Učinkovitost algoritma NSGA II za višekriterijski problem trgovačkog putnika

Paunović, Patrik

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Učinkovitost algoritma NSGA II za višekriterijski problem trgovačkog putnika

Rijeka, srpanj 2023.

Patrik Paunović
0069089098

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Učinkovitost algoritma NSGA II za višekriterijski problem trgovačkog putnika

Mentor: izv.prof.dr.sc. Goran Mauša

Rijeka, srpanj 2023.

Patrik Paunović
0069089098

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14. ožujka 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Programsko inženjerstvo**
Grana: **2.09.04 umjetna inteligencija**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Patrik Paunović (0069089098)**
Studij: Sveučilišni prijediplomski studij računarstva

Zadatak: **Učinkovitost algoritma NSGA II za višekriterijski problem trgovačkog putnika // Efficiency of NSGA II algorithm for multi-objective travelling salesman problem**

Opis zadatka:

Istražiti osnovne operatore algoritma NSGA II te njihove varijante za reprezentaciju koja je pogodna za rješavanje višekriterijskog problema trgovačkog putnika. Implementirati programsko rješenje koje provodi optimizaciju rute trgovačkog putnika s dva proturječna kriterija. Istražiti utjecaj varijanti operatora mutacije, rekombinacije i odabira na učinkovitost i uspješnost pronalaska rješenja.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Doc. Goran Mauša, dipl. ing.

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, srpanj 2023.

Ime Prezime

Zahvala

Velika zahvala mentoru doc. dr. sc. Goranu Mauši na konzistentnoj podršci i odličnome mentorstvu u izradi ovoga završnog rada. Zahvaljujem obitelji i kolegama u podržavanju tijekom studiranja.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Strojno učenje predstavlja granu umjetne inteligencije koja se bavi razvojem algoritama i sustava koji su sposobni učiti iz podataka i donositi zaključke na temelju tih podataka. Koncept strojnog učenja potječe još iz pedesetih godina prošlog stoljeća, no tek nedavni značajan napredak u računalnoj snazi i dostupnosti velikih količina podataka doveli su do eksplozije primjena strojnog učenja u različitim područjima, od preporučiteljskih sustava, preko autonomnih vozila, pa sve do medicinske dijagnostike.

Razvoj algoritama za optimizaciju i strojno učenje datira još od ranog 20. stoljeća, kada je riječ o teoriji optimalnog upravljanja, statističkoj analizi i teoriji igara. Međutim, s pojavom računala i sve većim količinama podataka koje je potrebno obraditi, razvoj ovih algoritama postao je još značajniji. Danas su algoritmi strojnog učenja u širokoj primjeni u različitim područjima, poput medicinske dijagnostike, financija, preporučivanja proizvoda i usluga, prepoznavanja govora i slike, te brojnih drugih.

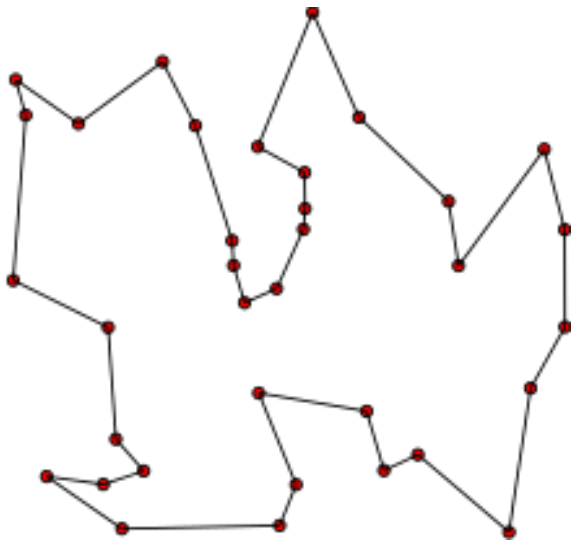
Algoritmi za optimizaciju i strojno učenje imaju značajnu ulogu u rješavanju složenih problema i postizanju optimalnih rješenja u različitim domenama. Uz njihovu pomoć, moguće je učinkovito obraditi velike količine podataka, automatizirati procese i poboljšati kvalitetu donošenja odluka [1].

Velika količina dostupnih podataka i rastuća složenost problema koje rješavamo, dovodi do pitanja pitanja održivosti algoritama koje koristimo. U ovom radu istražiti

će se osnovne operatore algoritma NSGA II(eng. Non-Dominated Sorting Genetic Algorithm II) te njihove varijante kako bismo ustanovili njihov utjecaj na učinkovitost višekriterijske optimizacije. Istraživanje je provedeno u okviru ERASMUS+ projekta *Promoting Sustainability as a Fundamental Driver in Software development Training and Education* sa oznakom 2020-1-PT01-KA203-078646.

Provedena analiza je temeljena na višekriterijskom problemu trgovačkog putnika (TSP, eng. Traveling Salesman Problem) [2]. Riječ je o poznatom problemu optimizacije koji se bavi pronalaskom najkraćeg puta koji povezuje skup gradova, pri čemu svaki grad mora biti posjećen jednom i na kraju se vraćamo u početni grad. TSP spada u NP-teške probleme [3], što znači da ne postoji brz način rješavanja ovog problema za veće skupove gradova, ali postoje algoritmi koji daju približna rješenja.

TSP je važan problem u mnogim područjima, poput logistike, transporta, proizvodnje, telekomunikacija, genetike i drugih. Rješavanje TSP-a može smanjiti troškove, vrijeme putovanja i povećati efikasnost poslovanja.



Slika 1.1 Prikaz rješenja TSP-a gdje crna crta povezuje crvene točke na nakraći mogući način

Izvor: Travelling salesman problem — Wikipedia, the free encyclopedia [4]

Poglavlje 2

Metodologija

2.1 Skup podataka

Skup podataka(eng. dataset) koji sam koristio za ovaj rad su bili berlin52 i eil51.

Za problem putovanja trgovaca s instancom "berlin52" i "eil51", koristi se dataset koji predstavlja Berlin52 i Eil51 TSP problem. U tom slučaju, "berlin52" se odnosi na instancu s 52 grada u Berlinu dok se "eil51" odnosi na instancu s 51 gradom u Eil-u.

Broj mogućih kombinacija za problem trgovačkog putnika ovisi o broju gradova i varijanti problema. U klasičnom TSP-u, broj mogućih kombinacija raste faktorijelno s brojem gradova. Dakle, za "berlin52" problem, broj mogućih kombinacija je približno 52 faktorijela dok za "eil51" broj mogućih kombinacija iznosi 51 faktorijela. U tablici 2.1 prikazan je broj mogućih kombinacija. Ove vrijednosti predstavljaju ogroman broj mogućih kombinacija, što ukazuje na složenost problema putovanja trgovaca.

Tablica 2.1 Skup podataka i broj mogućih kombinacija

Skup podataka	Veličina	Broj mogućih kombinacija
berlin52	52 grada	52! $\approx$ 8.0658175e+67
eil51	51 grad	51! $\approx$ 1.5511188e+66

2.2 Korištene tehnologije

2.2.1 Python

Python je programski jezik koji je popularan zbog svoje jednostavnosti i intuitivnosti. Python ima jasnu i čitljivu sintaksu, a osim toga je i izrazito fleksibilan i može se koristiti za različite vrste programiranja, od web aplikacija do analize podataka i strojnog učenja.

Python se često koristi u različitim znanstvenim i akademskim područjima, uključujući biologiju, fiziku, matematiku i računalnu znanost. Koristi se za programiranje algoritama i različitih vrsta simulacija, kao i za izradu skripti i automatizaciju zadataka.

Python se koristi u genetičkim algoritmima jer je jednostavan za učenje, a ima i veliku podršku i zajednicu koja razvija biblioteke i module koji su korisni za ove algoritme. Postoji mnogo biblioteka poput NumPy-a, SciPy-a, Pandas-a i Scikit-learn-a koji nude podršku za matematičke operacije, optimizaciju, obradu podataka i strojno učenje, što je korisno pri izradi genetičkih algoritama.

Python također ima veliku količinu gotovih rješenja i otvorenih izvornih kodova, što znači da se može koristiti za brzo prototipiranje algoritama i različitih aplikacija. Također, Python se može koristiti za različite vrste izvedbi genetičkih algoritama, od klasifikacije, pretraživanja optimizacije, do problema putujućeg trgovca, jer je vrlo fleksibilan i prilagodljiv. [5]

2.2.2 PyRAPL

PyRAPL [6] je softverski alat za mjerenje potrošnje energije prilikom izvršavanja određenog Python koda. PyRAPL koristi Intel-ov "Running Avergage Power Limit" (RAPL) [7] tehnologiju koja računa potrošnju energije procesora. Može mjeriti potrošnju energije sljedećih procesorskih domena:

- Podnožje procesora
- DRAM (za arhitekturu servera)

- GPU (za arhitekturu klijenta)

Ukoliko želimo instalirati pyRAPL možemo to učiti sa naredbom **"pip install pyRAPL"**

2.2.3 Tkinter

Tkinter je standardni Python modul za izradu grafičkog korisničkog sučelja (GUI). To je jednostavan način za stvaranje prozora, gumba, okvira, teksta, itd. u Pythonu, i lako ga je integrirati s drugim Python bibliotekama. Tkinter se temelji na biblioteci Tk koja je napisana u programskom jeziku Tcl, a koja je dostupna na svim operacijskim sustavima. Tkinter se obično koristi za izradu jednostavnih GUI aplikacija i alata, a njegova jednostavnost i brzina ga čine popularnim među početnicima u Pythonu.

2.2.4 Matplotlib

Matplotlib je jedna od najpopularnijih biblioteka u Pythonu koja se koristi za vizualizaciju podataka. Ova biblioteka omogućuje korisnicima stvaranje različitih vrsta grafova, dijagrama, histograma, slika, itd. Uz to, matplotlib je vrlo prilagodljiv i omogućuje korisnicima podešavanje svih aspekata vizualizacije, uključujući osi, oznake, boje, fontove i mnoge druge parametre.

Matplotlib se može koristiti i kao interaktivna biblioteka za crtanje grafova, koja omogućuje korisnicima da pomiču grafikon, zumiraju, rotiraju i mijenjaju njegov izgled.

Matplotlib se često koristi u znanstvenim i istraživačkim područjima, kao i u industriji za vizualizaciju velikih skupova podataka, strojno učenje i analizu podataka.

2.2.5 Tqdm

Tqdm je Python paket koji omogućava jednostavno stvaranje progresnih traka (engl. progress bars) za iteracije kroz liste ili datoteke. Progresna traka daje korisnicima vizualni prikaz napretka njihovog programa, što može biti korisno za prikaz dugo-

trajnih operacija poput obrade velikih datoteka ili slanja velikih količina podataka preko mreže.

Kada se tqdm koristi u petlji, korisniku će biti prikazan vizualni prikaz napretka u stvarnom vremenu. tqdm automatski izračunava procenat završetka iteracije i procjenjuje vrijeme koje je preostalo do završetka. Također omogućava podešavanje stila i drugih parametara progresne trake kako bi se prilagodili potrebama korisnika.

Poglavlje 3

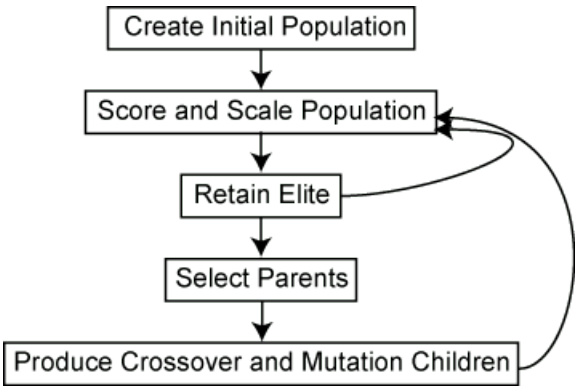
Genetski algoritam

3.1 Pojam genetskog algoritma

Genetski algoritam [8] je tehnika koja se koristi za rješavanje problema optimizacije, bez obzira na to jesu li ograničeni ili neograničeni, koristeći princip prirodnog odabira te je inspiriran biološkom evolucijom.

Genetski algoritam provodi višestruke modifikacije skupa pojedinačnih rješenja, pri čemu se u svakom koraku odabiru određeni pojedinci iz trenutnog skupa kako bi postali roditelji i njihova kombinacija stvara novu generaciju. Kroz niz generacija, populacija "evoluira" prema optimalnom rješenju.

Primjena genetskog algoritma može biti korisna za rješavanje različitih problema optimizacije koji ne odgovaraju dobro standardnim algoritmima. Ova tehnika je posebno korisna za probleme s diskontinuiranom, nediferencijabilnom, stohastičkom ili izrazito nelinearnom ciljnom funkcijom.

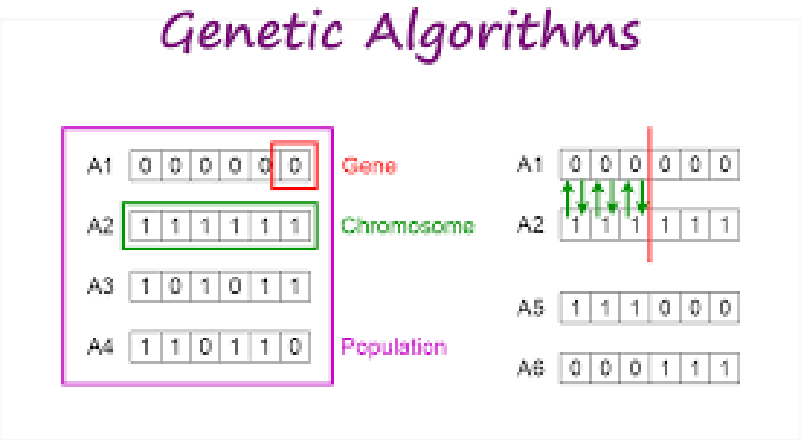


Slika 3.1 Dijagram toka koji prikazuje glavne algoritamske korake u genetskom algoritmu

Izvor: What is the genetic algorithm? — MathWorks [9]

U genetskim algoritmima, gen je obično predstavljen kao dio kromosoma koji kodira karakteristike rješenja problema koji se pokušava riješiti. Svaki kromosom predstavlja jedno rješenje problema, a sastoji se od niza gena koji predstavljaju različite varijable ili parametre u rješenju.

Populacija se sastoji od skupa kromosoma koji predstavljaju potencijalna rješenja problema. Svaki kromosom u populaciji je potencijalno rješenje problema, a njegova kvaliteta se ocjenjuje pomoću funkcije dobrote (engl. fitness function). Genetski algoritam koristi operator odabira, križanje i mutaciju kako bi generirao nove kromosome koji su potom zamijenjeni s najlošijim kromosomima iz trenutne populacije. Ovaj proces se ponavlja sve dok se ne postigne zadani cilj ili se ne ispuni kriterij zaustavljanja [10].



Slika 3.2 Prikaz prethodno navedenih parametara
Izvor: Introduction to genetic algorithms [11]

3.2 Pojam prirodnog odabira

Proces prirodnog odabira(eng. natural selection) počinje odabirom najsposobnijih jedinki iz populacije. One proizvode potomke koji nasljeđuju karakteristike roditelja te će biti dodani sljedećoj generaciji. Ako roditeli imaju bolju vrijednost dobrote, njihovi potomci će biti bolji od njih te će imati veće šanse za opstanak. Ovaj proces se ponavlja više puta te na kraju će se stvoriti generacija sa najboljim jedinkama.

U genetskom algoritmu postoji pet faza:

1. Inicijaliziranje populacije
2. Funkcija dobrote
3. Operator odabira
4. Križanje
5. Mutacija

3.3 Inicijaliziranje populacije

Proces započinje sa skupom jediniki koje se nazivaju populacija. Pojedinca karakterizira skup parametara(varijabli) poznatih kao geni. Geni su spojeni u niz kako bi formatirali kromosom(soluciju). U genetskom algoritmu, skup gena pojedinca predstavljen je pomoću stringa. Obično se koriste binarne vrijednosti(niz 1 i 0). Kažemo da kodiramo gene u kromosomu.

3.4 Funkcija dobrote

Funkcija dobrote određuje koliko je pojedinac u formi(sposobnost pojedinca da se natječe s drugim pojedincima). Svakom pojedincu daje ocjenu kondicije. Vjerojatnost da će jedinka biti odabrana za reprodukciju temelji se na ocjeni njezine sposobnosti.

Funkcija dobrote se obično definira u obliku matematičke funkcije, koja uzima rješenje kao ulazni parametar i vraća jednu numeričku vrijednost. Ta vrijednost predstavlja kvalitetu rješenja u odnosu na cilj optimizacije, a najbolja rješenja imaju najveću vrijednost dobrote.

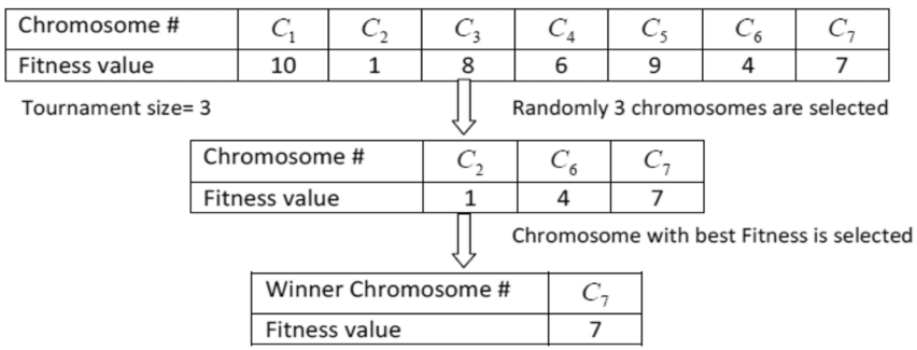
3.5 Operator odabira

Ideja procesa operatora odabira(eng. selection) je odabrati najspremnije jedinke i pustiti ih da prenesu svoje gene sljedećoj generaciji. Dva para pojedinaca(roditelja) odabiru se na temelju njihovih rezultata njihove dobrote. Pojedinci s većom vrijednosti dobrote imaju veće šanse da budu odabrani za reprodukciju.

U završnom radu je korišten tournament odabir te roulette odabir [12]. Ove metode su popularne jer su jednostavne za implementaciju i mogu se koristiti za bilo koju vrstu problema.

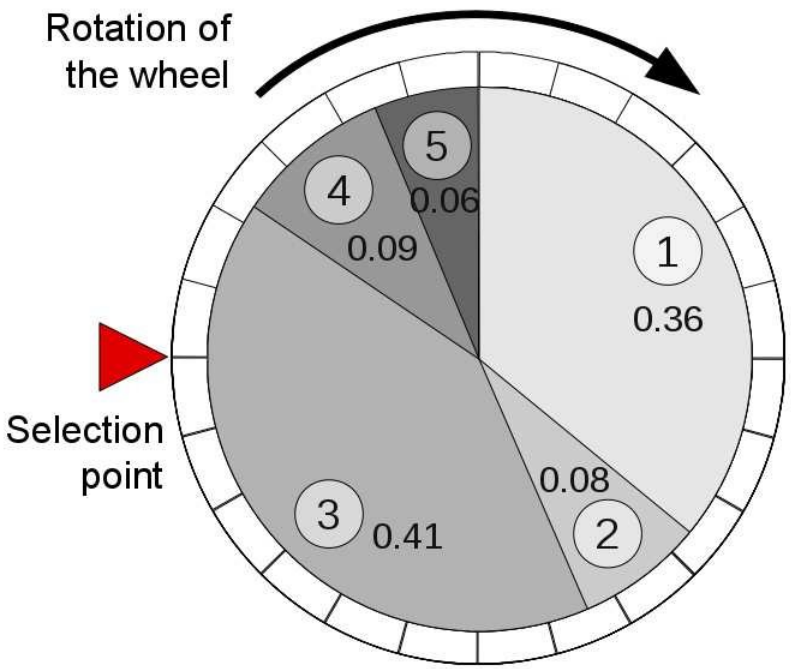
Kod tournament odabira postupak se sastoji od odabira nasumičnog broja jedinki iz populacije, obično manjeg broja od ukupnog broja jedinki u populaciji. Te jedinke se zatim natječu u turniru, gdje se izabire jedinka s najboljim fitnesom. Odabrane jedinke postaju roditeljske jedinke za stvaranje nove populacije u sljedećoj generaciji.

Kako bi se povećala raznolikost i spriječilo zaglavljivanje u lokalnom optimumu, turnir se obično provodi više puta s različitim veličinama turnira i najboljim jedinkama se dodjeljuju veće vjerojatnosti za odabir. Osim toga, turnir odabir se može koristiti i s dodatnim parametrom za podešavanje vjerojatnosti odabira jedinke s manjim fitnesom, kako bi se spriječila prevelika koncentracija dobrih rješenja u populaciji.



Slika 3.3 Prikaz tournament odabira
Izvor: A novel robust soft-computed range-free localization algorithm against malicious anchor nodes [13]

Osnovna ideja iza roulette odabira je da se pojedincima dodjeljuje vjerojatnost odabira koja je proporcionalna njihovoj dobroti (fitness vrijednosti). Pojedinci s većom prilagodbom imaju veću vjerojatnost da budu odabrani za reprodukciju. Roulette odabir omogućava pojedincima s većom prilagodbom da imaju veću vjerojatnost da budu odabrani, ali ne jamči njihov odabir. Pojedinci s manjom prilagodbom također mogu biti odabrani, iako s manjom vjerojatnošću. Ovaj odabirni mehanizam doprinosi očuvanju raznolikosti u populaciji i omogućuje bolje pretraživanje prostora rješenja.



Slika 3.4 Prikaz roulette odabira
Izvor: Genetic algorithm-based calibration of reduced order galerkin models [14]

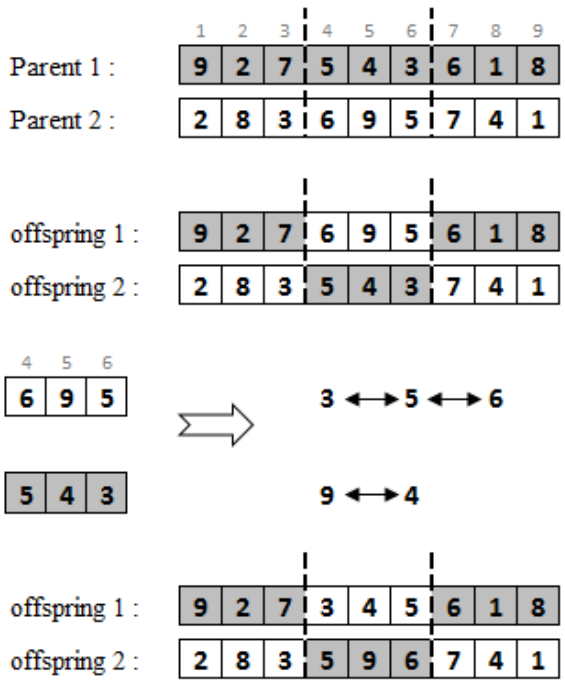
3.6 Križanje

Križanje(crossover) je najbitniji dio procesa u genetskom algoritmu. Za svaki par roditelja koji će se križati, točka križanja(eng. crossover point) je odabrana nasumično iz gena. U ovom završnom radu koristiti ćemo križanja poput pmx(partially mapped crossover), hx(hierarchical crossover), two-point križanje, uniform križanje.

Kod hijerarhijskog križanja, populacija rješenja se grupira u hijerarhijsku strukturu koja se sastoji od više razina. Križanje se provodi na različitim razinama hijerarhije, ovisno o postavljenim pravilima. Najčešće se koristi "bottom-up" pristup, gdje se križanje provodi od donjih razina prema višim razinama. U tom pristupu, najniže razine hijerarhije se prvo križaju, a zatim se dobiveni potomci kombiniraju sa susjednim rješenjima na višoj razini.

Kod pmx križanja se Nasumično odabiru dva mjesta (engl. cut points) u roditeljskim kromosomima koji određuju segmente koji će biti zamijenjeni između roditelja. Uzima se segment između dva odabrana mjesta u prvom roditelju i kopira se u po-

tomstvo. Zatim se segment između istih mjesta u drugom roditelju pregledava jedan po jedan gen. Ako se gen iz segmenta drugog roditelja nalazi unutar segmenta koji je već kopiran iz prvog roditelja, taj gen se preskače. Ako se gen iz segmenta drugog roditelja ne nalazi u segmentu kopiranom iz prvog roditelja, on se zamjenjuje s istim genom u prvom roditelju, a zatim se traži drugi gen iz segmenta drugog roditelja. Ovaj postupak se nastavlja dok se ne popuni cijeli potomak.



Slika 3.5 Prikaz pmx križanja
Izvor: New crossover operator for genetic algorithm to resolve the fixed charge transportation problem [15]

Two-point križanje je operator genetskog križanja koji se koristi u genetskim algoritmima za kombiniranje dvaju roditeljskih kromosoma i stvaranje potomaka. Križanje se provodi tako da se nasumično odaberu dva mjesta prekida u kromosomima roditelja. Nakon toga, dijelovi kromosoma između ta dva mjesta zamijene se između roditelja kako bi se stvorila dva nova potomka. Ovaj postupak se može primijeniti na bilo koju vrstu kromosoma, uključujući i one s fiksnom duljinom, različitim duljinama, permutacijskim i drugim varijantama.

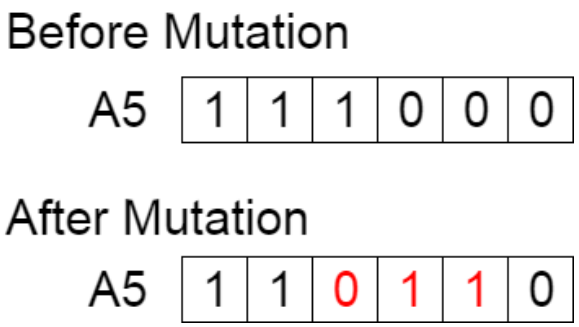
Uniformno križanje (engl. uniform crossover) je operator križanja u genetskom

algoritmu koji služi za stvaranje novih jedinki iz dva roditeljska genoma.

Kod uniformnog križanja, jedinke se stvaraju tako da se za svaki gen slučajno bira od kojeg će roditelja biti preuzet. To znači da će svaki gen nasljednika biti jednako vjerojatno preuzet od jednog od roditelja.

3.7 Mutacija

U određenim novoformiranim potomcima neki od njihovih gena mogu biti podvrgnuti mutaciji s malom slučajnom vjerojatnošću. U programu je ta vjerojatnost podešena na 5%. To implicira da se neki od bitova u nizu bitova mogu okrenuti. Razlog zašto to radimo je da bi održali raznolikost unutar populacije i kako bi spriječili preranu konvergenciju.



Slika 3.6 Prikaz mutacije
Izvor: Introduction to genetic algorithms [11]

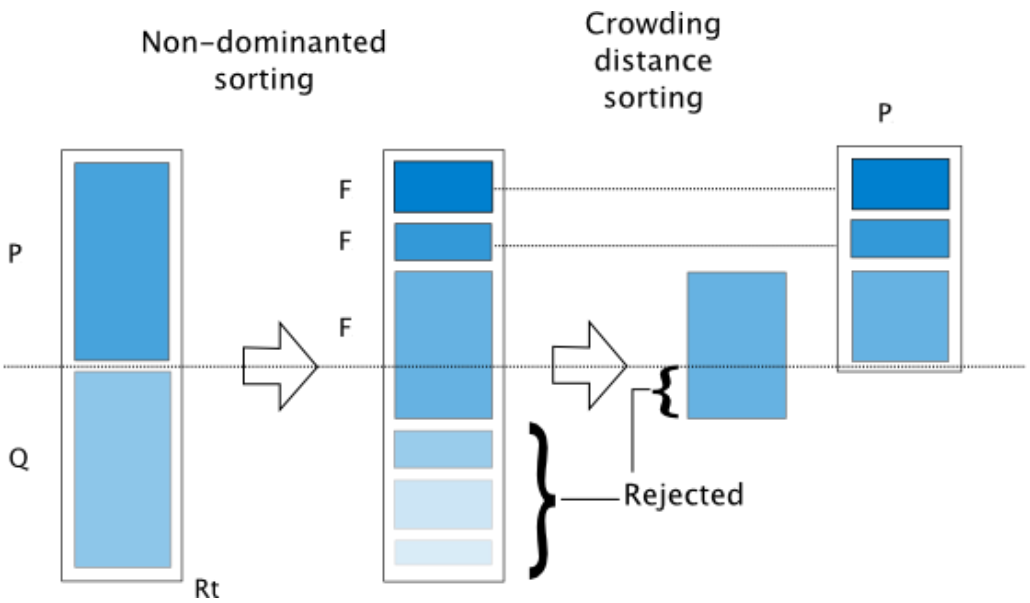
3.8 NSGA-II

NSGA-II (Nondominated Sorting Genetic Algorithm II) [16] je algoritam za višekriterijsku optimizaciju koji se široko koristi u inženjerstvu, računarstvu i drugim područjima. Cilj NSGA-II je pronaći skup rješenja koja su međusobno nedominantna, što znači da nijedno rješenje u skupu nije lošije od bilo kojeg drugog rješenja u skupu u odnosu na sva kriterijska ograničenja.

NSGA-II radi korištenjem genetskog algoritma za razvoj populacije potencijalnih rješenja tijekom više generacija. Algoritam počinje generiranjem početne populacije slučajnih rješenja. Svako se rješenje ocjenjuje na temelju toga koliko dobro zadovoljava ciljeve koji se optimiziraju. NSGA-II koristi brzu i učinkovitu tehniku sortiranja koja se naziva "nedominirano sortiranje" za rangiranje rješenja na temelju njihovih odnosa dominacije.

U NSGA-II, skup rješenja se smatra nedominantnim ako nijedno rješenje u skupu nije lošije od bilo kojeg drugog rješenja u skupu u odnosu na sva kriterijska ograničenja. Algoritam održava populaciju nedominantnih rješenja i koristi metriku "razmaka gužve" (eng. crowding distance) kako bi potaknuo diverzitet u populaciji.

NSGA-II je široko korišten algoritam za optimizaciju s više ciljeva jer je učinkovit, robustan i može se nositi s velikim brojem ciljeva. Korišten je za rješavanje raznih problema u inženjerstvu, financijama i drugim područjima.



Slika 3.7 Prikaz NSGA-II
Izvor: Nsga-II [17]

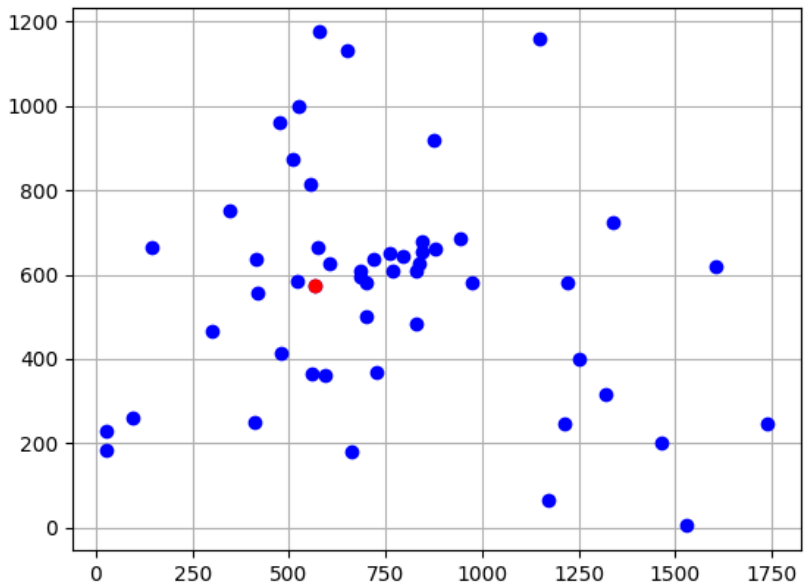
Poglavlje 4

Rezultati

U ovom djelu ćemo prikazati rezultate izvršenih mjerenja. Mape po kojima će trgovac ići su berlin52 i eil51, koristiti ćemo 4 vrste križanja a to su: hx, pmx, two-point i uniform. Operatori odabira su: roulette i tournament. Broj iteracija i populacije na početku je 100 neovisno o broju ponavljanja pa ćemo kasnije izbalansirati broj iteracija i populacije ovisno o broju ponavljanja. Mjerimo vrijeme potrebno da se kod izvrši te energetsku potrošnju procesora i dinamičke memorije.

4.1 Križanje

U 100 iteracija je odrađena hijerarhijsko, pmx, two-point te uniform križanje metode uz 1,2 ili 3 ponavljanja mjerenja. Mapa sa unaprijed određenim koordinatama koja je korištena se zove berlin52 [18] uz tournament selekciju. U prilogu ću prikazati mjerenja te grafove usporedbi vremenskog izvođenja te potrošnje energije različitih križanja.



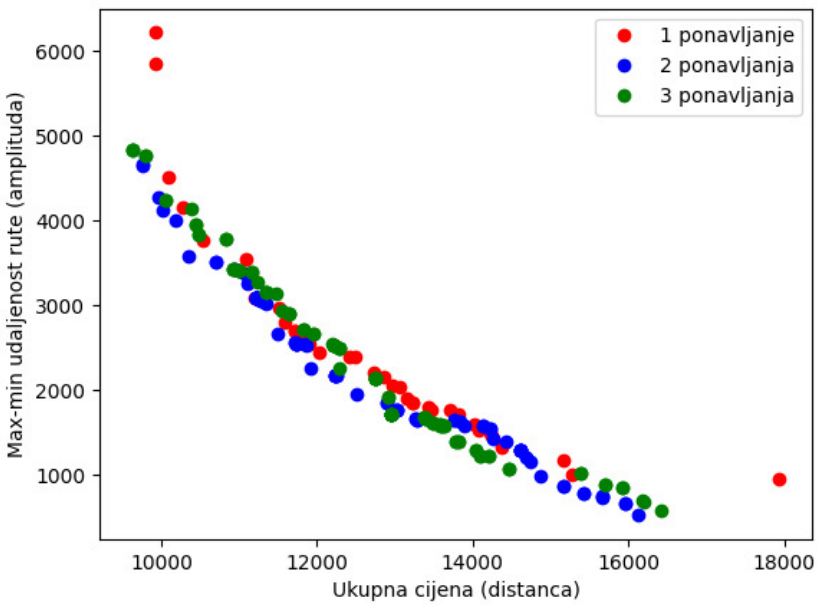
Slika 4.1 Prikaz berlin52 mape gdje crvena točka označuje početnu lokaciju a plave točkice znače lokacije koje treba posjetiti

4.1.1 Hijerarhijsko križanje

Tablica 4.1 prikazuje mjerenja postignuta sa hijerarhijskim križanjem korištenjem berlin52 mape (slika 4.1). Na slici 4.2 x-os predstavlja ukupnu cijenu puta što u prijevodu znači ukupan trošak uložen za prijeđen put dok y-os predstavlja razliku između najdalje i najbliže rute.

Tablica 4.1 Mjerenja sa hijerarhijskim križanjem

Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	26.19	154.05	20.59	berlin52	hx
2	41.84	241.56	27.89	berlin52	hx
3	57.29	352.79	39.80	berlin52	hx



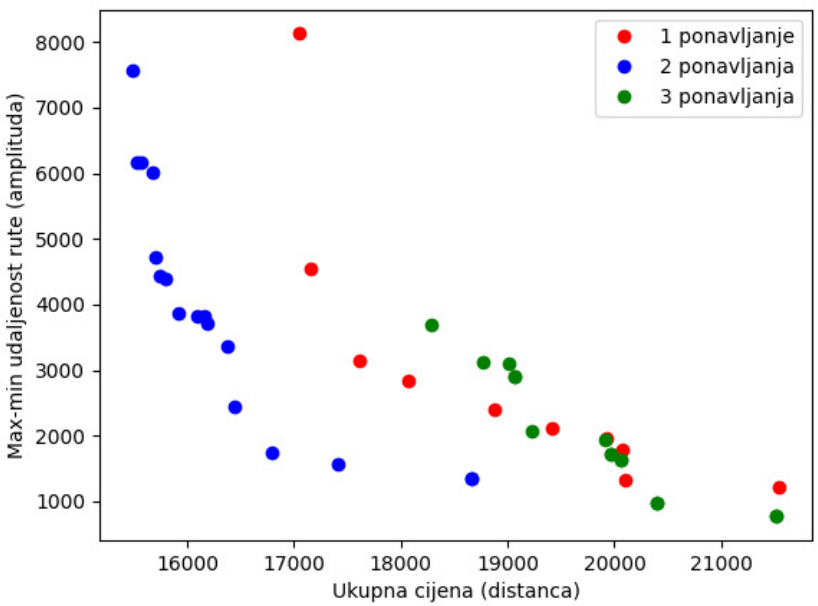
Slika 4.2 Prikaz hijerarhijskog križanja sa različitim brojem ponavljanja

4.1.2 Pmx križanje

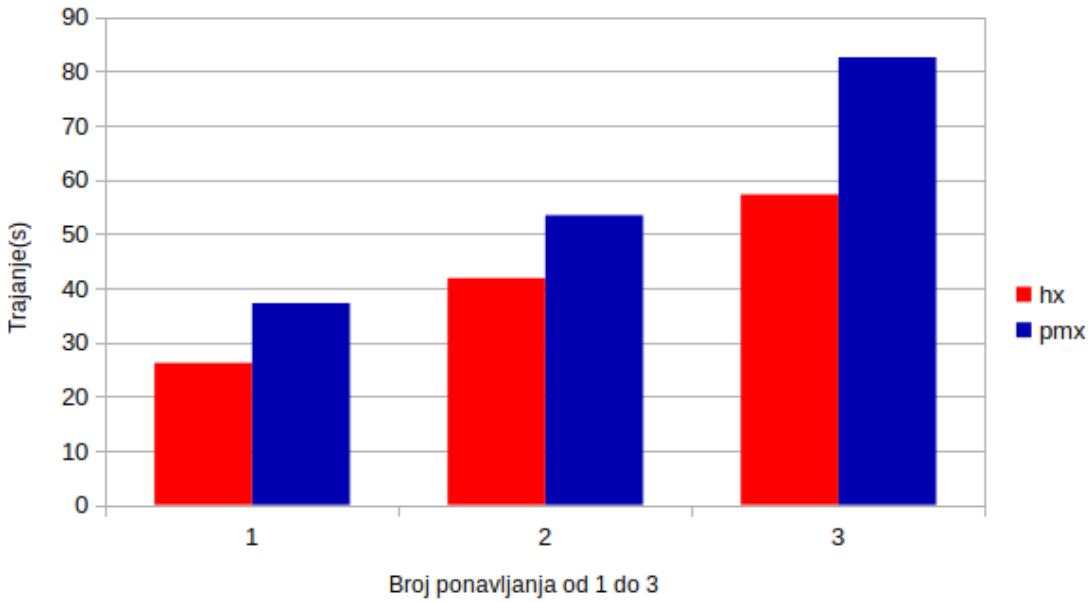
U tablici 4.2 možemo vidjeti mjerenja postignuta sa pmx križanjem te usporedimo li tablicu 4.1 sa 4.2 možemo zaključiti kako pmx križanje troši više energije i duže se vremenski izvodi ali grafički prikaz mjerenja na slici 4.3 nema poklapajuće rezultate kao slika 4.2.

Tablica 4.2 Mjerenja sa pmx križanjem

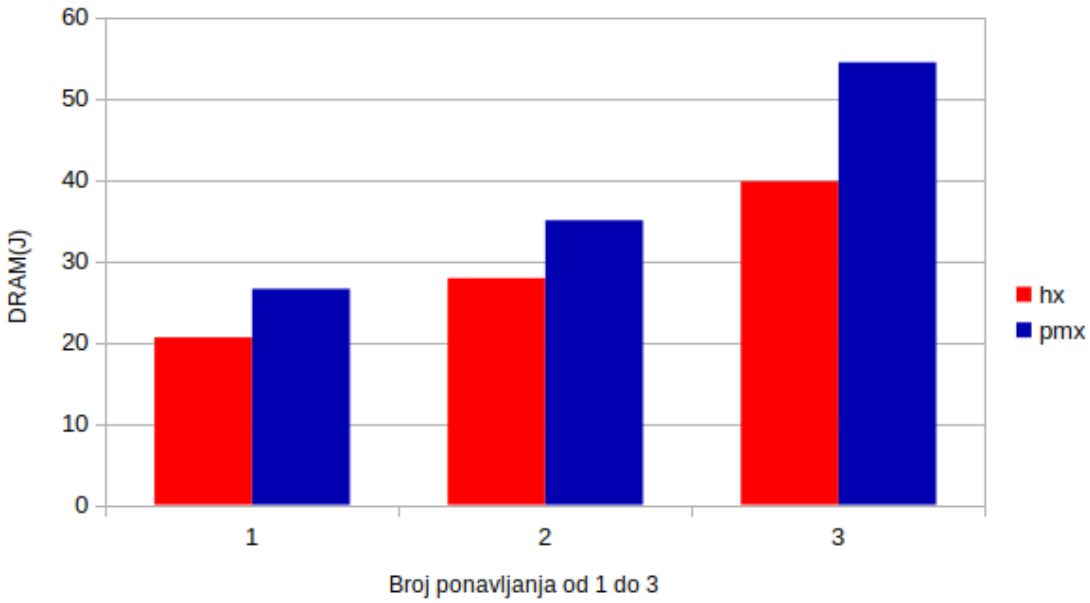
Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	37.22	214.71	26.57	berlin52	pmx
2	53.44	302.37	35.01	berlin52	pmx
3	82.59	487.97	54.44	berlin52	pmx



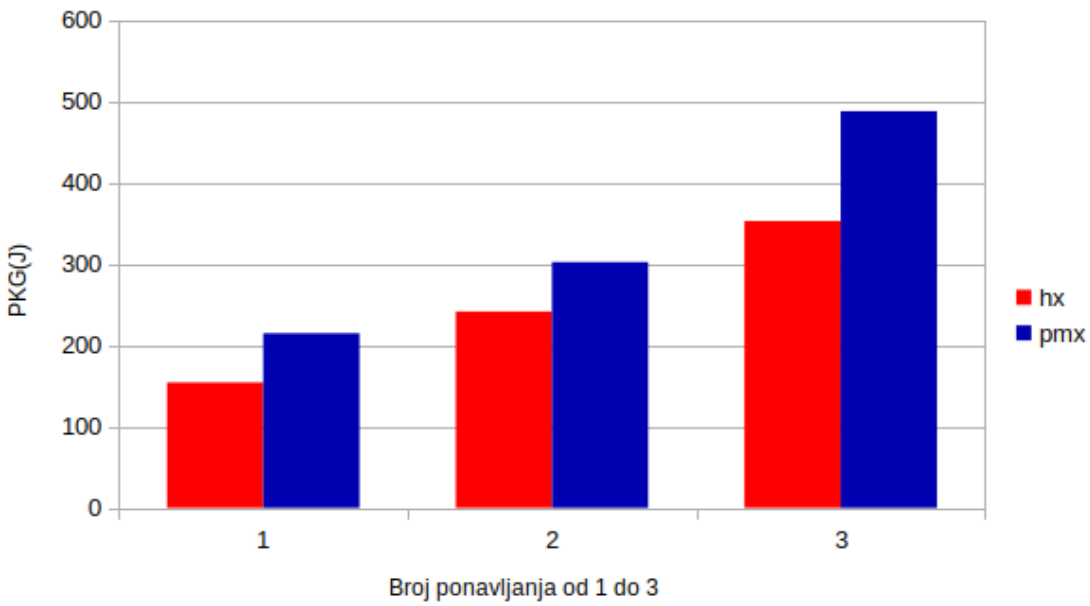
Slika 4.3 Prikaz pmx križanja sa različitim brojem ponavljanja



Slika 4.4 Graf usporedbe hx i pmx križanja u vremenu



Slika 4.5 Graf usporedbe hx i pmx križanja u DRAM-u



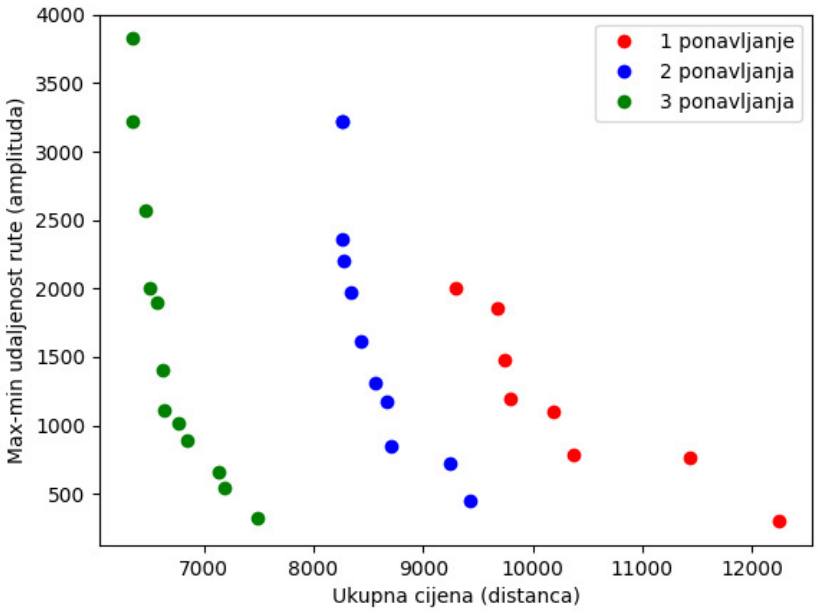
Slika 4.6 Graf usporedbe hx i pmx križanja u PKG-u

4.1.3 Two-point križanje

Kod two point križanja iz tablice 4.3 i slike 4.7 možemo zaključiti kako sa većim brojem ponavljanja dolazimo do boljih rezultata ali naravno uz veći utrošak energije i vremena.

Tablica 4.3 Mjerenja sa two-point križanjem

Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	23.33	118.74	17.39	berlin52	two-point
2	38.15	207.50	27.89	berlin52	two-point
3	53.55	299.33	37.07	berlin52	two-point



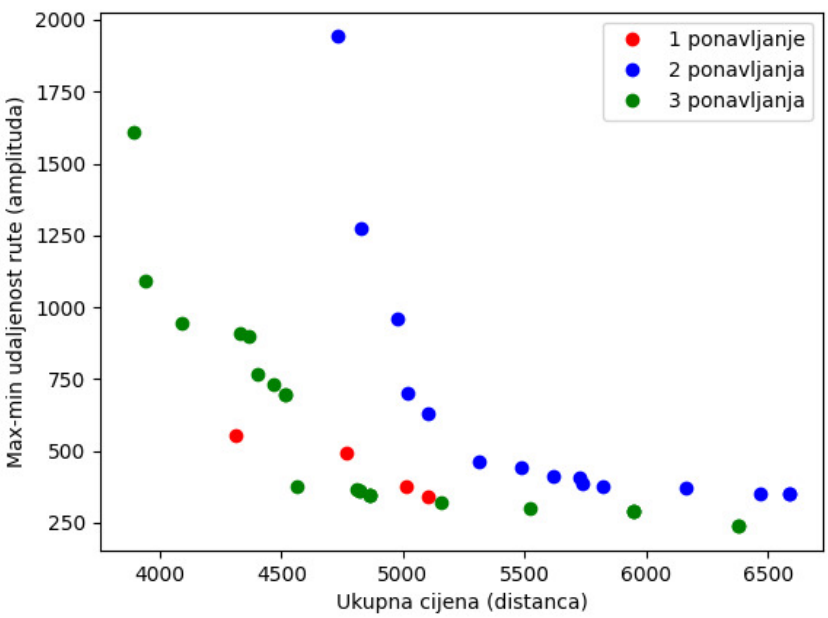
Slika 4.7 Prikaz two-point križanja kod različitih broja ponavljanja

4.1.4 Uniform križanje

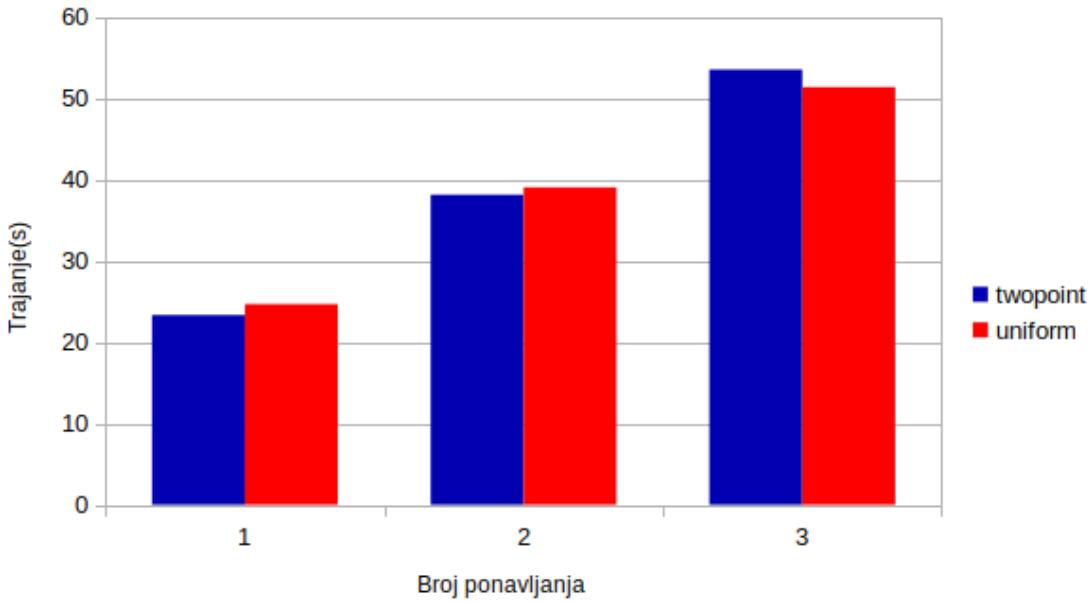
Kod uniform križanja dobivamo još bolje rezultate što se može vidjeti na slici 4.8 uz gotovo jednak utrošak energije i vremena kao i kod two-point križanja što možemo vidjeti kod grafa usporedbi na slikama 4.9, 4.10 i 4.11.

Tablica 4.4 Mjerenja sa uniform križanjem

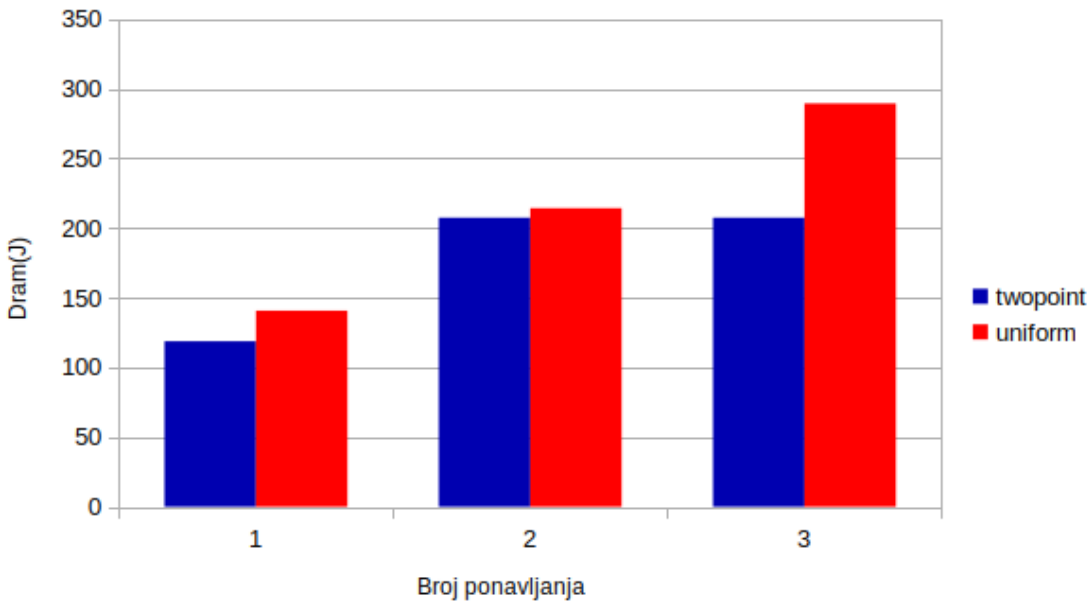
Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	24.68	140.73	20.40	berlin52	uniform
2	39.05	214.34	27.95	berlin52	uniform
3	51.39	289.48	35.16	berlin52	uniform



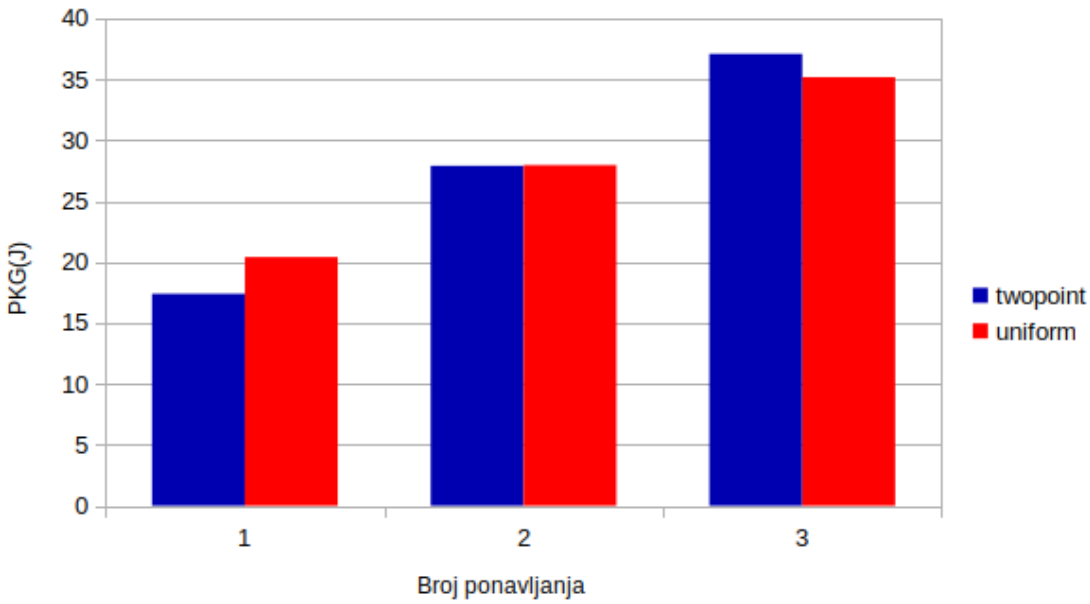
Slika 4.8 Prikaz uniform križanja kod različitih broja ponavljanja



Slika 4.9 Graf usporedbe two-point i uniform križanja u vremenu



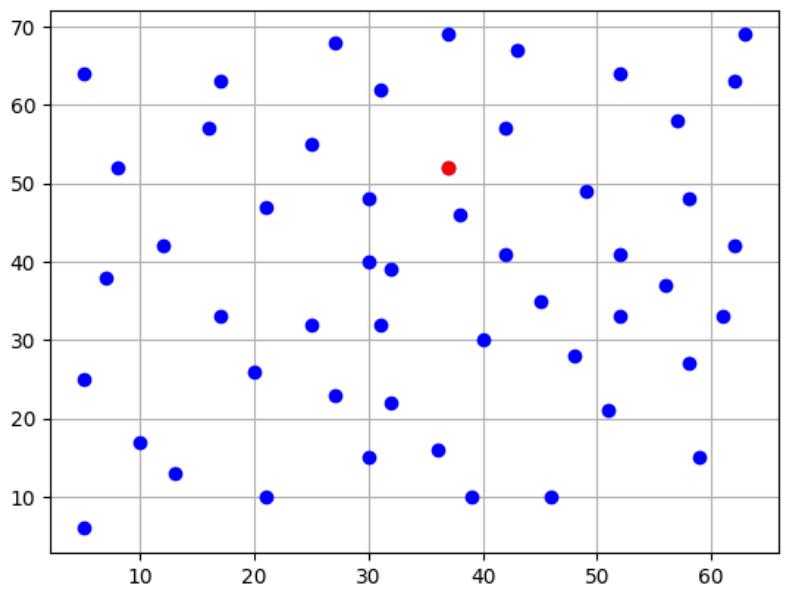
Slika 4.10 Graf usporedbe two-point i uniform križanja u DRAM-u



Slika 4.11 Graf usporedbe two-point i uniform križanja u PKG-u

4.2 Promjena operatora odabira i mape

U sljedećem djelu je promijenjen operator odabira sa tournament na roulette te berlin52 datoteka sa kordinatama na eil51 datoteku [19]. Na slici 4.12 možemo vidjeti prikaz eil51 mape sa označenim lokacijama.

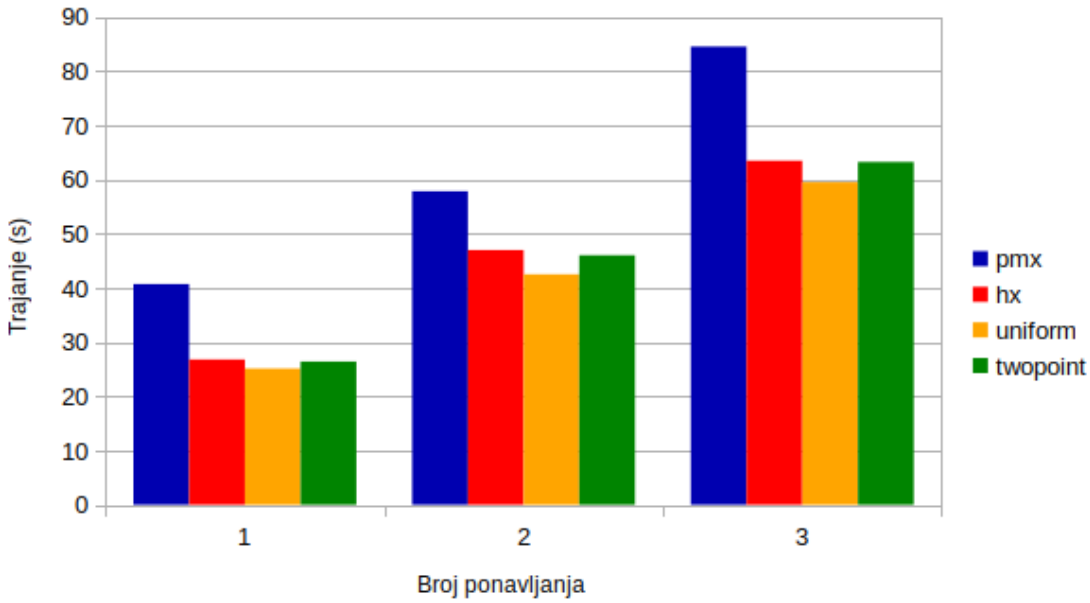


Slika 4.12 Prikaz eil51 mape gdje crvena točka označuje početnu lokaciju a plave točkice znače lokacije koje treba posjetiti

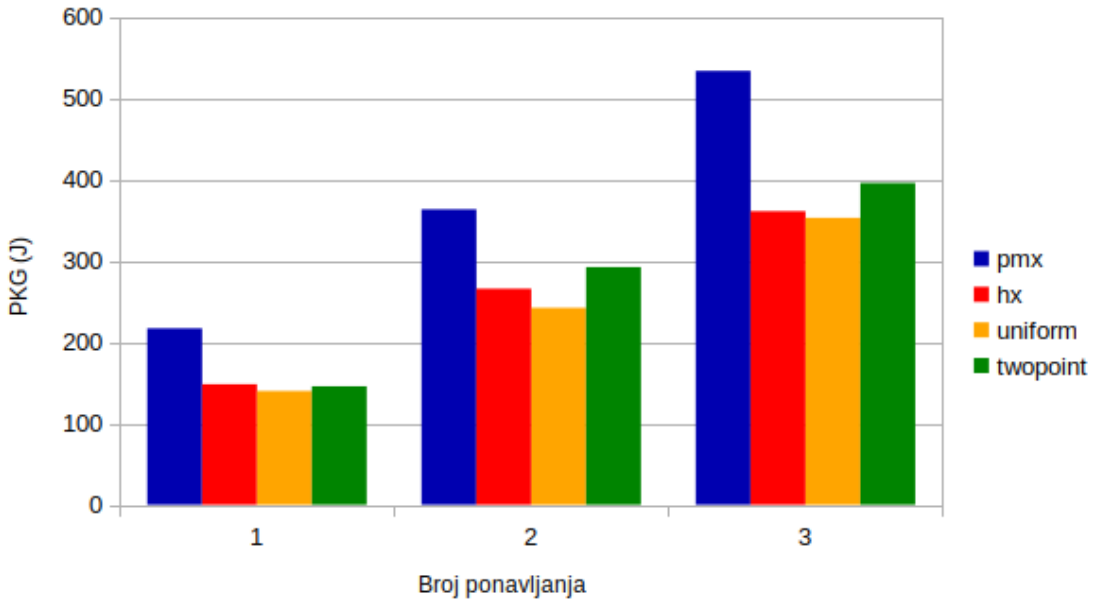
Iz tablice 4.5 primjećujemo kako pmx križanje je najzahtjevnije što se tiče vremenskog izvođenja i potrošnje energije što možemo također vidjeti kod grafa usporedbi na slikama 4.13, 4.14 i 4.15.

Tablica 4.5 Rezultati mjerenja kod roulette selekcije

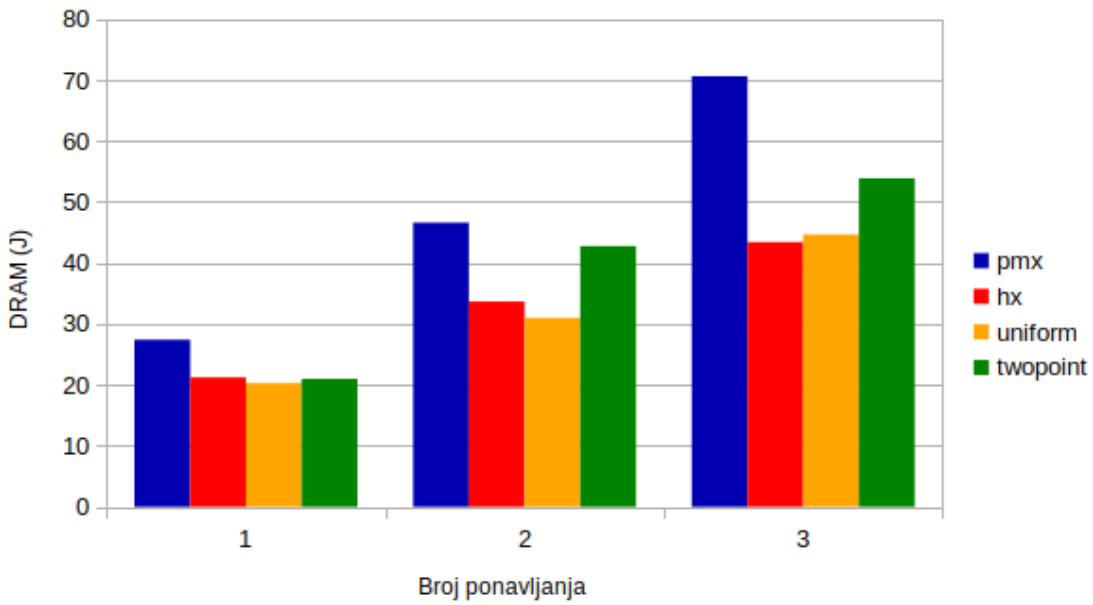
Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	40.74	216.91	27.41	eil51	pmx
1	26.80	148.30	21.23	eil51	hx
1	25.13	139.88	20.27	eil51	uniform
1	26.42	145.67	20.98	eil51	two-point
2	57.88	363.48	46.61	eil51	pmx
2	47.01	265.83	33.68	eil51	hx
2	42.53	242.35	30.98	eil51	uniform
2	46.05	292.39	42.78	eil51	two-point
3	84.58	533.81	70.63	eil51	pmx
3	63.50	361.21	43.42	eil51	hx
3	59.52	353.04	44.68	eil51	uniform
3	63.28	395.93	53.89	eil51	two-point



Slika 4.13 Graf usporedbe svih križanja prema vremenu



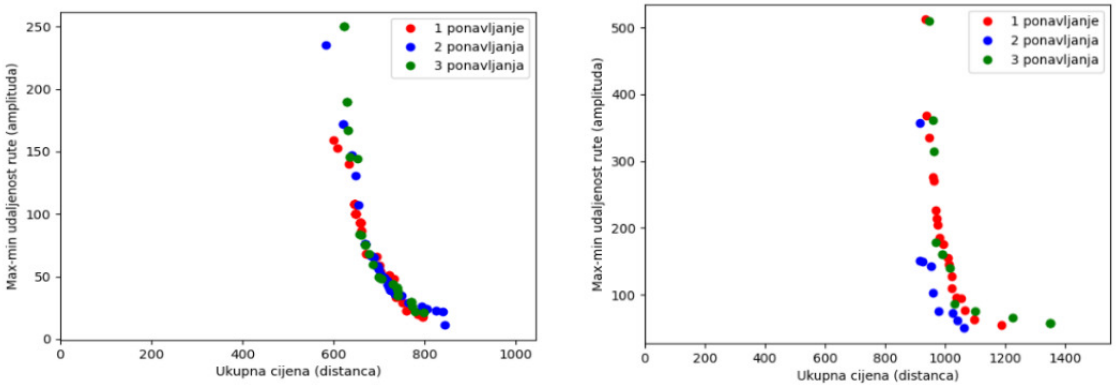
Slika 4.14 Graf usporedbe svih križanja prema PKG-u



Slika 4.15 Graf usporedbe svih križanja prema DRAM-u

4.3 Balansiranje iteriranja

Nakon dobivenih rezultata pokušano je balansiranje mjerenja tako što je za 1 ponavljanje stavljeno 300 iteracija, za 2 ponavljanja 150 iteracija te za 3 ponavljanja je stavljeno 100 iteracija. Iz rezultata na slici 4.16 i tablice 4.6 možemo vidjeti kako kod hx i pmx križanja dobivamo podjednake rezultate što je bilo i za očekivati pošto smo dali jednake uvjete za svako mjerenje te time možemo zaključiti da broj ponavljanja ne utječe na rezultate mjerenja ukoliko damo svakom ponavljanju jednake uvjete. Populacija je stavljena na 50 za svako ponavljanje.



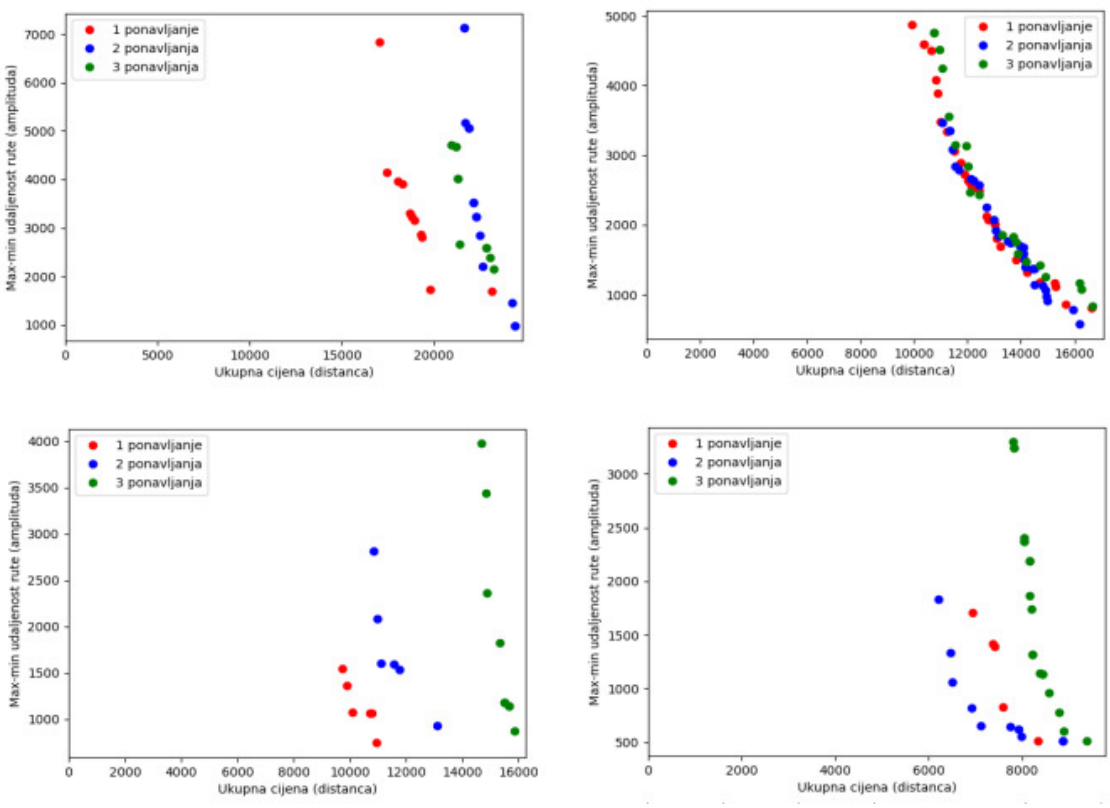
Slika 4.16 Prikaz hx(lijevo) i pmx(desno) križanja sa balansiranim uvjetima

Tablica 4.6 Rezultati mjerenja

Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	33.21	193.56	20.98	berlin52	pmx
1	25.55	151.29	20.08	berlin52	hx
2	33.33	190.20	23.21	berlin52	pmx
2	27.73	156.09	24.11	berlin52	hx
3	35.56	197.58	25.56	berlin52	pmx
3	28.45	159.11	22.00	berlin52	hx

4.4 Balansiranje populacije

Kako je u prethodnome djelu zaključeno kako balansiranjem iteriranja sa brojem ponavljanja donosi jednake rezultate, u ovom djelu je izbalansiran broj ponavljanja sa populacijom gdje za 1 ponavljanje je stavljena populacija 60, za 2 ponavljanja je stavljena 30 te za 3 ponavljanja 20. U ovom slučaju su se mjerenja kod hx križanja poklapala dok kod pmx, two-point te uniform križanja su imali drugačije rezultate te to možemo vidjeti na slici 4.17 dok mjerenja možemo vidjeti u tablici 4.7.



Slika 4.17 Prikaz pmx(gore lijevo), hx(gore desno), two-point(dolje lijevo) i uniform(dolje desno) križanja

Tablica 4.7 Rezultati mjerenja

Broj ponavljanja	Trajanje(s)	PKG(J)	DRAM(J)	mapa	križanje
1	21.71	120.81	18.31	berlin52	pmx
1	18.71	119.24	21.42	berlin52	hx
1	16.32	97.47	17.25	berlin52	uniform
1	15.34	84.87	14.79	berlin52	two-point
2	18.43	109.82	18.22	berlin52	pmx
2	15.14	85.83	15.09	berlin52	hx
2	13.72	74.47	13.71	berlin52	uniform
2	13.34	70.81	12.92	berlin52	two-point
3	15.95	91.56	15.81	berlin52	pmx
3	14.85	83.81	14.95	berlin52	hx
3	12.62	64.77	12.49	berlin52	uniform
3	12.73	66.25	12.63	berlin52	two-point

Poglavlje 5

Zaključak

U ovom završnom radu sam istražio NSGA II algoritam i primijenio ga kod problema trgovačkog putnika te zaključio da mijenjanjem raznih varijanti poput operatora mutacije, selekcije, populacije možemo dobiti drugačija rješenja te na nama je da odlučimo koje nam rješenje najviše odgovara.

Također ukoliko izbalansiramo određene dijelove mjerenja dobivati ćemo jednake rezultate neovisno o broju ponavljanja što se u jednu mjeru i treba očekivati. Programski jezik u kojemu izvodimo program je također bitan ukoliko želimo pripaziti na potrošnju energije i uštede vremena te u tom slučaju Python u kojemu sam ja izvodio svoj rad nije najefikasniji programski jezik.

NSGA II algoritam je učinkovit genetski algoritam za višekriterijsku optimizaciju te ukoliko se dobro implementira može se prilagoditi i primijeniti na različite probleme višekriterijske optimizacije u raznim područjima kao što su inženjerstvo, ekonomija, logistika, financije i drugi.

Bibliografija

[1] Wikipedia, "Machine learning — wikipedia, the free encyclopedia," 2023, [Online; accessed 8-April-2023]. , s Interneta, https://en.wikipedia.org/wiki/Machine_learning

[2] E. Osaba, X.-S. Yang, and J. Del Ser, "Chapter 9 - traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics," in *Nature-Inspired Computation and Swarm Intelligence*, X.-S. Yang, Ed. Academic Press, 2020, pp. 135–164. , s Interneta, https://www.sciencedirect.com/science/article/pii/B9780128197141000208

[3] GeeksforGeeks, "Proof that traveling salesman problem is np-hard," 2023, [Online; accessed 8-April-2023]. , s Interneta, https://www.geeksforgeeks.org/proof-that-traveling-salesman-problem-is-np-hard/

[4] Wikipedia, "Travelling salesman problem — Wikipedia, the free encyclopedia," 2023, [Online; accessed 8-May-2023]. , s Interneta, https://en.wikipedia.org/wiki/Travelling_salesman_problem

[5] Tutorialspoint, "Python overview — tutorialspoint," 2023, [Online; accessed 8-April-2023]. , s Interneta, https://www.tutorialspoint.com/python/python_overview.htm

[6] PyRAPL, "Pyrapl: Python interface for rapl power measurement," 2023, [Online; accessed 8-April-2023]. , s Interneta, https://pypi.org/project/pyRAPL/

[7] I. Corporation, "Running average power limit (rapl) energy reporting," 2023, [Online; accessed 8-April-2023]. , s Interneta, https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html

[8] J. Carr, "The traveling salesman problem," 2014, [Online; accessed 7-May-2023]. , s Interneta, https://www.whitman.edu/documents/academics/mathematics/2014/carrjk.pdf

[9] MathWorks, "What is the genetic algorithm? — MathWorks," 2023, [Online; accessed 8-May-2023]. , s Interneta, https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

[10] D. A. Coley, "An introduction to genetic algorithms for scientists and engineers," 2015, [Online; accessed 8-April-2023]. , s Interneta, http://ftp.demec.ufpr.br/CFD/bibliografia/an_introduction_to_genetic_algorithms_for_scientists_and_engineers_coley.pdf

[11] Towards Data Science, "Introduction to genetic algorithms (including example code) — Towards Data Science," 2023, [Online; accessed 8-May-2023]. , s Interneta, https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[12] J. Zhong, X. Hu, J. Zhang, and M. Gu, "Comparison of performance between different selection strategies on simple genetic algorithms." vol. 2, 01 2005, pp. 1115–1121.

[13] S. Banihashemian and F. Adibnia, "A novel robust soft-computed range-free localization algorithm against malicious anchor nodes," *Cognitive Computation*, vol. 13, 07 2021.

[14] W. Stankiewicz, R. Roszak, and M. Morzynski, "Genetic algorithm-based calibration of reduced order galerkin models," *Mathematical Modelling and Analysis*, vol. 16, pp. 233–247, 06 2011.

[15] A. Lahjouji El Idrissi, C. Tajani, and M. Sabbane, "New crossover operator for genetic algorithm to resolve the fixed charge transportation problem," *Journal of Theoretical and Applied Information Technology*, vol. 95, 05 2017.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," pp. 182–197, 2002.

[17] pymoo, "Nsga-ii," 2023, [Online; accessed 7-May-2023]. , s Interneta, https://pymoo.org/algorithms/moo/nsga2.html

[18] A. Otman, C. Tajani, and J. Abouchabaka, "Hybridizing psm and rsm operator for solving np-complete problems: Application to travelling salesman problem," *International Journal of Computer Science Issues*, vol. 9, 03 2012.

[19] M. Shokouhifar, A. Jalali, and H. Torfehnejad, "Optimal routing in traveling salesman problem using artificial bee colony and simulated annealing," 01 2015.

Sažetak

U ovom završnom radu istražena je učinkovitost algoritma NSGA II za višekriterijski problem trgovačkog putnika. Problemsko rješenje koje sam implementirao imalo je dva kriterija kroz koje je provodilo optimizaciju rute. Pomoću programskog jezika Python te softverskog alata pyRAPL mjerio sam potrošnju energije procesora kao i vrijeme potrebno za izvršenje programa. Kroz rad su također objašnjeni osnovni pojmovi vezani za korištene alate te za genetske algoritme.

Ključne riječi **— NSGA II, Genetski algoritmi, Problem trgovačkog putnika, Python, pyRAPL**

Abstract

In this thesis, the effectiveness of the NSGA II algorithm for the multi-objective traveling salesman problem was investigated. The problem solution I implemented had two criteria through which the route optimization was conducted. Using the Python programming language and the software toolkit pyRAPL, I measured the processor energy consumption as well as the time required for program execution. The paper also provides explanations of the basic concepts related to the tools used and genetic algorithms.

Keywords **— NSGA II, Genetic algorithms, Travelling salesman problem, Python, pyRAPL**