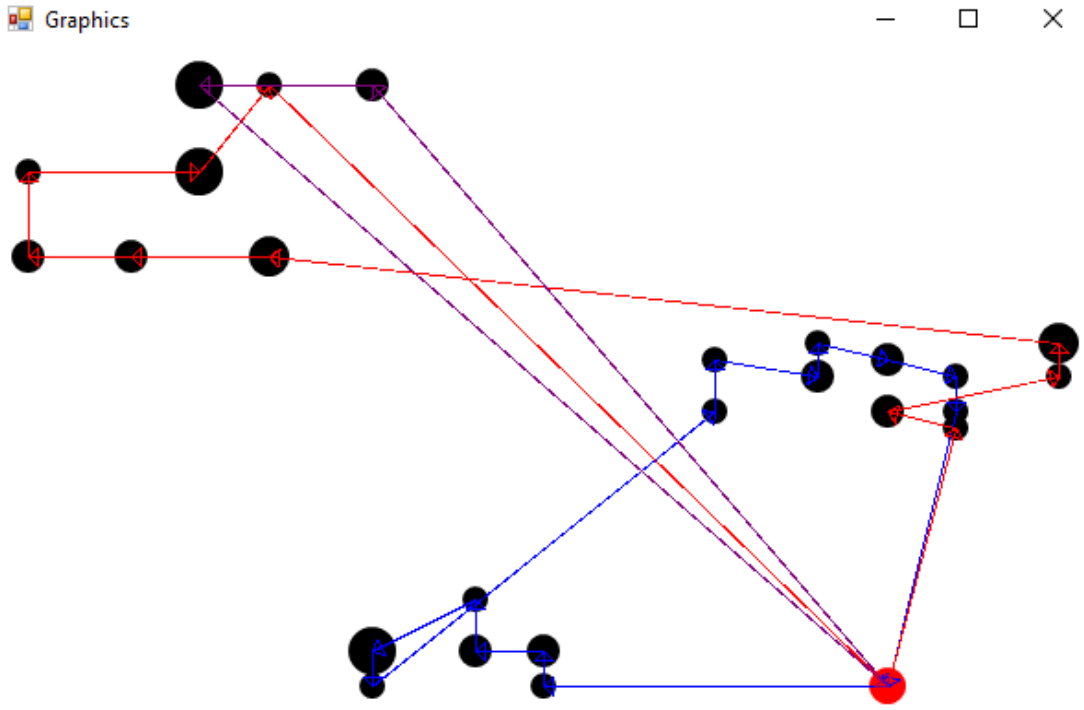
Tablica 1: Koraci algoritma za dobivanje početnog rješenja

Za početno rješenje program u konzolu ispisuje broj vozila koji je potreban da bi se obišli svi korisnici, ukupno prijeđeno vrijeme putovanja i ukupnu prijeđenu udaljenost. Program prikazuje i grafičko rješenje problema.

```
Broj vozila : 3!
Vrijeme putovanja : 617.144842841648!
Udaljenost : 266.293371776189!
```

Slika 17: Ispis konzole početnog rješenja za 25 korisnika

Slika 17 prikazuje ispis u konzoli nakon početnog rješenja za 25 korisnika. Iz slike se može zaključiti da je za obavljen problem trebalo 3 vozila, ukupno vrijeme putovanja iznosi 617.144842841648 jedinica vremena, a ukupna prijeđena udaljenost iznosi 266.29331716189.



Slika 18: Grafički prikaz početnog rješenja za 25 korisnika

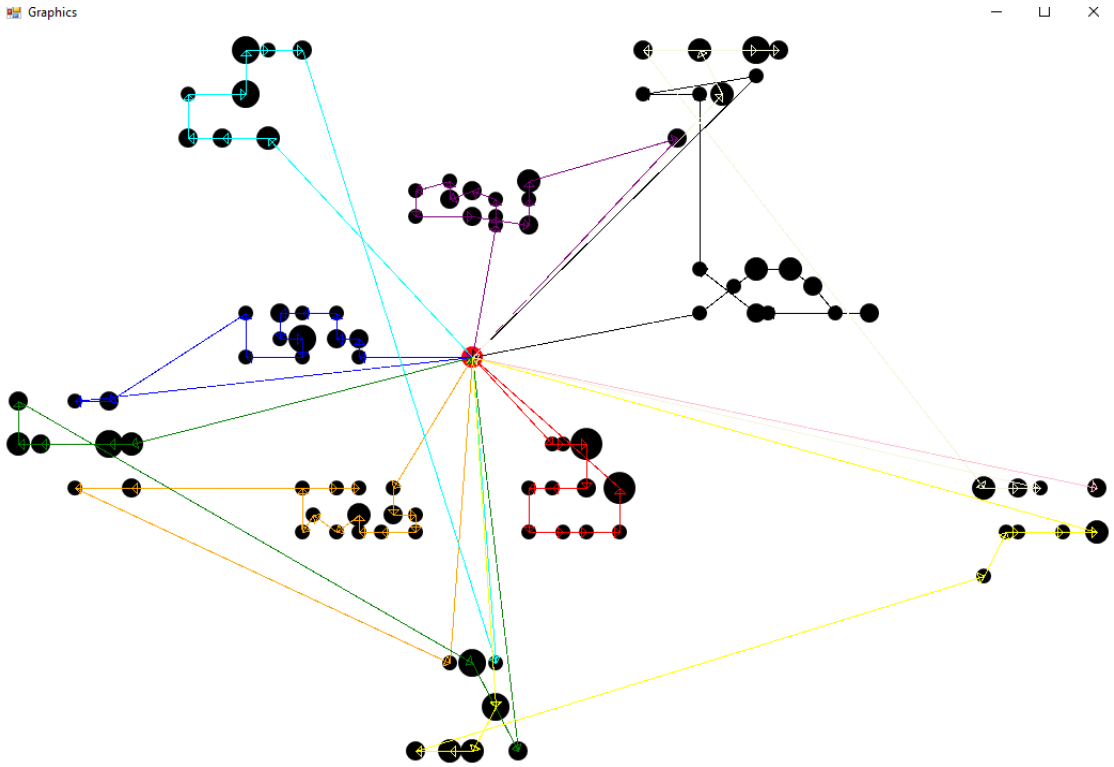
Slika 18 prikazuje grafičko početno rješenje problema s 25 korisnika. Iz slike je vidljivo da je za početno rješenje s 25 korisnika uključujući i skladište potrebno 3 vozila. Svako vozilo ima svoju rutu i svaka ruta ima svoju boju. Skladište je prikazano kao crvena točka. Korisnici su prikazani kao crne točke, te što je veća potražnja korisnika to je točka na slici veća. Strelice na linijama označavaju smjer kretanja vozila, odnosno redoslijed posluživanja korisnika.

```
Broj vozila : 10!
Vrijeme putovanja : 2633.5669763925!
Udaljenost : 1311.49987031026!
```

Slika 19: Ispis konzole nakon početnog rješenja za 100 korisnika

Slika 19 prikazuje ispis konzole za početno rješenje s 100 korisnika. Broj vozila potreban da bi se obišlo 100 korisnika iznosi 10, ukupno vrijeme putovanja iznosi 2633.5669763925 vremenskih jedinca, a ukupni prijeđeni put iznosi 1311.49987031026.

Usporedbom ispisa u konzoloma s primjerom od 25 korisnika i 100 korisnika vidi se da s povećanjem broja korisnika raste i broj vozila potrebnih za obavljanje posluživanja, te raste i ukupno vrijeme putovanja i ukupna prijeđena udaljenost.



Slika 20: Grafički prikaz početnog rješenja za 100 korisnika

Slika 20. prikazuje grafički prikaz rješenja za 100 korisnika. Na prikazu su vidljiva preklapanja, odnosno križanja u rutama vozila koja utječu na povećanje vremena putovanja i ukupnu udaljenost. Taj problem se riješava pomoću inter i intra operatora lokalne pretrage koji su objašnjeni u daljnjem tekstu.

3.3. Lokalna pretraga (Local Search)

Stvoreno početno rješenje nije optimalno rješenje problema. Pretraživanjem susjedstva početnog rješenja može se doći do boljeg rješenja koje se zove lokalno rješenje. Lokalno rješenje dobiva se računajući razlike u vremenu putovanja koje naprave vozila nakon zamjene korisnika između sebe ili ubacivanjem i izbacivanjem korisnika iz određenih ruta vozila.

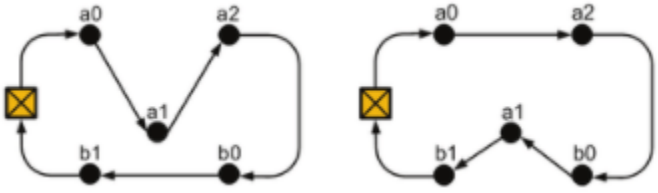
U ovome radu koriste se Intra i Inter operatori koji daju bolje rješenje. Nakon primjene svakog od operatora na trenutnom rješenju, program pokazuje uštedu u putu i vremenu koja se ostvari primjenom operatora. Svi operatori lokalne pretrage korišteni u programu biti će navedeni i objašnjeni, [7].

3.3.1 Intra operatori

Za promijene redoslijeda korisnika unutar iste rute koriste se Intra operatori. Ograničenje kapaciteta vozila kod Intra operatora nije potrebno provjeravati zbog rotacije korisnika unutar iste rute, [13].

Intra – Relocate

Intra Relocate je operator koji premiješta korisnika iz jedne pozicije u ruti na neko drugo mjesto unutar iste rute. Ušteda se dobiva na vremenu/udaljenosti i definira se kao razlika udaljenosti prije promjene i poslije promjene. Do promjene dolazi samo u jednom dijelu rute dok ostatak rute ostaje nepromijenjen, [13].



Slika 21: Prikaz Intra-Relocate operatora

Lukovi koji utječu na promjenu udaljenosti prije promjene:

$$D_1 = d_{a0,a1} + d_{a1,a2} + d_{b0,b1} \quad (4)$$

Lukovi koji utječu na promjenu nakon promjene:

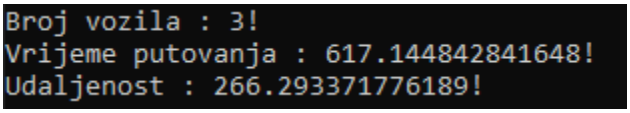
$$D_2 = d_{a0,a2} + d_{b0,a1} + d_{a1,b1} \quad (5)$$

Ušteda D definira se kao:

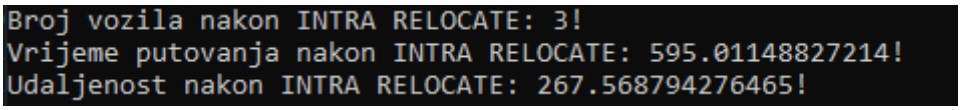
$$D = D_1 - D_2 \tag{6}$$

Do promjene u ruti dolazi ako je D pozitivan što znači da je došlo do smanjenja udaljenosti u ruti vozila.

Nakon izvršavanja operatora unutar programa prikazuje se slika problema prije i poslije izvršavanja operatora. Program u konzolu također ispisuje i udaljenosti prijeđene prije i poslije promjene. Usporedba početnog rješenja i rješenja nakon Intra Relocate prikazane su slikama 22 i 23.



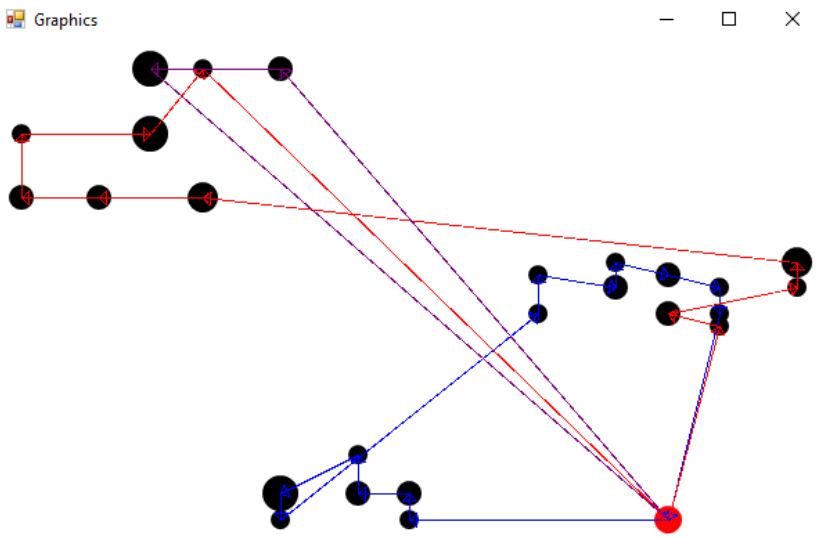
Slika 22: Prikaz početnog rješenja sa 25 korisnika



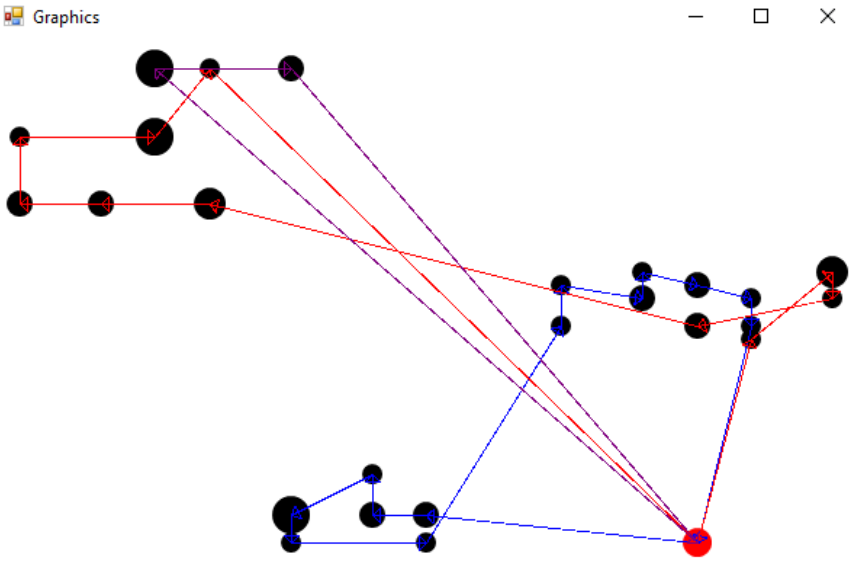
Slika 23: Prikaz konzole nakon prolaska kroz Intra-Relocate operator sa 25 korisnika

Usporedbom početnog rješenja i rješenja nakon Intra Relocate operatora primjećuje se da je vrijeme putovanja smanjeno, ali da se je udaljenost koju su vozila prošla povećala. To znači da je vozilo putovalo duljom rutom da bi prije stiglo. Ovaj primjer može se povezati s putovanjem kroz grad Zagreb tijekom vršnih sati. Ako se putuje sa istoka grada na zapad, najkraća prostorna ruta je ona koja koristi Slavonsku i Zagrebačku aveniju, dok je vremenski najkraća ruta oko Zagreba, po autocesti, ali je dulja skoro 10 kilometara. Ovo dokazuje da je u određenim slučajevima dulji put vodi do odredišta u kraćem vremenu nego kraći put.

Slikama 24 i 25 prikazano je grafičko početno rješenje i rješenje nakon Intra-Relocate operatora.



Slika 24: Grafičko početno rješenje za 25 korisnika

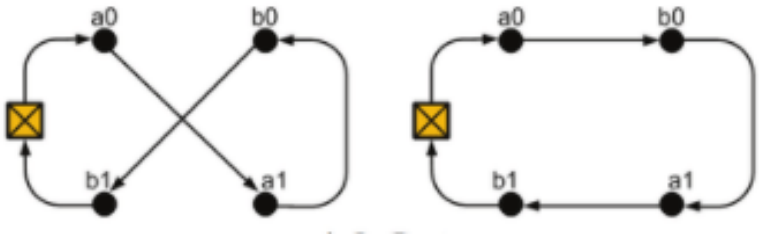


Slika 25: Grafičko rješenje problema nakon prolaska kroz Inter-Relocate operator

Usporedba slika 24 i 25 prikazuje djelovanje Intra-Relocate operatora. Zamjena korisnika unutar iste rute može se primjetiti na plavoj i crvenoj ruti vozila.

Intra 2-Opt

Intra 2-Opt operator služi za uklanjanje križanja lukova unutar rute. Navedeni primjer može se primjeniti samo u slučaju simetričnog problema, u slučaju asimetričnog problema potrebno je izračunati i sve ostale korisnike kojima se mijenja ruta, [13].



Slika 26: Prikaz Intra 2-Opt operatora, [13]

Lukovi koji utječu na udaljenost prije promjene:

$$D_1 = d_{a0,a1} + d_{b0,b1} \tag{7}$$

Lukovi koji utječu na udaljenost nakon promjene:

$$D_2 = d_{a0,b0} + d_{a1,b1} \tag{8}$$

Ušteda se definira izrazom 6.

Usporedba početnog rješenja i rješenja nakon Intra 2-Opt prikazane su slikama 27 i 28.

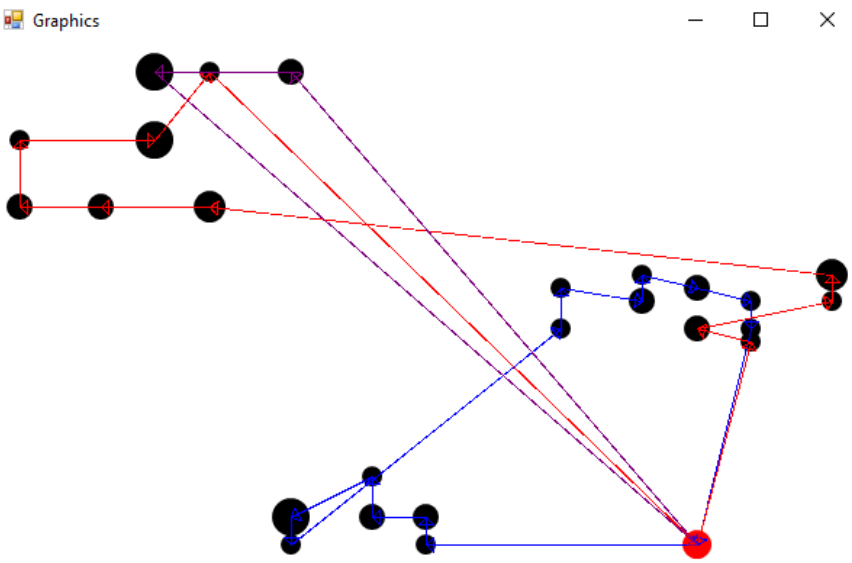
```
Broj vozila : 3!
Vrijeme putovanja : 617.144842841648!
Udaljenost : 266.293371776189!
```

Slika 27: Prikaz početnog rješenja s 25 korisnika

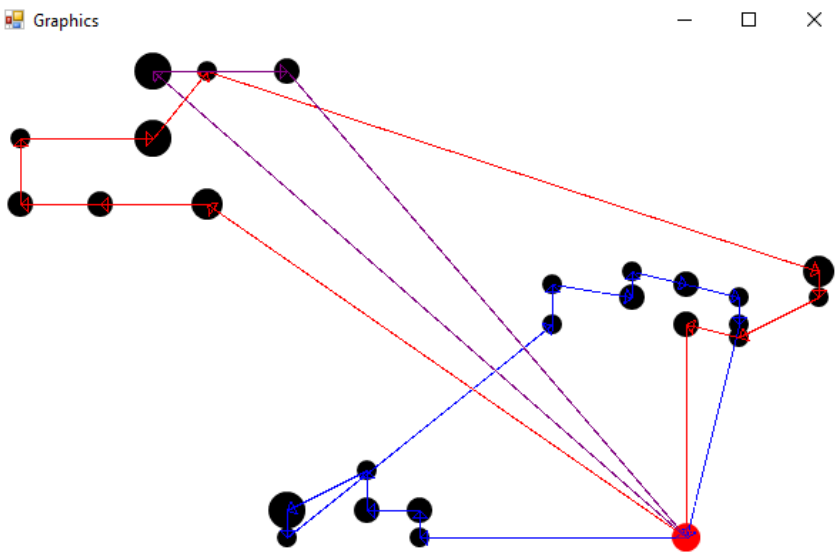
```
Broj vozila Final: 3!
Vrijeme putovanja Final: 357.640734562546!
Udaljenost Final: 259.72589990407!
```

Slika 28: Prikaz konzole nakon prolaska kroz Intra 2-Opt operator s 25 korisnika

Grafički prikaz početnog rješenja i rješenja nakon Intra 2-Opt operatora prikazan je slikama 29 i 30.



Slika 29: Grafičko početno rješenje za 25 korisnika



Slika 30: Grafičko rješenje problema nakon prolaska kroz Intra 2-Opt operator

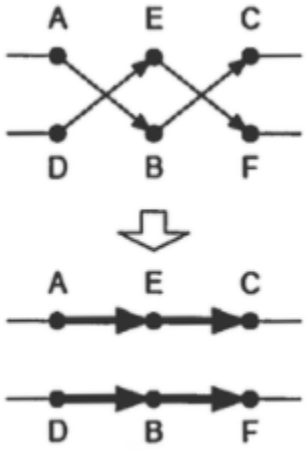
Usporedbom slika 29 i 30 vidi se djelovanje Intra 2-Opt operatora. Intra 2-Opt operator rješava problem križanja u ruti što se vidi na crvenoj ruti vozila.

3.3.2. Inter operatori

Inter operatori koriste se za premještanje korisnika između različitih ruta. Kod Inter operatora obavezno je provjeravati ograničenje kapaciteta vozila. Zamjenu korisnika moguće je napraviti samo ako to dopušta ograničenje kapaciteta, [14].

Inter-Exchange

Inter-Exchange operator izvršava zamjenu dva korisnika u različitim rutama. Zamjena se izvršava samo ako se ne prelazi iznad granice ograničenja kapaciteta vozila i ako se ostvari ušteda u vremenu, [14].



Slika 31: Prikaz Inter-Exchange operatora, [14]

Lukovi koji utječu na udaljenost prije promjene:

$$D_1 = d_{A,B} + d_{B,C} + d_{D,E} + d_{E,F} \quad (9)$$

Lukovi koji utječu na udaljenost nakon promjene:

$$D_2 = d_{A,E} + d_{E,C} + d_{D,B} + d_{B,F} \quad (10)$$

Ušteda se definira izrazom 6.

Usporedba početnog rješenja i rješenja nakon Inter-Exchange operatora prikazana je slikama 32 i 33.

```
Broj vozila : 3!
Vrijeme putovanja : 617.144842841648!
Udaljenost : 266.293371776189!
```

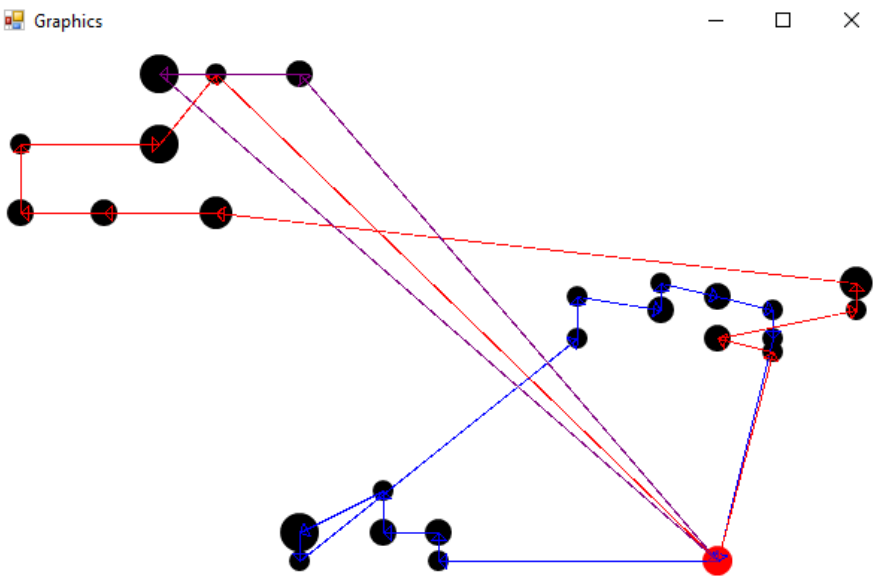
Slika 32: Prikaz početnog rješenja s 25 korisnika

```
Broj vozila Final: 3!
Vrijeme putovanja Final: 506.078288639423!
Udaljenost Final: 284.670734617532!
```

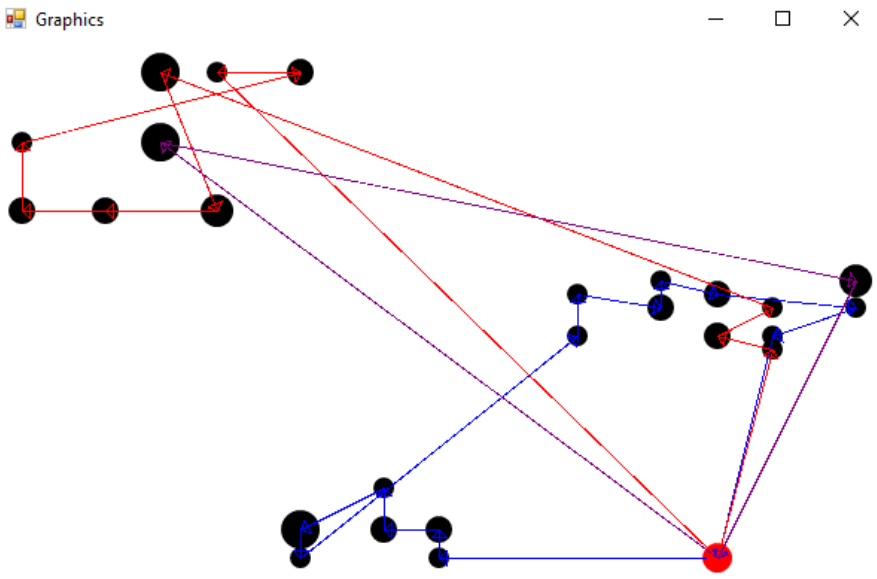
Slika 33: Prikaz konzole nakon prolaska kroz Inter-Exchange operator s 25 korisnika

Usporedbom slika 32 i 33 vidi se da Inter-Exchange operator kao i Intra-Relocate operator koristi dulji put da bi smanjio vrijeme putovanja, tj. da bi prije došao na određenu lokaciju.

Grafički prikaz početnog rješenja i rješenja nakon Inter-Exchange operatora prikazan je slikama 34 i 35.



Slika 34: Grafičko početno rješenje za 25 korisnika

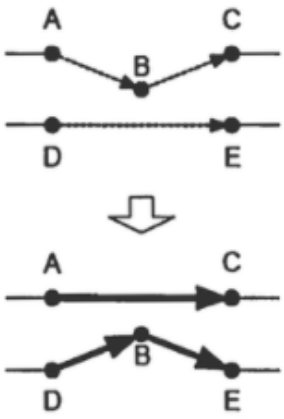


Slika 35: Grafičko rješenje problema nakon prolaska kroz Inter-Exchange operator s 25 korisnika

Usporedbom slika 34 i 35 vidi se djelovanje Inter-Excgange operatora. Operator zamjenjuje dva korisnika u različitim rutama što se dobro vidi u gornjem lijevom kutu u crvenoj i ljubičastoj ruti vozila.

Inter-Relocate

Inter-Relocate operator izvršava zamjenu korisnika iz jedne pozicije u ruti na drugu poziciju u nekoj drugoj ruti. Zamjena se izvršava samo ako se ne prelazi iznad ograničenja kapaciteta vozila i ako se ostvari ušteda u vremenu, [14].



Slika 36: Prikaz Inter-Relocate operatora, [14]

Lukovi koji utječu na udaljenost prije promjene:

$$D_1 = d_{A,B} + d_{B,C} + d_{D,E} \tag{11}$$

Lukovi koji utječu na udaljenost nakon promjene:

$$D_2 = d_{A,C} + d_{D,B} + d_{B,E} \tag{12}$$

Ušteda se definira izrazom 6.

Usporedba početnog rješenja i rješenja nakon Inter-Relocate operatora prikazane su slikama 30 i 31.

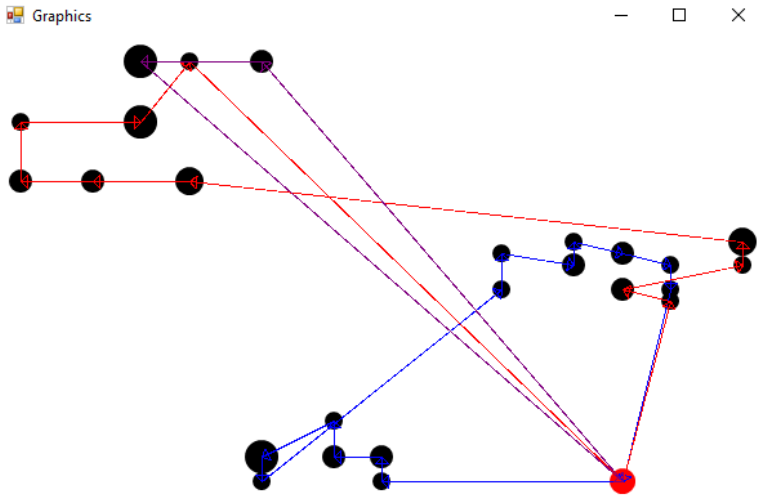
```
Broj vozila : 3!
Vrijeme putovanja : 617.144842841648!
Udaljenost : 266.293371776189!
```

Slika 37: Prikaz početnog rješenja s 25 korisnika

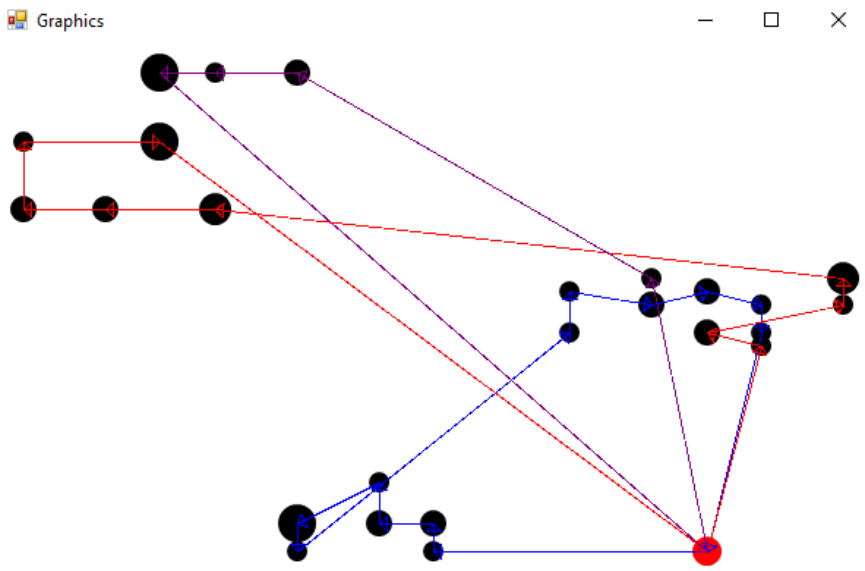
```
Broj vozila Final: 3!
Vrijeme putovanja Final: 507.724733127752!
Udaljenost Final: 257.476701369587!
```

Slika 38: Prikaz konzole nakon prolaska kroz Inter-Relocate operator s 25 korisnika

Grafički prikaz početnog rješenja i rješenja nakon Inter-Relocate operatora prikazan je na slikama 39 i 40.



Slika 39: Grafičko početno rješenje za 25 korisnika



Slika 40: Grafičko rješenje problema nakon prolaska kroz Inter-Relocate operator sa 25 korisnika

Usporedbom slika 39 i 40 vidi se zamjena korisnika između ljubičaste i plave rute vozila što je rezultat Inter-Relocate operatora

Gore prikazani primjeri prikazuju usporedbu početnog rješenja i rješenja nakon prolaska kroz samo jednog operatora lokalne pretrage. Nakon prolaska rješenja redom kroz sve operatore lokalne pretrage vrijeme putovanja i udaljenost bitno su smanjeni. Usporedba početnog rješenja i završnog lokalnog rješenja prikazane su slikama 41 i 42.

```
Broj vozila : 3!
Vrijeme putovanja : 617.144842841648!
Udaljenost : 266.293371776189!
```

Slika 41: Prikaz početnog rješenja

```
Broj vozila First local search solution: 3!
Vrijeme putovanja First local search solution: 481.369097333786!
Udaljenost First local search solution: 256.552992749714!
```

Slika 42: Prikaz konzole nakon prolaska početnog rješenja kroz sve operatore lokalne pretrage

Na slici 42 mogu se primjetiti konačne uštede u ukupnom vremenu i ukupnom prijeđenom putu. Zbog lakšeg prikaza uštede, dobivene vrijednosti s konzole zaokružene su na tri decimale. Ukupno vrijeme putovanja na početku je iznosilo 617.144, a nakon lokalne pretrage vrijeme iznosi 481.369. Oduzimanjem vremena prije i vremena poslije lokalne pretrage dobiva se ušteda u vremenu koju je napravila lokalna pretraga:

$$617.144 - 481.369 = 135.775 \quad (2)$$

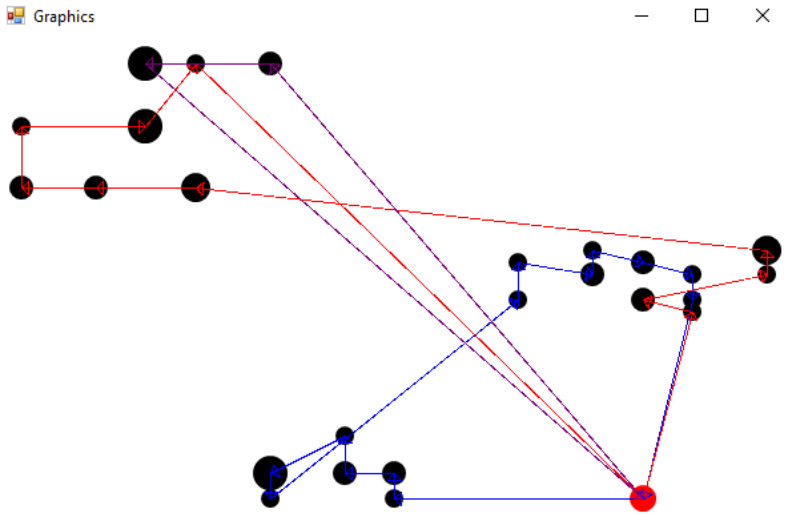
Izraz 13 prikazuju uštedu vremena danu lokalnom pretragom koja iznosi 135.775.

Isti postupak može se primijeniti i na prijeđenu udaljenost prije i nakon lokalne pretrage.

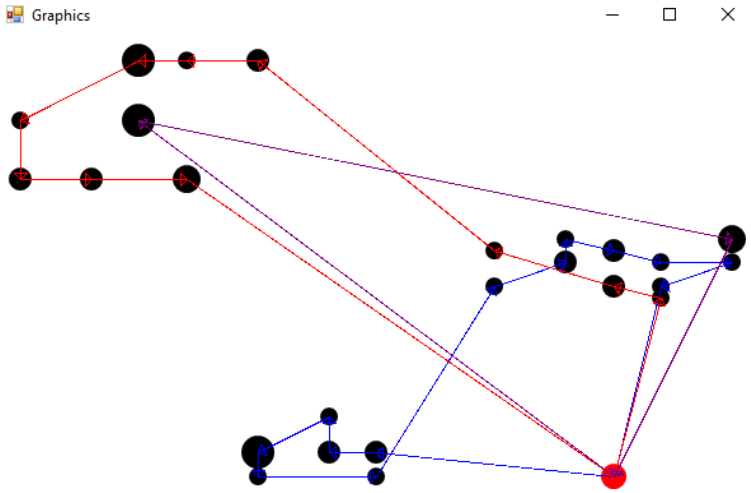
$$266.293 - 256.553 = 9.74 \quad (3)$$

Jednadžba 14 prikazuje uštedu udaljenosti danu lokalnom pretragom koja iznosi 9.74.

Slike 43 i 44 prikazuju grafičku razliku između početnog rješenja i rješenja nakon lokalne pretrage.



Slika 43: Prikaz početnog rješenja s 25 korisnika



Slika 44: Prikaz rješenja s 25 korisnika nakon lokalne pretrage

3.4. Iterativan bijeg iz lokalnog optimuma

Lokalno rješenje problema moguće je još više poboljšati slučajnim ubacivanjem i izbacivanjem korisnika. U programu svako vozilo ima spremljenu listu korisnika u ruti vozila. Iz lista se na slučajan način izabere 10% korisnika i izbacuje ih se. Slučajni izbačeni korisnici pokušavaju se ubaciti na 10 slučajnih mjesta liste nekih drugih vozila. Pri zamijeni korisnika u vozilima obavezna je provjera ograničenja kapacita vozila. U slučaju da dolazi do prekoračenja kapaciteta, zamjena korisnika u listama se neće izvršiti i program će pokušati zamijeniti sljedeće korisnike. Nakon uspješnih zamjena ponovno se provodi lokalna pretraga i pokušava se naći bolje rješenje. Postupak se pomoću while petlje ponovlja 30 puta.

Algoritam za bijeg iz lokalnog optimuma		
Ulaz	:	Rješenje nakon prve lokalne pretrage
Izlaz	:	Konačno rješenje
Korak 1	:	Iz svake liste korisnika u svakom od vozila izbacuje se sveukupno 10% korisnika slučajnim odabirom
Korak 2	:	Izbačeni korisnici pokušavaju se ubaciti na 10 slučajnih mjesta liste drugih vozila
Korak 3	:	Prije zamjene obavlja se provjera kapaciteta
Korak 4	:	Prekoračenje kapaciteta vozila –izvršavaju se koraci 5 i 6
Korak 5	:	Zamjena korisnika se ne izvršava
Korak 6	:	Program pokušava zamjeniti sljedeće korisnike
Korak 7	:	Nema prekoračenja kapaciteta – izvršava se korak 8
Korak 8	:	Zamjeni korisnike
Korak 9	:	Nakon uspješnih zamjena provodi se lokalna pretraga
Korak 10	:	Potupak se ponavlja 30 puta

Tablica 2: Koraci algoritma za bijeg iz lokalnog optimuma

```
Broj vozila First local search solution: 3!
Vrijeme putovanja First local search solution: 481.369097333786!
Udaljenost First local search solution: 256.552992749714!
```

Slika 45: Prikaz konzole nakon prolaska kroz lokalnu pretragu

```
Broj vozila Final: 3!
Vrijeme putovanja Final: 433.885339094596!
Udaljenost Final: 215.167720779082!
```

Slika 46: Prikaz konzole za konačno rješenje

Slike 45 i 46 prikazuju razliku u uštedi koju daje završno rješenje dobiveno slučajnim ubacivanjem i izbacivanjem korisnika. Zbog lakšeg prikaza uštede dobivene vrijednosti konzole zaokružene su na tri decimale. Ukupno vrijeme putovanja u rješenju prije slučajnog izbacivanja i ubacivanja korisnika iznosi 481.369, a ukupno vrijeme putovanja nakon iznosi 433.885.

$$481.369 - 433.885 = 47.484 \quad (4)$$

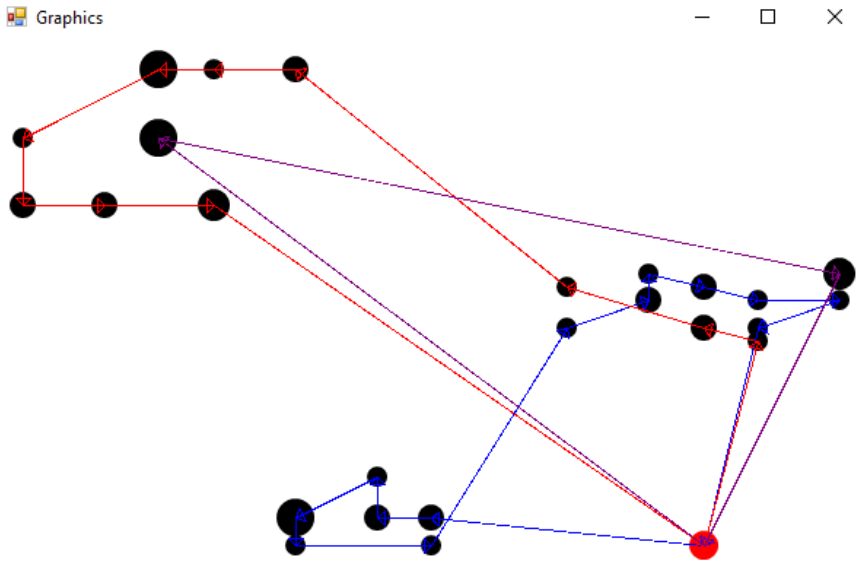
Izraz 15 prikazuje uštedu vremena nakon slučajnog ubacivanja i izbacivanja korisnika koja iznosi 47.484.

Isti postupak može se primjeniti i na prijeđenu udaljenost.

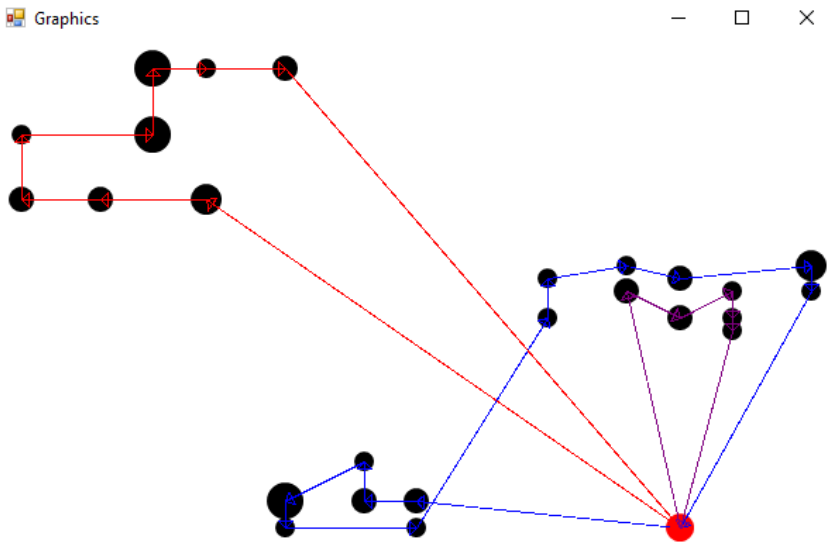
$$256.552 - 215.168 = 41.384 \quad (5)$$

Izraz 16 prikazuje uštedu prijeđene udaljenosti koja iznosi 41.384.

Zbog slučajnog biranja korisnika koji se zamjenjuju nakon svakog ponovnog pokretanja programa doći će do drugačijeg rješenja koje može biti bolje nego rješenje dobiveno nakon prve lokalne pretrage. Bitno je za napomenuti da se ovim postupkom pretražuje prostor rješenja dalje od lokalnog optimuma, kako bi se pronašlo bolje rješenje koje onda možda može biti i globalno optimalno rješenje, ali to nije zagarantirano.



Slika 47: Grafički prikaz nakon lokalne pretrage



Slika 48: završni grafički prikaz rješenja s 25 korisnika

Nakon prolaska većeg primjera problema, onoga sa 100 korisnika kroz lokalnu pretragu i kroz slučajno ubacivanje i izbacivanje korisnika dobiva se bolje rješenje nego početno. Primjer početnog rješenja, rješenja nakon lokalne pretrage i završnog rješenja za problem s 100 korisnika prikazan je slikama 49, 50 i 51.

```
Broj vozila : 10!
Vrijeme putovanja : 2633.5669763925!
Udaljenost : 1311.49987031026!
```

Slika 49: Prikaz početnog rješenja sa 100 korisnika

```
Broj vozila First local search solution: 10!
Vrijeme putovanja First local search solution: 1840.08975335878!
Udaljenost First local search solution: 1172.07216944079!
```

Slika 50: Prikaz rješenja sa 100 korisnika nakon lokalne pretrage

```
Broj vozila Final: 10!
Vrijeme putovanja Final: 1794.17060315747!
Udaljenost Final: 1120.03162629213!
```

Slika 51: Prikaz konačnog rješenja sa 100 korisnika

Usporedbom slika 49, 50 i 51 vidi se postupno smanjenje trajanja algoritma. Za početno rješenje vrijeme putovanja iznosilo je 2633.567, nakon lokalne pretrage vrijeme izvršavanja iznosi 1840.09, a nakon konačnog rješenja vrijeme izvršavanja iznosi 1794.171. Iz navedenoga mogu se odrediti uštede u vremenu koju daje prvo lokalna pretraga, a zatim slučajno izbacivanje i ubacivanje korisnika.

$$2633.567 - 1840.09 = 793.477 \quad (17)$$

Izraz 17 pokazuje uštedu u vremenu kod problema sa 100 korisnika nakon lokalne pretrage koja iznosi 793.477.

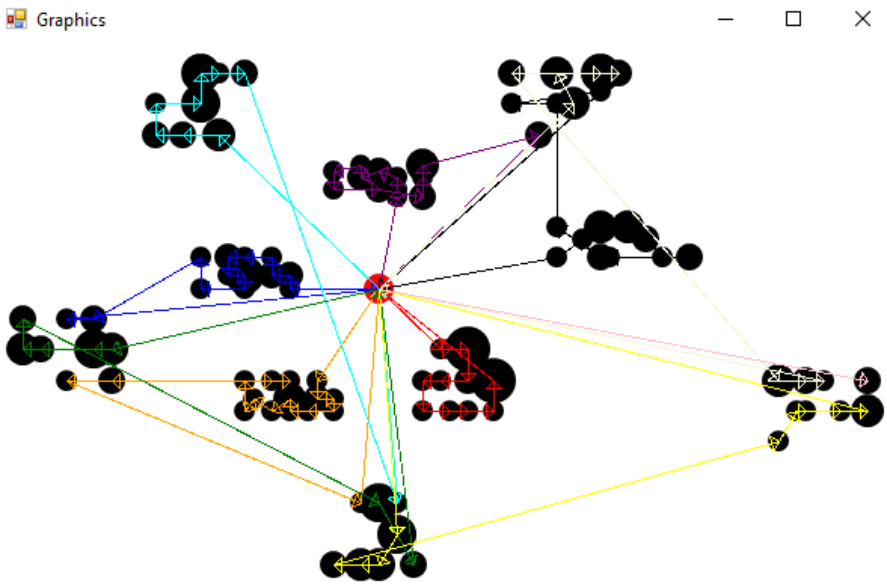
$$1840.09 - 1794.171 = 45.919 \quad (18)$$

Izraz 18 pokazuje uštedu u vremenu kod problema sa 100 korisnika nakon lokalne pretrage i zatim nakon slučajnog ubacivanja i izbacivanja korisnika koja iznosi 45.919.

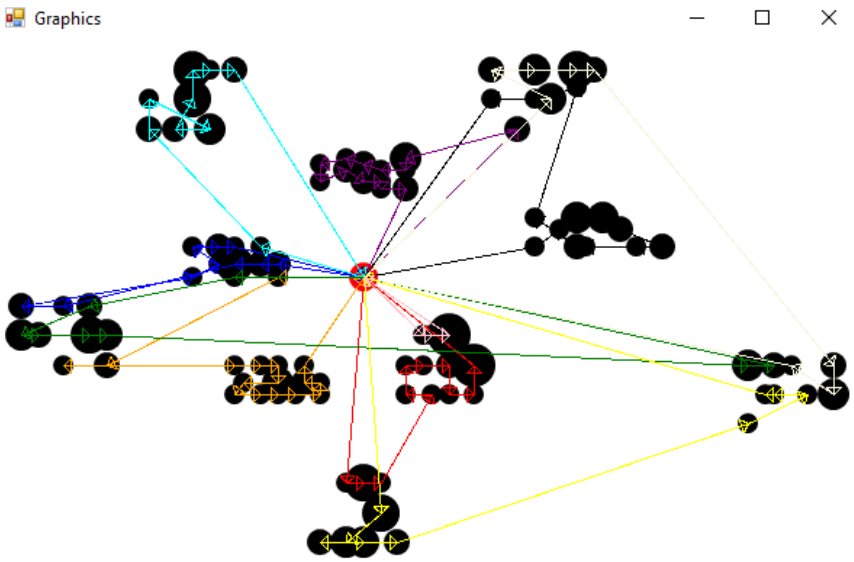
Znači da se sveukupno trajanje problema od početnog da krajnjeg rješenja smanjilo za 839.396 što je prikazano izrazom 19.

$$2633.567 - 1794.171 = 839.396 \quad (19)$$

Slike 52 i 43 prikazuju grafički prikaz rješenja sa 100 korisnika koje daje početno rješenje i koje daje završno rješenje.



Slika 52: Prikaz grafičkog početnog rješenja sa 100 korisnika



Slika 53: Prikaz završnog grafičkog rješenja sa 100 korisnika

Uspoređujući grafičko početno i završno rješenje mogu se uočiti rezultati intra i inter operatora. Primjer je intra 2 – Opt operator koji rješava problem križanja unutar jedne rute. Njegov primjer jasno se vidi na plavoj ruti vozila uspoređujući je na početnom i završnom rješenju problema.

4.Utjecaj vremenskih intervala s različitim brzinama na konačno riješenje

Prikaz utjecaja vremenskih intervala s različitim brzinama na konačno rješenje prikazan je sljedećim primjerima. U program su učitana 2 primjera A i B. Svaki ima po 5 različitih vremenskih intervala. Svakom od 5 elemenata dana je drugačija brzina kojom se giba vozilo u tome intervalu. Brzine učitane u program, u vremenskim intervalima A i B iznose: 1.00, 0.2, 0.3, 0.2, 1.00.

$$public\ double[\ ]\ speeds = new\ double[5]\{1.00, 0.2, 0.3, 0.2, 1.00\} \tag{20}$$

Izrazom 20 prikazan je kôd u Visual Studiu kojem je zadan prikaz brzina učitanih u program za vremenske intervale.

Brojevi dani programu za vremenske intervale množe se s radnim vremenom skladišta zbog toga što se sva vozila moraju vratiti u skladište prije zatvaranja. Vremenski intervali za primjer A iznose: 0.05, 0.15, 0.2, 0.4 i 100. Radno vrijeme skladišta u ovome primjeru podijeljeno je sa 4 kako bi se još iskazala razlika u odnosu na primjer B.

$$\begin{aligned} double\ SkladisteDueDate &= p.customers[0].\_dueDate\ /\ 4; \\ p.maxTime = new\ double[5\ ]\{&0.05 * SkladisteDueDate, 0.15 \\ &* SkladisteDueDate, 0.2 * SkladisteDueDate, 0.4 \\ &* SkladisteDueDate, 100 * SkladisteDueDate\}; \end{aligned} \tag{21}$$

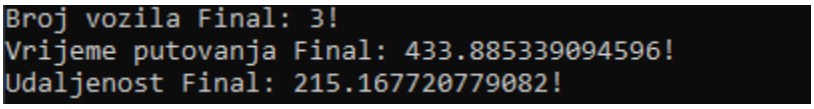
Izrazom 21 prikazan je kôd koji prikazuje vremenske intervale za primjer A. Vozilo u intervalu do 0.05 * *SkladisteDueDate* ima brzinu gibanja 1.00, za sljedeći interval vrijedi brzina 0.2, itd.

$$\begin{aligned} double\ SkladisteDueDate &= p.customers[0]._{dueDate}; \\ p.maxTime = new\ double[5]\{&1 * SkladisteDueDate, 0.2 \\ &* SkladisteDueDate, 0.5 * SkladisteDueDate, 0.1 \\ &* SkladisteDueDate, 1 * skladisteDueDate\}; \end{aligned} \tag{22}$$

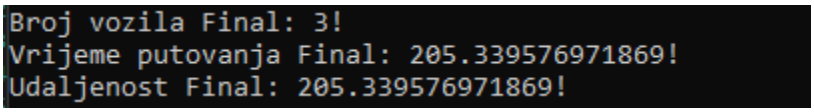
Izrazom 22 prikazan je kôd koji prikazuje vremenske intervale za primjer B. Vozilo u intervalu do 1 * SkladisteDueDate ima brzinu kretanja 1.00, za sljedeći interval vrijedi brzina 0.2 itd. Bitno je za napomenuti da će se u B primjeru sva posluživanja korisnika odviti u prvom intervalu, jer vrijeme trajanja rute nikad neće premašiti vrijeme rada skladišta.

Ovisno u kojem vremenskom trenutku vozilo dode do korisnika, tom brzinom se kreće do sljedećeg korisnika.

Da bi se pokazala važnost vremenskog trenutka dolaska vozila do korisnika prikazani su rezultati programa s primjerom A i s primjerom B.



Slika 54: Prikaz konačnog rješenja s vremenskim intervalom A



Slika 55: Prikaz konačnog rješenja s vremenskim intervalom B

Slika 52 i 55 prikazuju razliku dobivenu u vremenu i prijeđenom putu za različite vremenske intervale.

$$433.885 - 205.34 = 228.545 \tag{23}$$

Izrazom 23 prikazana je razlika u vremenu putovanja između korisnika zbog promjene vremenskih intervala koja iznosi 228.545.

$$215.168 - 205.34 = 9.828 \tag{24}$$

Izrazom 24 prikazana je razlika u prijeđenoj udaljenosti između korisnika zbog promjene vremenskih intervala koja iznosi 9.828.

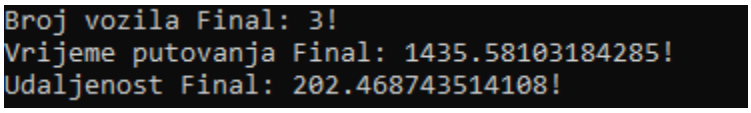
Sljedeći primjer na kojemu je prikazan utjecaj vremenskih intervala s različitim brzima prikazuje promjenu dobivenog rješenja u tri slučaja. U svakom slučaju koristiti će vremenske intervale dane primjerom A, samo će u svakome od slučaja biti promijenjenje unesene brzine. Na taj način prikazana su tri slučaja:

1. Primjer sa konstantnom gužvom na prometnici
2. Primjer sa vršnim satima
3. Primjer kada nema gužve na prometnici

Za primjer 1 sa konstantnom gužvom na prometnici učitane brzine su jako male zbog pretpostavke da se sva vozila u svim vremenskim intervalima kreću sporo zbog gužva. Brzine učitane za ovoj primjer prikazane su izrazom 25.

$$public\ double\ [\quad]speeds = new\ double\ [5]\{0.1, 0.2, 0.2, 0.2, 0.1\}; \tag{25}$$

Izrazom 25 vidi se da su za prvi primjer učitane vrlo male brzine. Završni rezultat programa za ovaj primjer prikazan je slikom 56.

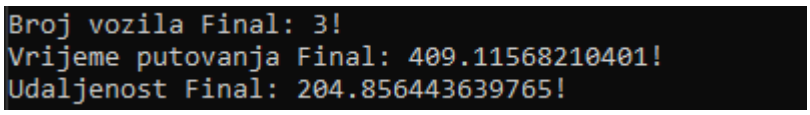


Slika 56: Završno rješenje programa sa konstantnom gužvom na prometnici

Za primjer 2 uvode se brzine unutar vršnih sati. Vršni sati javljaju se oko 08:00 sati ujutro i oko 16:00 sati popodne. U program se učitava pet brzina, stoga će druga i četvrta brzina biti jako male i predstavljati će brzine tijekom vršnih sati dok će ostale tri brzine biti jednake gibanju u razdoblju bez gužve na prometnici. Brzine učitane za ovaj primjer prikazane su izrazom 26.

$$public\ double\ [\quad]speeds = new\ double\ [5]\{0.9, 0.1, 0.7, 0.1, 0.9\}; \tag{26}$$

Izrazom 26 vidi se da su tijekom dva vršna sata na prometnici brzine gibanja vozila jako male, dok su u ostalim vremenskim intervalima brzine gibanja jednake slobodnom toku na prometnici. Završni rezultat programa za ovaj primjer prikazan je slikom 57.

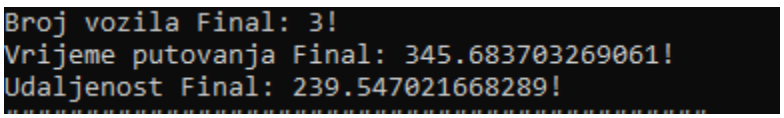


Slika 57: Završno rješenje programa sa vršnim satima na prometnici

Za primjer 3 uvode se brzine kada nema gužva na prometnicama i pretpostavlja se da se vozila kreću tijekom svih vremenskih intervala približno brzinom slobodnog toka. Brzine učitane za ovaj primjer prikazane su izrazom 27.

$$public\ double\ [\quad]speeds = new\ double\ [5]\{0.8, 0.6, 0.7, 0.6, 0.8\}; \tag{27}$$

Izrazom 27 vidi se da su dane brzine tijekom svih vremenskih intervala jednake brzinama slobodnog toka na prometnicama bez zagušenja. Završni rezultat programa za ovaj primjer prikazan je slikom 58.



Slika 58: Završno rješenje programa bez zagušenja na prometnici

Iz predstavljena tri primjera i usporedbom slika 56, 57 i 58 može se zaključiti kako je vrijeme putovanja najmanje u slučaju kada uopće nema zagušenja na prometnicama. Dani primjeri jasno objašnjavaju situacije s gužvom na prometnici cijelo vrijeme, gužvu samo u vršnim satima i bez gužve na prometnici. Navedeni primjeri pokazuju koliko promjena vremenskog trenutka utječe na određene varijable. Navedeni primjeri mogu se primjeniti i u stvarnome svijetu. Vrijeme putovanja i prijeđena udaljenost neće biti isti ukoliko se krene u jutarnjem i večernjem vremenu. U stvarnome svijetu teško je odrediti točno vrijeme i udaljenost zbog brojnih nepoznatih čimbenika poput stanja na cestama, vremenskih utjecaja i radnog vremena korisnika.

5. Zaključak

Završni rad prikazuje razvoj modela za rješavanje problema vremenski ovisnog usmjeravanja vozila s ograničenim kapacitetom. U radu su prikazani i ostali srodni problemi vezani za problem usmjeravanja vozila. Prikazani primjeri ukazuju kako se u različitim vremenskim intervalima, vozila kreću različitim brzinama, što se može primjeniti u stvarnome svijetu. Vrijeme i prijeđeni put između određenog broja lokacija neće biti isti u različitim vremenskim intervalima unutar jednog dana. Poznavanjem stvarno-vremenskih čimbenika poput stanja na prometnicama, brzininama kretanja na određenoj prometnici u određenom dobu dana i udaljenosti između korisnika, koristeći primjere navedene u radu, može se zaključiti na koji način doći do uštede u vremenu, prijeđenom putu i broju korištenih vozila u opskrbi korisnika teretom i putovanju između lokacija.

Kroz poglavlja ovoga rada opisani su postupci i metode dobivanja početnog rješenja, njegovog poboljšanja i konačno dobivanja najboljeg krajnjeg rješenja. Početno rješenje poboljšano je lokalnom pretragom pomoću inter i intra operatora čiji su primjeri djelovanja objašnjeni i grafički prikazani slikama prije i poslje njihove upotrebe. Završno rješenje dobiveno je pomoću algoritma slučajnog izbacivanja i ubacivanja korisnika između ruta različih vozila. Navedena rješenja prikazana su na dvije različite količine korisnika, količine od 25 i 100 korisnika. Kroz rad prikazana su rješenja i poboljšanja za oba slučaja količine korisnika. Svako od navedenih poboljšanja vodilo je do konačnog rješenja na kojemu je uspješno pokazan utjecaj vremenskih intervala sa različitim brzinama. Utjecaj je prikazan pomoću više primjera u kojima su se mijenjale vrijednosti intervala ili pripadnih brzina. Mijenjanjem vrijednosti prikazane su promjene koje se mogu usporediti sa primjerima zagušenja ili vršnih sati na prometnicama iz stvarnoga svijeta. Izrađeni program prikazuje rješenje pomoću kojega se do odredišta dolazi u najkraćem mogućem vremenu u slučaju navedenih zagušenja.

Završni rezultat rada prikazuje uštedu u vremenu i udaljenosti ovisno o brzini koje se može primijeniti u mnogim današnjim prometnim problemima opskrbe korisnika, prijevoza robe te smještaja električnih i benzinskih pumpi. Primjenom rada u stvarnome svijetu može se rezultirati ekonomska ušteda u poslovanju što je cilj svakog poduzeća koje se bavi navedenim problemima.

6. Literatura

[1] Vehicle Routing Problem. Preuzeto sa: https://en.wikipedia.org/wiki/Vehicle_routing_problem [Pristupljeno: srpanj 2020.]

[2] Hasle, K., Modeling and solving complex vehicle routing problems, Završni rad, Technical University of Denmark, 1992.

[3] Problem trgovačkog putnika. Preuzeto sa: https://hr.wikipedia.org/wiki/Problem_trgova%C4%8Dkog_putnika [Pristupljeno: srpanj 2020.]

[4] Travelling salesman problem. Preuzeto sa: https://en.wikipedia.org/wiki/Travelling_salesman_problem [Pristupljeno: srpanj 2020.]

[5] Hrvatska pošta - vozni park. Preuzeto sa: https://www.posta.hr/vozni-park/6472 [Pristupljeno: kolovoz 2020.]

[6] Erdelić, T., Problem usmjeravanja električnih vozila, Kvalifikacijski doktorski ispit, Fakultet prometnih znanosti,Zagreb, 19. srpnja 2018.

[7] Erdelić, T., Carić, T., A Survey on the Electric Vehicle Routin Problem: Variants and Solution Approaches. Journal of Advanced Transportation, Volume 2019, https://doi.org/10.1155/2019/5075671, [Pristupljeno: kolovoz 2020.]

[8] Fošner, M., Kramberger, T., Teorija grafova i logistika, Hrvatski matematički elektronički časopis, Preuzeto sa: http://e.math.hr/math_e_article/br14/fosner_kramberger [Pristupljeno: kolovoz 2020.]

[9] Figliozzi, M. A., The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transportation Research Part E Logistics and Transportation Review 48(3):616–636,DOI: 10.1016/j.tre.2011.11.006, [Pristupljeno: kolovoz 2020.]

[10] Euklidska udaljenost. Preuzeto sa: https://bs.wikipedia.org/wiki/Euklidska_udaljenost [Pristupljeno: kolovoz 2020.]

[11] Microsoft Visual Studio. Preuzeto sa: https://visualstudio.microsoft.com/ [Pristupljeno: srpanj 2020.]

[12] Sintef, Solomon benchmark instances, Preuzeto sa: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/ [Pristupljeno: lipanj 2020.]

[13] Watanabe, S., Sakakibara, K., A Multiobjectivization Approach for Vehicle Routing Problems. Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007, Preuzeto sa: https://www.researchgate.net/figure/Intra-Route-operators-for-CVRPTW_fig3_221228536 [Pristupljeno: srpanj 2020]

[14] Layeb, A., Ammi, M, Chikhi, S., GRASP Algorithm Based on New Randomized Heuristic for Vehicle Routing Problem. Journal of Computing and Information Technology 21(1):35–46, Preuzeto sa: https://www.researchgate.net/figure/Inter-route-operators_fig7_250309690 [Pristupljeno: srpanj 2020]

7. Popis Slika

8. Popis tablica

Sveučilište U Zagrebu

Fakultet prometnih znanosti

Vukelićeva 4, 10000 Zagreb

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je ovaj završni rad isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog rada pod naslovom

Izrada programskog sučelja za rješavanje problema vremenski ovisnog usmjeravanja vozila, u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student/ica

U Zagrebu 08.09.2020.

(potpis)