SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2099

Rješavanje višekriterijskog problema usmjeravanja vozila evolucijskim algoritmima

Vjeko Kužina

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 2. ožujka 2020.

Predmet: **Neizrazito, evolucijsko i neuro računarstvo**
Polje: **2.09 Računarstvo**

DIPLOMSKI ZADATAK br. 2099

Pristupnik: **Vjeko Kužina (0036491223)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Rješavanje višekriterijskog problema usmjeravanja vozila evolucijskim algoritmima**

Opis zadatka:

Definirati višekriterijski problem usmjeravanja vozila s kapacitetnim i vremenskim ograničenjima. Navesti algoritme za višekriterijsku optimizaciju. Posebnu pažnju posvetiti evolucijskim algoritmima i detaljno opisati nedominantni sortirajući genetski algoritam (engl. Non-dominated Sorting Genetic Algorithm-II, NSGA-II). Ostvariti programski sustav za ispitivanje učinkovitosti algoritama za rješavanje višekriterijskog problema usmjeravanja vozila i grafički prikaz rezultata višekriterijskog optimiranja. Podesiti parametre algoritma. Ispitati ponašanje algoritma na nekoliko problema s različitim svojstvima. Uz rad priložiti izvorne tekstove programa i citirati korištenu literaturu.

Zadatak uručen pristupniku: 6. ožujka 2020.
Rok za predaju rada: 30. lipnja 2020.

Mentor:
Prof. dr. sc. Marin Golub

Predsjednik odbora za diplomski rad profila:
Doc. dr. sc. Marko Čupić

Djelovođa:
Izv. prof. dr. sc. Tomislav Hrkać

Zahvaljujem mentoru, prof. dr. sc. Marinu Golubu, na pruženim savjetima tijekom izrade ovoga rada.

SADRŽAJ

1. Uvod

Problem usmjeravanja vozila (eng. *Vehicle routing problem*, VRP) je jedan od najzastupljenijih problema u logistici i upravljanju dostave. VRP je problem planiranja ruta vozila koji uz sebe donosi određena ograničenja koja moraju biti zadovoljena te pokušava minimizirati trošak koji će time biti prouzročen. Općenito je u upravljanju prijevoza dobara potrebno preuzimati sva dobra na jednoj lokaciji, takozvanom skladištu opskrbe (eng. *Supply depot*) te ih distribuirati do potrošača na više različitih odredišta koja mogu biti geografski vrlo raspršena te zbog raspršenosti može doći do velikih troškova dostavljača.

Na osnovni problem se u stvarnosti kao prvo ograničenje nadodaje ograničenje kapaciteta vozila, kojime se za svako vozilo propisuje koliki je maksimalni kapacitet koje ono može prevoziti u nekome trenutku. Taj se problem u literaturi naziva Kapacitetni problem usmjeravanja vozila (*Capacitated vehicle routing problem*, CVRP).

Na osnovni problem se u stvarnosti nadodaje dodatno ograničenje koje dopušta preuzimanje dobara samo u određenim vremenskim intervalima koje svaki potrošač zasebno propisuje. Također se uzima u obzir i potrebno vrijeme za preuzimanje dobara. Taj se problem u literaturi naziva Problemom usmjeravanja vozila s vremenskim ograničenjima (eng. Vehicle routing problem with time windows, VRPTW).

Za sada definirani problem je jednokriterijski jer je trenutni cilj minimizirati ukupnu prijeđenu udaljenost kako bi se smanjilo ukupni trošak, a time povećao konačni profit. Takvim pristupom se nailazi na nove probleme koji se javljaju u stvarnome svijetu jer se rasporedi pojedinih vozila, a time i njihovih vozača, mogu značajno vremenski razlikovati, štoviše smanjenjem razlike duljina prijeđenih putanja vozila većinom se dolazi do duljeg prijeđenog puta, a time se i profit smanjuje. No kako će vozači eventualno saznati rasporede ostalih vozača, tako će uočiti drastične razlike u rasporedima te će to dovesti do nezadovoljstva vozača, a time i do smanjenja kvalitete rada te u konač-

nici vjerojatno i manjeg profita. Iz tog razloga se sve više vremena i pažnje posvećuje zadovoljstvu zaposlenika, a to je moguće postići dodavanjem dodatnog kriterija pri optimizaciji problema usmjeravanja vozila koji će pokušati balansirati putanje vozila tako što će minimizirati razliku udaljenosti koju pojedina vozila prelaze.

2. Postavljanje problema

Osnovni jednokriterijski problem usmjeravanja vozila jedan je od najraširenijih NP teških problema zbog svoje složenosti a uz to i ekonomske isplativosti. U njemu je potrebno odrediti koje potrošače će posjetiti i uslužiti koje vozilo te kojom putanjom će to učiniti. Također, dobra je moguće isporučiti samo u određenim vremenskim intervalima za svaku pojedinu lokaciju te istovar dobara ima određeno trajanje. Pri tome se pokušava minimizirati trošak prijevoza, bez prekoračenja maksimalnog kapaciteta vozila.

VRP definiran je kao neusmjereni potpuni graf G = (V, E). Graf G se sastoji od lokacija $V = \{0, ..., n\}$ koji označavaju skladište te potrošače kojima dobra trebaju biti isporučena i od bridova $E = (i, j), i, j \in V$, pri čemu je n broj potrošača. Problem je predstavljen sljedećim uvjetima:

Notacija:
C_{ij} - cijena prelaska puta od lokacije i do lokacije j.
d_j - potražnja potrošača j.
a_j - najranije vrijeme u koje potrošač na lokaciji j prihvaća isporuku.
b_j - najkasnije vrijeme u koje potrošač na lokaciji j prihvaća isporuku.
K - broj dostupnih vozila.
N - broj potrošača plus skladište, pri čemu skladište ima indeks 0, a potrošači indekse od 1 do N. Q_k - kapacitet vozila k
S_{kj} - vrijeme u koje vozilo k opskrbljuje potrošača na lokaciji j.
X_{ij}^k - varijabla odluke, istinita u slučaju da vozilo k putuje s lokacije i na lokaciju j, neistinita u suprotnom

Minimizirati

$$g = \sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{j=0}^{N} X_{ij}^k C_{ij} \tag{2.1}$$

$$X_{ii}^k = 0 \qquad (\forall i \in \{0, ..., N\}, \forall k \in \{1, ..., K\}) \tag{2.2}$$

$$X_{ij}^k \in \{0,1\} \qquad (\forall i,j \in \{0,...,N\}, \forall k \in \{1,...,K\}) \tag{2.3}$$

$$\sum_{k=1}^{K} \sum_{i=0, i\neq j}^{N} X_{ij}^k = 1 (\forall j \in \{1,...,N\}) \tag{2.4}$$

$$\sum_{j=1}^{N} d_j \sum_{i=0, i\neq j}^{N} X_{ij}^k \leq Q_k (\forall k \in \{1,...,K\}) \tag{2.5}$$

$$\sum_{k=1}^{K} \sum_{j=1}^{N} X_{0j}^k \leq K \tag{2.6}$$

$$\sum_{j=1}^{N} X_{0j}^k - \sum_{j=1}^{N} X_{j0}^k = 0 (\forall k \in \{1,...,K\}) \tag{2.7}$$

$$a_j < s_{kj} < b_j \qquad (\forall j \in \{1,...,N\}, \forall k \in \{1,...,K\}) \tag{2.8}$$

$$s_{ki} + C_{ij} \leq s_{kj} \qquad (\forall i,j \in \{0,...,N\}, \forall k \in \{1,...,K\}) \tag{2.9}$$

Izraz (2.1) predstavlja funkciju dobrote jednokriterijskog problema usmjeravanja vozila koju se pokušava minimizirati, pri čemu je ona manja što je cijena ukupnoga puta manja. Izraz (2.2) isključuje mogućnost putovanja s jedne lokacije na istu tu lokaciju te tako omogućuje samo posjet neke druge lokacije. Izraz (2.3) označava povezanost 2 lokacije unutar jedne putanje time što će u slučaju kada je $X_{ij}^k = 1$ vozilo putovati od lokacije i prema lokaciji j, a u suprotnom neće. Izraz (2.4) stavlja ograničenje prema kojem 2 različita vozila ne mogu posjetiti istu lokaciju te vozilo koje posjećuje neku lokaciju smije tu lokaciju posjetiti samo jednom. Izraz (2.5) postavlja uvjet koji onemogućava da suma težina zahtijevanih isporuka prekorači ukupni kapacitet vozila. Izraz (2.6) ograničava ukupan broj odlazaka iz skladišta na broj vozila, time onemogućivši višestruki odlazak pojedinog vozila iz skladišta. Izraz (2.7) postavlja potrebu početka i kraja svake putanje u skladištu.

Izraz (2.8) nalaže kako se svaka isporuka na pojedinoj lokaciji mora dogoditi unutar dozvoljenog vremena isporuke te lokacije, pri čemu se vrijeme posjeta lokacije izračunava kao vrijeme posjete prijašnje lokacije zbrojeno s vremenom istovara te zbrojeno s udaljenošću koju je potrebno prijeći kako bi se došlo od prošle do sadašnje lokacije. Izraz (2.9) ustvrđuje kako nakon što je od vozila k opskrbljeno mjesto i u vrijeme s_{ki} nije moguće opskrbiti mjesto j vozilom k u vrijeme s_{kj} ako je razlika tih vremena manja od udaljenosti C_{ij} između mjesta i i mjesta j.

U ovome radu se jednokriterijski problem pretvara u višekriterijski time što se uz minimizaciju kriterija ukupne cijene (prijeđene udaljenosti) dodaje optimizacija kriterija neuravnoteženosti putanja, kojim će se pokušati smanjiti razlika pojedinih putanja unutar rješenja prema određenom mjerilu.

Neuravnoteženost putanja se može iskazati jednim od sljedećih mjerila:

1. Broj lokacija koje putanja posjećuje.
2. Suma težina dobara koje vozilo na određenoj putanji prenosi.
3. Vrijeme potrebno za prijeći putanju.
4. Vrijeme čekanja na putanji.
5. Duljina puta prijeđena na pojedinoj putanji.

U ovome radu odabrana je duljina puta (5) kao drugi kriterij optimizacije. Time se na prije navedenu funkciju (2.1) koju optimiramo nadodaje sljedeće.

$$BL_d = max(\sum_{i=0}^{N}\sum_{j=1}^{N} X_{ij}^k C_{ij}) - min(\sum_{i=0}^{N}\sum_{j=1}^{N} X_{ij}^k C_{ij}, (\forall k \in \{1, ..., K\}) \tag{2.10}$$

$$BL_m = \frac{1}{k}\sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k C_{ij} \tag{2.11}$$

$$BF = \frac{BL_d}{BL_m} \tag{2.12}$$

Te se ukupna funkcija dobrote pretvara u višekriterijsku:

$$f((\sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k C_{ij}), (\frac{BL_d}{BL_m})) \tag{2.13}$$

Pri čemu je izraz (2.10) razlika udaljenosti najdulje i najkraće putanje, a time i sama neuravnoteženost udaljenosti. Izraz (2.11) je srednja vrijednost prijeđenih udaljenosti

pojedinih ruta. Izraz (2.12) povezuje izraze (2.10) i (2.11) kao omjer te time dobiva ukupnu mjeru stupnja balansiranosti. Izraz (2.13) je konačna funkcija koja objedinjuje oba kriterija te koju je potrebno optimizirati.

3. Višekriterijska optimizacija

U višekriterijskoj optimizaciji promatraju se problemi u kojima je potrebno optimizirati više kriterijskih funkcija istovremeno. Do takvih se problema dolazi kada je cilj donijeti optimalne odluke u prisustvu 2 ili više sukobljenih kriterija. Takva funkcija formulirana je na ovaj način $min(f_1(x), f_2(x), ..., f_k(x)), x \in X$ pri čemu je X skup ostvarljivih vektora i $k \geq 2$.

Postoji više mogućih pristupa rješavanju ovoga problema među kojima su:

1. Prioritiziranje kriterija
2. Svođenje na jednokriterijski problem
3. Pareto dominacija

3.1. Prioritizacija kriterija

Kao prva ideja za rješavanje problema više kriterija nameće se prioritizacija kriterija kojom se određuje poredak usporedbe pojedinih kriterija te čim se prema nekom kriteriju jedno rješenje pokaže boljim, odmah se cijelo rješenje smatra boljim, a ostali kriteriji se ne uzimaju u obzir.

Ovaj pristup ima očiglednu manu, a to je da neko rješenje po jednom kriteriju može biti malo lošije a u drugome mnogo bolje no i dalje će se ukupno smatrati kao lošije rješenje.

3.2. Svođenje na jednokriterijski problem

Pri pokušaju poboljšanja prijašnje metode, dolazi se do metode prema kojoj se uzima svaka kriterijska funkcija u obzir te tako napravi arbitrarna funkcija koja je u gene-

ralnom obliku prikazana u izrazu (3.1), pri čemu je g rezultantna funkcija, N broj kriterija, f_i funkcija i-tog kriterija te ω_i težinski faktor koji pridjeljujemo i-toj funkciji. Takva funkcija povezuje sve kriterije tako što će svakome kriteriju pridružiti određeni faktor koji predstavlja koliko je taj kriterij važan u konačnome rješenju te time iz više kriterija stvori samo jedan kriterij koji ih ipak sve u nekoj mjeri uzima u obzir.

$$g = \sum_{i}^{N} \omega_i f_i \tag{3.1}$$

Kao rezultat se dobiva skalar. Skalare, a time i pojedina rješenja, možemo jednostavno uspoređivati te odlučiti koja su rješenja bolja.

No kako je takve faktore jako teško odrediti te oni ne moraju nužno biti univerzalno dobri, proizlazi kako takav pristup na netrivijalnim problemima većinom ne rezultira prihvatljivim rješenjima.

3.3. Pareto dominacija

Budući da želimo sve kriterije uzeti u obzir, a ti kriteriji mogu biti kontradiktorni te ne možemo odrediti koliko nam je neki od kriterija važan u odnosu na ostale, dolazimo do metode pomoću koje možemo predstaviti više mogućih rješenja kao najbolja. Najbolja rješenja sadržavati će i rješenja koja su po nekom od kriterija najbolja, a u ostalim kriterijima loša i rješenja koja su u svim kriterijima relativno dobra. Na kraju će osobi koja želi saznati najbolje rješenje biti predstavljeno više rješenja te će ona sama moći odlučiti koje joj najviše odgovara. Kako bi se pak odredilo koja su rješenja najbolja, uvodi se pojam Pareto dominacije.

3.3.1. Dominacija

Neko rješenje $x^1 \in X$ dominira nad nekim drugim rješenjem $x^2 \in X$ ako vrijede iduća 2 uvjeta:

1. $$\forall i \in \{1, 2, ..., k\}, f_i(x^1) \leq f_i(x^2)$$

2. $$\exists i \in \{1, 2, ..., k\}, f_i(x^1) < f_i(x^2)$$

Iz tih uvjeta slijedi kako je za dominaciju jednog rješenja nad drugim potreban bolji rezultat funkcije prema barem jednom kriteriju, te bolji ili jednako dobar rezultat prema svim ostalim kriterijima. Time se osigurava da će rješenje koje po nekom kriteriju postiže najbolji rezultat uvijek biti smatrano jednim od najboljih rješenja iako su mu rezultati ostalih kriterija moguće vrlo loši.

Tako 2 rješenja mogu biti u 3 odnosa - prvo dominira nad drugim ili drugo dominira nad prvim ili se rješenja međusobno ne dominiraju, a najbolja rješenja bit će ona koja pripadaju takozvanom nedominiranom skupu.

3.3.2. Nedominirani skup

Nedominirani skup je skup rješenja nad kojima niti jedno drugo rješenje ne dominira, jer ne postoji neko drugo rješenje koje je prema svim kriterijima jednako ili bolje od njega. Skup se dobiva izvođenjem nedominiranog sortiranja prikazanog pseudokodom u nastavku.

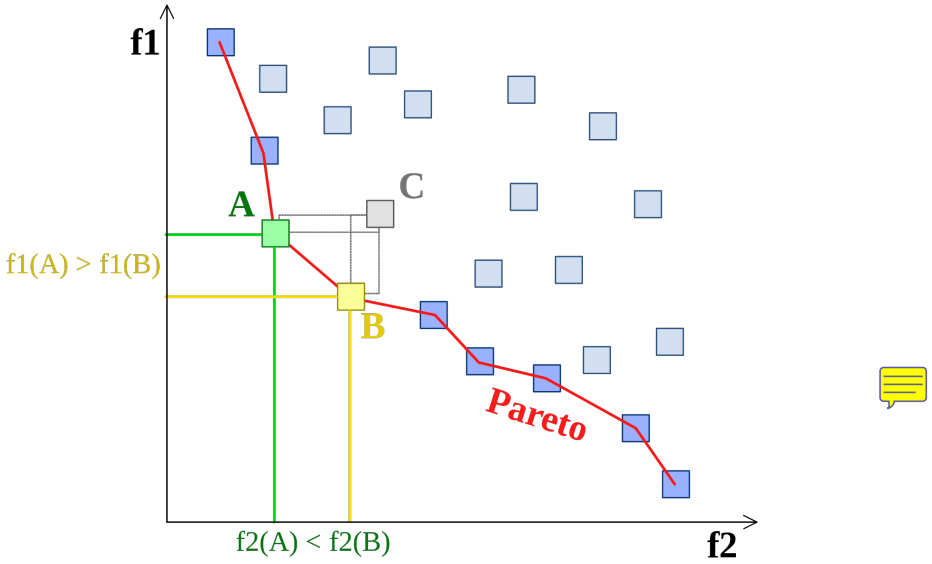
Nedominirano sortiraj populaciju P
Inicijaliziraj P' na prazan skup.
for rješenje n u skupu P
    dominiran = **false**
    for rješenje m u skupu P
        if dominira(m,n)
            dominiran = **true**
            break
        end if
    end for
    if dominiran == **false**
        dodaj x u P'
    end if
end for

Cilj Pareto optimizacije je aproksimirati nedominirani skup cijelog skupa mogućih rješenja, takozvanog globalnog Pareto-optimalnog skupa, nedominiranim skupom trenutne populacije te time postići rješenja što bliže optimalnima.

Za daljnju obradu i separaciju podataka potrebno je podijeliti skup dobivenih rješenja na Pareto fronte koristeći pri tome nedominirane skupove.

3.3.3. Pareto fronta

Pareto fronte se stvaraju ponavljanjem postupka određivanja nedominiranog skupa, postavljanjem tog skupa u trenutnu Pareto frontu te naknadnim izbacivanjem tog skupa iz populacije dok se ne ostane bez rješenja u populaciji.



Slika 3.1: Pareto fronta za dva kriterija [10]

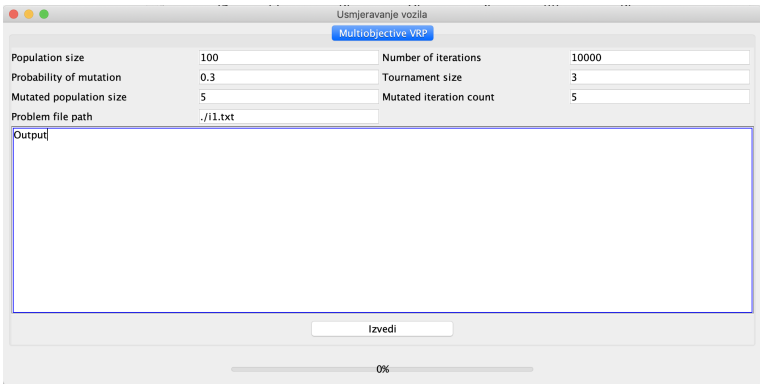
Slika (3.1) prikazuje Pareto frontu koja se sastoji od svih kvadrata povezanih crvenom linijom, dok su ostali kvadrati rješenja koji nisu u najboljoj Pareto fronti. Također se može vidjeti kako rješenje A i rješenje B ne dominiraju jedno nad drugim jer rješenje A ima manji iznos prema jednom kriteriju, a veći iznos prema drugom kriteriju nego rješenje B, dok oba rješenja dominiraju rješenje C te ono zato nije zajedno s njima u fronti.

4. Programsko ostvarenje

Algoritam izrađen u sklopu ovog završnog rada izrađen je u programskom jeziku Java.

Pri pokretanju programa otvara se ~~implementirano~~ grafičko sučelje kao što je prikazano na slici (4.1), u njemu su inicijalno postavljene vrijednosti 7 parametara potrebnih za izvođenje programa među kojima su:

1. ~~V~~Veličina populacije (eng. *Population size*)
2. Broj iteracija (eng. *Number of iterations*)
3. Vjerojatnost mutacije (eng. *Probability of mutation*)
4. Veličina turnira (eng. *Tournament size*)
5. Veličina mutirane populacije (eng. *Mutated population size*)
6. Broj iteracija mutirane populacije (eng. *Mutated iteration count*)
7. Putanja do datoteke problema (eng. *Problem file path*)

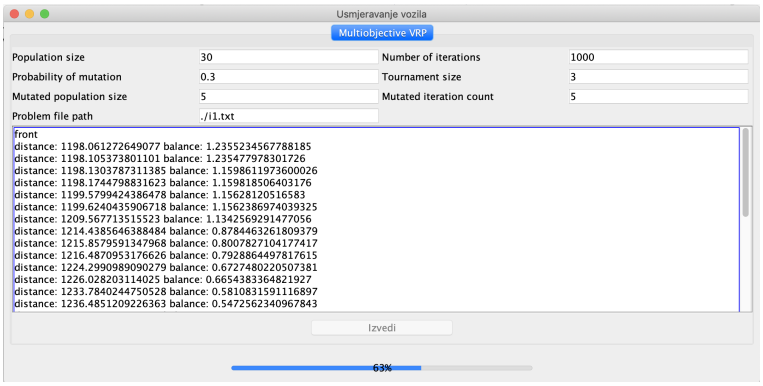


Slika 4.1: Grafičko sučelje pri pokretanju programa

Uz to u grafičkom sučelju prikazan je i prostor za rješenje te gumb, a po pritisku na taj gumb (*Izvedi*) pokreće se program koji učitava dane parametre, pokreće ~~implementiran~~ parser ulaznih datoteka te prikazuje stanje programa koje se svakih 10 iteracija

osvježava trenutno najboljom Pareto frontom rješenja, prikazujući iznos obje kriterijske funkcije za svako rješenje (4.2).

Također je na dnu grafičkog sučelja traka za učitavanje koje se puni kako se program izvodi te prikazuje postotak izvršenosti programa, računajući ga kao omjer trenutne iteracije te ukupnog broja iteracija izvođenja programa.



Slika 4.2: Grafičko sučelje tijekom izvođenja programa

4.1. Dohvat podataka

Putanja do datoteke koju je potrebno predati programu mora biti specifičnog formata. Ta datoteka u zaglavlju mora definirati:

1. Maksimalni dozvoljeni broj vozila (*Vehicle number*)
2. Kapacitet (*Capacity*)

Te nakon toga u svakome retku mora sadržavati podatke o jednoj lokaciji a ti podatci su redom:

1. Indeks (*Cust no.*)
2. Koordinata na osi X (*XCOORD.*)
3. Koordinata na osi Y (*YCOORD.*)
4. Iznos potražnje dobara (eng. *Demand*)
5. Početno vrijeme prihvaćanja dobara (*Ready time*)
6. Završno vrijeme prihvaćanja dobara (*Due date*)

7. Vrijeme istovara (*Service time*)

Primjer takve datoteke dan je na slici (4.3)

```
VEHICLE
NUMBER     CAPACITY
  25         200

CUSTOMER
CUST NO.   XCOORD.   YCOORD.   DEMAND   READY TIME   DUE DATE   SERVICE TIME

    0      35         35         0          0         230          0
    1      41         49        10          0         204         10
    2      35         17         7          0         202         10
    3      55         45        13          0         197         10
    4      55         20        19        139         169         10
    5      15         30        26          0         199         10
    6      25         30         3         89         119         10
    7      20         50         5          0         198         10
    8      10         43         9         85         115         10
    9      55         60        16         87         117         10
   10      30         60        16        114         144         10
   11      20         65        12         57          87         10
   12      50         35        19          0         205         10
   13      30         25        23        149         179         10
   14      15         10        20          0         187         10
   15      30          5         8         51          81         10
   16      10         20        19          0         190         10
   17       5         30         2        147         177         10
   18      20         40        12          0         204         10
   19      15         60        17          0         187         10
   20      45         65         9          0         188         10
   21      45         20        11          0         201         10
   22      45         10        18         87         117         10
   23      55          5        29         58          88         10
   24      65         35         3          0         190         10
   25      65         20         6        156         186         10
   26      45         30        17          0         208         10
   27      35         40        16         27          57         10
   28      41         37        16          0         213         10
   29      64         42         9          0         190         10
   30      40         60        21         61          91         10
   31      31         52        27          0         202         10
   32      35         69        23          0         186         10
   33      53         52        11         27          57         10
   34      65         55        14          0         183         10
```

Slika 4.3: Ulazna datoteka

4.2. Prikaz rješenja

Rješenje je vidljivo na slici (4.4), te prikazuje rješenje tako što u prvome retku ispisuje ukupni broj korištenih vozila n te u idućih n redaka ispisuje indeks pojedinog vozila nakon kojega slijedi ruta toga vozila.

Ruta vozila prikazana je kao niz indeksa lokacija te u zagradi navedenih vremena dolaska na lokaciju međusobno odvojenih strelicama.

Na kraju je ispisani ukupna udaljenost prijeđena od svih vozila te stupanj balansiranosti ruta unutar rješenja.

```
15
1: 0(0)->31(18)->19(46)->64(73)->49(98)->47(121)->46(141)->48(163)->52(190)->0(212)
2: 0(0)->12(15)->3(37)->76(63)->79(83)->78(99)->55(139)->24(162)->77(187)->0(217)
3: 0(0)->2(18)->2(28)->2(38)->2(48)->15(71)->22(97)->75(112)->56(127)->74(145)->73(160)->2(179)->2(189)->0(217)
4: 0(0)->96(16)->92(31)->61(52)->99(73)->53(101)->13(149)->2(169)->2(179)->58(199)->0(219)
5: 0(0)->33(27)->65(64)->34(97)->29(121)->68(140)->80(168)->28(193)->0(210)
6: 0(0)->2(18)->39(54)->23(73)->41(97)->21(121)->2(142)->2(152)->2(162)->2(172)->2(182)->2(192)->0(220)
7: 0(0)->94(13)->37(33)->14(55)->44(71)->38(92)->86(116)->91(138)->85(152)->93(169)->97(187)->0(215)
8: 0(0)->69(40)->40(75)->2(95)->2(105)->72(128)->25(157)->26(190)->0(212)
9: 0(0)->62(48)->90(70)->32(85)->51(113)->71(136)->35(153)->50(188)->0(215)
10: 0(0)->59(18)->42(41)->57(59)->87(83)->6(106)->98(127)->16(147)->100(166)->95(186)->0(211)
11: 0(0)->54(23)->67(73)->4(139)->0(174)
12: 0(0)->88(64)->84(100)->43(141)->2(170)->0(198)
13: 0(0)->36(42)->11(71)->63(90)->10(114)->60(154)->0(183)
14: 0(0)->27(27)->30(61)->81(88)->9(104)->66(129)->20(148)->70(171)->1(190)->0(216)
15: 0(0)->18(16)->45(41)->7(70)->82(86)->8(103)->5(127)->17(147)->83(169)->89(192)->0(211)
Distance: 1657.5843515365073
Balance: 0.3067272348428845
```

Slika 4.4: Prikaz rješenja

4.3. Stvaranje rješenja

Kada bi pri stvaranju ili nadopunjavanju postojećih rješenja nasumično dodavali lokacije u prvu moguću putanju, u većini slučajeva dobivali bi ili rješenja koja ne zadovoljavaju postojeće uvjete ili bi nam ponestalo vozila za stvaranje nove putanje, a u preostalim slučajevima dobili bi ispravna rješenja koja bi pak bila vrlo loša prema kriterijima koje pokušavamo minimizirati.

Stoga je u ovome radu ~~implementirana~~ Umetajuća heuristika (eng. *Insertion heuristic*, 4.5) koja pri izradi rješenja pokušava ubacivati lokacije u putanje koristeći znanja o domeni i logiku, ali pri tome zadržavajući i veliku razinu slučajnosti kako bi se što bolje istražio prostor rješenja.

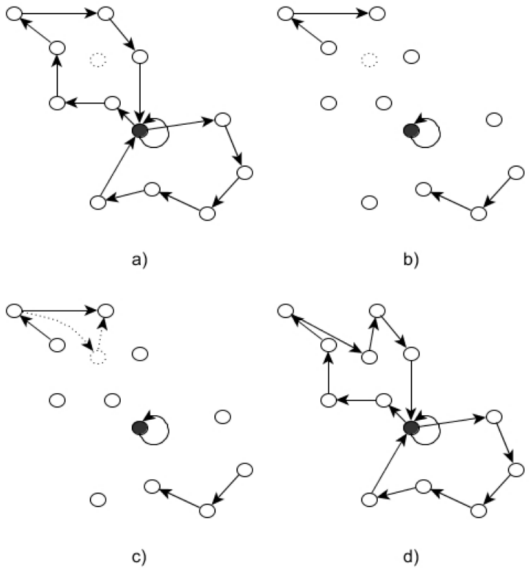
4.3.1. Umetajuća heuristika

Umetajuća heuristika pri stvaranju jedinke prvotno stvori novo rješenje koje se sastoji od samo jednog vozila koje u svojoj putanji sadrži samo lokaciju skladišta te nakon toga iterativno ponavlja postupak dodavanja lokacije.

U postupku dodavanja lokacije odabire se lokacija iz skupa neposjećenih lokacija te se tu lokaciju pokušava dodati tako da se pokušava pronaći vozilo koje zadovoljava sljedeće uvjete:

1. Kapacitet vozila neće biti prekoračen ako mu se nadoda kapacitet isporuke na odabranu lokaciju
2. Vozilo ima mogućnost doći do lokacije u vremenskom intervalu u kojemu odabrana lokacija dozvoljava posjete, uz uvjet da su vremenska ograničenja lokacija koja su već bila u putanji ostala zadovoljena
3. Dodaje najmanju udaljenost ukupnom rješenju pri umetanju lokacije u rutu.

Ako nije moguće pronaći vozilo koje zadovoljava prva 2 uvjeta, onda se dodaje novo vozilo u koje se umeće odabrana lokacija, a ako je maksimalan broj vozila već dosegnut, rješenje se odbacuje te se kreće u ponovni pokušaj stvaranja rješenja.



Slika 4.5: Umetajuća heuristika: a) Trenutne putanje; b) Bridovi u koje je moguće dodati čvor; c) Potraga za minimalnom cijenom umetanja; d) putanje s ubačenim čvorom [2]

4.4. Evolucija algoritma

Kao glavni algoritam ovoga rada implementiran je prilagođeni Genetski algoritam sortiranja bez dominacije - 2 (eng. *Non-Dominated Sorting Genetic Algorithm - 2*, NSGA-II), što je implementacija genetskog algoritma koja izravno koristi koncept dominacije na višekriterijskim problemima optimizacije.

Prvi korak algoritma je stvoriti inicijalnu populaciju jedinki nakon čega se ona evaluira te kreće proces evolucije koji traje zadani broj iteracija.

4.4.1. Stvaranje nove populacije

Kao prvi korak NSGA-II algoritma stvara se nova populacija jedinki. Nova populacija stvara se tako što se operatorom selekcije odaberu 2 roditelja, nakon čega se roditelji križaju te nastaju 2 djeteta koja se oba dodaju u novonastalu populaciju.

Osim toga, će s određenom vjerojatnošću prvo dijete mutirati s nasumično odabranim operatorom mutacije, koji će izbaciti nekoliko čvorova iz putanja toga djeteta te će se od ostataka putanja i izbačenih čvorova generirati jedna cijela populacija mutiranih jedinki, koja bi trebala biti puno manja u odnosu na glavnu populaciju, te će

izvoditi svoj vlastiti evolucijski algoritam. Također bi broj iteracija mutirane populacije idejno trebao biti puno manji nego broj iteracija glavnog algoritma na kraju kojega će iz nedominiranog skupa populacije pod-algoritma biti nasumično odabrana te dodana u novu populaciju glavnog algoritma.

Pop_1 - generiraj inicijalnu populaciju veličine PS_1 umetajućom heuristikom.
while broj iteracija < NOI_1
    $Front =$ **separate(**Pop_1**) - odvoji populaciju u Pareto fronte**
    for $i = 1..PL_1$
        x_{p11}, x_{p12} **– selektiraj roditelje iz** $Front$
        $x_{c11} = crossover(x_{p11}, x_{p12})$ **- križaj roditelje i generiraj potomka**
        $x_{c12} = crossover(x_{p12}, x_{p11})$ **- križaj roditelje i generiraj potomka**
        dodaj x_{c11} **i** x_{c12} **u** Pop_2
        if random$(0, 1) < MP$ **- primjeni mutaciju s vjerojatnošću MP**
            (x'_{m1}**,** N'_{m1}**) - procesom mutacije izbaci neke čvorove i generira novo djelomično rješenje s neizbačenim čvorovima**
            $MutPop_1$ **– generiraj populaciju veličine** PL_2 **umetanjem** N'_{m1} **u** x'_{m1}
            call($MutPop$**) - pokreni proces** $MutPop_1$
            x_{m1} **= odredi najbolje rješenje iz** $MutPop_1$
            dodaj x_{m1} **u** Pop_2
        end if
    end for
    dodaj Pop_2 ***u*** Pop_1
    $Front$ **=** ***separate(***Pop_1***)***
    $Pop1$ **=** ***groupSelect(***$Front$***)***
    clear (Pop_2***)***
end while

PROCESS $MutPop$***: while broj iteracija <*** NOI_2
    $MutFront =$ ***separate(***$MutPop_1$***)***
    for j=0...PL_2
        x_{p21}***,*** x_{p22} ***- selektiraj roditelje iz*** $MutFront$
        x_{c21} **=** ***crossover(***x_{p21}***,*** x_{p22}***)***
        x_{c22} **=** ***crossover(***x_{p22}***,*** x_{p21}***)***
        dodaj x_{c21} ***i*** x_{c22} ***u*** $MutPop_2$
        if random$(0, 1) < MP$

x_{m2} **- *mutiraj*** x_{c21}

dodaj x_{m2} ***u*** $MutPop_2$

end if

end for

dodaj $MutPop_2$ ***u*** $MutPop_1$

$MutFront$ **= *separate*(**$MutPop_1$***)***

$MutPop1$ **= *groupSelect*(**$MutFront$***)***

clear($MutPop_2$***)***

end while

Parametri:

Pop_1 - trenutna populacija

PS_1 - veličina populacije

$Front$ - Populacija odvojena u fronte

NOI_1 - Ukupan broj iteracija

x_{p1y} - Roditelj broj y

x_{c1y} - Dijete broj y

Pop_2 - nova populacija (za iduću generaciju)

MP - vjerojatnost mutacije

x'_{m1} - parcijalna jedinka

N'_{m1} - čvorovi izbačeni iz parcijalne jedinke

$MutPop_1$ - trenutna populacija sastavljena od mutiranih jedinki

PL_2 - veličina mutirane populacije

$MutFront$ - Mutirana populacija odvojena u fronte

x_{m1} - najbolje rješenje iz mutirane populacije

NOI_2 - ukupan broj iteracija mutirane populacije

x_{p2y} - Roditelj broj y u mutiranoj populacij

x_{c2y} - Dijete broj y u mutiranoj populacij

$MutPop_2$ - nova populacija mutiranih jedinki (za iduću generaciju)

x_{m2} - mutirana jedinka u procesu evolucije mutiranih jedinki

4.5. Selekcija

Pri odabiru rješenja koja će predstavljati roditelje te otići u proces križanja koristi se K-Turnirska selekcija čiju veličinu K korisnik određuje u parametrima algoritma.

K-Turnirska selekcija postupak je u kojem se iz populacije nasumično odabire K različitih rješenja te se ona međusobno usporede nakon čega se uzima najbolje od njih.

Budući da je problem u ovome radu višekriterijski, u slučaju kada 2 rješenja jedno drugo ne dominiraju nije moguće na tradicionalni način usporedbe funkcija dobrota odrediti koje je od tih rješenja bolje. Tako se pri usporedbi 2 rješenja prvo uspoređuje iz kojih su fronta rješenja, pri čemu će rješenje iz fronte manjeg indeksa biti bolje, a u slučaju iste fronte uvodi se pojam Udaljenosti grupiranja koji donosi odluku.

4.5.1. Udaljenost grupiranja

Udaljenost grupiranja je mjera rješenja kojoj je cilj pridijeliti veću važnost rješenjima koja su raspršena što znači da se nalaze što dalje od ostalih rješenja te se time pokušava dobiti što različitija rješenja, koja su i dalje najbolja unutar te fronte.

Kako bi se to postiglo, rubnim rješenjima, rješenjima s najvećim i najmanjim iznosom prema svakom od kriterija, pridaje se udaljenost grupiranja od beskonačno, a ostalima se izračunava kao suma normiranih udaljenosti do najbliža 2 rješenja, pri čemu je jedno manjeg a drugo većeg iznosa.

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m(\overrightarrow{x}^{I_{j+1}^m}) - f_m(\overrightarrow{x}^{I_{j-1}^m})}{f_m^{max} - f_m^{min}} \tag{4.1}$$

Pri čemu je m kriterijska funkcija $m \in 1, ..., M$, $\overrightarrow{x}^n$ n-to rješenje, a I^m vektor indeksa rješenja sortiranih po kriterijskoj funkciji m.

4.6. Križanje

U programskom ostvarenju operatora križanja nastoji se sačuvati zajedničke dijelove roditelja. Prvo je potrebno odlučiti što će se uopće smatrati zajedničkim unutar 2 jedinke, nakon čega je potrebno identificirati koji su dijelovi zajednički, sačuvati ih te ostatak rekonstruirati u djetetu koristeći umetajuću heuristiku.

U nastavku je prikazan pseudokod koji gore opisano izvodi:

x_i - prvi roditelj
x_j - drugi roditelj
$x_0^{'}$ - generiraj dijete
for svaku putanju $r_i \in R_i$
    dodaj (**call** (**Pronađi_Zajednički_Dio** (x_i, x_j))) u $x_0^{'}$
end for
dodaj(**Neposjećeni**(x_i)) u $x_0^{'}$

Parametri:
x_y - roditelj y
$x_0^{'}$ - dijete
r_i - putanja
R_i - lista svih putanja

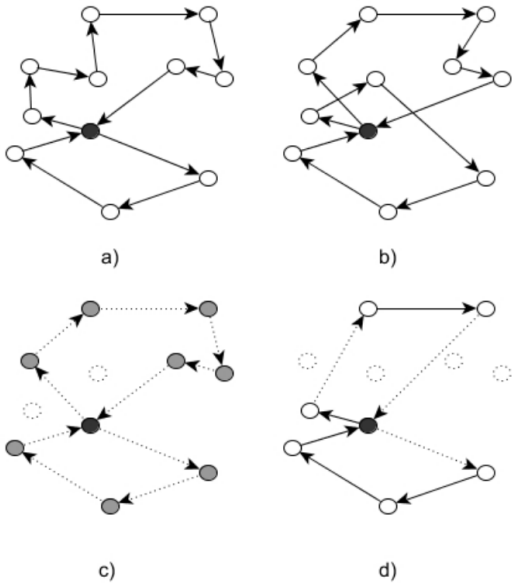
Programski su ostvarena 3 operatora križanja koji na drugačiji način određuju što se smatra zajedničkim unutar dvije jedinke:

1) Križanje zajedničkih čvorova (eng. *Common nodes crossover*, CNX)

Ideja je pronaći putanje unutar roditelja koje imaju najveći broj zajedničkih čvorova, sačuvati te zajedničke čvorove unutar jedne putanje i predati ju djetetu te ostatak te putanje zajedno sa ostalim putanjama popuniti umetajućom heuristikom.

2) Križanje zajedničkih lukova (eng. *Common arcs crossover*, CAX)

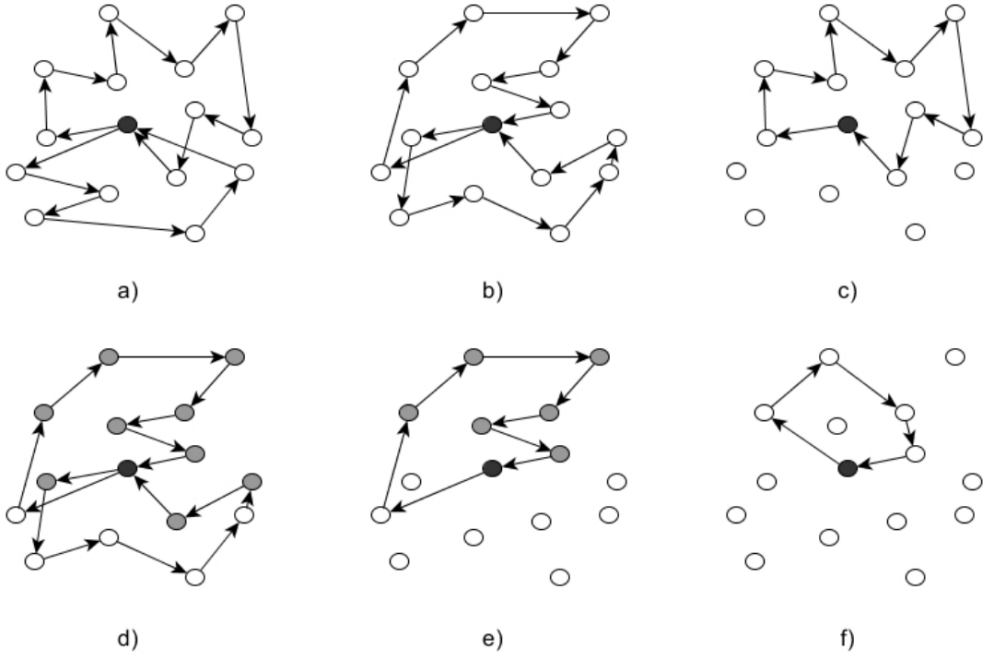
Lukovi prvog roditelja će se zadržati istim redoslijedom u djetetu ako isti takvi lukovi postoje i u drugome roditelju. Time se podrazumijeva da imaju isti početni i završni čvor te povezuju čvorove istim redoslijedom.



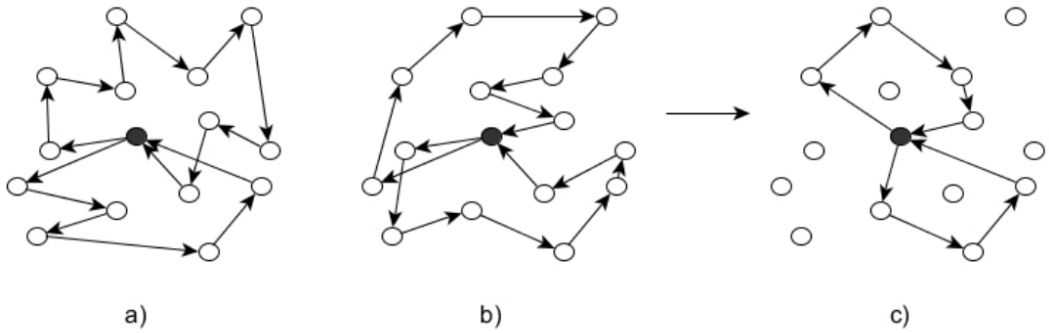
Slika 4.6: Križanja zajedničkih čvorova i lukova: a) Prvi roditelj; b) Drugi roditelj; c) Križanje zajedničkih čvorova; d) Križanje zajedničkih lukova [2]

3) Križanje zajedničkih najdužih sekvenci (eng. *Longest common sequence crossover*, LCSX)

Za svaku putanju prvog roditelja pronađe se putanja u drugom roditelju s najviše istih čvorova, te se ti zajednički čvorovi prenesu djetetu istim redoslijedom kojim su prisutni u prvom roditelju.



Slika 4.7: Pronalazak najdulje sekvence u svim putanjama: a) Prvi roditelj; b) Drugi roditelj; c) Prva određena putanja prvog roditelja za evaluaciju; d) Identificirane putanje s istim čvorovima kao u putanji u c); e) Određena putanja s najviše zajedničkih čvorova iz d); f) Najdulja zajednička sekvenca iz c) i e) je pronađena [2]



Slika 4.8: Križanje zajedničkih najdužih sekvenci: a) Prvi roditelj; b) Drugi roditelj; c) Pronađene najdulje sekvence [2]

4.7. Mutacija

U programskom ostvarenju mutacije princip je bio suprotan križanju, uz to što se izvodi na jednom rješenju, prema nekom kriteriju izdvojiti čvorove koji će se preurediti, a ostatak rješenja ostaviti istim.

Vjerojatnost mutacije u algoritmu selekcije predaje se u argumentima te će zbog načina programskog ostvarenja algoritma biti podosta veća nego u tradicionalnim evolucijskim algoritmima. Naime, inače dva odabrana roditelja križanjem stvaraju dijete koje još može i mutirati te se tek onda dodaje u populaciju, a u ovome algoritmu stvara se dvoje djece te se odmah oboje dodaju u populaciju. Nakon toga postoji vjerojatnost da prvo dijete mutira te se, u slučaju da se to dogodi i mutirana jedinka dodaje u populaciju. Zbog toga je udio mutacije puno manji nego u tradicionalnim algoritmima pa je vjerojatnost mutacije veća kako bi to nadoknadila.

U algoritmu eliminacije vjerojatnost mutacije se dinamički mijenja ovisno o tome koliko je populacija konvergirala. Tako će vjerojatnost mutacije biti veća ako populacija više konvergira, a manja ako populacija više divergira[4].

Prema kriteriju izdvajanja čvorova programski su ostvarena 3 algoritma:

1) Mutacija nasumičnih čvorova

Nasumično se biraju čvorovi koji će biti izbačeni iz jedinke te naknadno dodani umetajućom heuristikom.

2) Mutacija bliskih čvorova

Nasumično se odabere jedan čvor koji se izbaci te se za daljnje izbacivanje odabiru čvorovi najbliži prvom izbačenom čvoru.

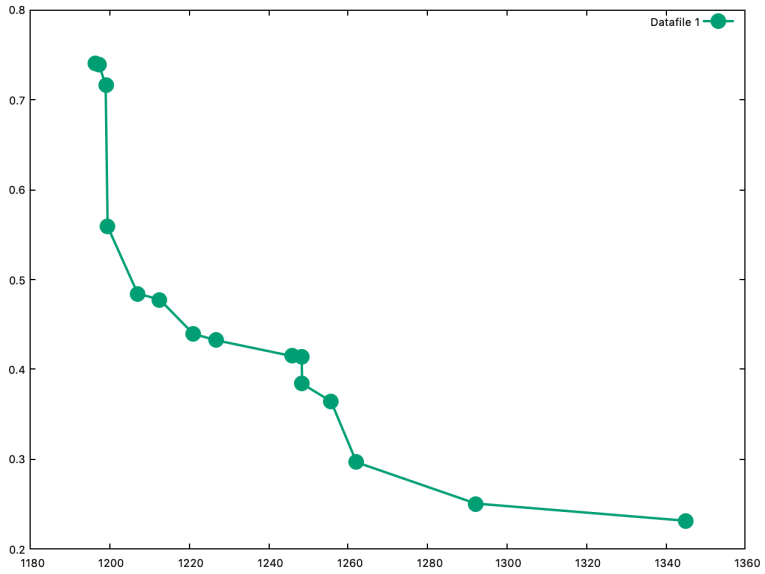
3) Mutacija najvećeg obilaska

Izabiru se čvorovi koji maksimiziraju funkciju $l_r(n_r) = l(n_{r-1}, n_r) + l(n_r, n_{r+1}) - l(n_{r-1}, n_{r+1})$ gdje l označava udaljenost. Time se izdvajaju čvorovi koji uzrokuju najveći zaobilazak u trenutnom rasporedu čvorova.

Količina izabranih čvorova ograničena je izrazom **0.5z|N|** gdje je **z** nasumično odabran broj iz intervala **[0,1]**.

4.8. Iscrtavanje rješenja

Pri završetku algoritma pokreće se gnuplot, program za iscrtavanje grafova, na kojemu se prikazuje najbolja Pareto fronta dobivena u konačnom rješenju (4.9), pri čemu svaka točka predstavlja jedno rješenje.



Slika 4.9: Najbolja Pareto fronta konačnog rješenja prikazana u programu gnuplot, na osi X nalazi se udaljenost, a na osi Y stupanj balansiranosti.

5. Eksperimentalni rezultati

Statističkim rezultatima pokušavaju se prikazati utjecaji pojedinih parametara na izvođenja samog algoritma te time pronaći najbolje parametre za algoritam.

Budući da je riječ o višekriterijskom problemu, usporedbu rješenja nije jednako lako napraviti kao usporedbu rješenja u jednokriterijskim problemima. U jednokriterijskim problemima se jednostavno mogu usporediti 2 skalara koja su dobivena u najboljim rješenjima, a u višekriterijskim problemima to nije moguće, budući da je ukupni rezultat skup rješenja, čija je veličina uz to i varijabilna.

5.1. Usporedba rješenja

U literaturi se mogu pronaći razni načini usporedbe koji kao rezultat usporedbe dobivaju jednu ili više vrijednosti kvalitete rješenja. Kako u višekriterijskoj optimizaciji još uvijek ne postoji jedna mjera koja se smatra najboljom, odlučeno je eksperimente provoditi jednom od najčešće korištenih mjera, indikatorom hipervolumena (eng. *Hypervolume indicator*, HV indicator).

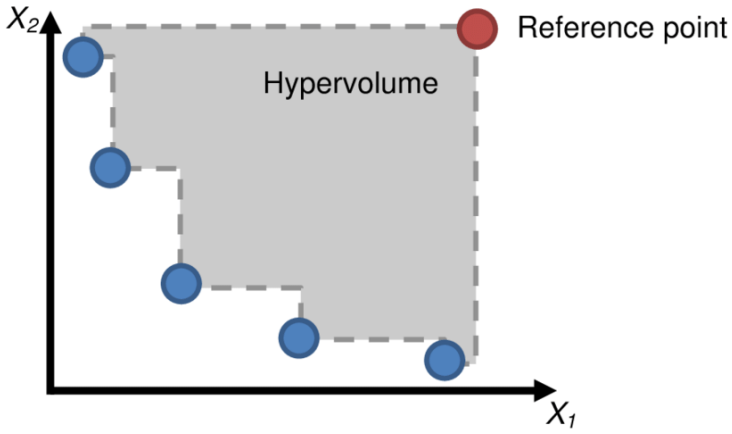
5.2. Indikator hipervolumena

Indikator hipervolumena mjera je uspješnosti koja svakoj fronti pridruži jednu vrijednost koja predstavlja hipervolumen relevantnih rješenja koje ta fronta dominira

U višedimenzijskom prostoru računa se hipervolumen, dok se u trodimenzijskom i dvodimenzijskom prostoru računaju volumen i površina. Opisat će se način izračuna u

dvodimenzijskom prostoru zbog jednostavnosti te bez smanjenja općenitosti.

Potrebno je odrediti referentu točku (*Reference point*) koja će za iznos svakog kriterija imati vrijednost veću ili jednaku najvećoj vrijednosti tog kriterija u svim rješenjima iz Pareto fronti. Nadalje se za svaku frontu izračunava površina geometrijskog tijela koje sadrži sva rješenja koja su dominirana od bilo kojeg rješenja trenutne fronte ograničeno vrijednostima referentne točke, kao što je prikazano na slici (5.1).



Slika 5.1: Hipervolumen Pareto fronte, na svakoj od osi je jedan kriterij, plave točke su rješenja jedne Pareto fronte, crvena točka je referentna točka, a sivo tijelo je hipervolumen. [9]

Nakon što se svakoj Pareto fronti pridijeli hipervolumen, moguće je fronte uspoređivati pri čemu je bolja fronta ona koja ima veću vrijednost hipervolumena jer dominira više relevantnih rješenja.

5.3. Provođenje eksperimenata

Pri provođenju eksperimenata cilj je pronaći što bolje parametre za izvođenje programa. Svi eksperimenti provode se na istome problemu s 25 dostupnih vozila te s potrebom isporuke na 100 lokacija [15]. Za svaki parametar koji se pokušava optimizirati fiksiraju se svi ostali parametri te se samo parametar trenutnog eksperimenta mijenja. Inicijalno su parametri postavljeni na sljedeće vrijednosti:

1. Veličina populacije = 60
2. Broj iteracija = 5000

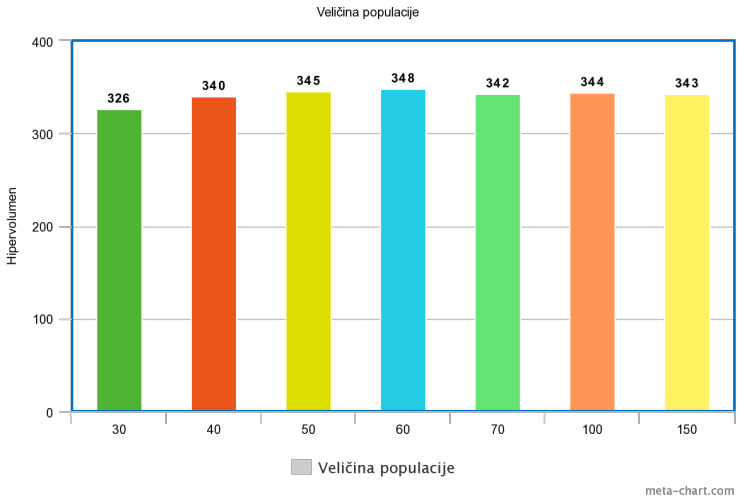
3. Vjerojatnost mutacije = 0.3
4. Veličina turnira = 3
5. Veličina mutirane populacije = 5
6. Broj iteracija mutirane populacije = 5
7. Operatori križanja = [LCSX, CAX]
8. Operatori mutacije = [Nasumična mutacija, Mutacija bliskih čvorova]

Svaki eksperiment se provodi 30 puta sa svim predočenim vrijednostima parametra koji se trenutno optimizira te se uzima prosječna vrijednost hipervolumena kao mjera za iste.

Potrebno je naglasiti kako se u svakom eksperimentu određuje nova referentna točka čije su vrijednosti maksimumi svih rješenja koja sudjeluju u eksperimentu za svaki kriteriji zasebno. Stoga će se iznosi hipervolumena potencijalno razlikovati od eksperimenta do eksperimenta te se ne bi trebali uspoređivati međusobno.

5.3.1. Veličina populacije

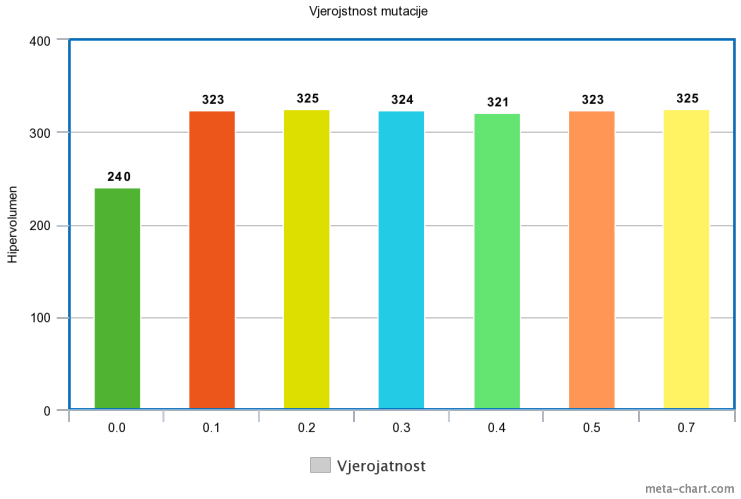
Veličina populacije utječe na količinu genetskog materijala, što je populacija veća to je više genetskog materijala prisutno, a što je populacija manja to bolja rješenja više dolaze do izražaja. Tako se za ovaj problem najbolja veličina populacije pokazala oko 60 jedinki jer uspijeva balansirati količinu genetskog materijala i selekcijski pritisak.



Slika 5.2

5.3.2. Vjerojatnost mutacije

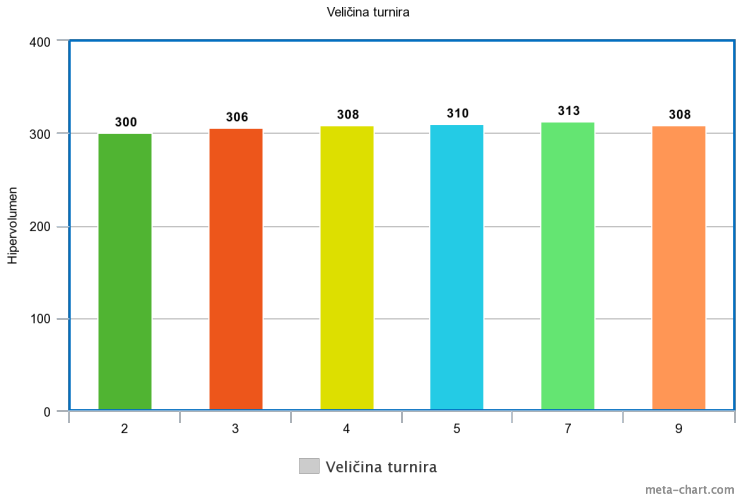
Graf (5.3) pokazuje kako je prisutnost mutacije jako važna unutar algoritma te znatno poboljšava rješenja, no ne prikazuju se velike razlike između isprobanih vjerojatnosti. Do toga vjerojatno dolazi zbog slabijeg utjecaja mutacije u algoritmu jer se uvijek obje jedinke dobivene križanjem dodaju u novu populaciju te se s vjerojatnosti mutacije dodaje još i mutirana prva jedinka dobivena križanjem.



Slika 5.3

5.3.3. Veličina turnira

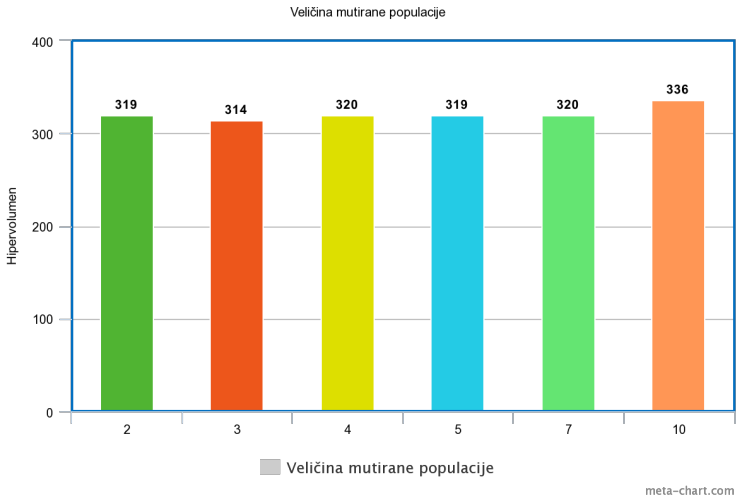
Veličina turnira utječe na selekcijski pritisak - što je veća to se veći selekcijski pritisak stvara te će češće biti odabrana bolja rješenja. Tako se na grafu (5.4) može vidjeti da su rješenja bolja kako se povećava veličina turnira do veličine od 7, a nakon toga počinju padati jer je selekcijski pritisak postao prevelik.



Slika 5.4

5.3.4. Veličina mutirane populacije

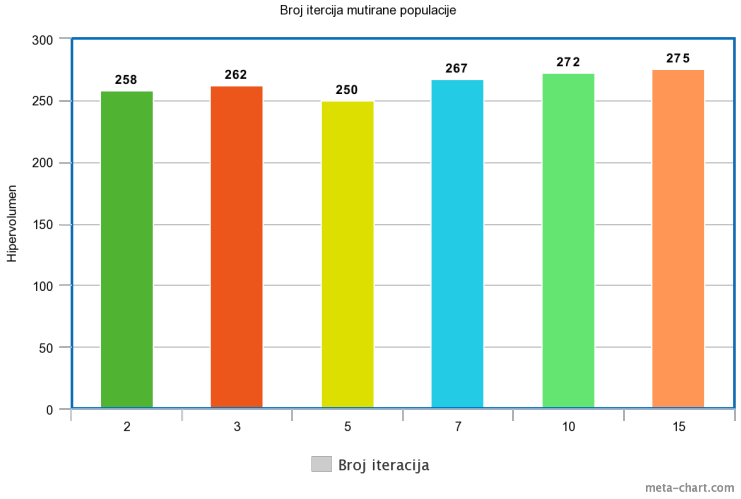
Iz grafa (5.5) može se vidjeti trend poboljšanja rješenja s povećanjem veličine mutirane populacije. No potrebno je uzeti u obzir da će povećanjem veličine mutirane populacije vrijeme izvođenja programa biti znatno produljeno.



Slika 5.5

5.3.5. Broj iteracija mutirane populacije

Iz grafa (5.6) može se također vidjeti trend rasta kvalitete rješenja s povećanjem broja iteracija mutirane populacije što je očekivano, no što je veći broj iteracija mutirane populacije to je duže izvođenje programa, a isto tako će i rješenja generalno biti bolja ako povećamo broj iteracija glavnog algoritma.

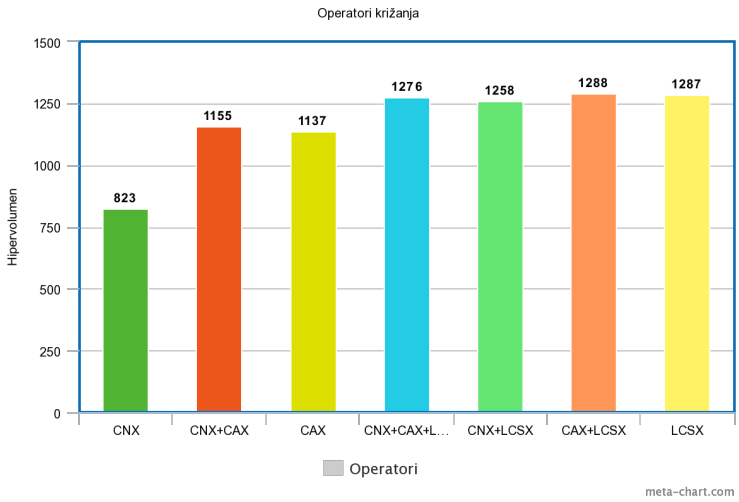


Slika 5.6

5.3.6. Operatori križanja

Iz rezultata s grafa (5.7), pokazalo se kako CAX i CNX sami po sebi ili u kombinaciji jedan s drugim ne uspijevaju doći do zadovoljavajućih rješenja, no kada se započne ko-

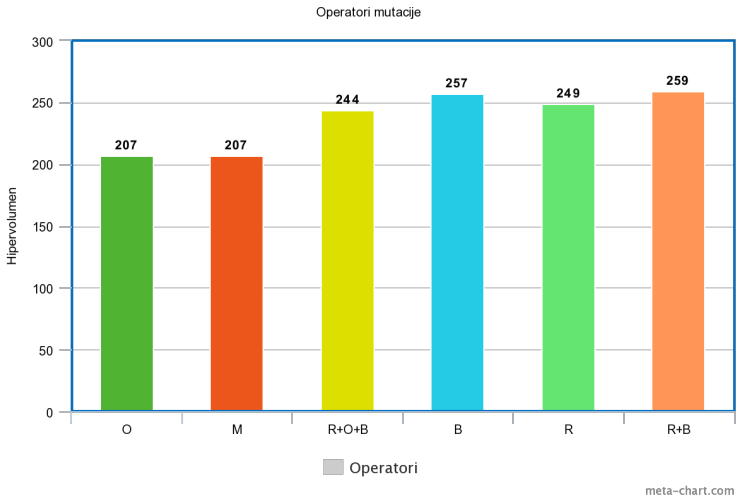
ristiti LCSX, rješenja postaju značajno bolja. Iz toga se može zaključiti da je vjerojatno najbolja taktika čuvanja najduljih sekvenci unutar roditelja te da oni nose najkvalitetnije i najpoželjnije informacije, a uz to je poželjno ponekad ubaciti i CAX budući da zajedno daju najbolja rješenja.



Slika 5.7: CNX - Križanje zajedničkih čvorova; CAX - Križanje zajedničkih lukova; LCSX - Križanje zajedničkih najduljih sekvenci

5.3.7. Operatori mutacije

Mini mutacija nasumično izbacuje samo jedan čvor te ga ponovno dodaje umetajućom heuristikom, a ostale mutacije su opisane u odjeljku mutacija. Kao što je vidljivo iz grafa (5.8), mini mutacija radi minimalne promjene te kao takva ne dobiva pretjerano dobre rezultate. Zbog toga se niti mutacija najvećeg obilaska nije pokazala kao dobar izbor, dok su se nasumična mutacija i mutacija bliskih čvorova pokazale kao dobar izbor, a njihova kombinacija kao najbolji.



Slika 5.8: O - Mutacija najvećeg obilaska; M - Mini mutacija; R - Nasumična mutacija; B - Mutacija bliskih čvorova

6. Zaključak

Višekriterijski problem usmjeravanja vozila je NP složen problem iz stvarnoga svijeta za koji postoji više pristupa rješavanju. U ovome radu odabran je pristup evolucijskim algoritmom. Programski je ostvareno grafičko sučelje za odabir parametara, ispis rezultata te pokretanje algoritma, parsiranje ulaznih datoteka s problemom, umetajuća heuristika, ostvarena je Pareto separacija, grupirajuća selekcija, 3 operatora križanja i 3 operatora mutacije te adaptirani algoritam NSGA-2.

Kao najbolji parametri algoritma pokazali su se veličina populacije od 60 jedinki, vjerojatnost mutacije od 20%, kombinacija dvaju operatora križanja, LCSX i CAX, kombinacija dvaju operatora mutacije, nasumične mutacije i mutacije bliskih čvorova, 7 turnirska selekcija.

7. Literatura

[1] Golub, M., Genetski algoritam, prvi dio, Zagreb, 2004.

[2] Vaira, G., Genetic algorithm for vehicle routing problem, Vilinus, 2014.

[3] Gold, H., Carić, T., Vehicle routing problem, Zagreb, 2008.

[4] Jakobović, D., Adaptive genetic operators in elimination genetic algorithm, Zagreb.

[5] Vehicle routing problem, 6.11.2006., https://en.wikipedia.org/wiki/Vehicle_routing_problem, 1.6.2020.

[6] Auger, Anne and Bader, Johannes and Brockhoff, Dimo and Zitzler, Eckart, Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice of the Reference Point, Orlando, Florida, USA , Association for Computing Machinery, 2009.

[7] Zhou,W., Song, T., He, F. And Liu, X. (2013) Multiobjective Vehicle Routing Problem with Route Balance Based on Genetic Algorithm. Discrete Dynamics in Nature and Society, Volume 2013, 9 pages.

[8] Čupić, M., Prirodom inspirirani optimizacijski algoritmi, Metaheuristike, Zagreb, 2013

[9] A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, i M. McGinnity.Advances in Computational Intelligence Systems: Contributions Presented at the 18th UKWorkshop on Computational Intelligence, September 5-7, 2018, Nottingham, UK.Advances in Intelligent Systems and Computing. Springer International Publishing,2018.

[10] Multi-objective optimization, 25.3.2007.,
https://en.wikipedia.org/wiki/Multi-objective_optimization, 31.5.2020.

[11] Zhou, W., Song, T., Multiobjective Vehicle Routing Problem with Route Balance Based on Genetic Algorithm, Wuhan, 2013. [12] Chen, J., Chen, S., Optimization of Vehicle Routing Problem with Load Balancing
and Time Windows in Distribution, Jinan, 2008. [13] Lee, T., Ueng, J. , A study of

vehicle routing problems with load-balancing, Taichung, 1999.

[14] Pareto efficiency, https://en.wikipedia.org/wiki/Pareto_efficiency, 2019.

[15] VRPTW, 27.2.2008., https://www.sintef.no/projectweb/top/vrptw/

Rješavanje višekriterijskog problema usmjeravanja vozila evolucijskim algoritmima

Sažetak

Opisan je višekriterijski problem usmjeravanja vozila i korišteni algoritam NSGA-II. Programski je ostvaren genetski algoritam za rješavanje višekriterijskog problema usmjeravanja vozila. U sklopu toga su ostvarena 3 operatora mutacije, 3 operatora križanja, grupirajuća selekcija, umetajuća heuristika, grafičko sučelje, Pareto separator i parser za dokumente koji zadaju problem. Izvedeni su eksperimenti i prikazani rezultati algoritma pri varijaciji pojedinih parametara i operatora.

Ključne riječi: Višekriterijski problem usmjeravanja vozila, evolucijski algoritam, Pareto optimizacija, križanje, mutacija, selekcija, jedinka, populacija.

Evolutionary Algorithm for Vehicle Routing

Abstract

The multiobjective vehicle routing problem and the NSGA-II algorithm are described. A genetic algorithm for solving the multiobjective vehicle routing problem was implemented. 3 mutation operators, 3 crossover operators, a grouping selection, an insertion heuristic, a graphical interface, and a parser for documents that define the problem were implemented. Statistics have been made showing how successful the algorithm is for variations of parameters and operators.

Keywords: Multiobjective vehicle routing problem, evolutionary algorithm, Pareto optimisation, crossover, mutation, selection, unit, population.