Optimiranje robotskih trajektorija primjenom biološki inspiriranih algoritama

Čehulić, Lovro

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:330089

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Lovro Čehulić

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Doc. dr. sc. Petar Ćurković, dipl. ing.

Student:

Lovro Čehulić

Zagreb, 2018.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svome mentoru, doc.dr.sc Petru Ćurkoviću na pružanoj podršci, potpori i savjetima tijekom izrade i pisanja ovog rada, ali i zato što me uveo u svijet evolucijskih algoritma. Bez njegovih predavanja iz umjetne inteligencije i mentorstva na završnom projektu preddiplomskog studija vjerojatno nikada ne bih niti saznao za njihovo postojanje.

Isto tako, zahvalio bih se svim prijateljima i kolegama koji su cijelo trajanje studija strojarstva bili uz mene i učinili ovo putovanje zabavnim.

Na kraju, zahvaljujem se svojoj obitelji što su mi omogućili studiranje sve ove godine. Bez njihove bezuvjetne podrške ne bih bio u mogućnosti završiti ovaj studij. Hvala Vam na svemu!

Lovro Čehulić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **LOVRO ČEHULIĆ** Mat. br.: 0035190265

Naslov rada na hrvatskom jeziku: **Optimiranje robotskih trajektorija primjenom biološki inspiriranih algoritama**

Naslov rada na engleskom jeziku: **Robot trajectory optimization based on bio-inspired algorithms**

Opis zadatka:

Planiranje kretanja referentne točke industrijskog robota u prostoru koji sadrži prepreke težak je optimizacijski problem kojega u općem slučaju nije moguće riješiti u polinomnom vremenu. Prirodom inspirirani algoritmi optimiranja pokazuju vrlo dobre rezultate u pronalaženju rješenja ovakvih složenih procesa.

U radu je potrebno osmisliti modul za planiranje temeljen na biološki inspiriranim algoritmima, koji za poznatu početnu i konačnu poziciju referentne točke i položaj prepreka u radnom prostoru, na izlazu daje trajektoriju optimiranu po kriteriju duljine, uz ograničenja zakrivljenosti i broja kolizija. Ispitati prednosti i nedostatke po dijelovima linearne trajektorije u odnosu na trajektoriju određenu polinomom općeg reda.

U drugom dijelu rada, potrebno je povezati algoritam za planiranje s fizičkim modelom robota te omogućiti kretanje robota po trajektorijama preuzetim s modula za planiranje.

U radu je potrebno navesti literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
27. rujna 2018.

Rok predaje rada:
29. studenog 2018.

Predviđeni datum obrane:
05. prosinca 2018.
06. prosinca 2018.
07. prosinca 2018.

Zadatak zadao:
doc. dr. sc. Petar Ćurković

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK V
SUMMARY VI
1. UVOD 1
1.1. Cilj i plan rada 1
1.2. Pregled poglavlja 1
2. EVOLUCIJSKI ALGORITMI 3
2.1. Razvoj 4
2.2. Inspiracije iz biologije 4
2.2.1. Teorija evolucije 4
2.2.2. Prirodna genetika 6
2.3. Razlike prirodne i umjetne evolucije 7
2.4. Osnovni elementi evolucijskih algoritama 8
2.4.1. Prikazivanje jedinki 11
2.4.2. Odabir roditelja 12
2.4.3. Rekombinacija 13
2.4.4. Mutacija 15
2.4.5. Zamjena populacije 16
2.5. Očuvanje raznolikosti 17
2.5.1. Fitness sharing 17
2.5.2. Crowding 18
3. OPTIMIRANJE PUTANJE UNUTAR ISPITNOG OKOLIŠA 20
3.1. Planiranje putanje 20
3.2. Linearna putanja po koordinatama 21
3.2.1. Rezultati testiranja algoritama 25
3.3. Polinomna putanja 29
3.3.1. Rezultati testiranja algoritama 35
4. SIMULACIJA PRAĆENJA TRAJEKTORIJE 41
4.1. Kinematički model robota rotacijske strukture 41
4.2. Simulacija praćenja putanje 46
5. ZAKLJUČAK 48
LITERATURA 50
PRILOZI 52
Prilog II: Matlab kodovi algoritama 53

POPIS SLIKA

Slika 1. Adaptivni krajolik s dva obilježja. 5
Slika 2. Pseudokod tipičnog evolucijskog algoritma. [4] 8
Slika 3. Dijagram toka opće sheme evolucijskog algoritma. [4] 11
Slika 4. Ruletno pravilo s četiri člana. [1] 12
Slika 5. Križanje u jednoj točki. 13
Slika 6. Križanje u više točaka, slučaj s dvije točke. 14
Slika 7. Ujednačeno križanje. 14
Slika 8. Aritmetičko križanje: $\alpha = 0{,}5$. 15
Slika 9. Nasumična mutacija. 15
Slika 10. Idealizirana distribucija populacije za fitness sharing. [4] 18
Slika 11. Idealizirana distribucija populacije za crowding. [4] 19
Slika 12. Grafički prikaz kodiranja genoma na kretnje. 22
Slika 13. Primjer otkrivene linearne putanje po koordinatama. 23
Slika 14. Rezultati testiranja osnovnog 2D algoritma za linearnu putanju. 26
Slika 15. Rezultati testiranja fitness sharing 2D algoritma za linearnu putanju. 26
Slika 16. Rezultati testiranja crowding 2D algoritma za linearnu putanju. 27
Slika 17. Rezultati testiranja osnovnog 3D algoritma za linearnu putanju. 28
Slika 18. Rezultati testiranja fitness sharing 3D algoritma za linearnu putanju. 28
Slika 19. Rezultati testiranja crowding 3D algoritma za linearnu putanju. 29
Slika 20. Primjer otkrivene polinomne putanje. 32
Slika 21. Ispitni okoliši s mogućim putanjama. 35
Slika 22. Prikaz prostorne putanje u xy i xz ravnini. 35
Slika 23. Rezultati testiranja osnovnog 2D algoritma za polinomnu putanju. 37
Slika 24. Rezultati testiranja fitness sharing 2D algoritma za polinomnu putanju. 37
Slika 25. Rezultati testiranja crowding 2D algoritma za polinomnu putanju. 38
Slika 26. Rezultati testiranja osnovnog 3D algoritma za polinomnu putanju. 38
Slika 27. Rezultati testiranja fitness sharing 3D algoritma za polinomnu putanju. 39
Slika 28. Rezultati testiranja crowding 3D algoritma za polinomnu putanju. 39
Slika 29. Robot RRR strukture (lijevo) i vezani koordinatni sustavi (desno). [21] 41
Slika 30. Opći prikaz RRR robota. 45
Slika 31. Prikaz praćenja putanje. 47

POPIS TABLICA

Tablica 1. Razlike između prirodne i umjetne evolucije. [4] 7
Tablica 2. Postavke i parametri algoritama za linearnu putanju. 25
Tablica 3. Rezultati testiranja algoritama za linearnu putanju. 29
Tablica 4. Postavke i parametri algoritama za polinomnu putanju. 36
Tablica 5. Rezultati testiranja algoritma za polinomnu putanju. 40

POPIS OZNAKA

Oznaka	Jedinica	Opis
a_i	-	i-ti koeficijent polinomne funkcije u xy koordinatnom sustavu
$\boldsymbol{A}$	-	Matrica homogenih transformacija
$\boldsymbol{A_i}$	-	Matrica prijelaza
b_i	-	i-ti koeficijent polinomne funkcije u xz koordinatnom sustavu
d	-	Udaljenost fitnesa dva pojedinca
$f1$	-	Nagrada za udaljenost dosegnute pozicije od kraja labirinta
$f2$	-	Nagrada za sudare s preprekama
F	-	Vrijednost fitnesa
F'	-	Prilagođena vrijednost fitnesa
L	-	Donja vrijednost domene problema
m	-	Broj članova populacije
n	-	Duljina genoma
p	-	Oznaka potomaka
pc	-	Faktor vjerojatnosti križanja
pm	-	Faktor vjerojatnosti mutacije
$\boldsymbol{PF}$	-	Matrica populacijske funkcije
q_i	-	Kut zakreta i-tog članka robota
r	-	Oznaka roditelja
rp	-	Kazna za vraćanje
$\mathbb{R}$	-	Skup realnih brojeva
sh	-	Funkcija dijeljenja
$\boldsymbol{T_i}$	-	Matrica prijenosa iz i-tog u nepokretni koordinatni sustav
U	-	Gornja vrijednost domene problema
v	-	Relativna vrijednost fitnesa
$\boldsymbol{X}$	-	Matrica diskretiziranog prostora u smjeru osi x
α	-	Faktor aritmetičkog križanja
β	-	Faktor intenziteta mutacije
γ	-	Kontrolni parametar funkcije dijeljenja
σ_{share}	-	Područje dijeljenja

Kratica	Opis
2D	Dvodimenzionalno
3D	Trodimenzionalno
DNA	Deoksiribonukleinska kiselina

SAŽETAK

Tema ovog diplomskog rada je izrada modula za planiranje kretanja koji se temelji na biološki inspiriranim algoritmima. Planirane trajektorije smještene su u 2D i 3D prostoru. Prostor je okoliš koji je korisnik definirao ovisno o željenom broju i položaju prepreka. Provode se različite metode kako bi se ispitao njihov utjecaj na performanse i na razinu konvergencije algoritma. Pokazano je da svaka verzija algoritma ima svoje prednosti i nedostatke. Za svaku verziju algoritma provodi se temeljita statistička analiza, a rezultati su dokumentirani i prezentirani u radu.

Ključne riječi: Evolucijski algoritmi; planiranje kretanja; umjetna inteligencija.

SUMMARY

The subject matter of this master's thesis is a design of a motion planning module based on a biologically inspired algorithm. Planned trajectories are placed in both 2D and 3D space. Space is a user-defined environment which contains an arbitrary number of placed obstacles. Different methods are implemented to examine their impact on the performance and the level of convergence of the algorithm. It is shown that each instance of the algorithm has its benefits and shortcomings. A meticulous statistical analysis is performed for each version of the algorithm, and the results are documented and presented in the thesis.

Key words: Evolutionary algorithms; motion planning; artificial intelligence.

1. UVOD

U svjetskoj, ali i domaćoj industriji već godinama traje trend sve veće primjene industrijskih robota kod proizvodnih i pomoćnih procesa. Uz sve širu primjenu industrijskih robota potrebno je iskoristiti njihove mogućnosti u što većoj mjeri kako bi se investicija u industrijskog robota isplatila, ali i kako bi se zadržala konkurentnost na tržištu. Jedan od načina na koji se može povećati učinkovitost industrijskih robota je optimizacija kretanja. Ako se za primjer uzme najjednostavniji „*pick and place*" problem s kojim se često susrećemo, a pogotovo kod paletiranja, postavlja se nekoliko pitanja: „Koliko visoko iznad predmeta hvatanja treba postaviti točku prilaska?", „Gdje u prostoru treba postaviti interpolacijsku točku?". Naravno, što su te točke postavljene bliže predmetu hvatanja i putanja robota između početne i krajnje pozicije bit će kraća, a time će i vrijeme trajanja ciklusa biti kraće. Upravljački sustavi modernih robota određuju putanju interpolacijom između tih točaka, ali ona ovisi o točkama koje smo sami odabrali. Ako uz to u radnom prostoru postoje određene prepreke, treba jako paziti na položaje odabranih točaka u prostoru kako ne bi došlo do štete uzrokovane sudarom. Planiranje kretanja referentne točke industrijskog robota u prostoru koji sadrži prepreke težak je optimizacijski problem.

1.1. Cilj i plan rada

Cilj ovog rada je osmisliti modul za planiranje, temeljen na biološki inspiriranim algoritmima, koji, za poznatu početnu i krajnju poziciju referentne točke i položaj prepreka u radnom prostoru, na izlazu daje trajektoriju optimiranu po kriteriju duljine.

Algoritmi će biti programirani u programskom okružju MATLAB. Prvo će se pronaći linearne trajektorije gibanja po koordinatama, nakon toga će se tražiti trajektorije koje su određene polinomom općeg reda. Na kraju će biti simulirano praćenje pronađene trajektorije.

1.2. Pregled poglavlja

Rad je strukturiran u sljedeća poglavlja:

- Prvo poglavlje daje kratki uvod u temu koja je obrađena u radu.
- Drugo poglavlje, *Evolucijski algoritmi,* ukratko prikazuje povijesni razvoj područja evolucijskog računarstva. Opisani su izvori inspiracije za nastajanje ovog područja

znanosti u kontekstu evolucije i njenih procesa. Nakon toga je opisan princip rada evolucijskih algoritama, pri čemu su objašnjeni svi njegovi osnovni dijelovi.

- Treće poglavlje, *Optimiranje putanje unutar ispitnog okoliša*, sadrži glavni zadatak ovog rada. Detaljno su opisani osmišljeni algoritmi i njihovi dijelovi te su prikazani rezultati ispitivanja istih algoritama na konkretnom problemu.
- Četvrto poglavlje, *Simulacija praćenja trajektorije*, sadrži drugi zadatak ovog rada. Osmišljeni algoritam za planiranje je povezan s modelom robota te je simulirano praćenje pronađenih trajektorija.
- Peto poglavlje daje zaključak obrađene teme, prikazani su i komentirani dobiveni rezultati te je predložen daljnji smjer razvoja teme.
- Na kraju je priložen popis korištene literature, a u prilogu se nalaze MATLAB kodovi algoritama.

2. EVOLUCIJSKI ALGORITMI

Trenutno poglavlje bit će posvećeno evolucijskim algoritmima, njihovoj povijesti te na kojem principu rade. Ovo poglavlje je već obrađeno u mojem završnom radu [1], a budući da je ovaj rad nastavak teme završnog rada, i samo poglavlje je preuzeto i prilagođeno trenutnoj temi.

Evolucijski algoritmi su grana evolucijskog računarstva koja se bavi proučavanjem računalnih metoda inspiriranih procesima i mehanizmima evolucije. Evolucijski algoritmi bave se istraživanjem računarskih sustava koji sliče pojednostavljenim inačicama procesa i mehanizama evolucije kako bi se postigli njihovi rezultati, odnosno kako bi se razvili prilagodljivi sustavi [2]. U procesima istraživanja prirodnih zakona u znanosti i proizvodnje korisnih proizvoda u inženjerstvu, često se susrećemo s određenim problemima. Od tih problema zanimljivi su nam problem optimizacije i problem učenja. Kod optimizacije želja nam je pronaći optimalno rješenje unutar domene varijabli. Problemi optimizacije i učenja često su međusobno povezani, jer ponekad je optimizacijski problem toliko složen da ne možemo izračunati rješenja za sve kombinacije varijabli. Tada možemo iskoristiti evolucijske algoritme koji će nam olakšati zadatak [3].

Izbor evolucije kao izvor inspiracije ne iznenađuje kada se u obzir uzme raznolikost biljnog i životinjskog svijeta i njihova prilagođenost za preživljavanje u okolišu u kojemu se nalaze. Osnovni princip rada evolucijskih algoritama proizlazi iz načina na koji evolucija rješava probleme, a zove se metoda pokušaja i pogrešaka [1], [4]. Može se navesti velik broj izuma koji su rezultat oponašanja prirode, oponašali smo ribe i izumili podmornice ili šišmiše kako bi izumili radar. Evoluciju različitih vrsta možemo gledati kao proces učenja, kako se prilagoditi svojem okolišu [3]. U prirodi, određeni okoliš sadrži populaciju individualnih jedinki koje teže preživljavanju i razmnožavanju. Sposobnost ili fitnes ovih jedinki određen je okolišem i ovisi o tome koliko su uspješne u postizanju svojih ciljeva, odnosno predstavlja njihovu vjerojatnost za preživljavanje i razmnožavanje. U kontekstu stohastičke metode pokušaja i pogrešaka kao načina rješavanja problema, imamo skupinu mogućih rješenja. Njihova kvaliteta, odnosno sposobnost rješavanja problema, određuje kolika će biti vjerojatnost da ih zadržimo i iskoristimo u procesu dobivanja novih, poželjno još boljih, mogućih rješenja [4].

2.1. Razvoj

Ideja primjene Darwinovih principa u automatskom rješavanju problema datira iz četrdesetih godina prošlog stoljeća, odnosno prije proboja računala. 1948. godine Turing je predložio „genetsko ili evolucijsko traženje“, a do 1962. godine Bremermann je izveo računalne pokuse na „optimizaciji kroz evoluciju i križanje“. Od tada pa do devedesetih godina znanstvenici su razvijali različite implementacije osnovne ideje: evolucijsko programiranje, Fogel, Owens i Walsh [5]; genetski algoritam, Holland [6]; strategije evolucije, Rechenberg i Schwefel [7]; genetsko programiranje, Koza [8], koje su kasnije ujedinjene u područje evolucijskog računarstva. Svi algoritmi koji pripadaju području evolucijskog računarstva jednim nazivom zovu se evolucijski algoritmi, a navedena područja smatraju se različitim verzijama unutar istog područja [4].

Otkad je 1985. godine održana prva međunarodna konferencija posvećena evolucijskim algoritmima (ICGA), održane su brojne međunarodne konferencije, pokrenuti su časopisi te su izdane tisuće članaka i radova, sve na temu evolucijskog računarstva [4].

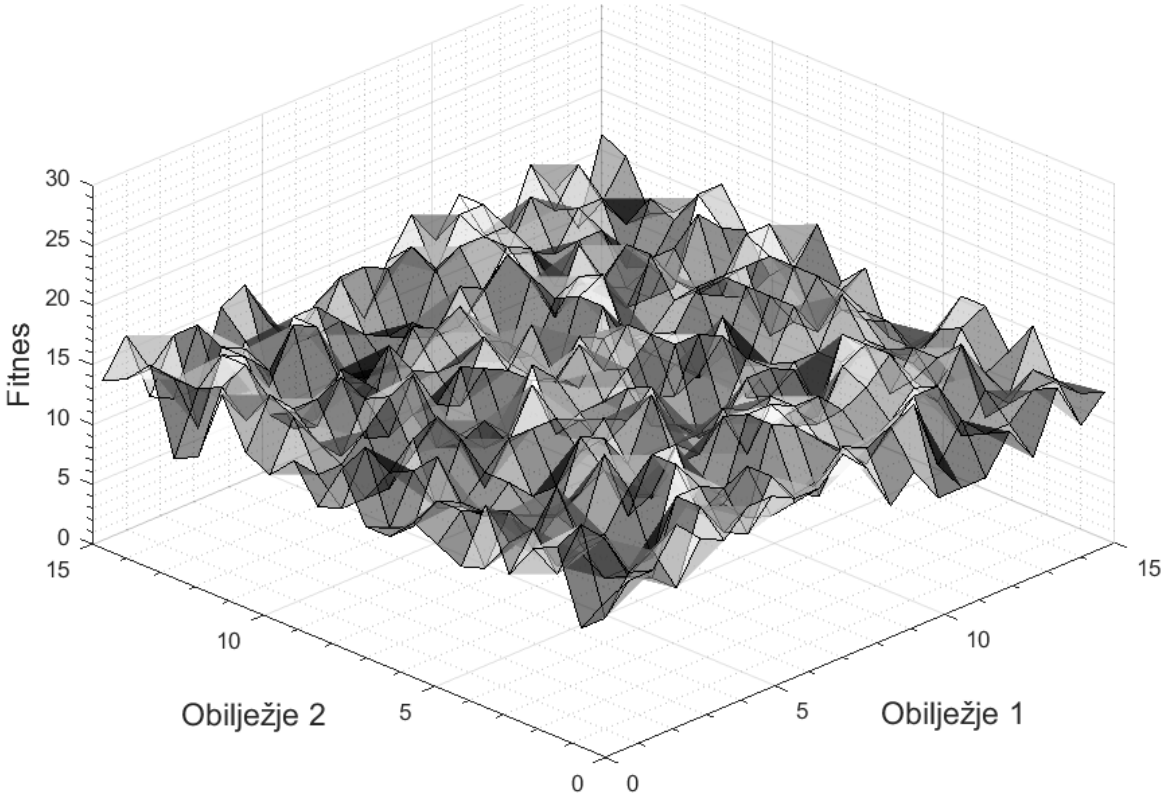
2.2. Inspiracije iz biologije

2.2.1. Teorija evolucije

Darwinova teorija evolucije [9] nudi nam objašnjenje o podrijetlu biološke raznolikosti i osnovnih mehanizama evolucije. U makroskopskom pogledu na evoluciju, glavnu ulogu ima prirodna selekcija. Svaka pojedina okolina sadrži ograničenu količinu resursa što znači da može opskrbljivati ograničen broj individualnih jedinki. Kako je osnovni instinkt svih živih bića usmjeren prema reprodukciji, selekcija postaje neizbježna budući da populacija ne može rasti eksponencijalno. Preživjet će jedino oni članovi populacije koji su najefikasniji u borbi za dostupne resurse, odnosno oni koji su najbolje prilagođeni uvjetima okoliša u kojemu se nalaze. Ovaj fenomen poznat je pod nazivom preživljavanje najsposobnijih i predstavlja jedan od dva osnovna principa evolucije. Darwin predlaže i drugi princip koji proizlazi iz različitih fenotipskih obilježja između članova populacije. Fenotipska obilježja su one osobine fizičkog izgleda i ponašanja članova populacije koje direktno utječu na odziv prema okolini i drugim članovima, što određuje njihov fitnes. Svaki član populacije sadrži jedinstvenu kombinaciju fenotipskih obilježja koju ocjenjuje okolina. U slučaju pozitivne ocjene rezultiraju povećanom šansom za reprodukciju, u suprotnom pojedinac se odbacuje i umire bez potomaka. Neka fenotipska obilježja su nasljedna tako da se ona pozitivna mogu prenijeti na sljedeću generaciju.

Darwin je primijetio da se prilikom reprodukcije mogu dogoditi male, nasumične varijacije odnosno mutacije u fenotipskim obilježjima potomaka. Kroz te varijacije dolazi do novih kombinacija obilježja koja se ponovno ocjenjuju te se najbolja dalje reproduciraju i preživljavaju. Kako proces evolucije napreduje, dolazi do potpune promjene u strukturi populacije, što znači da je populacija jedinica evolucije [1], [4], [9].

Ovaj proces opisan je metaforom adaptivnog krajolika ili adaptivne površine. U prikazanom adaptivnom krajoliku (Slika 1), dimenzija visine predstavlja fitnes: više uzvišenje predstavlja bolji fitnes. U ovom slučaju, druge dvije dimenzije predstavljaju biološke osobine. Općenito, adaptivni krajolik može prikazati populaciju s n obilježja u $n + 1$ dimenzionalnom prostoru pri čemu se jedna dimenzija odnosi na fitnes, a ostale na biološke osobine. Populacija je skup točaka u krajoliku, a svaka jedinka predstavlja jednu kombinaciju obilježja, odnosno jednu točku na krajoliku. Evolucija je proces postepenog pomaka populacije prema područjima višeg fitnesa, što ovisi o mutacijama i prirodnoj selekciji [1], [4].



Slika 1. Adaptivni krajolik s dva obilježja.

Povezanost evolucije i optimizacijskih problema nije jednostavna kao što se na prvi pogled čini zato što evolucija nije proces koji uvijek ide u istom smjeru. Budući da populacija ima ograničenu veličinu, a prilikom selekcije i mutacije se događaju nasumični odabiri, pojavljuje se fenomen genetskog drifta. Ovaj fenomen predstavlja situaciju u kojoj populacija gubi raznolikost određenih obilježja i nestanu članovi s visokim fitnesom. Pri tome se može dogoditi da cijela populacija napusti lokalne optimume i uđe u područja niskog fitnesa. Kombinacija selekcije i globalnih posljedica genetskog drifta omogućava populaciji da se pomiče i uzbrdo i nizbrdo, ali ne postoji jamstvo da će se populacija vratiti na isti lokalni optimum, što znači da postoji mogućnost pronalaska globalnog optimuma [4].

2.2.2. Prirodna genetika

Proces evolucije se može opisati na mikroskopskoj razini zahvaljujući molekularnoj genetici koja nam objašnjava procese koji su podloga vidljivim fenotipskim obilježjima, a koji su povezani s nasljedstvom. Osnovno opažanje iz genetike je da svaki pojedinac ima vanjska odnosno fenotipska svojstva koja su predstavljena na unutrašnjoj odnosno genotipskoj razini. Drugim riječima, genotip nekog pojedinca kodira njegov fenotip, čime geni postaju jedinice nasljeđa koje određuju fenotipska svojstva. Geni utječu na fenotipska svojstva složenim preslikavanjem, jedan gen može utjecati na više fenotipskih obilježja (pleiotropija), ali i više gena može utjecati na jedno fenotipsko obilježje (poligenija). Varijacije u fenotipskim obilježjima (npr. visina, boja kose, boja očiju) su uvijek uzrokovane varijacijama gena koje su nadalje posljedica mutacija gena ili križanja gena prilikom reprodukcije. Pojam genom odnosi se na cjelokupni genetski materijal neke jedinke. Ovaj genetski materijal, odnosno svi geni u organizmu, nalazi se u dijelovima DNA koji se nazivaju kromosomi. Ljudske stanice sadrže 46 kromosoma koji su raspoređeni u 23 para, 23 kromosoma dobijemo od oca, a 23 kromosoma dobijemo od majke te oni zajedno definiraju naša fizička svojstva. Mejozom stanica dolazi do križanja kromosoma pri čemu se par kromosoma duplicira te se jedan od nova dva para križa, čime dobijemo jedan par koji je kopija kromosoma od oca i majke i jedan novi par kromosoma s potpuno novim svojstvima. Povremeno se pojedini dijelovi genetskog koda promjene, čime dolazi do novih dijelova gena koji nisu naslijeđeni od roditelja. Ovaj proces nazivamo mutacija, a njegova posljedica može biti: negativna zbog čega potomak nije sposoban za život; neutralna čime nova obilježja nemaju nikakav utjecaj; pozitivna gdje dolazi do dobrih obilježja [1], [4].

Darwinova teorija evolucije i saznanja iz genetike mogu se povezati kako bi se razjasnila dinamika iza raznolikosti života na Zemlji. Svako živo biće sadrži svoj nevidljivi genetski kod,

genotip, i vidljive vanjske značajke, fenotip. Njegova vjerojatnost za preživljavanje i razmnožavanje ovisi o njegovim fenotipskim obilježjima, na primjer: dobar sluh, oštri zubi krzno prilagođeno okolišu, socijalno ponašanje, itd.. Pri tome je ključna reprodukcija koja je uvjetovana prirodnom selekcijom i odabirom partnera ovisno o fenotipskim obilježjima. Naravno pri tome implicitno utječe i na razina genotipa. Novi pojedinci mogu imati jednog roditelja (aseksualna reprodukcija) ili dva roditelja (seksualna reprodukcija). U svakom slučaju, genom novog pojedinca nije identičan njegovim roditeljima zbog malih varijacija uzrokovanih kombiniranjem genoma. Na ovaj način dolazi do raznolikosti u genotipu koje dovode do raznolikosti u fenotipu. Zbog tih razlika potomci se fizički razlikuju od svojih roditelja. Geni se mogu smatrati predmetom preživljavanja i evolucije. Tako da se, umjesto borbe unutar populacije pojedinaca, može razmatrati natjecanje genetskog materijala populacije, koje se odvija i ocjenjuje unutar različitih pojedinaca [4], [10].

2.3. Razlike prirodne i umjetne evolucije

Tablica 1. Razlike između prirodne i umjetne evolucije. [4]

	Prirodna evolucija	**Umjetna evolucija**
Fitnes	Promatrana veličina čiji se utjecaj na selekciju vidi tek nakon nje.	Preddefinirana veličina koja vodi postupak selekcije.
Selekcija	Složen višeparametarski proces koji ovisi o uvjetima okoliša, drugim jedinkama iste populacije i ostalih populacija. Sposobnost se testira kontinuirano, a reproduktivnost u diskretnom vremenu.	Nasumični operator kojemu je vjerojatnost odabira ovisna o vrijednosti fitnesa. Odabir roditelja i zamjena populacije odvijaju se u diskretnom vremenu.
Kodiranje	Vrlo složen biološki proces na koji utječe okoliš.	Jednostavna matematička transformacija parametara.
Varijacija	Potomci su dobiveni od jednog ili dva roditelja	Potomci se mogu generirati od jednog, dva ili više roditelja
Smrtnost	Paralelno izvođenje, događaji rođenja i smrti nisu sinkronizirani.	Tipično se odumiranje stare populacije i stvaranje nove odvija sinkronizirano.
Populacija	Prostorno strukturirane populacije. Veličina populacije mijenja se ovisno o relativnom broju smrti i rođenja.	Općenito nestrukturirana, veličina populacije se održava konstantnom sinkronizacijom broja novih i odbačenih članova.

Razvoj evolucijskog računarstva može se smatrati velikim prijelazom principa evolucije iz domene biologije u domenu računala. Korištenjem računala kao instrumenta za izradu virtualnih okružja, mogu se napraviti puno fleksibilnija i upravljivija od stvarnog svijeta u kojem živimo. Kada se tome pridoda sve dublje razumijevanje genetskih mehanizama koji su podloga evoluciji, ljudi su počeli aktivno konstruirati i izvršavati vlastite evolucijske procese [4]. Može se reći da evolucijski algoritmi nisu vjerni modeli prirodne evolucije, ali ipak jesu određeni oblik evolucije. (Tablica 1) prikazuje usporedbu prirodne evolucije i umjetne evolucije koja se koristi u evolucijskim algoritmima.

2.4. Osnovni elementi evolucijskih algoritama

Evolucijski algoritmi dijele svojstva prilagodbe kroz iterativne procese koji akumuliraju i proširuju pozitivne varijacije kroz metodu pokušaja i pogrešaka. Moguća rješenja predstavljaju članove virtualne populacije koja teži preživljavanju u okolišu koji je definiran problemom. Mehanizmi evolucijskih algoritama oponašaju procese evolucije, koristeći zamjenske procese kao što su genetska rekombinacija (križanje) i mutacija [2].

Kao što je već spomenuto, postoji nekoliko različitih verzija evolucijskih algoritama. Zajednička ideja iza svih različitih vrsta je jednaka: zadana populacija pojedinaca unutar istog okoliša s ograničenim resursima natječe se za te resurse što uzrokuje selekciju. Natjecanje nadalje uzrokuje rast fitnesa populacije. Generalna shema evolucijskog algoritma prikazana je pseudokodom, (Slika 2) i dijagramom toka, (Slika 3).

```
POČETAK
    INICIJALIZIRAJ populaciju slučajnih kandidata;
    OCIJENI svakog kandidata;
    PONAVLJAJ DO (UVJET PREKIDA ZADOVOLJEN)
        1 IZABERI roditelje;
        2 REKOMBINIRAJ parove roditelja;
        3 MUTIRAJ potomke;
        4 OCIJENI nove kandidate;
        5 IZABERI jedinke za novu generaciju;
    prekid
KRAJ
```

Slika 2. Pseudokod tipičnog evolucijskog algoritma. [4]

Evolucijski algoritmi sadrže određene osnovne dijelove, procedure i operatore koji ih definiraju i kao takvi moraju biti točno specificirani:

1. Prikazivanje jedinki: prvi korak prilikom izrade algoritma definira način na koji će fenotip članova populacije biti prikazan u algoritmu u obliku genotipa. Fenotipski prostor se nalazi na strani konteksta problema dok se na strani evolucijskog algoritma nalazi njegov genotipski prikaz. Genotip mora biti specificiran na način kojim računalo može manipulirati. Na primjer, ako se fenotipska značajka poput boje definira u RGB zapisu, taj zapis je genotip iste boje. U ovom smislu, prikazivanje je sinonim za kodiranje, prebacivanje fenotipa u genotip, a inverzno mapiranje, odnosno prebacivanje genotipa u fenotip zove se dekodiranje.

2. Funkcija procjene kvalitete (fitnes funkcija): predstavlja zahtjeve okoline koje populacija mora zadovoljiti i zbog koje se populacija mora prilagoditi. Fitnes funkcija stvara osnovu za selekciju i omogućava napredak, odnosno definira što je to napredak. Ova funkcija svakom genotipu dodjeljuje jedinstvenu kvalitativnu mjeru kvalitete (fitnes) koja se mjeri u fenotipskom prostoru.

3. Populacija: zadaća populacije je čuvati genotipski prikaz pojedinaca koji predstavljaju moguća rješenja. Pojedinci su statički objekti koji se ne mijenjaju, ono što se mijenja i prilagođava je populacija te je zato ona jedinica evolucije. U većini evolucijskih algoritama, populacija je konstantne veličine, a njezina raznolikost je mjera koja prikazuje broj različitih fitnesa i genotipa prisutnih u njoj.

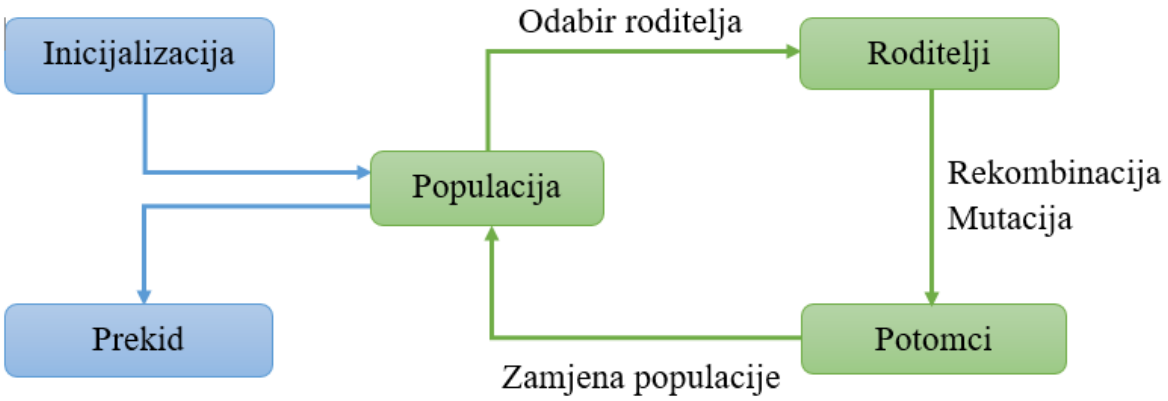
4. Mehanizam selekcije (odabir roditelja): uloga ovog mehanizma je izdvojiti pojedine članove populacije ovisno o njihovom fitnesu i omogućiti boljim pojedincima da postanu roditelji sljedećoj generaciji. Odabir roditelja je jedan od mehanizama koji su zaduženi za poboljšanje kvalitete populacije. Odabir roditelja se obično temelji na vjerojatnosti pri čemu kvalitetniji pojedinci imaju veću vjerojatnost da postanu roditelji, ali čak i najlošiji član ima malu vjerojatnost, inače bi populacija mogla zapeti u lokalnom optimumu.

5. Operatori varijacije: služe kako bi stvorili nove članove populacije iz starih članova. Operatori varijacije kod evolucijskih algoritama dijele se u dvije vrste: operator rekombinacije ili križanja i operator mutacije. Rekombinacija spaja genotipske značajke

roditelja kako bi se stvorila dva potomka koji posjeduju značajke svojih roditelja. Rekombinacija je stohastički operator što znači da su kombinirani dijelovi gena odabrani nasumično. Mutacija se primjenjuje na jednom genotipu te ga (malo) mijenja stvarajući nove značajke koje prije nisu postojale. Operator mutacije je uvijek stohastički i stvara nasumičnu nepristranu promjenu.

6. Mehanizam zamjene populacije: određuje koji će članovi populacije biti uključeni u sljedećoj generaciji ovisno o njihovoj kvaliteti. Za razliku od odabira roditelja koji je stohastički proces, mehanizam zamjene je gotovo uvijek deterministički proces. Zamjena je uglavnom bazirana na vrijednosti fitnesa članova pri čemu se favoriziraju oni članovi s boljim fitnesom. Zamjena može biti isključivo ovisna o fitnesu gdje se odabiru najbolji članovi roditelja i potomaka, ili može ovisiti i o starosti gdje se uzimaju samo najbolji potomci.

Uz prethodno navedene osnovne dijelove algoritma, potrebno je navesti i inicijalizaciju algoritma i uvijete prekida. Inicijalizacija algoritma se uglavnom zadržava jednostavnom, prva populacija odabire se nasumično unutar zadanih granica, pri čemu geni moraju biti ravnomjerno raspoređeni i pomiješani. Ponekad se mogu iskoristiti određene heurističke metode kojima će početna populacija imati nešto veći fitnes. Razlikujemo dva različita slučaja prekida algoritma. Ako naš problem ima poznato optimalno rješenje, tada bi u idealnoj situaciji izvođenje algoritma prekinuli kada se to rješenje dosegne. Međutim, evolucijski algoritmi su stohastički i kada rješavamo realne probleme ne postoji garancija da ćemo doseći takav optimum, tako da se taj uvjet nikada neće doseći i algoritam možda nikada ne stane. Zato je potrebno proširiti uvjet prekida kako bi bili sigurni da će algoritam stati. Iz tog razloga uvjet prekida se izvršava kada: protječe maksimalno dozvoljeno vrijeme; algoritam dosegne zadani broj generacija; raznovrsnost genoma padne ispod određene granice; vrijednost fitnesa ne ostvari poboljšanje unutar određenog broja generacija [1, 4, 11].



Slika 3. Dijagram toka opće sheme evolucijskog algoritma. [4]

2.4.1. Prikazivanje jedinki

Prvi korak kod izrade evolucijskog algoritma je odlučivanje o prikazu genotipa populacije. Ovaj korak uključuje definiranje genotipa i mapiranje između genotipa i fenotipa. Kod odabira načina prikazivanja, potrebno je odabrati onaj najprikladniji trenutnom problemu. Odabir prikladnog načina prikaza prilikom konstruiranja genetskog algoritma je jedan od najtežih koraka [1, 4, 11].

U algoritmima koji su korišteni u izradi ovog rada, upotrijebljene su dvije vrste prikaza. Prvi je prikazivanje cijelim vrijednostima, dok je drugi prikazivanje realnim vrijednostima.

Kod prikazivanja cijelim vrijednostima genotip se sastoji od niza cijelih brojeva te jedan broj predstavlja jedan gen. Za svaku primjenu treba odlučiti koliko će taj niz brojeva biti dugačak i kako će se interpretirati kako bi se dobio fenotip. Kod odabira mapiranja iz genotipskog u fenotipski prostor i obratno, potrebno je pobrinuti se da svaka moguća kombinacija daje valjanu vrijednost u fenotipskom prostoru [4, 11].

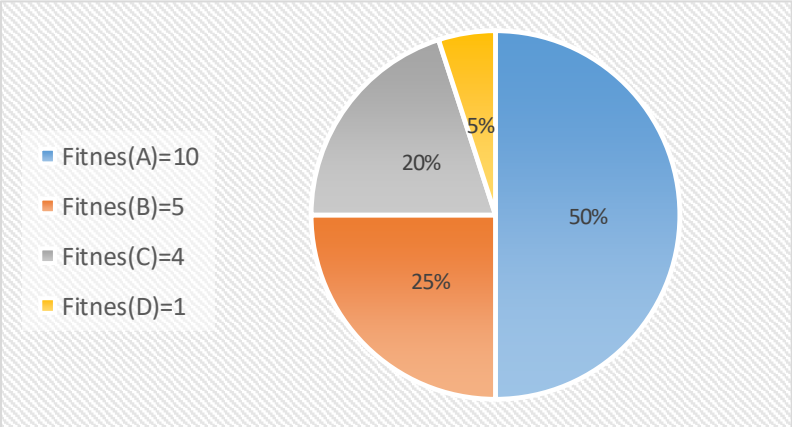
Najčešće je najprigodniji način prikazivanja korištenjem niza realnih vrijednosti. Pri tome se misli na situacije u kojima vrijednosti koje želimo prikazati dolaze iz kontinuiranog, umjesto diskretnog skupa. Kao i kod prikazivanja cijelim vrijednostima, potrebno se odlučiti za duljinu genoma i za način mapiranja. Kod prikazivanja realnim vrijednostima genom rješenja s n gena postaje vektor $\langle x_1, \dots, x_n \rangle$ gdje je $x_i \in \mathbb{R}$ [4, 11].

2.4.2. Odabir roditelja

Nakon inicijalizacije početne populacije, algoritam ulazi u svoju glavnu petlju te počinje proces odabira roditelja. U procesu odabira kopira se prirodna selekcija tako što se onim članovima koji imaju bolji fitnes daje veća vjerojatnost za razmnožavanje [3]. Ako odabir roditelja vršimo proporcionalno vrijednosti fitnesa članova, tada oni najbolji pojedinci brzo prevladaju cijelom populacijom. Ovakva situacija može dovesti do fokusiranja procesa traženja rješenja i manje je vjerojatno da će algoritam pretražiti cijeli prostor mogućih rješenja. Situacija u kojoj algoritam zapne u lokalnom optimumu, zbog izbacivanja članova s lošijim fitnesom u ranim stadijima algoritma, naziva se preuranjena konvergencija. Kako bi spriječili tu pojavu, umjesto da strogo odabiremo najbolje članove, potrebno je implementirati određene probabilističke metode odabira članova [4]. Kao što je ranije spomenuto, bolji pojedinci imaju prednosti kod odabira roditelja. Definira se relativna vrijednost fitnesa v za svakog pojedinca i kao što slijedi:

$$v_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{2.1}$$

pri čemu se fitnes i-tog člana F_i dijeli sa sumom svih fitnesa u populaciji koja sadrži n članova. Jednostavno se provjeri da suma svih relativnih fitnesa iznosi 1, što znači da se relativni fitnes člana može smatrati njegovom vjerojatnošću da će biti odabran za roditelja. Dobivenu vjerojatnost možemo implementirati u algoritam korištenjem ruletnog pravila [3]. (Slika 4) prikazuje nam primjer u kojemu se fitnes četiri člana preslikao u njihove vjerojatnosti za odabir na krugu ruleta.



Slika 4. Ruletno pravilo s četiri člana. [1]

Odabir ruletnim pravilom implementira se u algoritam tako da se rotacija ruleta simulira procesom akumulacije relativnih fitnesa članova. Odabere se nasumični realni broj iz intervala $\langle 0,1 \rangle$, nakon toga se traži pojedinac koji zadovoljava sljedeću nejednakost:

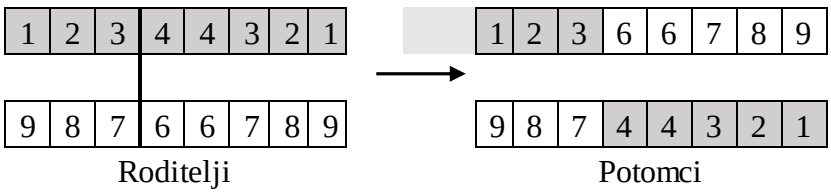
$$rand < \sum_{i=1}^{k} v_i \qquad \textbf{(2.2)}$$

pri čemu je $k \leq n$. Kada se pojavi pojedinac koji zadovoljava prethodnu nejednakost odabire se novi nasumični broj te se ovaj proces ponavlja dok se ne odabere zadani broj roditelja. Na ovaj način, neki pojedinci će biti odabrani više puta dok neki uopće neće biti odabrani [3].

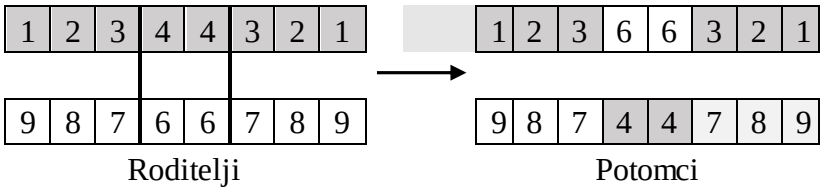
2.4.3. Rekombinacija

Rekombinacija ili križanje je operator varijacije kod kojega se odvija zamjena gena između dva pojedinca. Postoje dva načina kako odabrati koja će se dva pojedinca iz populacije roditelja križati. Prvi način je da se iz populacije redom u parove združe prvi i drugi član, treći i četvrti, itd. Drugi način je da se parovi slože nasumičnom permutacijom članova. Uobičajeno je da se uvede faktor vjerojatnosti križanja pc, kako bi imali kontrolu nad učestalosti križanja [3]. Faktor vjerojatnosti križanja određuje korisnik, te se on uspoređuje s nasumičnim vrijednosti iz intervala [0,1]. Ako je nasumična vrijednost manja od faktora vjerojatnosti križanja doći će do križanja, u suprotnom roditelji će se bez izmjene proglasiti potomcima [12].

Kod križanja cjelobrojnih vrijednosti općenito se koriste tri standardna načina križanja. Kod svih se od dva roditelja stvaraju dva potomka, iako postoje iznimne situacije kada se koristi više roditelja ili se dobije jedan potomak. Prvi način je križanje u jednoj točki [6, 13] gdje se odabere nasumični broj iz intervala $[1, l-1]$ (l je duljina gena), zatim se genomi roditelja razdvoje u toj točki te im se zamijene repovi.



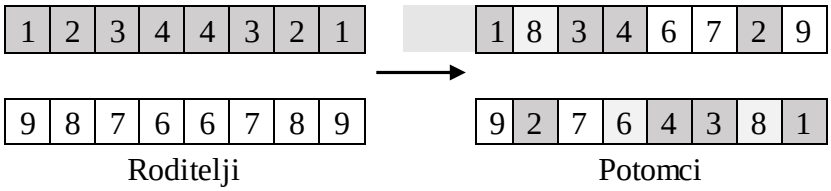
Slika 5. Križanje u jednoj točki.



Slika 6. Križanje u više točaka, slučaj s dvije točke.

Drugi način je križanje u više točaka. Ovaj način je generalizirani prikaz križanja u jednoj točki, pri čemu se genom razdvoji na više od dva odsječka kontinuiranih gena. Potomci se dobiju naizmjeničnom zamjenom odsječaka [4].

U prethodna dva načina roditelji su podijeljeni u nekoliko dijelova kontinuiranih gena nakon čega su geni presloženi kako bi se dobili potomci. Za razliku od toga, kod ujednačenog križanja [14] svaki se gen gleda zasebno te se korištenjem faktora vjerojatnosti križanja odredi od kojeg će roditelja potomak naslijediti pojedini gen.

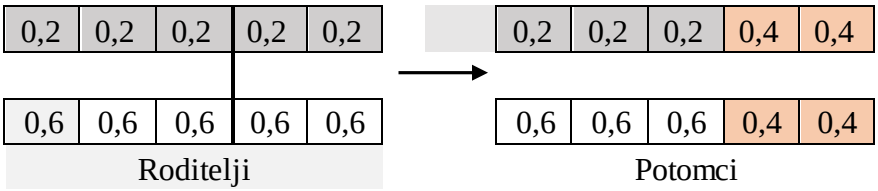


Slika 7. Ujednačeno križanje.

Kod križanja genoma koji sadrže realne vrijednosti jedan od mogućih operatora je aritmetičko križanje [15]. Ovaj operator na svakom odabranom genu unutar genoma stvara novu vrijednost koja je nastala kombiniranjem roditeljskih vrijednost prema formuli:

$$z_i = \alpha x_i + (1 - \alpha) y_i \qquad \textbf{(2.3)}$$

pri čemu je z_i i-ti gen potomka, x_i i y_i su i-ti geni roditelja, a α je faktor aritmetičkog križanja čiju vrijednost odabire korisnik iz intervala $[0,1]$. Iako se na ovaj način križanjem dobije novi genetski materijal, nedostatak je to što se kao rezultat dobije uprosječena vrijednost gena [3, 4, 15].

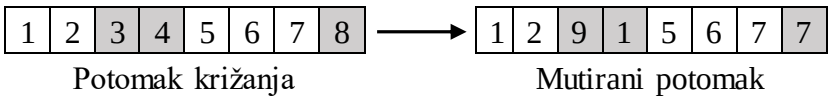


Slika 8. Aritmetičko križanje: $\alpha = 0,5$.

2.4.4. Mutacija

Mutacija je operator varijacije kod kojega dolazi do samostalne promjene određenih gena unutar genoma pojedinca. Mutacija se odvija direktno na potomcima dobivenim procesom križanja. Slično kao i kod križanja, kod mutacije se uvodi faktor vjerojatnosti mutacije pm koji je uglavnom manji od faktora vjerojatnosti križanja pc [3, 4]. Način mutacije odabire se ovisno o načinu prikazivanja genoma jedinki.

Kod mutacije genoma koji sadrži cjelobrojne vrijednosti najprikladnija metoda je nasumična mutacija. Za svaki odabrani gen, ovisno o faktoru vjerojatnosti mutacije, odabire se nova nasumična vrijednost iz domene dozvoljenih vrijednosti. Druga moguća metoda određenim genima, ovisno o faktoru vjerojatnosti mutacije, dodaje malu (pozitivnu ili negativnu) vrijednost. Ovaj način mutacije uvodi manje promjene od nasumične mutacije, a potrebno je kontrolirati veći broj parametara zbog domene dozvoljenih vrijednosti. Iz tog razloga je teško pronaći situacije za koje je prikladna [3, 4, 11, 12].



Slika 9. Nasumična mutacija.

Kod mutiranja genoma koji sadrži realne vrijednosti, geni se nasumično mijenjaju unutar intervala ograničenog donjom L i gornjom U vrijednosti domene problema, što za genotip s n gena rezultira sljedećom transformacijom:

$$\langle x_1, \dots, x_n \rangle \rightarrow \langle x'_1, \dots, x'_n \rangle, \qquad gdje\ su: x_i, x'_i \in [L, U] \tag{2.4}$$

Operator kod kojega se vrijednosti x'_i odabiru nasumično iz intervala $[L, U]$, zove se uniformna mutacija. Ovo je najizravnija opcija, koja je analogna nasumičnoj mutaciji kod cjelobrojnog kodiranja. Kao i kod cjelobrojnog kodiranja, koristi se uz faktor vjerojatnosti mutacije. Uz uniformnu mutaciju koristi se i neuniformna mutacija. Najčešći oblik neuniformne mutacije za genome s realnim vrijednostima sličan je drugoj navedenoj metodi za mutiranje genoma s cjelobrojnim vrijednostima. Ovaj operator, ovisno o faktoru vjerojatnosti mutacije, genu dodaje malu nasumičnu vrijednost [4].

2.4.5. Zamjena populacije

Proces zamjene populacije odvije se nakon procesa križanja i mutacije, odnosno nakon što su od roditelja dobiveni potomci. Kao što je ranije bilo rečeno, u prirodi okoliš ima dovoljno resursa samo za ograničen broj jedinki, tako da je veličina populacije konstantna. Zbog toga treba donijeti odluku o tome koji će članovi iz skupina roditelja i potomaka biti dio nove populacije. Strategije zamjene mogu se podijeliti ovisno da li diskriminiraju po kriteriju starosti ili kriteriju fitnesa [4, 11].

Metode zamjene ovisno o starosti, prilikom odabira članova ne uzimaju u obzir njihov fitnes. Umjesto toga, ove metode osmišljene su tako da svaki pojedinac postoji u populaciji jednak broj iteracija algoritma. Kako je broj roditelja jednak broju potomaka, svaki član postoji samo jednu generaciju te se svi roditelji odbacuju i zamjenjuju cijelom generacijom potomaka. Ovo je generalni model, ali postoje varijacije kod kojih je broj potomaka manji od broja roditelja pa se zamjenjuju ili nasumični roditelji ili oni najstariji [4, 11].

Postoji velik broj metoda zamjene koje ovisno o fitnesu odlučuju koji će se roditelji i potomci prenijeti u sljedeću generaciju, a neke čak uključuju i starost kao dodatni kriterij odabira. Jedna od mogućih metoda je mehanizam turnira u kojem se roditelji i djeca poslože u parove i natječu ovisno o tome koji član ima veći fitnes. Oni koji pobijede prelaze u sljedeću generaciju. Elitizam je metoda koja se često koristi u kombinaciji s metodama zamjene ovisno o starosti, kako bi se u populaciji zadržali članovi s najboljim fitnesom [4]. Ova metoda je potrebna ako želimo jamstvo da će nam najbolji članovi preživjeti iz generacije u generaciju. Populacije roditelja i potomaka se sortiraju ovisno o fitnesu te se željeni broj najboljih članova zadržava unutar populacije [12]. U nekim od korištenih algoritmima koristi se ova metoda uz zadržavanje jednog člana s najboljim fitnesom, elitni član se zadržava iz generacije u generaciju dok se jedan nasumični potomak odbacuje.

2.5. Očuvanje raznolikosti

Ideja očuvanja raznolikosti je prisiliti populaciju da zadržava raznolikost dok se obavljaju odabir roditelja i zamjena članova. Dvije najčešće korištene metode su fitness sharing [16], gdje se vrijednost fitnesa svakog člana prilagodi prije odabira roditelja kako bi se članovi razdvojili u skupine te se odabir odvija u ovisnosti o fitnesu skupine, i crowding [13], gdje su članovi raspoređeni ravnomjerno među skupinama korištenjem selekcije bazirane na udaljenosti između vrijednosti članova. Potrebno je naglasiti da obje metode koriste globalni odabir roditelja i ništa ne sprječava rekombinaciju roditelja iz različitih skupina. Obje metode pripadaju općem pristupu očuvanja raznolikosti koji je od krajnje važnosti algoritmima inspiriranim evolucijom zbog potreba uspješnog rješavanja zahtjevnih optimizacijskih zadataka [17, 18].

2.5.1. Fitness sharing

Ova metoda se zasniva na ideji da dijeljenje vrijednosti fitnesa pojedinaca prije odabira roditelja kontrolira broj pojedinaca u svakoj skupini [16]. Kod ove metode razmatra se svaki mogući par pojedinaca i te j, a nakon toga se računa udaljenost između njihovih vrijednosti. Vrijednost fitnesa F svakog pojedinca i se nakon toga prilagođava ovisno o broju pojedinaca koji su udaljeni od njega za neku pred definiranu udaljenost σ_{share} koristeći power-law distribuciju:

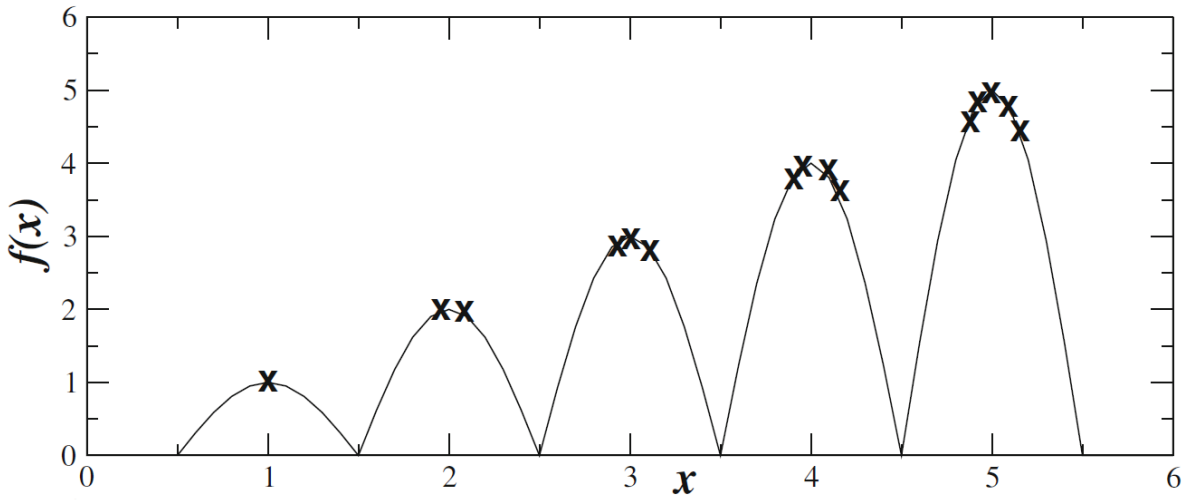
$$F'(i) = \frac{F(i)}{\sum_j sh(d(i,j))} \tag{2.5}$$

$$sh(d) = \begin{cases} 1 - \left(d/\sigma_{share}\right)^{\gamma} & ako\ je\ d \leq \sigma_{share}, \\ 0 & inače. \end{cases} \tag{2.6}$$

Funkcija dijeljenja $sh(d)$ je funkcija udaljenosti d. Kontrolni parametar funkcije dijeljenja γ određuje oblik funkcije dijeljenja i odabrana je vrijednost $\gamma = 1$ kako bi se funkcija napravila linearnom. Zadnji parametar je područje dijeljenja σ_{share} koje definira koliko će postojati skupina. Preporučeno je da vrijednost σ_{share} bude u rasponu 5 do 10 [19], u korištenim algoritmima koristi se vrijednost 10 kako bi se dobila veća raznolikost. Iako se preferira korištenje udaljenosti d kao fenotipske vrijednosti [4], metodom pokušaja i pogrešaka

zaključeno je da se korištenjem razlike između fitnesa pojedinaca kao udaljenosti među njima dobivaju bolji rezultati i brža konvergencija algoritma [18].

(Slika 10) pokazuje okoliš s pet uzvišenja čije su visine vrijednosti fitnesa (1,2,3,4,5), a veličina populacije iznosi 15 članova. Fitness sharing raspoređuje pojedince u vrhove u odnosu na vrijednost njihovog fitnesa. Za ovakav idealizirani slučaj, uz korištenje jednadžbe (2.5) dobije se da je prilagođena vrijednost fitnesa svakog pojedinca jednaka 1, čime imaju jednaku vjerojatnost odabira. Nedostatak metode je parametar područja dijeljenja σ_{share} koji je teško odrediti za stvarne probleme i pretpostavlja se jednakim za svakog člana u okolišu, što nije realno za stvarne probleme [3].



Slika 10. Idealizirana distribucija populacije za fitness sharing. [4]

Korištenjem fitness sharing metode, članovi s manjom vrijednosti fitnesa imaju veću mogućnost da postanu roditelji nego što je to slučaj kod osnovnog algoritma. Ovako imamo veću količinu gena što direktno povećava raznolikost populacije [18].

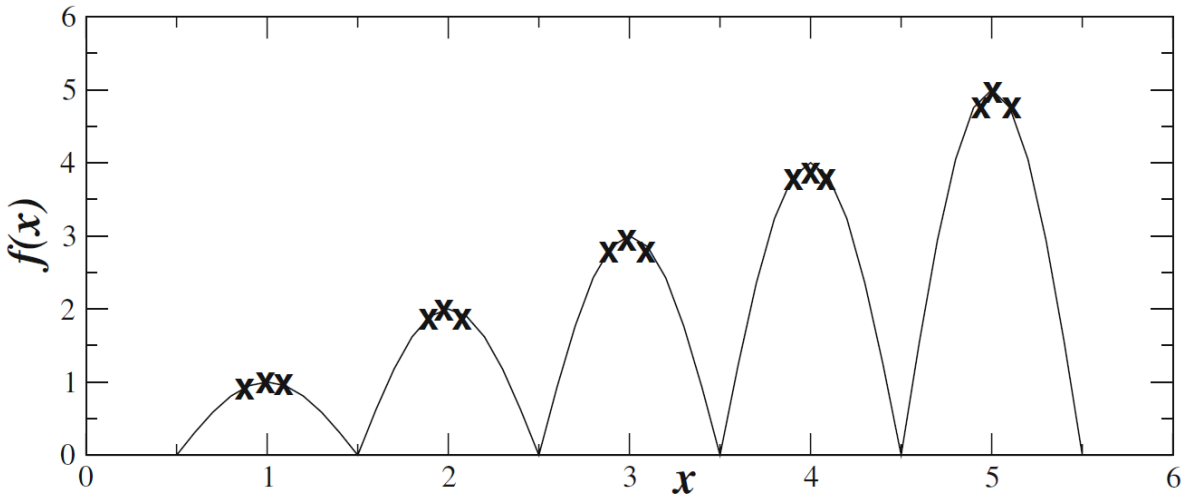
2.5.2. Crowding

Ova metoda se zasniva na činjenici da će potomci vjerojatno biti slični svojim roditeljima. U ovoj metodi, populacija roditelja se ocjenjuje te se nakon toga nasumično uparuje neovisno o vrijednosti fitnesa. Svaki par roditelja proizvede par potomaka rekombinacijom, nakon čega potomci mutiraju te se ocjenjuju. Ako roditelje označimo slovom r, a potomke slovom p, potrebno je izračunati udaljenosti $d(r_1,p_1)$, $d(r_1,p_2)$, $d(r_2,p_1)$ i $d(r_2,p_2)$ između roditelja i

potomaka. Kao i kod fitnes sharing-a, udaljenosti se mogu računati pomoću fenotipskih ili genotipskih vrijednosti. Nakon toga se odabiru parovi za usporedbu i natjecanje:

$$[d(r_1, p_1) + d(r_2, p_2)] \leq [d(r_1, p_2) + d(r_2, p_1)] \quad \textbf{(2.7)}$$

Ako je nejednakost (2.7) istinita, odvija se natjecanje između parova $r_1 \leftrightarrow p_1$ i $r_2 \leftrightarrow p_2$, u suprotnom se odvija natjecanje između parova $r_1 \leftrightarrow p_2$ i $r_2 \leftrightarrow p_1$. Ne postoji natjecanje između potomaka, potomci se natječu samo s najsličnijim roditeljem. Nakon natjecanja, pojedinci s većom vrijednosti fitnesa ostaju u populaciji dok se oni s nižom vrijednosti odbacuju [4, 13, 18].



Slika 11. Idealizirana distribucija populacije za crowding. [4]

(Slika 11) pokazuje okoliš s pet uzvišenja čije su visine vrijednosti fitnesa (1,2,3,4,5), a veličina populacije iznosi 15 članova. Crowding raspoređuje populaciju ravnomjerno između uzvišenja.

3. OPTIMIRANJE PUTANJE UNUTAR ISPITNOG OKOLIŠA

Riječ „*optimum*“ dolazi iz latinskog jezika a znači „*najbolji“*. Stoga, optimirati znači dovesti problem s kojim se bavimo u najbolje moguće stanje. U ovom poglavlju cilj je upravo to, pronaći najbolju moguću putanju unutar okoliša ispunjenog preprekama. Metoda traženja navedenih putanja je pomoću evolucijskog algoritma koje je opisan u prethodnom poglavlju. Tražit će se dvije različite putanje, jedna je linearna putanja po koordinatama, dok je druga definirana polinomom općeg reda. Okoliš može biti dvodimenzionalan ili trodimenzionalan prostor, a prepreke su zadane proizvoljno. Ispitat će se tri verzije algoritma, osnovna verzija te dvije koje koriste fitness sharing i crowding metode očuvanja raznolikosti. Time se dobiva 12 različitih algoritama čiji su rezultati prikazani i statistički obrađeni.

3.1. Planiranje putanje

Planiranje je pojam koji ima različito značenje u različitim područjima. Robotika primjenjuje automatizaciju na mehaničke sustave koji imaju senzorske, aktuatorske i računalne mogućnosti, odnosno autonomne sustave. Osnovna potreba u robotici su algoritmi koji pretvaraju zadatke s visokim zahtjevima na tehničke podatke u jednostavne naredbe kako se kretati. Za ovakve probleme često se koriste izrazi *planiranje kretanja* i *planiranje putanja*. Planiranje putanje obično se odnosi na problem kod kojega se uzima rješenje dobiveno od algoritma za planiranje gibanja, nakon čega se utvrđuje kako će se pratiti dobiveno rješenje poštivajući mehanička ograničenja robota. Kod umjetne inteligencije, pojam planiranja ima malo drugačije značenje te se odnosi na zadatak poput rješavanja zagonetke ili rješavanje diskretno postavljenog problema. Iako se takvi problemi mogu riješiti u kontinuiranom prostoru, prirodnije je definirati ograničeni skup radnji koje se mogu primijeniti na diskretni skup stanja, kako bi se konstruiralo rješenje zadavanjem prikladnog niza radnji [20].

Algoritmi za planiranje imaju široku primjenu te se koriste u nekoliko različitih industrijskih i akademskih disciplina koje uključuju: robotiku, proizvodnju, svemirsku tehnologiju i mnoge druge. Navigiranje mobilnih robota je klasičan zadatak pri kojemu se od robota traži snalaženje u zatvorenom prostoru. Robotu se može zadati mapiranje prostora određivanje njegove točne lokacije na karti ili dolazak na određeno mjesto. Prikupljanje i obrada podataka sa senzora može biti jako izazovna te većina robota funkcionira unatoč velikoj

nesigurnosti. Još jedna od mogućih primjena algoritama za planiranje je navigacija autonomnih letjelica u prostoru ispunjenom preprekama [20].

Teme koje se bave planiranjem sadrže nekoliko osnovnih dijelova koji se pojavljuju u gotovo svim primjenama:

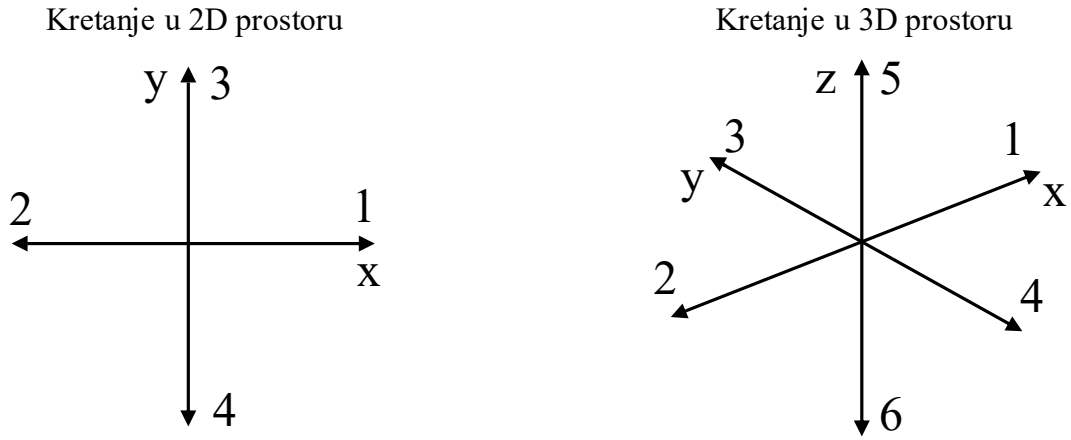
1. Stanje: Planiranje uključuje prostor stanja koji može predstavljati poziciju i orijentaciju industrijskog robota ili poziciju i brzinu letjelice. Koriste se i diskretni i kontinuirani prostori stanja. Zajednička pojava je ta da algoritam za planiranje prostor stanja prikazuje implicitno zato što je u većini slučajeva prevelik da bi se mogao prikazati eksplicitno.
2. Vrijeme: Svi algoritmi za planiranje uključuju niz odluka koje se eksplicitno događaju u vremenu. Alternativno, vrijeme može biti implicitno što odražava da se radnje moraju odvijati uzastopno. Vrijeme može biti diskretno ili kontinuirano.
3. Radnje: Plan generira radnje koje manipuliraju stanjem. Termini radnje i operatori su česti kod umjetne inteligencije, dok se u robotici koriste srodni izrazi ulazi i kontrole. Potrebno je specificirati kako se korištenjem radnji mijenja stanje, što se može izraziti funkcijom vrijednosti stanja za diskretno vrijeme ili običnom diferencijalnom jednadžbom za kontinuirano vrijeme.
4. Početno i krajnje stanje: Problem planiranja uključuje neko početno stanje, a radnje se odabiru kako bi se dostiglo specificirano krajnje stanje.
5. Kriterij: Kriterij uključuje željeni rezultat planiranja u okviru stanja i odrađenih radnji. Kriterij može biti baziran na izvedivosti, gdje se traži dolazak u željeno stanje neovisno o efikasnosti, i optimalnosti, gdje se traži izvediv plan koji je optimiziran po odabranom kriteriju.
6. Plan: Općenito, plan nameće specifičnu strategiju ili ponašanje. Plan može jednostavno odrediti niz potrebnih radnji, ali može biti i kompliciraniji. Ako je nemoguće predvidjeti buduća stanja onda plan određuje radnje kao funkciju stanja. U ovom slučaju, neovisno o budućim stanjima, određene su prikladne radnje [20].

3.2. Linearna putanja po koordinatama

Trenutno potpoglavlje rezultat je mojeg projektnog zadatka koji je kasnije i objavljen u sklopu DAAAM međunarodne konferencije [18]. Budući da se ostatak rada nadovezuje na temu obrađenu u projektom zadatku, ovo potpoglavlje je preuzeto i prilagođeno trenutnoj temi.

Kako bi se ispitao utjecaj metoda za očuvanje raznolikosti na učinkovitost algoritma, napravljene su i uspoređene tri verzije algoritma. Prva je osnovni evolucijski algoritam dok ostale dvije koriste fitness sharing, odnosno crowding metode kontrole raznolikosti populacije. Svi elementi algoritama su identični kako bi rezultati bili usporedivi, jedine razlike među algoritmima proizlaze iz različitih metoda očuvanja raznolikosti populacije. Osim toga, svi algoritmi su testirani na istim labirintima, s jednakim rasporedom prepreka te jednakim početnim i krajnjim pozicijama [18].

Populacija je predstavljena matricama, (3.1) i (3.2), gdje m predstavlja broj pojedinaca u populaciji, a n predstavlja duljinu genoma. Veličina populacije m postavljena je na 50 članova u svakom algoritmu, dok je duljina genoma n postavljena na 10 što predstavlja potreban broj koraka od početka do kraja labirinta bez sudara s preprekama.



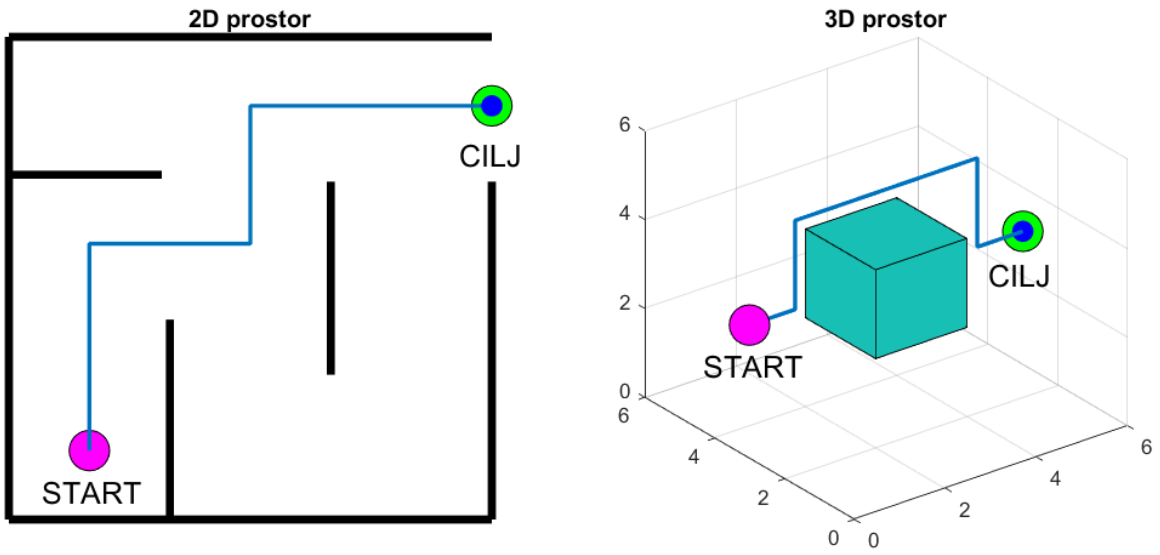
Slika 12. Grafički prikaz kodiranja genoma na kretnje.

Za 2D labirint, genom je sastavljen od nasumično generiranih cijelih brojeva u rasponu od 1 do 4, (3.1). Svaki broj predstavlja različito gibanje kroz labirint, gdje 1 i 2 predstavljaju gibanja u pozitivnom i negativnom smjeru x osi, dok su 3 i 4 gibanja u pozitivnom i negativnom smjeru y osi.

$$\boldsymbol{populacija} = \begin{bmatrix} rand(4)_{11} & rand(4)_{12} & \cdots & rand(4)_{1n} \\ rand(4)_{21} & rand(4)_{22} & \cdots & rand(4)_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ rand(4)_{m1} & rand(4)_{m2} & \cdots & rand(4)_{mn} \end{bmatrix} \quad \textbf{(3.1)}$$

$$\boldsymbol{populacija} = \begin{bmatrix} rand(6)_{11} & rand(6)_{12} & \cdots & rand(6)_{1n} \\ rand(6)_{21} & rand(6)_{22} & \cdots & rand(6)_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ rand(6)_{m1} & rand(6)_{m2} & \cdots & rand(6)_{mn} \end{bmatrix} \quad \textbf{(3.2)}$$

Za 3D labirint, genom sadrži dodatne brojeve 5 i 6, oni opisuju gibanje u pozitivnom i negativnom smjeru z osi (3.2).



Slika 13. Primjer otkrivene linearne putanje po koordinatama.

(Slika 13) prikazuje jedan primjer od nekoliko mogućih putanja za koje ne postoje kolizije za 2D i za 3D prostor. U početnim fazama odvijanja algoritma, velik broj članova populacije ne dosegne cilj, a kasnije kada ga dosegnu najčešće postoje kolizije. Zato se svaki član i populacije ocjenjuje i uspoređuje, koristeći njegovu vrijednost fitnesa F. Fitnes F računa se pomoću fitnes funkcije (3.3), ovisno o fenotipskim svojstvima članova. Vrijednost fitnesa ovisi o udaljenosti dosegnute pozicije od kraja labirinta i broju kolizija, uz to dodaje se mala kazna rp onim članovima koji su se vratili na poziciju na kojoj su se već prethodno nalazili [18].

$$F(i) = f1(i) + f2(i) - rp(i) \quad \textbf{(3.3)}$$

$$f1 = \frac{60 + udaljenost(i)}{1 + udaljenost(i)} \tag{3.4}$$

Jednadžba (3.4) opisuje nagradu $f1$ za udaljenost dostignute pozicije od kraja labirinta. Što je udaljenost manja, nagrada je veća do maksimalne nagrade od 60 ako je udaljenost 0.

$$f2(i) = 40 - \frac{40 \cdot (broj\ sudara)}{n} + \begin{cases} 0 \\ \frac{40}{n} \cdot \frac{prvi\ korak\ bez\ sudara}{n} \end{cases} \tag{3.5}$$

Jednadžba (3.5) prikazuje kako se nagrađuju sudari s preprekama, pojedinci koji se sudare s manjim brojem prepreka dobit će veću nagradu, a uz to, oni pojedinci koji se sudare s preprekom u kasnijem koraku dobit će bolju nagradu. Maksimalna vrijednost $f2$ je 40, čime najveća moguća vrijednost fitnesa F pojedinca može biti 100. Kazna uslijed vraćanja rp, (3.6), dodana je zato što je primijećeno kako ubrzava rad algoritma [18].

$$rp(i) = \begin{cases} 0, & ako\ nema\ vraćanja, \\ 10, & ako\ ima\ vraćanja. \end{cases} \tag{3.6}$$

Što se tiče odabira roditelja, kod osnovnog algoritma koristi se ruletno pravilo s vjerojatnošću odabira koja je jednaka relativnoj vrijednosti fitnesa, (2.1), na način kako je opisano u prethodnom poglavlju. Algoritam koji koristi fitness sharing metodu ima drugačiji način odabira roditelja. Također se koristi ruletno pravilo, ali umjesto relativne vrijednosti fitnesa, svaki član ima vjerojatnost odabira koja je jednaka njegovoj prilagođenoj vrijednosti fitnesa (2.5). Za crowding algoritam, prilikom odabira roditelja ne koristi se ruletno pravilo nego se iz populacije roditelja svaki član nasumično odabire i upari s drugim nasumičnim članom.

Svi algoritmi koriste standardni operator rekombinacije za cjelobrojne vrijednosti s križanjem u jednoj točki i operator nasumične mutacije.

Kod osnovnog i fitness sharing algoritma, odabir nove populacije odvija se prema starosti, odnosno potomci zamjenjuju roditelje pri čemu se koristi elitizam, odabire se jedan elitni član koji se zadržava u populaciji. U usporedbi s prijašnja dva algoritma, crowding algoritam koristi drugačiji model odabira zamjene populacije. Kod crowding metode, odabir nove populacije je

prema vrijednosti fitnesa umjesto prema starosti, odvija se turnir između roditelja i djece kako je to i opisano u prethodnom poglavlju.

Svaki algoritam je ograničen na 500 generacija. Ako cilj nije dosegnut unutar 500 generacija algoritam se zaustavlja i smatra se da je bio neuspješan iako je potencijalno mogao doseći cilj u nekoj od kasnijih generacija.

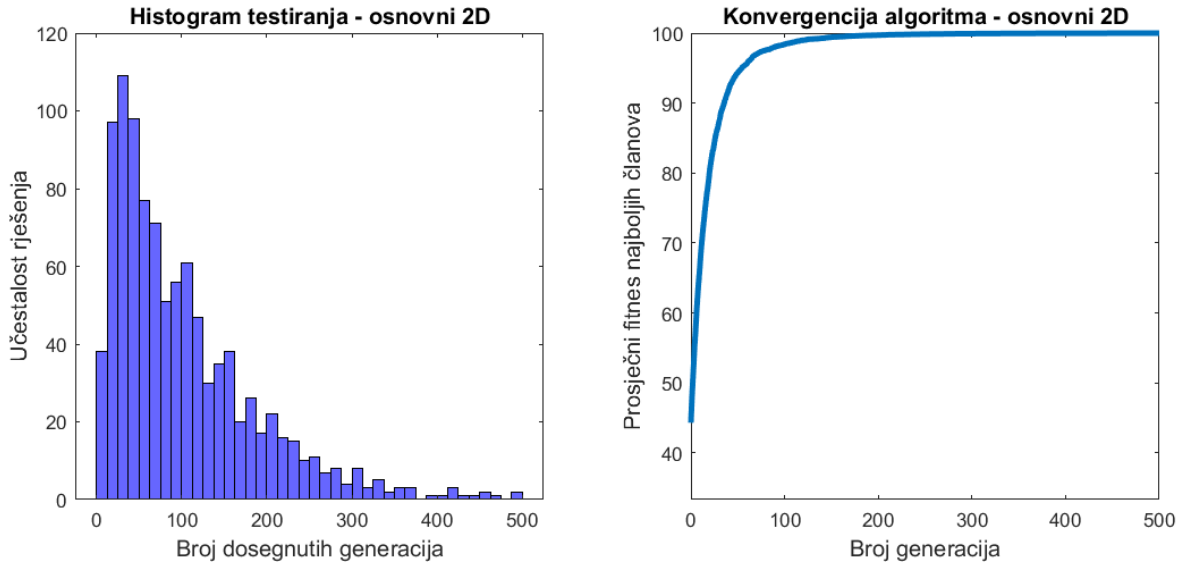
3.2.1. Rezultati testiranja algoritama

Potrebno je naglasiti kako su svi algoritmi, simulacije i rezultati napravljeni i prikazani programom MATLAB, verzija R2017b. Kako bi se usporedile različite verzije algoritama, svaki algoritam je simuliran 1000 puta, a rezultati su prikazani uz pomoć histograma. Proces simuliranja svih 6 algoritama trajao je 38 minuta.

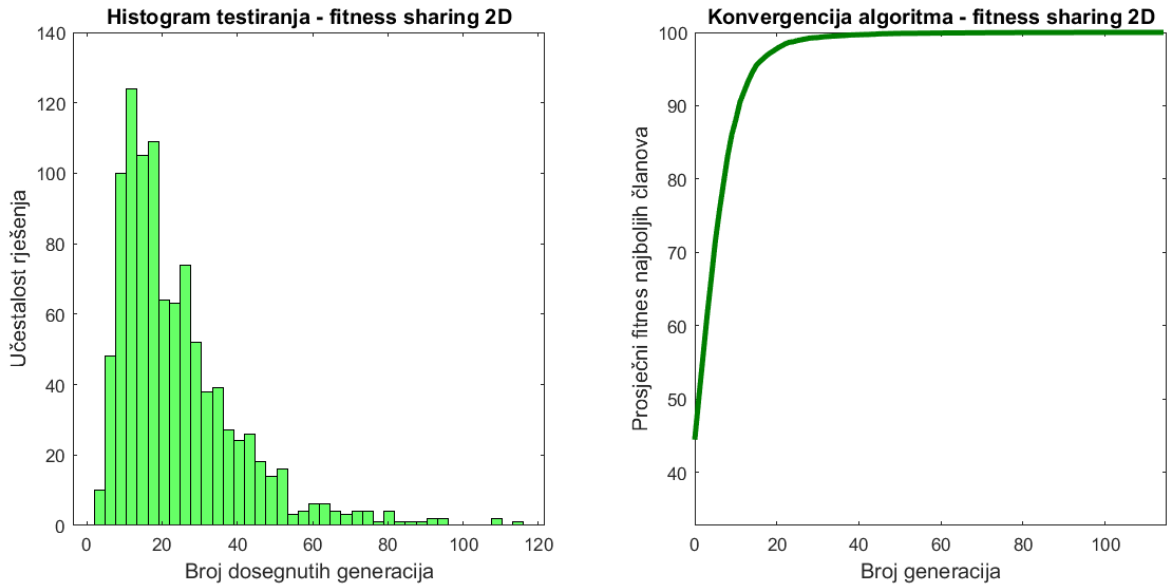
Tablica 2. Postavke i parametri algoritama za linearnu putanju.

Algoritam	**Osnovni**	**Fitness sharing**	**Crowding**
Prikazivanje jedinki	Prikazivanje cijelim brojevima		
Rekombinacija	Križanje u jednoj točki, vjerojatnost križanja $pc = 0{,}7$		
Mutacija	Nasumična mutacija, vjerojatnost mutacije $pm = 0{,}4$		
Odabir roditelja	Ruletno pravilo	Fitness sharing	Nasumično uparivanje
Zamjena populacije	Ovisno o starosti uz elitizam		Ovisno o fitnesu
Veličina populacije	50 članova		
Inicijalizacija	Početna populacija je odabrana nasumično		
Uvjet prekida	Dosegnut maksimalni fitnes ili dosegnuto 500 generacija		

Svaki histogram prikazuje koliko je puta algoritam pronašao rješenje u određenoj generaciji ili koliko je puta zaustavljen zbog toga što je dosegao generacijsko ograničenje. Visina stupaca histograma predstavlja broj simulacija u kojima je algoritam zaustavljen u generacijskom intervalu koji odgovara širini istog stupca. Uz histogram, prikazana je i prosječna konvergencija fitnesa najboljih članova kroz generacije. Prosjek je izračunat iz svih 1000 simulacija pri čemu su uzete u obzir i one simulacije u kojima nije pronađeno rješenje.



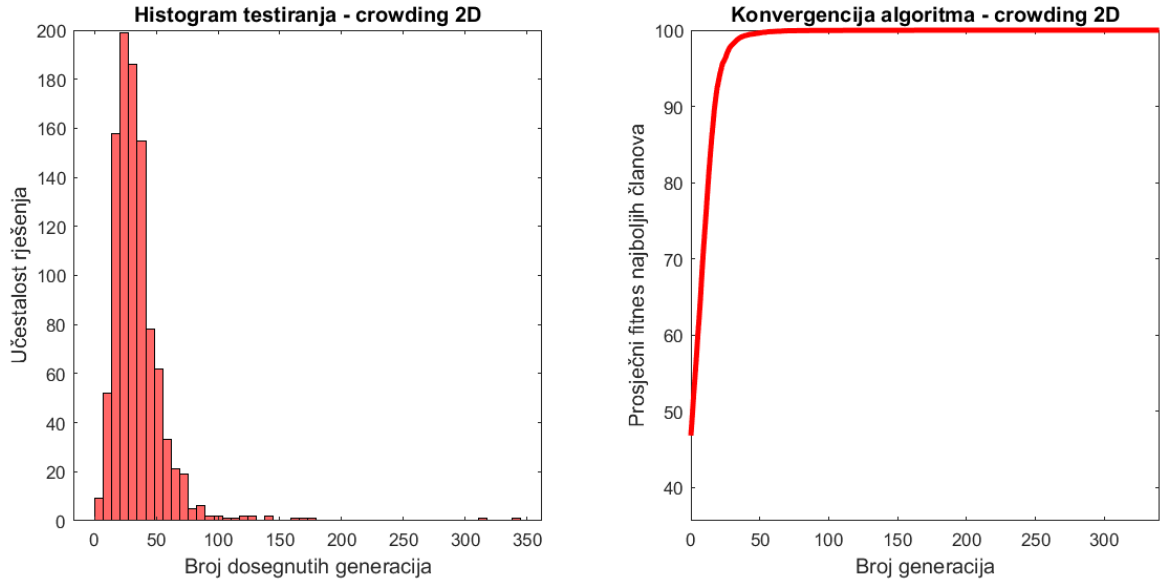
Slika 14. Rezultati testiranja osnovnog 2D algoritma za linearnu putanju.



Slika 15. Rezultati testiranja fitness sharing 2D algoritma za linearnu putanju.

(Slika 14), (Slika 15) i (Slika 16) pokazuju rezultate testiranja algoritama koji koriste linearnu putanju u 2D prostoru. Iz slika se vidi kako su svi algoritmi pronašli cilj, i putanju do njega bez sudara s preprekama, unutar zadanog ograničenja od 500 generacija. Iako su svi algoritmi bili uspješni u svojem radu, nisu svi jednako učinkoviti. Osnovnom algoritmu, (Slika 14), trebalo je puno više generacija da dođe do zadovoljavajućeg rješenja nego što je to trebalo algoritmima koji koriste metode očuvanja raznolikosti populacije. Ako se usporede rezultati

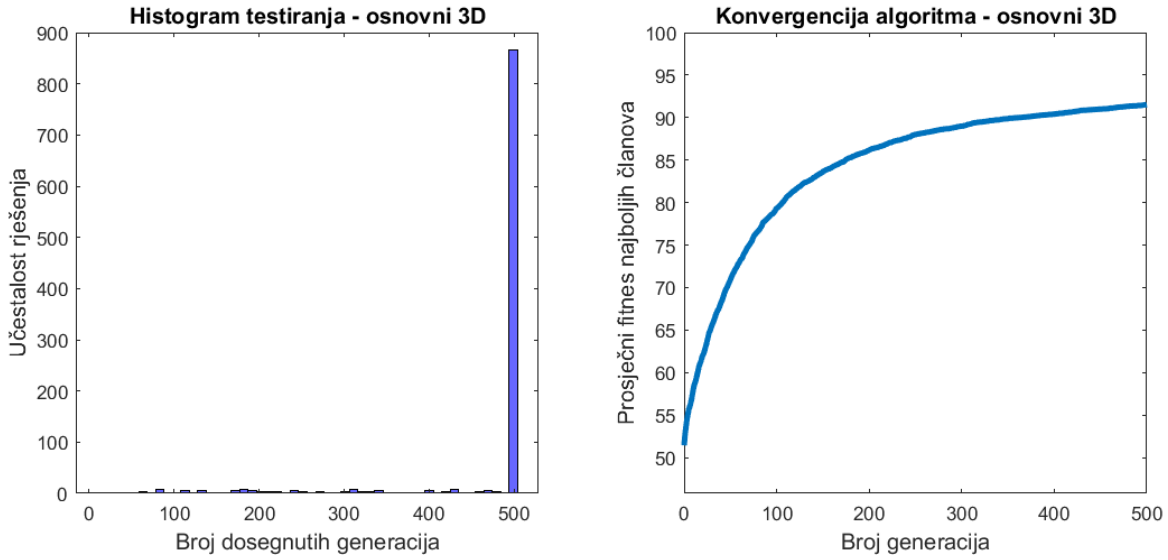
fitness sharing , (Slika 15) i crowding algoritama, (Slika 16), može se primijetiti da fitness sharing algoritam ima malo bržu konvergenciju fitnesa i manji prosječni broj generacija, (Tablica 3).



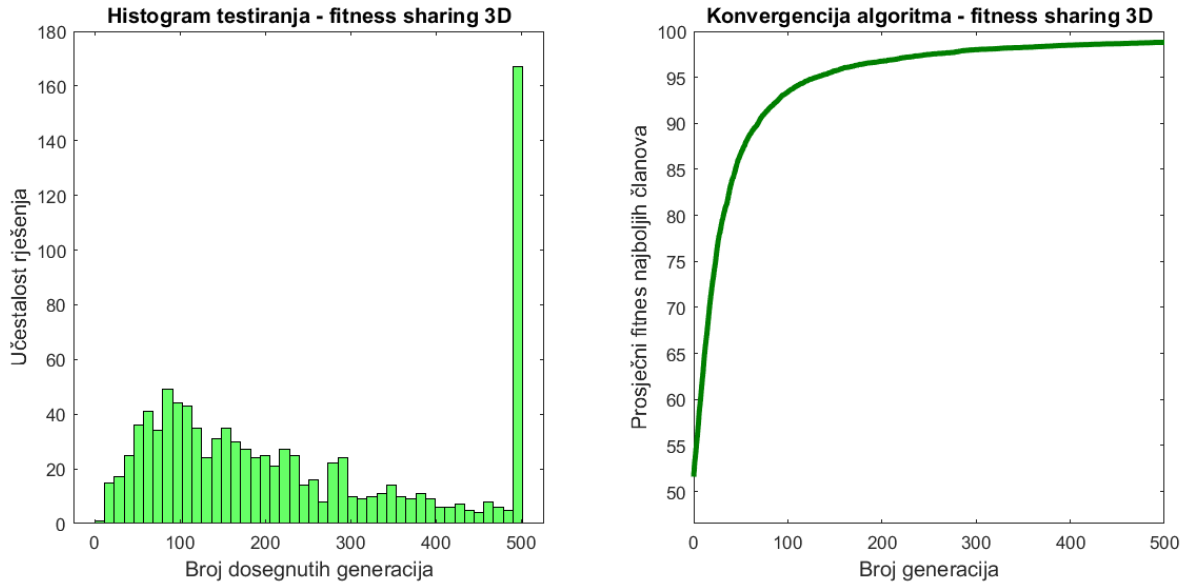
Slika 16. Rezultati testiranja crowding 2D algoritma za linearnu putanju.

(Slika 17), (Slika 18) i (Slika 19) pokazuju rezultate testiranja algoritama koji koriste linearnu putanju u 3D prostoru. 3D problem je znatno zahtjevniji za rješavanje od 2D problema, kao što se vidi iz rezultata. Iako je 3D problem namjerno postavljen kao najjednostavniji mogući, sa samo jednom preprekom, (Slika 13), osnovni algoritam, (Slika 17), nije uspio pronaći rješenje unutar zadanog ograničenja od 500 generacija u 86,4% simulacija, (Tablica 3). Iako je moguće da bi ih pronašao u nekoj kasnijoj generaciji da nije bio prekinut, iz prikaza konvergencije, (Slika 17), može se zaključiti da bi mu za to trebao velik broj generacija ili uopće ne bi pronašao rješenje. Iako je u 13,6% simulacija pronašao rješenje, zbog takvih rezultata može se smatrati neprikladnim za ovakvu vrstu problema.

Pozitivni utjecaj metoda za očuvanje raznolikosti dolazi do izražaja tek u 3D prostoru. Fitness sharing algoritam, (Slika 18), pronašao je zadovoljavajuće rješenje unutar zadanog ograničenja od 500 generacija u 84% slučajeva (Tablica 3). Osim toga, gledajući konvergenciju fitnesa, može se zaključiti da bi vjerojatno pronašao rješenja i u kasnijim generacijama da nije bio prekinut.

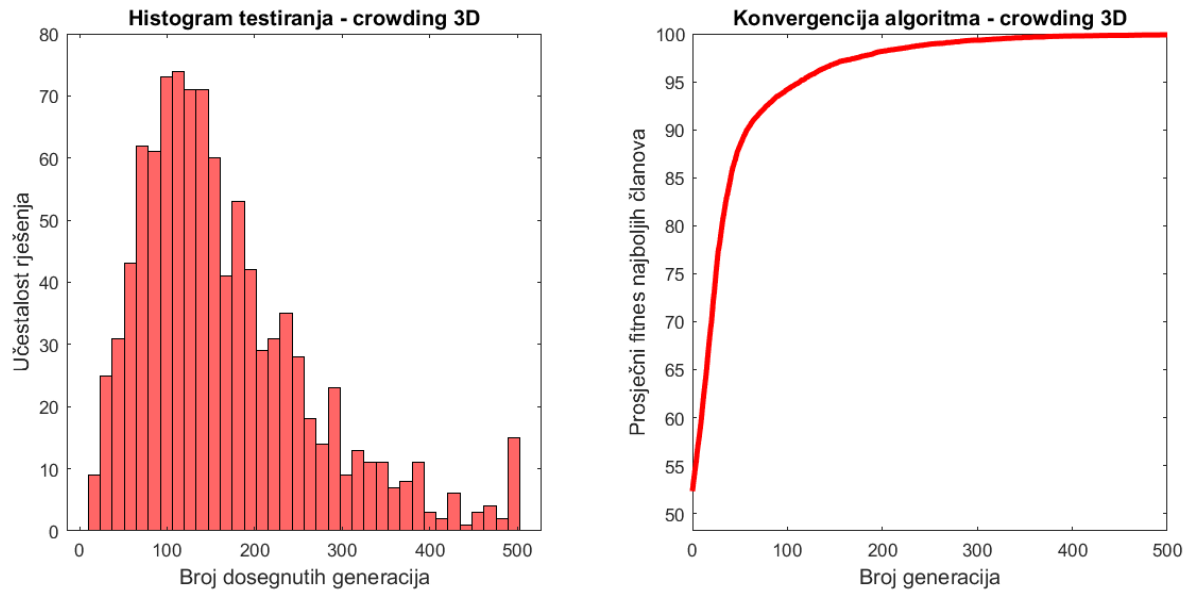


Slika 17. Rezultati testiranja osnovnog 3D algoritma za linearnu putanju.



Slika 18. Rezultati testiranja fitness sharing 3D algoritma za linearnu putanju.

Rezultati testiranja crowding algoritma u 3D prostoru, (Slika 19), pokazuju kako crowding metoda ima veliku prednost nad fitness sharing metodom kada se radi o zahtjevnijim problemima. Kod 3D problema, crowding algoritam nije pronašao rješenje unutar limita od 500 generacija u samo 14 simulacija, (Tablica 3), što pokazuje pouzdanost i robusnost ove metode.



Slika 19. Rezultati testiranja crowding 3D algoritma za linearnu putanju.

Tablica 3. Rezultati testiranja algoritama za linearnu putanju.

Algoritam	Broj simulacija	Broj zaustavljenih simulacija	Najmanja dosegnuta generacija	Najveća dosegnuta generacija	Prosječni broj generacija
Osnovni 2D	1000	2	3	500	101,73
Fitness sharing 2D	1000	0	3	116	24
Crowding 2D	1000	0	3	341	34,453
Osnovni 3D	1000	864	12	500	467,98
Fitness sharing 3D	1000	160	6	500	233,535
Crowding 3D	1000	14	14	500	167,088

3.3. Polinomna putanja

Slično kao i u prethodnom potpoglavlju, u ovom potpoglavlju bit će objašnjene tri različite verzije algoritama, također sve tri verzije algoritma su ispitane na dvodimenzionalnom i trodimenzionalnom prostoru. Isto kao i u prethodnom potpoglavlju, tri verzije se odnose na osnovni algoritam i algoritme koji koriste fitness sharing i crowding metode očuvanja raznolikosti populacije. Kako bi se mogle napraviti usporedbe o učinkovitosti, dijelovi algoritama koji nisu povezani s očuvanjem raznolikosti su im jednaki. Svi algoritmi su testirani

na istim labirintima s jednakim rasporedom prepreka te jednakim početnim i krajnjim pozicijama.

Za razliku od prethodnog potpoglavlja, putanja se više ne odvija linearno po koordinatama nego je ona polinomna funkcija proizvoljnog reda. To znači da se promijenio fenotipski prostor članova populacije, a samim time i njegov genotip i način kodiranja između ta dva prostora. Polinomna funkcija je funkcija definirana polinomom:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x^1 + a_0 \quad , a_i \in \mathbb{R}, i = 1 \dots n \tag{3.7}$$

ili, kraće:

$$P(x) = \sum_{i=0}^{n} a_i x^i \tag{3.8}$$

Najprikladniji oblik gena za prikaz polinomnog fenotipskog prostora su realni brojevi. Tako je genom članova složen od niza realnih brojeva koji ustvari predstavljaju koeficijente a_i polinomne funkcije. Budući da se početna točka kod svakog algoritma nalazi u nuli, slobodni koeficijent a_0 je izbačen, a fenotipski prikaz je jednak jednadžbi (3.9):

$$P(x) = \sum_{i=1}^{n} a_i x^i \tag{3.9}$$

Populacija je predstavljena matricama, (3.10), (3.12) i (3.13), gdje m predstavlja broj članova populacije, a n predstavlja duljinu genoma. Veličina populacije m postavljena je na 50 članova u svakom algoritmu , dok je duljina genoma postavljena na 7 što je ujedno i red polinoma n iz jednadžbe (3.9). Jednadžba (3.10) prikazuje izgled populacije.

$$\boldsymbol{populacija} = \begin{bmatrix} a_{11} = rand & a_{12} = rand & \cdots & a_{1n} = rand \\ a_{21} = rand & a_{22} = rand & \cdots & a_{2n} = rand \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} = rand & a_{m2} = rand & \cdots & a_{mn} = rand \end{bmatrix} \tag{3.10}$$

Svaki red matrice je jedan član, a u svakom redu stupci su redom geni koji odgovaraju koeficijentima polinoma a_i. Geni su prilikom inicijalizacije algoritama odabrani nasumično pri čemu operator $rand$ traži nasumične realne brojeve iz intervala $[-5, 5]$.

Za traženje putanje u trodimenzionalnom prostoru, 3D problem je sveden na dva 2D problema. To je izvedeno tako da se traže dvije 2D polinomne krivulje koje se kasnije povežu u prostornu krivulju. U jednadžbi (3.11), $f1(x)$ predstavlja polinomnu funkciju u xy ravnini dok $f2(x)$ predstavlja polinomnu funkciju u xz ravnini:

$$P1(x) = \sum_{i=1}^{n} a_i x^i \quad , \quad P2(x) = \sum_{i=1}^{n} b_i x^i \tag{3.11}$$

Samim time potrebne su i dvije populacije:

$$\boldsymbol{populacija_y} = \begin{bmatrix} a_{11} = rand & a_{12} = rand & \cdots & a_{1n} = rand \\ a_{21} = rand & a_{22} = rand & \cdots & a_{2n} = rand \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} = rand & a_{m2} = rand & \cdots & a_{mn} = rand \end{bmatrix} \tag{3.12}$$

$$\boldsymbol{populacija_z} = \begin{bmatrix} b_{11} = rand & b_{12} = rand & \cdots & b_{1n} = rand \\ b_{21} = rand & b_{22} = rand & \cdots & b_{2n} = rand \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} = rand & b_{m2} = rand & \cdots & b_{mn} = rand \end{bmatrix} \tag{3.13}$$

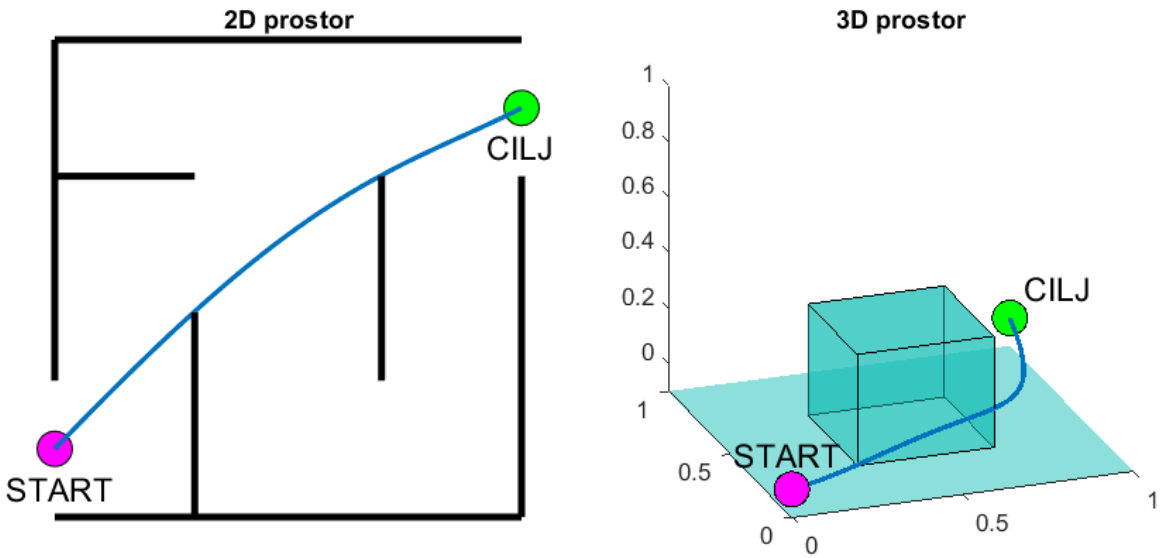
Kako se polinomi protežu u intervalu $\langle -\infty, \infty \rangle$ bilo bi nemoguće prikazati putanju u cijelom intervalu, zato se polinom promatra u ograničenom intervalu: $x = [0,1], x \in \mathbb{R}$. Pri tome je navedeni interval diskretiziran na 100 dijelova koji su složeni u vektor od kojega se kasnije radi matrica:

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_{100} \\ x_1^2 & x_2^2 & \cdots & x_{100}^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_{100}^n \end{bmatrix} = \begin{bmatrix} 0 & x_2 & x_3 & \cdots & x_{99} & 1 \\ 0 & x_2^2 & x_3^2 & \cdots & x_{99}^2 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & x_2^n & x_3^n & \cdots & x_{99}^n & 1 \end{bmatrix} \tag{3.14}$$

pri čemu je n duljina gena, odnosno red polinoma. Dobivena matrica množi se s matricom populacije, (3.10), te se dobiva matica koja sadrži vrijednosti diskretiziranih polinomnih funkcija:

$$\boldsymbol{PF = populacija \cdot X} = \begin{bmatrix} 0 & P_1(x_2) & P_1(x_3) & \cdots & P_1(x_{99}) & 1 \\ 0 & P_2(x_2) & P_2(x_3) & \cdots & P_2(x_{99}) & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & P_m(x_2) & P_m(x_3) & \cdots & P_m(x_{99}) & 1 \end{bmatrix} \tag{3.15}$$

Dobivena matrica sadrži m redova, što odgovara broju članova u populaciji, a vrijednosti vektora polinoma, koje se nalaze u redovima, jednostavno se mogu iskoristiti kako bi se nacrtale dobivene putanje.



Slika 20. Primjer otkrivene polinomne putanje.

(Slika 20) pokazuje primjer moguće polinomne putanje od početne do krajnje pozicije za 2D i 3D prostore. Raspored prepreka je sličan onom koji se koristio za testiranje algoritama za linearnu putanju, ali je malo prilagođen algoritmima za polinomnu putanju. Za razliku od algoritama koji su tražili linearnu putanju po koordinatama, gdje je algoritmu jedan od zadataka pronaći cilj, algoritmi s polinomnom putanjom imaju određenu početnu i krajnju poziciju. To znači da se polinom mora prilagoditi tako da prolazi kroz početnu poziciju čije su koordinate $(0,0)$ i krajnju poziciju čije su koordinate $(1,1)$. Za početnu poziciju, jednostavno se u vektoru

polinoma prva vrijednost postavi na nulu, kao što je već prikazano. Za krajnju poziciju nije dovoljno postaviti vrijednost zadnjeg stupca u vektoru polinoma na 1, uz to je potrebno namjestiti jedan od koeficijenata polinoma. To je izvedeno tako da je u jednadžbu (3.7) uvrštena vrijednost $P(1) = 1$, iz čega sređivanjem slijedi:

$$\begin{aligned} 1 &= a_n + a_{n-1} + \cdots a_i + \cdots + a_2 + a_1 \\ a_i &= 1 - (a_1 + \cdots + a_{i-1}) - (a_{i+1} + \cdots + a_n) \end{aligned} \tag{3.16}$$

Pri čemu je i nasumično odabrani broj iz intervala $\langle 1, n \rangle$. Ovisno o željenoj krajnjoj poziciji potrebno je uvrstiti drugačiju vrijednost, na primjer kod algoritama koji traže 3D putanju, z koordinata krajnje pozicije iznosi $z = 0$, tako da se kod namještanja koeficijenata tog polinoma slično dobije:

$$b_i = -(b_1 + \cdots + b_{i-1}) - (b_{i+1} + \cdots + b_n) \tag{3.17}$$

U početnim fazama odvijanja algoritma, velik broj članova populacije ima koliziju s postavljenim preprekama. Unatoč tome članovi populacije se ocjenjuju njihovim vrijednostima fitnesa F. Fitnes F računa se pomoću fitnes funkcije (3.18), ovisno o fenotipskim svojstvima članova. Vrijednost fitnesa ovisi o duljini polinoma između početne i krajnje pozicije i kazni koja ovisi o kolizijama s preprekama:

$$F(i) = \frac{1000}{1 + kazna(i) + duljina(i)} \tag{3.18}$$

Kazna se uvećava za svaki sudar s preprekom, pri čemu povećanje iznosi 90 za sudar s vertikalnom preprekom i 100 za sudar s horizontalnom preprekom, gledano ovisno o provjeravanoj ravnini.

Što se tiče odabira roditelja, izveden je jednako kao i kod algoritama koji traže linearnu putanju po koordinatama. Kod osnovnog algoritma koristi se ruletno pravilo s vjerojatnošću odabira koja je jednaka relativnoj vrijednosti fitnesa, (2.1). Kod algoritma koji koristi fitness sharing metodu također se koristi ruletno pravilo, ali umjesto relativne vrijednosti fitnesa, svaki

član ima vjerojatnost odabira koja je jednaka njegovoj prilagođenoj vrijednosti fitnesa (2.5). Za crowding algoritam, prilikom odabira roditelja ne koristi se ruletno pravilo nego se iz populacije roditelja svaki član nasumično odabire i upari s drugim nasumičnim članom.

Kako genom sadrži realne vrijednosti, operator rekombinacije koji se koristi u ovim algoritmima je aritmetičko križanje, (2.3). Prilikom križanja svakog para gena provjerava se hoće li doći do križanja ovisno o faktoru vjerojatnosti križanja pc koji je postavljen na $0{,}7$ što znači da će se vjerojatno križati 70% gena. Ako dolazi do križanja faktor aritmetičkog križanja α odabire se nasumično iz intervala $[0{,}1]$, što znači da je faktor aritmetičkog križanja vjerojatno drugačiji za svaki par gena. Mutacija se odvija na svakom genu zasebno ovisno o faktoru vjerojatnosti mutacije pm koji je postavljen na $0{,}5$, što znači da će vjerojatno mutirati 50% gena. Implementirani operator mutacije je izmijenjeni oblik neuniformne mutacije za genome s realnim vrijednostima:

$$a_i = a_i(1 - \beta + 2\beta \cdot rand) \tag{3.19}$$

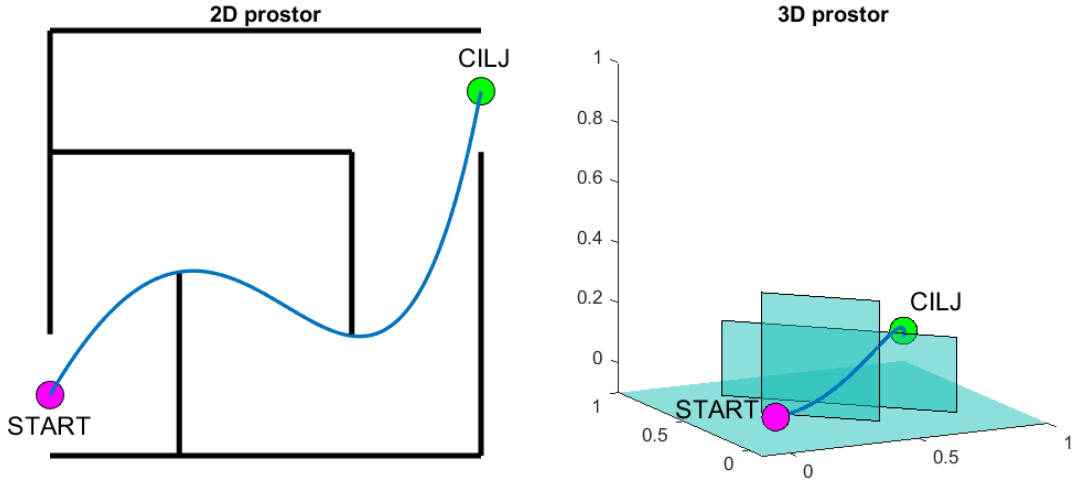
Pri čemu se nasumični broj $rand$, kao i uvijek, odabire iz intervala $[0{,}1]$, a faktor intenziteta mutacije β je postavljen na $0{,}5$. Iz jednadžbe (3.19) može se vidjeti da se vrijednost gena množi s nasumičnim brojem iz intervala $[0{,}5, 1{,}5]$, što dalje konkretno znači da je iznos mutacije najviše $\pm 50\%$, čime je u određenoj mjeri ispravljen nedostatak aritmetičkog križanja.

Kod osnovnog i fitness sharing algoritma, odabir nove populacije odvija se prema starosti, odnosno potomci zamjenjuju roditelje pri čemu se koristi elitizam, odabire se jedan elitni član koji se zadržava u populaciji. Crowding algoritam koristi drugačiji model odabira zamjene populacije. Kod crowding metode, odabir nove populacije je prema vrijednosti fitnesa umjesto prema starosti, odvija se turnir između roditelja i djece te se oni s većom vrijednosti fitnesa zadržavaju u populaciji, a lošiji se odbacuju. Kod 2D algoritmima se zbog primjene malo složenijeg rasporeda prepreka može dogoditi da algoritam zapne i izgubi raznolikost populacije prije pronalaska putanje bez sudara. Iz tog razloga se kod 2D algoritama, u slučaju da se ne pronađe putanja bez sudara unutar 50 generacija, pola populacije briše i zamjenjuje s novim nasumičnim članovima. Novi članovi, iako vjerojatno imaju loše vrijednosti fitnesa, unose novi genetski materijal u populaciju, što povećava raznolikost populacije i značajno poboljšava rad algoritama.

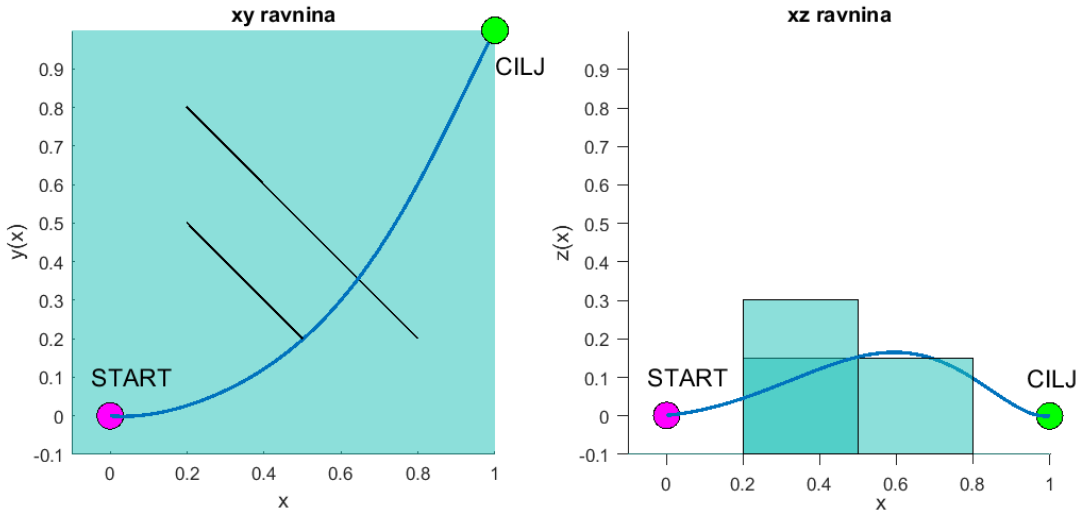
Svaki algoritam je ograničen na 500 generacija. Nakon 500 generacija algoritam se zaustavlja i uzima se najbolja pronađena putanja, iako je u kasnijim generacijama algoritam mogao dodatno optimizirati pronađenu putanju. Ako algoritam ne pronađe putanju bez sudara, simulacija se smatra neuspješnom. Broj generacija može se postaviti na željenu vrijednost ovisno o dostupnom vremenu i željenom stupnju optimiziranosti putanje.

3.3.1. Rezultati testiranja algoritama

Prilikom testiranja rada algoritma ispitni okoliš, odnosno raspored prepreka u prostoru, je izmijenjen. (Slika 21) prikazuje raspored prepreka koji se koristi prilikom testiranja algoritama u 2D i 3D prostoru.



Slika 21. Ispitni okoliši s mogućim putanjama.



Slika 22. Prikaz prostorne putanje u *xy* i *xz* ravnini.

Gledajući sliku ispitnog okoliša teško je prepoznati izgled prostorne krivulje, iz tog razloga (Slika 22) prikazuje kako krivulja izgleda gledajući xy i xz ravnine. Dvije prikazane polinomne funkcije su dobivene u svojim ravninama te kasnije povezane u jednu prostornu putanju.

Algoritmi koji su koristili linearnu putanju nisu imali zadanu poziciju cilja, nego su je morali sami pronaći. Kako algoritmi koji traže polinomnu putanju imaju zadanu krajnju poziciju, odlučio sam im zadati kompleksniji raspored prepreka od onog na prethodnoj slici, (Slika 20). Tako da za 2D prostor, umjesto jednog blagog skretanja, putanja ima dva značajna skretanja, a za 3D prostor, umjesto jedne velike prepreke i poda, putanja ne smije udariti u pod i dvije prepreke od čega prvu treba zaobići sa strane, a drugu preskočiti.

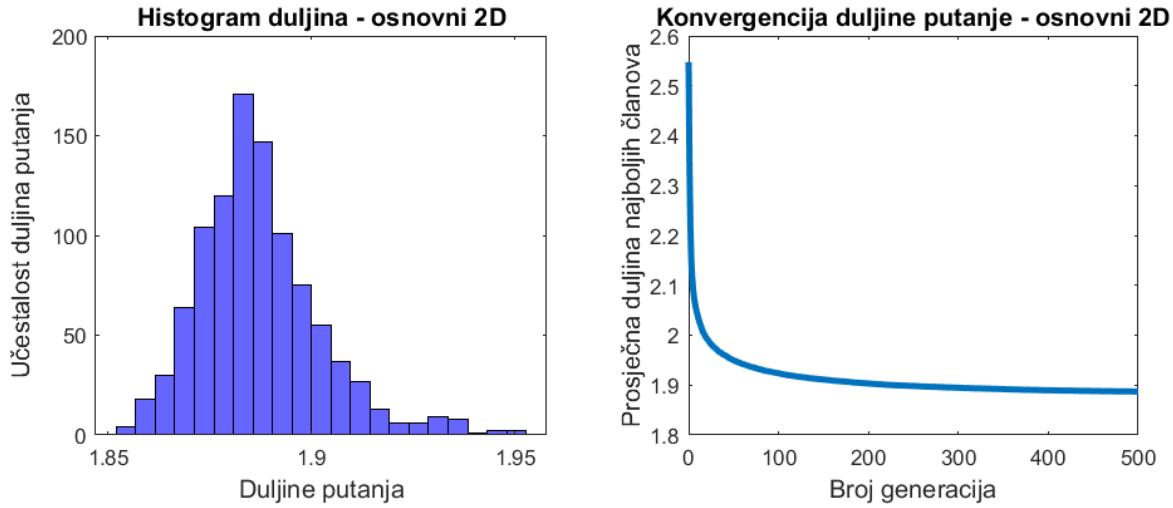
Tablica 4. Postavke i parametri algoritama za polinomnu putanju.

Algoritam	**Osnovni**	**Fitness sharing**	**Crowding**
Prikazivanje jedinki	Prikazivanje realnim brojevima		
Rekombinacija	Aritmetičko križanje, vjerojatnost križanja $pc = 0{,}7$		
Mutacija	Nasumična mutacija		
Parametri mutacije	Vjerojatnost mutacije $pm = 0{,}5$, intenzitet mutacije $\beta = 0{,}5$		
Odabir roditelja	Ruletno pravilo	Fitness sharing	Nasumično uparivanje
Zamjena populacije	Ovisno o starosti uz elitizam		Ovisno o fitnesu
Veličina populacije	50 članova		
Inicijalizacija	Početna populacija je odabrana nasumično		
Uvjet prekida	Dosegnuto 500 generacija		

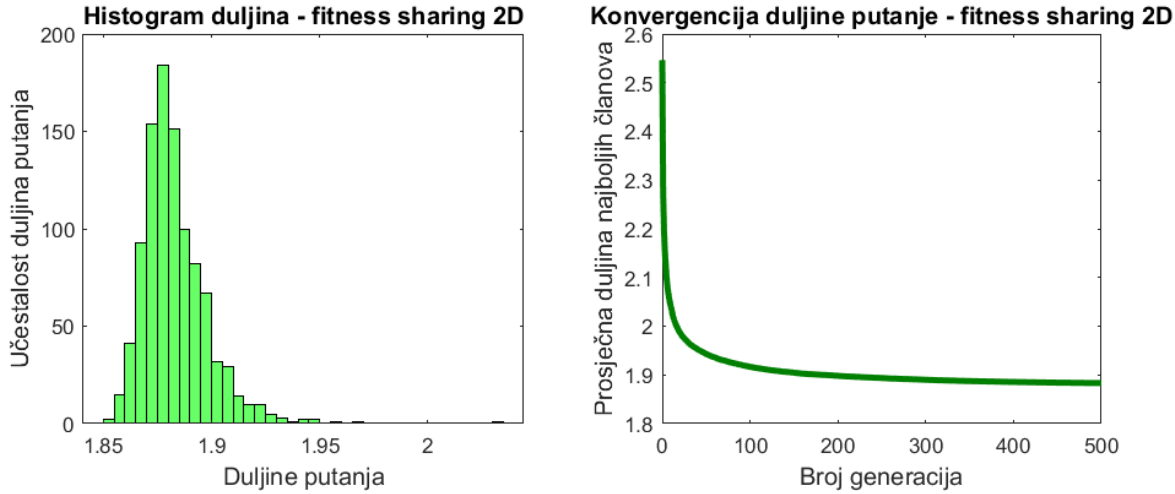
Kao i za algoritme koji koriste linearnu putanju, algoritmi s polinomnom putanjom te njihove simulacije i rezultati su napravljeni i prikazani programom MATLAB, verzija R2017b. Kako bi se usporedile različite verzije algoritama, svaki algoritam je i u ovom slučaju simuliran 1000 puta, a rezultati su također prikazani grafički pomoću histograma. Budući da je uporaba polinomnih putanja za računalo računski puno zahtjevniji problem od uporabe linearne putanje, proces simuliranja svih 6 algoritama trajao je 10 sati i 42 minute.

Sljedeći histogrami prikazuju koliko je puta pronađena putanja čija je duljina u rasponu koji odgovara širini stupca histograma. Uz histograme su prikazane prosječne konvergencije

duljina najboljih pronađenih putanja koje nemaju koliziju s preprekom. One simulacije kod kojih nije pronađena putanja bez sudara su isključene iz grafičkih prikaza.

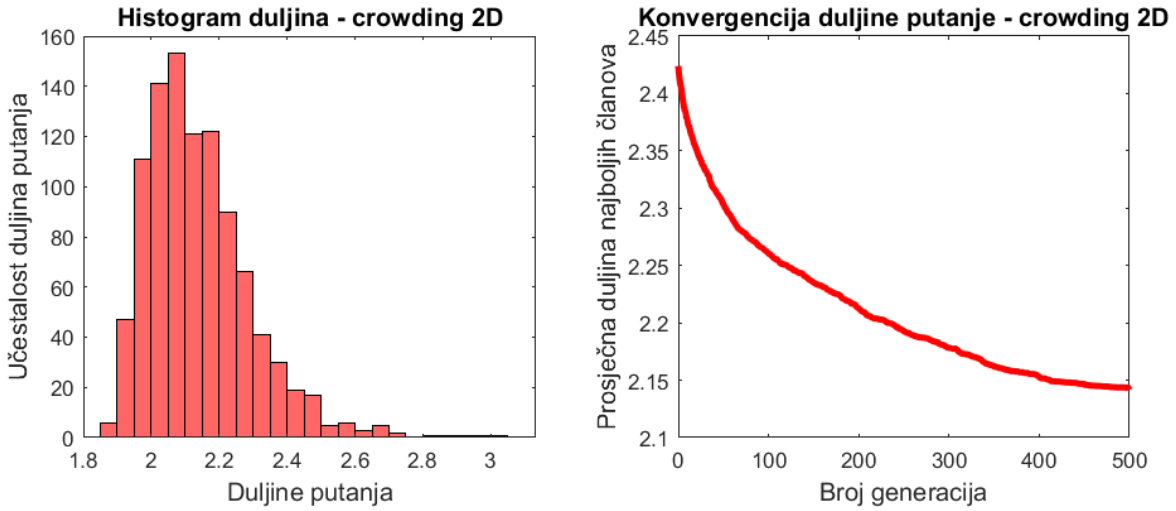


Slika 23. Rezultati testiranja osnovnog 2D algoritma za polinomnu putanju.



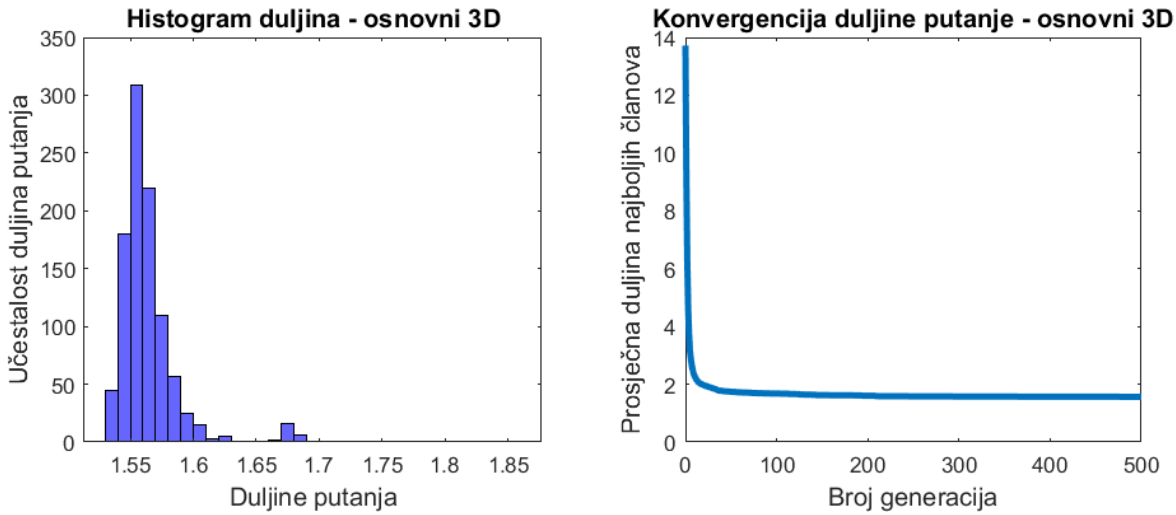
Slika 24. Rezultati testiranja fitness sharing 2D algoritma za polinomnu putanju.

(Slika 23), (Slika 24) pokazuju rezultate testiranja algoritama koji koriste polinomnu putanju u 2D prostoru. Gledajući njihove podatke, (Tablica 5), može se vidjeti kako su osnovni 2D algoritam i fitness sharing 2D algoritam pronašli putanju bez sudara s preprekom u svim provedenim simulacijama. Uz to se iz njihovih histograma može vidjeti da se pronađena rješenja rasipaju u uskom intervalu duljina, a prikaz konvergencije pokazuje kako algoritmi jako brzo konvergiraju optimalnom rješenju. Ono što pomalo iznenađuje su rezultati testiranja crowding 2D algoritma, (Slika 25).



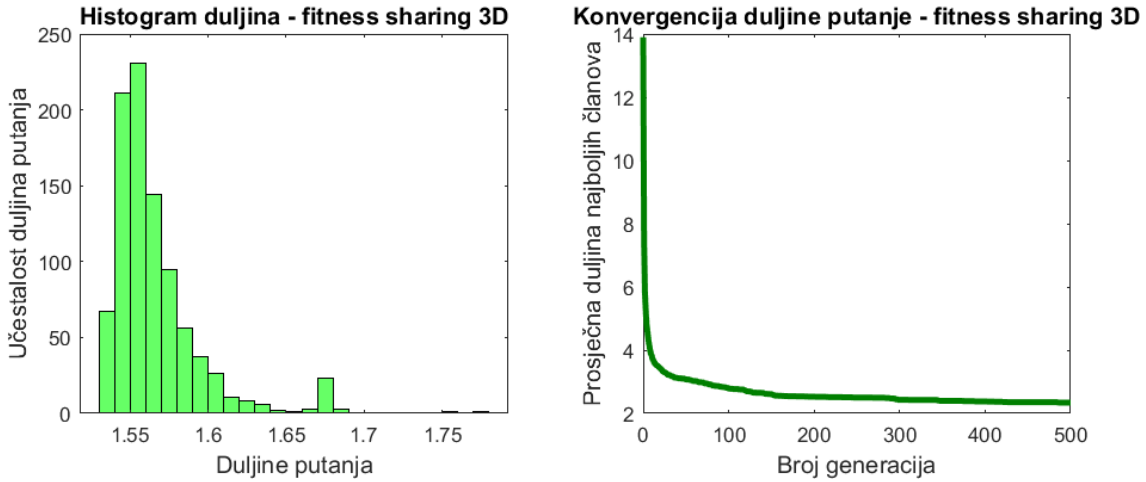
Slika 25. Rezultati testiranja crowding 2D algoritma za polinomnu putanju.

Algoritam koji je dao najbolje rezultate kod problema s linearnom putanjom za trenutni problem daje najgore rezultate. Gledajući podatke, (Tablica 5), vidi se da algoritam nije uspio pronaći rješenje u 1% simulacija, što pokazuje laganu nesigurnost prilikom rješavanja problema. Gledajući histogram i konvergenciju duljina putanja jasno se vidi koliko je algoritam sporiji. Crowding 2D algoritmu vjerojatno bi trebalo još barem 500 generacija da dosegne prosječnu duljinu putanja osnovnog 2D i fitness sharing 2D algoritma. Osim toga razlika prosječne duljine putanja između algoritama iznosi ~12%, što se može smatrati značajnim brojem kada se govori o optimizaciji duljine putanje.



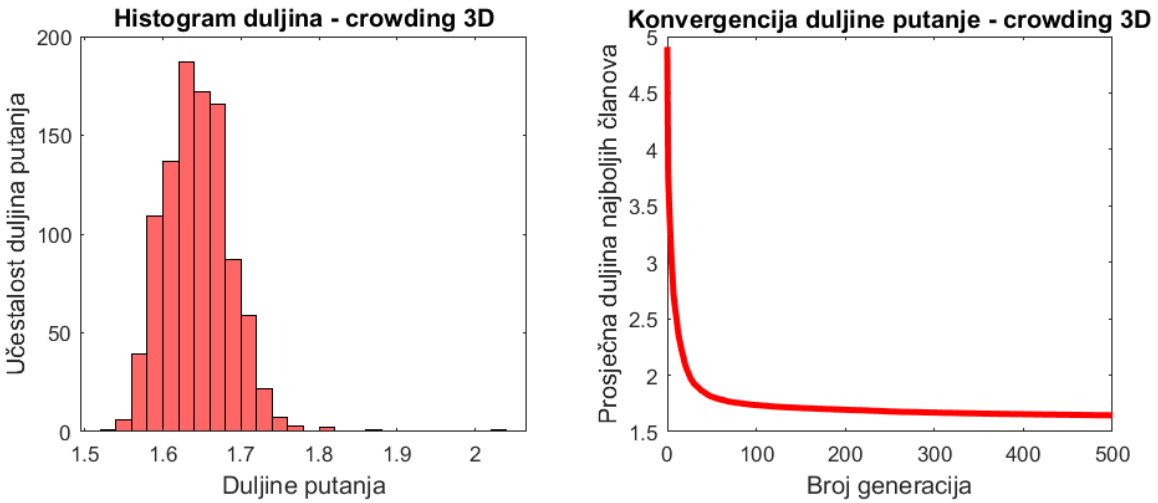
Slika 26. Rezultati testiranja osnovnog 3D algoritma za polinomnu putanju.

Gledajući rezultate testiranja osnovnog algoritma na 3D problemu, (Slika 26) i (Tablica 5), vidi se kako je ova verzija algoritma opet dala neočekivano dobre rezultate.



Slika 27. Rezultati testiranja fitness sharing 3D algoritma za polinomnu putanju.

Fitnes sharing algoritam nije dao tako dobre rezultate za 3D problem kao što su bili kod 2D problema. Rasipanje duljina mu je bilo tako veliko, da sam morao izbaciti 74 rezultata simulacije, čije su duljine iz intervala $\langle 4, 30 \rangle$, kako bih mogao prikazati histogram duljina, (Slika 27). Iako su to sve putanje bez sudara, duljine su im toliko velike da se mogu smatrati neuspješnim simulacijama. Ako se izbačene simulacije ne iskoriste prilikom računanja prosječne duljine, ona bi iznosila 1,5659 čime bi fitness sharing algoritam imao gotovo jednake rezultate kao osnovni algoritam za ovaj problem.



Slika 28. Rezultati testiranja crowding 3D algoritma za polinomnu putanju.

Unatoč slabijim rezultatima kod 2D problema, crowding algoritam je za 3D problem, (Slika 28), dao dobre rezultate koji se ne razlikuju puno od onih osnovne verzije algoritma.

Tablica 5. Rezultati testiranja algoritma za polinomnu putanju.

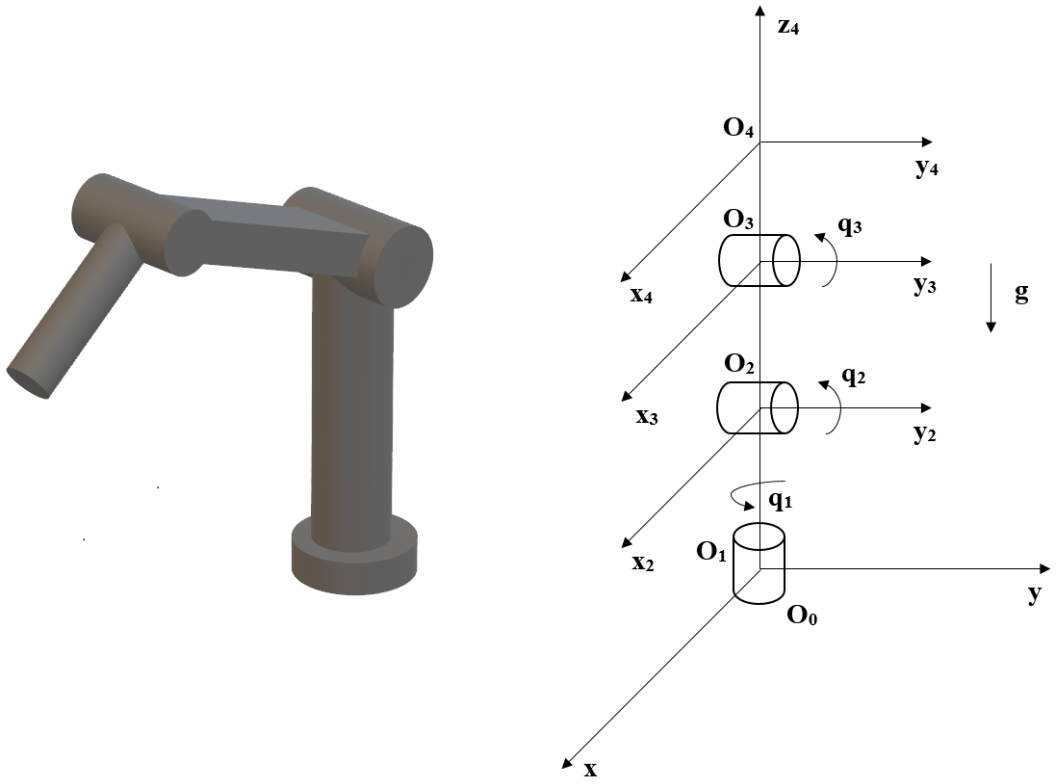
Algoritam	Broj simulacija	Broj neuspješnih simulacija	Najkraća pronađena putanja	Najdulja pronađena putanja	Prosječna duljina putanja
Osnovni 2D	1000	0	1,8536	1,9512	1,8871
Fitness sharing 2D	1000	0	1,8517	2,0311	1,8832
Crowding 2D	1000	10	1,8915	3,0043	2,1415
Osnovni 3D	1000	0	1,5319	1,8522	1,5649
Fitness sharing 3D	1000	0	1,5305	29,1408	2,3411
Crowding 3D	1000	0	1,5374	2,0355	1,6444

4. SIMULACIJA PRAĆENJA TRAJEKTORIJE

Optimizirane polinomne putanje koje su dobivene u prošlom poglavlju predstavljaju najkraći nelinearni put kroz zadani prostor od početne do ciljane pozicije. Primjena ovih putanja je široka, a jedan od mogućih primjera je kretanje središnje točke alata na prirubnici industrijskog robota. Industrijski roboti i manipulatori svake godine imaju sve širu primjenu u domaćoj industriji te polako zamjenjuju klasične metode korištene u prošlosti. Skraćivanjem duljine putanje, koju industrijski robot prijeđe prilikom izvođenja svojih operacija, značajno skraćuje trajanje njegovog radnog ciklusa. Iz tog razloga industrijski robot je odabran za simuliranje primjene algoritama iz prethodnog poglavlja.

4.1. Kinematički model robota rotacijske strukture

Simulacija praćenja pronađene putanje bit će prikazana na modelu rotacijskog robota s tri stupnja slobode gibanja. Prije nego što je moguće izvesti simulaciju, potrebno je izvesti direktni i inverzni kinematički model robota. (Slika 29) prikazuje primjer rotacijskog robota i raspored koordinatnih sustava članaka.



Slika 29. Robot RRR strukture (lijevo) i vezani koordinatni sustavi (desno). [21]

Kako bi se mogao riješiti direktni kinematički problem, potrebno je omogućiti jednostavan prijelaz između koordinatnih sustava, tj. zadani vektor iz jednog koordinatnog sustava preračunati u drugi. To se postiže matricom homogenih transformacija:

$$\boldsymbol{A} = \begin{bmatrix} i_x & j_x & k_x & p_x \\ i_y & j_y & k_y & p_y \\ i_z & j_z & k_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Prva tri stupca su projekcije jediničnih vektora definiranog koordinatnog sustava u odnosu na nepokretni (početni) koordinatni sustav, a posljednji stupac daje koordinate ishodišta definiranog koordinatnog sustava u odnosu prema nepokretnome. Dakle, prva tri vektora definiraju orijentaciju, a posljednji vektor definira položaj između koordinatnih sustava. S obzirom na to da se kod robota pojavljuju translacijski i rotacijski stupnjevi slobode gibanja, potrebno je definirati osnovne transformacije translacije i rotacije:

Transformacija translacije koordinatnog sustava:

$$Tran(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Transformacija rotacije oko x-osi za kut ϑ:

$$Rot(x,\vartheta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\vartheta & -sin\vartheta & 0 \\ 0 & sin\vartheta & cos\vartheta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Transformacija rotacije oko y-osi za kut φ:

$$Rot(y,\varphi) = \begin{bmatrix} cos\varphi & 0 & sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -sin\varphi & 0 & cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Transformacija rotacije oko z-osi za kut ψ:

$$Rot(z,\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 & 0 \\ sin\psi & cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Prema vezanim koordinatnim sustavima, (Slika 29), mogu se pisati sljedeće matrice prijelaza između koordinatnih sustava:

$$\boldsymbol{A}_1 = Rot(z,q_1) = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A}_2 = Tran(0,0,L_1)Rot(y,q_2) = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ 0 & 1 & 0 & 0 \\ -s_2 & 0 & c_2 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$\boldsymbol{A}_3 = Tran(0,0,L_2)Rot(y,q_3) = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ 0 & 1 & 0 & 0 \\ -s_3 & 0 & c_3 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A}_4 = Tran(0,0,L_3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pri čemu je zbog jednostavnijeg zapisa uvedena supstitucija $\sin(q_1) = s_1, cos(q_2) = c_2$, i slično.

Nakon uvrštavanja matrica prijelaza i sređivanja izraza, dobiju se sljedeće matrice prijenosa iz i-tog u nepokretni koordinatni sustav:

$${}^0\boldsymbol{T}_1 = \boldsymbol{A}_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$^0\boldsymbol{T}_2 = \boldsymbol{A}_1\boldsymbol{A}_2 = \begin{bmatrix} c_1c_2 & -s_1 & c_1s_2 & 0 \\ s_1c_2 & c_1 & s_1s_2 & 0 \\ -s_2 & 0 & c_2 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.8)}$$

$$^0\boldsymbol{T}_3 = \boldsymbol{A}_1\boldsymbol{A}_2\boldsymbol{A}_3 = \begin{bmatrix} c_1c_{23} & -s_1 & c_1s_{23} & c_1s_2L_2 \\ s_1c_{23} & c_1 & s_1s_{23} & s_1s_2L_2 \\ -s_{23} & 0 & c_{23} & L_1 + c_2L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.9)}$$

$$^0\boldsymbol{T}_4 = \boldsymbol{A}_1\boldsymbol{A}_2\boldsymbol{A}_3\boldsymbol{A}_4 = \begin{bmatrix} c_1c_{23} & -s_1 & c_1s_{23} & c_1(s_2L_2 + s_{23}L_3) \\ s_1c_{23} & c_1 & s_1s_{23} & s_1(s_2L_2 + s_{23}L_3) \\ -s_{23} & 0 & c_{23} & L_1 + c_2L_2 + c_{23}L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.10)}$$

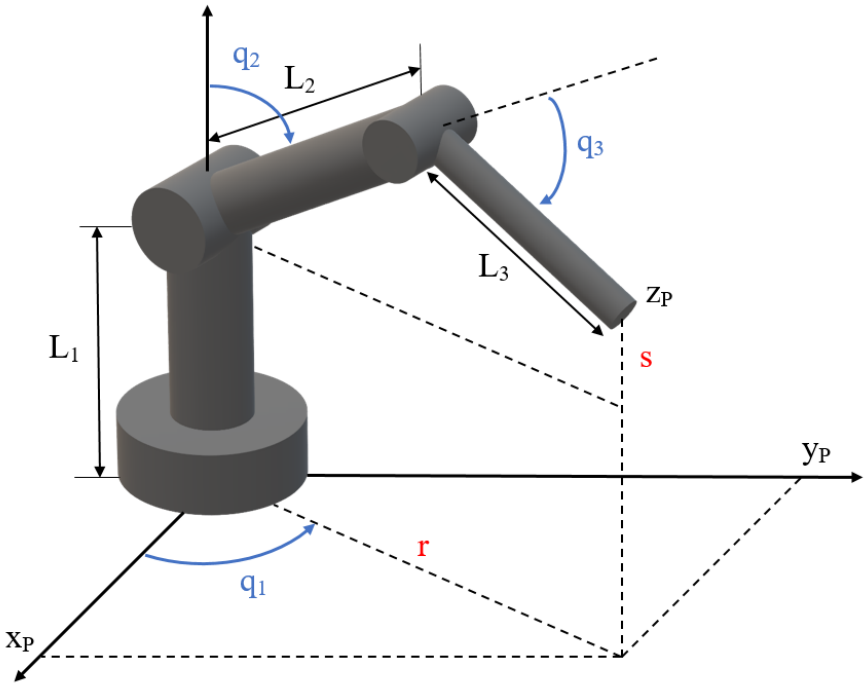
Gdje su $n_{x,y,z}$ komponente vektora normale, $o_{x,y,z}$ komponente vektora orijentacije, $a_{x,y,z}$ komponente vektora akcije i $p_{x,y,z}$ komponente vektora pozicije koji je poveznica između ishodišnog koordinatnog sustava i koordinatnog sustava x_4-y_4-z_4.

Čime je direktni kinematički problem riješen [21].

Rješavanjem direktnog kinematičkog problema dobivena je ovisnost vanjskih koordinata, odnosno položaj robota (p_x, p_y, p_z), o unutrašnjim koordinatama, odnosno rotacijama u pojedinim stupnjevima slobode gibanja (q_1, q_2, q_3). Ovime se ne može napraviti simulacija praćenja putanje budući da nije poznato koliki je iznos unutarnjih koordinata robota. Iz tog razloga treba riješiti inverzni kinematički problem.

Inverzna kinematika je problem određivanja unutarnjih koordinata robota iz zadane vanjske pozicije i orijentacije, to jest matrice $^0\boldsymbol{T}_4$. Inverzna kinematika je iznimno težak problem kojem kompleksnost raste ovisno o broju stupnjeva slobode jer problem nije jednoznačan, odnosno, postoji više kombinacija unutarnjih koordinata (tzv. konfiguracija robota) koje rezultiraju jednakom vanjskom pozicijom i orijentacijom. Postoji više pristupa rješavanja inverzne kinematike, npr. numerički alati, poput iterativnog rješavanja inverznog kinematičkog problema pomoću Jacobijeve matrice, analitičkih izvoda za jednadžbe kuteva zakreta stupnjeva slobode i geometrijskih pristupa.

Najjednostavniji način dobivanja kuteva zakreta je korištenjem geometrijske metode. Ovom metodom se, iz poznatih dimenzija robota i poznate vanjske pozicije, unutarnje koordinate računaju korištenjem trigonometrijskih funkcija. (Slika 30) prikazuje koje su sve dimenzije potrebne za rješavanje inverznog kinematičkog problema.



Slika 30. Opći prikaz RRR robota.

Kut zakreta prvog stupnja slobode gibanja dobije se jednostavno primjenom tangens funkcije:

$$\tan(q_1) = \frac{y_P}{x_P} \quad \longrightarrow \quad q_1 = \arctan\left(\frac{y_P}{x_P}\right) \tag{4.11}$$

Izračunati kut q_1 nije potreban da bi se izračunala preostala dva kuta zakreta, ali je potrebno izračunati pomoćne vrijednosti r i s, koje su označene crvenom bojom, (Slika 30):

$$r = \sqrt{x_P^2 + y_P^2}$$
$$s = z_P - L_1 \tag{4.12}$$

Od sljedeća dva kuta, prvo se računa kut zakreta trećeg stupnja slobode gibanja. Za to je iskorišten kosinusov poučak, a nakon sređivanja izraza dobije se:

$$q_3 = \arccos\left(\frac{r^2 + s^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \tag{4.13}$$

Kada je poznat kut zakreta trećeg stupnja slobode, može se izračunati i kut zakreta drugog stupnja slobode:

$$q_2 = \arctan\left(\frac{r}{s}\right) - \arctan\left(\frac{L_3 \sin q_3}{L_2 + L_3 \cos(q_3)}\right) \tag{4.14}$$

Kao što je već ranije spomenuto, budući da problem nije jednoznačan, postoji više kombinacija unutarnjih koordinata (tzv. konfiguracija robota) koje rezultiraju jednakom vanjskom pozicijom i orijentacijom. (Slika 30) prikazuje takozvanu *„Elbow up“* konfiguraciju robota. Kako bi se robot doveo u *„Elbow down“* konfiguraciju kutevi zakreta robota računaju se na isti način , ali je potrebno jednadžbu (4.13) izmijeniti tako da bude:

$$q_3 = -\arccos\left(\frac{r^2 + s^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \tag{4.15}$$

Kada su poznati svi kutevi zakreta sloboda gibanja, odnosno sve unutarnje koordinate robota, inverzni kinematički problem je riješen.

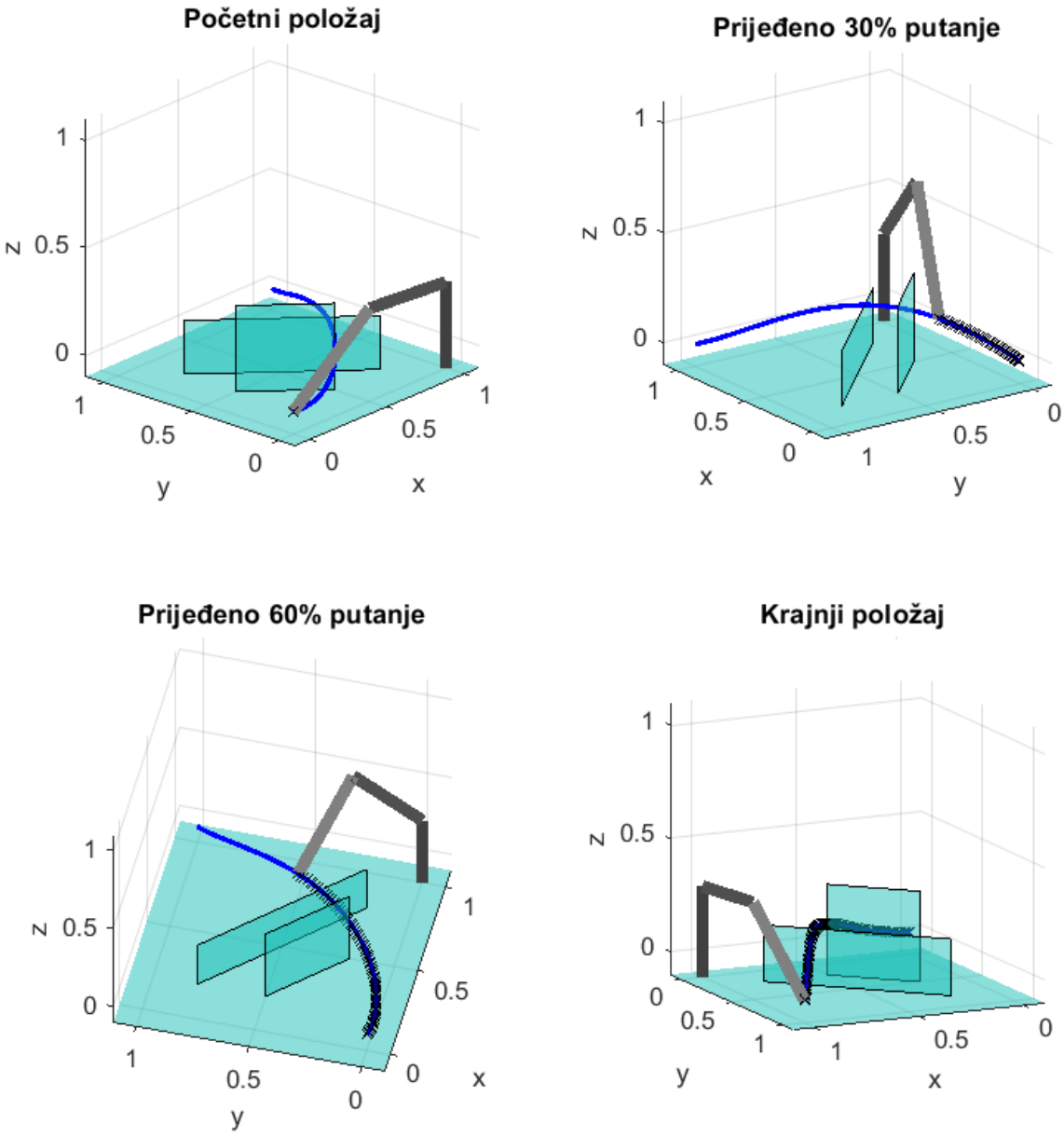
4.2. Simulacija praćenja putanje

Kao i algoritmi za traženje putanje, i simulacija praćenja putanje je napravljena i prikazana programom MATLAB, verzija R2017b.

Praćenje putanje prikazano je na 3D prostoru te je praćena putanja polinomna funkcija. Praćena funkcija pronađena je korištenjem osnovnog 3D algoritma za polinomnu putanju budući da je ta verzija algoritma dala najbolje rezultate za ovaj problem. Pronađena optimizirana polinomna funkcija putanje je diskretizirana na 100 dijelova čime su dobivene točke koje čine polinomnu krivulju. Svaka točka ima svoje prostorne koordinate koje su ustvari vanjske koordinate položaja robota (p_x, p_y, p_z). Na ovaj način simulacija se svodi na proces

vođenja rotacijskog robota po položaju, što se može jednostavno implementirati i na fizičke industrijske robote.

(Slika 31) prikazuje proces praćenja putanje pri čemu su prikazani trenutci: u samom početku praćenja, nakon prijeđenih 30% putanje, nakon prijeđenih 60% putanje i trenutak u kojem se proces praćenja zaustavlja zbog dostizanja cilja.



Slika 31. Prikaz praćenja putanje.

5. ZAKLJUČAK

U ovome radu uspješno je osmišljena metoda planiranja optimiziranih trajektorija kretanja, koja je inspirirana evolucijskim algoritmom. Radni prostor može biti organiziran kao dvodimenzionalan ili trodimenzionalan uz uvjet poznate početne i krajnje pozicije referentne točke i poznat raspored prepreka unutar prostora. Ukupno je izrađeno 12 algoritama od čega njih šest traži linearnu trajektoriju, tri u 2D prostoru i tri u 3D prostoru, a ostalih šest traži trajektoriju određenu polinomom općeg reda, tri u 2D prostoru i tri u 3D prostoru. Svi algoritmi su testirani te su rezultati testiranja analizirani i prikazani kako bi se vidjele prednosti i mane pojedinih algoritama.

Za slučaj primjene linearne putanje po koordinatama rezultati pokazuju kako metode očuvanja raznolikosti značajno utječu na učinkovitost evolucijskih algoritama. Razlika je očita čak i kod jednostavnog 2D problema gdje je osnovna verzija algoritma bila značajno sporija prilikom pronalaženja rješenja. Gledajući rezultate testiranja 3D problema, osnovni algoritam je prisilno zaustavljen u 86,4% simulacija zbog toga što nije uspio pronaći rješenje bez sudara. Može se zaključiti kako evolucijski algoritam u svojem najjednostavnijem obliku nije prikladan za rješavanje ovako definiranog 3D problem. Kod algoritma koji koristi fitness sharing metodu mogu se vidjeti dobri rezultati kod 2D problema. Štoviše ova verzija algoritma je dala najbolje rezultate za 2D problem. Unatoč tome, nije dala jednako dobre rezultate kod testiranja 3D problema, gdje je uspješno završeno 84% simulacija, u usporedbi s algoritmom koji koristi crowding metodu koji je uspješno završio 98,6% simulacija. Može se reći da je crowding metoda, kao metoda očuvanja raznolikosti, znatno učinkovitija i robusnija od fitness sharing metode za ovako definiran problem.

Rezultati testiranja za slučaj primjene putanje određene polinomom pokazuju kako je osnovni algoritam vrlo dobar za rješavanje ovakvog problema. Osnovni algoritam je i za 2D i za 3D prostor dao rješenja koja imaju mala odstupanja i brzu konvergenciju prema optimalnom rješenju. Fitness sharing algoritam imao je određene probleme kod 3D prostora, u 7,4% simulacija zapeo je u nekom od lokalnih optimuma. Zbog toga, iako je pronašao rješenja bez sudara, duljina takvih putanja je bila toliko velika da se mogu odbaciti kao uspješna rješenja. Ako zanemarimo ta rješenja dobiveni rezultati su bolji od onih osnovne verzije algoritma čime je fitness sharing algoritam, od testiranih algoritama, dao najbolje rezultate za zadani problem. Crowding algoritam, iako je dao daleko najbolje rezultate kod problema s linearnom putanjom

po koordinatama, ima najlošije rezultate kod putanje određene polinomom općeg reda. Unatoč tome, dobiveni rezultati nisu puno lošiji od ostala dva algoritma tako da je i dalje primjenjiv.

Nakon izvršene statističke analize dobivenih rezultata može se zaključiti kako svaki algoritam ima svoje prednosti i mane te da svaki ima određene primjene za koje će dati dobre, ali i loše rezultate. Nameće se pitanje zašto je crowding metoda dala toliko bolje rezultate za linearnu putanju, ali najlošije rezultate za polinomnu putanju? Razlika u performansama dolazi iz razlike u metodi odabira roditelja. Kod crowding algoritma svaki član populacije se koristi u populaciji roditelja, što znači da i oni članovi s lošim fitnesom postaju roditelji. Ova metoda pridonosi veću količinu genetskog materijala u populaciju, ali za posljedicu ima sporiju konvergenciju. Rad crowding algoritama potencijalno bi se mogao poboljšati zamjenom loših članova elitnim članovima, što treba napraviti u pažljivoj mjeri da se ne izgubi cijeli smisao crowding metode očuvanja raznolikosti. Kod osnovnog i fitness sharing algoritma odabir roditelja iz populacije proporcionalno je ovisan fitnesu članova uz dodatno korištenje ruletnog pravila. Treba napomenuti da iako odabir korištenjem ruletnog pravila daje vjerojatnost odabira ovisno o vrijednosti fitnesa članova, to je i dalje jako stohastička metoda kod koje član populacije s dobrim fitnesom može izgubiti svoje mjesto u populaciji roditelja. Ova metoda odabira roditelja dalje puno bržu konvergenciju koja za posljedicu ima gubitak raznolikosti genetskog materijala, što je reducirano kod fitness sharing metode. Ovi algoritmi imaju veliku slobodu prilikom istraživanja prostora, što im je dalo značajnu prednost kod traženja polinomne putanje zbog definirane početne i krajnje točke. Kod traženja linearne putanje algoritmi ne znaju gdje je cilj nego ga moraju pronaći i zato sloboda prilikom istraživanja prostora postaje nedostatak.

Simulacija praćenja pronađene trajektorije prikazana je na primjeru industrijskog robota, ali to nije granica primjene ove metode. Ova metoda planiranja ima široku potencijalnu primjenu koja osim za optimizaciju rada industrijskih robota uključuje planiranje kretanja mobilnih robota u 2D prostoru i letjelica u 3D prostoru.

Jedno od mogućih proširenja ove metode je primjena vizijskih sustava za mapiranje prostora kako bi se dobio raspored prepreka.

LITERATURA

[1] Čehulić, L.: *Usporedba rada evolucijskog algoritma i algoritma roja na standardnim testnim funkcijama*. Završni projekt preddiplomskog studija, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, 2017.

[2] Brownlee, J.: *Clever Algorithms: Nature-Inspired Programming Recipes*. Creative Commons, Australia, 2011.

[3] Xinjie Yu, Mitsuo Gen: *Introduction to Evolutionary Algorithms*. Springer, 2010.

[4] Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Second Edition, Springer, 2015.

[5] Fogel, L.J., Owens, A.J., Walsh, M.J.: *Artificial Intelligence through Simulated Evolution*. Wiley, Chichester, UK, 1966.

[6] Holland, J.H.: *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992.

[7] Schwefel, H.P.: *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[8] Koza, J.R.: *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[9] Darwin, C.: *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. John Murray, London, 1859.

[10] Dawkins, R.: *The Selfish Gene*. Oxford University Press, Oxford, UK, 1976.

[11] Goldberg, D.E.: *Genetic Algorithms in search, optimization, and machine learning*. Addison Wesley, 1989.

[12] Parkinson, A.R., Balling, R., Hedengren, J.D.: *Optimization Methods for Engineering Design*. Second Edition, Brigham Young University, 2018.

[13] De Jong, K.A.: *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[14] Syswerda, G.: *Uniform crossover in genetic algorithms*. Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco, 1989., pages 2–9.

[15] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1996

[16] Goldberg, D.E., Richardson, J.: *Genetic algorithms with sharing for multimodal function optimization*. Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, Hillsdale, New Jersey, 1987., pages 41–49.

[17] Squillero, G., Tonda, A.: *Promoting Diversity in Evolutionary Algorithms*., Proceedings of the 2016 conference on Genetic and Evolutionary Computation Conference, p.pp 934-944, Denver, CO, USA., 2016.

[18] Ćurković, P., Čehulić, L.: *Evolutionary Planner of 3D Trajectories*. The 29th DAAAM International Symposium on Intelligent Manufacturing and Automation, Zadar, Croatia, 24-27th October 2018.

[19] Deb, K.: *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[20] LaValle, S., M.: *Planning algorithms*. Cambridge University Press, 2006.

[21] Šurina, T., Crneković, M.: *Industrijski roboti*. Školska knjiga, Zagreb, 1990.

[22] https://www.mathworks.com/help/matlab/ (20.11.2018.)

PRILOZI

I. CD-R disc

II. Matlab kodovi algoritama

Prilog II: Matlab kodovi algoritama

```
%   Inicijalizacija algoritama za linearno gibanje po koordinatama 2D
% Glavni parametri
broj_clanova=50;
duljina_gena=10;
broj_generacija=500;
pc=0.7; % Vjerojatnost križanja
pm=0.4; % Vjerojatnost mutacije
kzv=10; % Kazna za vraćanje
nagrada1=60; % Nagrada za dolazak do cilja
nagrada2=40; % Nagrada za izbjegavanje zida

udaljenost=zeros(broj_clanova,1);
SUM=0;
fitnes3=200; % fitnes3 - za elitizam - udaljenost
fitnes4=0;       % fitnes4 - za elitizam - ukupno
fitnes_uk=0;
fitnes_uk_r=0;
fitnes_uk_d=0;
broj1=0; % za prekid
broj2=0; % za graf 1
broj3=0; % za graf 2

% Zidovi
zid_x=[0 0; 0 6; 6   6; 0 6;   4   4; 0 1.9; 2   2];
zid_y=[0 7; 0 0; 0 4.9; 7 7; 2.1 4.9; 5   5; 0 2.9];
[A,B]=size(zid_x);

start_x=1;    % Zadani početak kretanja robota
start_y=1;
cilj_x=6; % Željeni cilj kretanja robota
cilj_y=6;

% Inicijalizacija populacije
populacija=randi(4,[broj_clanova,duljina_gena]);    % postavljanje
populacije
elitni=populacija(1,:); % početni elitni
x_koord=ones(broj_clanova,1)*start_x;
y_koord=ones(broj_clanova,1)*start_y;
x_koord=ones(broj_clanova,1)*start_x;
y_koord=ones(broj_clanova,1)*start_y;
x_koord_r=ones(broj_clanova,1)*start_x;
y_koord_r=ones(broj_clanova,1)*start_y;
x_koord_d=ones(broj_clanova,1)*start_x;
y_koord_d=ones(broj_clanova,1)*start_y;

%   Algoritam za linearno gibanje - osnovni 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje za rješenje bez sudara
        broj1=broj1+1;
        if broj1==2
            break
        end
    end
    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord] = koordVekt2D(x_koord,y_koord,populacija);
```

```
    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2);
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - za
udaljenost
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:duljina_gena
                for k=1:A
                    if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))

funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
```

```
            end
        end
    end

    if max(fitnes_uk)==100 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk);
    for i=1:broj_clanova
        if  fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end

    % Vjerojatnost izbora člana populacije
    vjerojatnost=fitnes_uk/sum(fitnes_uk);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelj(i,:)=populacija(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelj(u:t,:); %parovi gena
        if rand<=pc
            p1=par_trenutni(1,:);
            p2=par_trenutni(2,:);
            qq=randi([1,duljina_gena-1]);
            P11=[p1(1:qq) p2(qq+1:duljina_gena)];
            P22=[p2(1:qq) p1(qq+1:duljina_gena)];
            par_trenutni_krizani=[P11;P22];
        else
            par_trenutni_krizani=par_trenutni; % ako nema križanja
        end
        nova_pop(u:t,:)=par_trenutni_krizani;
    end

    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                nova_pop(i,j)=randi(4);
            end
        end
    end
```

```
    % Slaganje nove populacije
    populacija=nova_pop;
    populacija(1,:)=elitni;

    % Graf
    figure(1)
    subplot(2,2,[1,3])
    plot(start_x,start_y,'ko','MarkerSize',20,'MarkerFaceColor','m'),hold
on
    str=char('START');
    text(0.5,0.4,0.9,str,'FontSize',15)
    plot(cilj_x,cilj_y,'ko','MarkerSize',20,'MarkerFaceColor','g')
    str=char('CILJ');
    text(6.4,6,0.9,str,'FontSize',15)
    for i=1:A
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',4)
    end
    plot(x_koord(1,:),y_koord(1,:),'Linewidth',2)

plot(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),'bo','MarkerSize',
10,'MarkerFaceColor','b')
    title('Putanja dobivena osnovnim algoritmom')
    axis([0 6 0 7]),axis off,hold off
    subplot(2,2,2)
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
end

%   Algoritam za linearno gibanje - fitness sharing 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje za rješenje bez sudara
        broj1=broj1+1;
        if broj1==2
            break
        end
    end
    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord] = koordVekt2D(x_koord,y_koord,populacija);
```

```
    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2);
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - ovisi o
udaljenosti
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:duljina_gena
                for k=1:A
                    if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))

funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
            end
```

```
        end
    end

    if max(fitnes_uk)==100 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk);

    for i=1:broj_clanova
        if fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end

    % Vjerojatnost izbora člana populacije
    % Fitnes sharing
    for i=1:broj_clanova
        for j=1:broj_clanova
            d=abs(fitnes_uk(i,:)-fitnes_uk(j,:));
            if d<10 % sigma vrijednost je postavljena na 10
                sh(i,j)=1-(d/10);
            else
                sh(i,j)=0;
            end
        end
    end
    fitnes_uk_sh=fitnes_uk./sum(sh,2);
    vjerojatnost_sh=fitnes_uk_sh/sum(fitnes_uk_sh);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost_sh(j);
            if SUM>r
                roditelj(i,:)=populacija(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelj(u:t,:); %parovi gena
        if rand<=pc
            p1=par_trenutni(1,:);
            p2=par_trenutni(2,:);
            qq=randi([1,duljina_gena-1]);
            P11=[p1(1:qq) p2(qq+1:duljina_gena)];
            P22=[p2(1:qq) p1(qq+1:duljina_gena)];
            par_trenutni_krizani=[P11;P22];
        else
            par_trenutni_krizani=par_trenutni; % ako nema križanja
        end
```

```
        nova_pop(u:t,:)=par_trenutni_krizani;
    end

    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                nova_pop(i,j)=randi(4);
            end
        end
    end

    % Slaganje nove populacije
    populacija=nova_pop;
    populacija(1,:)=elitni;

    % Graf
    figure(1)
    subplot(2,2,[1,3])
    plot(start_x,start_y,'ko','MarkerSize',20,'MarkerFaceColor','m'),hold
on
    str=char('START');
    text(0.5,0.4,0.9,str,'FontSize',15)
    plot(cilj_x,cilj_y,'ko','MarkerSize',20,'MarkerFaceColor','g')
    str=char('CILJ');
    text(6.4,6,0.9,str,'FontSize',15)
    for i=1:A
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',4)
    end
    plot(x_koord(1,:),y_koord(1,:),'Linewidth',2)

plot(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),'bo','MarkerSize',
10,'MarkerFaceColor','b')
    title('Putanja dobivena uz dodatak fitness sharing-a')
    axis([0 6 0 7]),hold off,axis off
    grid on
    subplot(2,2,2)
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
```

```
end

%   Algoritam za linearno gibanje - crowding 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje za rješenje bez sudara
        broj1=broj1+1;
        if broj1==2
            break
        end
    end

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=randi(broj_clanova+1-i);   % nasumično biranje roditelja
        roditelji(i,:)=populacija(r,:);
        populacija(r,:)=[];
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelji(u:t,:); %parovi gena
        p1=par_trenutni(1,:);
        p2=par_trenutni(2,:);
        qq=randi([1,duljina_gena-1]);
        P11=[p1(1:qq) p2(qq+1:duljina_gena)];
        P22=[p2(1:qq) p1(qq+1:duljina_gena)];
        par_trenutni_krizani=[P11;P22];
        djeca(u:t,:)=par_trenutni_krizani;
    end

    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                djeca(i,j)=randi(4);
            end
        end
    end

    % Postavljanje koordinata putanje u vektore
    [x_koord_r,y_koord_r] = koordVekt2D(x_koord_r,y_koord_r,roditelji);
    [x_koord_d,y_koord_d] = koordVekt2D(x_koord_d,y_koord_d,djeca);

    %Fitnes
    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova  % za roditelje
        udaljenost_r(i,:)=sqrt((cilj_x-
x_koord_r(i,duljina_gena+1))^2+(cilj_y-y_koord_r(i,duljina_gena+1))^2);
    end
    fitnes1_r=(nagrada1+udaljenost_r)./(1+udaljenost_r); % Fitnes 1 - ovisi
o udaljenosti

    for i=1:broj_clanova % za djecu
        udaljenost_d(i,:)=sqrt((cilj_x-
x_koord_d(i,duljina_gena+1))^2+(cilj_y-y_koord_d(i,duljina_gena+1))^2); %
za roditelje
    end
    fitnes1_d=(nagrada1+udaljenost_d)./(1+udaljenost_d); % Fitnes 1 - ovisi
o udaljenosti
```

```
    % Kažnjavanje udaranja u zid
    funkcija1_r=zeros(broj_clanova,duljina_gena); % za roditelje
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord_r(i,j+1)>=min(zid_x(k,:)) &&
x_koord_r(i,j+1)<=max(zid_x(k,:)) && y_koord_r(i,j+1)>=min(zid_y(k,:)) &&
y_koord_r(i,j+1)<=max(zid_y(k,:))
                    funkcija1_r(i,j)=1;
                end
            end
        end
    end
    funkcija2_r=sum(funkcija1_r,2);
    fitnes2_r=nagrada2-nagrada2*funkcija2_r./duljina_gena;  % Fitnes 2 -
ovisi o udaranju u zid
    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3_r=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2_r(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord_r(i,j+1)>=min(zid_x(k,:)) &&
x_koord_r(i,j+1)<=max(zid_x(k,:)) && y_koord_r(i,j+1)>=min(zid_y(k,:)) &&
y_koord_r(i,j+1)<=max(zid_y(k,:))

funkcija3_r(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3_r(i,j)>0
                    break
                end
            end
            funkcija4_r=sum(funkcija3_r,2);
            fitnes2_r(i,:)=fitnes2_r(i,:)+funkcija4_r(i,:);
        end
    end
    fitnes_uk_r=fitnes1_r+fitnes2_r; % Ukupni fitnes

    % kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord_r(i,j+1)==x_koord_r(i,j-1) &&
y_koord_r(i,j+1)==y_koord_r(i,j-1)
                fitnes_uk_r(i,:)=fitnes_uk_r(i,:)-kzv;
                break
            end
        end
    end
    funkcija1_d=zeros(broj_clanova,duljina_gena); % za djecu
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord_d(i,j+1)>=min(zid_x(k,:)) &&
x_koord_d(i,j+1)<=max(zid_x(k,:)) && y_koord_d(i,j+1)>=min(zid_y(k,:)) &&
y_koord_d(i,j+1)<=max(zid_y(k,:))
                    funkcija1_d(i,j)=1;
                end
            end
```

```
        end
    end
    funkcija2_d=sum(funkcija1_d,2);
    fitnes2_d=nagrada2-nagrada2*funkcija2_d./duljina_gena;  % Fitnes 2 -
ovisi o udaranju u zid
    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3_d=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2_d(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord_d(i,j+1)>=min(zid_x(k,:)) &&
x_koord_d(i,j+1)<=max(zid_x(k,:)) && y_koord_d(i,j+1)>=min(zid_y(k,:)) &&
y_koord_d(i,j+1)<=max(zid_y(k,:))

funkcija3_d(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3_d(i,j)>0
                    break
                end
            end
            funkcija4_d=sum(funkcija3_d,2);
            fitnes2_d(i,:)=fitnes2_d(i,:)+funkcija4_d(i,:);
        end
    end
    fitnes_uk_d=fitnes1_d+fitnes2_d; % Ukupni fitnes
    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord_d(i,j+1)==x_koord_d(i,j-1) &&
y_koord_d(i,j+1)==y_koord_d(i,j-1)
                fitnes_uk_d(i,:)=fitnes_uk_d(i,:)-kzv;
                break
            end
        end
    end

    %Usporedba
    rod_za_usp=bi2de(roditelji,10,'left-msb'); % kombinaciji gena se
dodjeljuje vrijednost koja se koristi za usporedbu
    dijete_za_usp=bi2de(djeca,10,'left-msb');
    for u=1:2:broj_clanova
        t=u+1;
        d1=abs(rod_za_usp(u)-dijete_za_usp(u)); % trazi se udaljenost
između djece i roditelja
        d2=abs(rod_za_usp(t)-dijete_za_usp(t));
        d3=abs(rod_za_usp(u)-dijete_za_usp(t));
        d4=abs(rod_za_usp(t)-dijete_za_usp(u));
        if (d1+d2)<=(d3+d4) % u ovom slučaju uspoređuju se r1-d1 i r2-d2
            if fitnes_uk_r(u)>=fitnes_uk_d(u)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(u,:);
            end
            if fitnes_uk_r(t)>=fitnes_uk_d(t)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(t,:);
```

```
            end
        else % u ovom slučaju uspoređuju se r1-d2 i r2-d1
            if fitnes_uk_r(u)>=fitnes_uk_d(t)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(t,:);
            end
            if fitnes_uk_r(t)>=fitnes_uk_d(u)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(u,:);
            end
        end
    end

    % Ubacivanje elitnog iz prošle generacije - samo zbog grafa
    populacija(1,:)=elitni;

    % Testiranje nove populacije
    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord] = koordVekt2D(x_koord,y_koord,populacija);

    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2);
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - ovisi o
udaljenosti
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija1;
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord(i,j+1)>=min(zid_x(k,:)) &&
x_koord(i,j+1)<=max(zid_x(k,:)) && y_koord(i,j+1)>=min(zid_y(k,:)) &&
y_koord(i,j+1)<=max(zid_y(k,:))
```

```
funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
            end
        end
    end

    if max(fitnes_uk)==100 && broj3==0 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk);
    for i=1:broj_clanova
        if fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end

    % Graf
    figure(1)
    subplot(2,2,[1,3])
    plot(start_x,start_y,'ko','MarkerSize',20,'MarkerFaceColor','m'),hold
on
    str=char('START');
    text(0.5,0.4,0.9,str,'FontSize',15)
    plot(cilj_x,cilj_y,'ko','MarkerSize',20,'MarkerFaceColor','g')
    str=char('CILJ');
    text(6.4,6,0.9,str,'FontSize',15)
    for i=1:A
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',4)
    end
    plot(x_koord(1,:),y_koord(1,:),'Linewidth',2)

plot(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),'bo','MarkerSize',
10,'MarkerFaceColor','b')
    title('Putanja dobivena uz dodatak crowding-a')
    axis([0 6 0 7]),axis off, hold off
    subplot(2,2,2)
```

```
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
end

%   Funkcija za postavljanje koordinata putanja u vektore za 2D prostor
function [x_koord,y_koord] = koordVekt2D(x_koord,y_koord,populacija)
%   koordVekt2D
    for i=1:size(populacija,1)
        for j=1:size(populacija,2)
            x_koord(i,j+1)=x_koord(i,j);
            y_koord(i,j+1)=y_koord(i,j);
            if populacija(i,j)==1
                x_koord(i,j+1)=x_koord(i,j+1)+1; % za 1 robot ide desno
            elseif populacija(i,j)==2
                x_koord(i,j+1)=x_koord(i,j+1)-1; % za 2 robot ide lijevo
            elseif populacija(i,j)==3
                y_koord(i,j+1)=y_koord(i,j+1)+1; % za 3 robot ide gore
            elseif populacija(i,j)==4
                y_koord(i,j+1)=y_koord(i,j+1)-1; % za 4 robot ide dolje
            end
        end
    end
end

%   Inicijalizacija algoritama za linearno gibanje po koordinatama 3D
% Glavni parametri
broj_clanova=50;
duljina_gena=10;
broj_generacija=500;
pc=0.7; % Vjerojatnost križanja
pm=0.4; % Vjerojatnost mutacije

% Početne vrijednosti nekih veličina
udaljenost=zeros(broj_clanova,1);
SUM=0;
fitnes3=200; % fitnes3 - za elitizam - udaljenost
fitnes4=0;       % fitnes4 - za elitizam - ukupno
fitnes_uk=0;
fitnes_uk_r=0;
```

```
fitnes_uk_d=0;
broj1=0; % za prekid
broj2=0; % za graf 1
broj3=0; % za graf 2
kzv=10; % Kazna za vraćanje
nagrada1=60; % Nagrada za dolazak do cilja
nagrada2=40; % Nagrada za izbjegavanje zida

start_x=0; % Zadani početak kretanja robota
start_y=3;
start_z=3;
cilj_x=6;   % Željeni cilj kretanja robota
cilj_y=3;
cilj_z=3;

% Zidovi
zid_x=[2 4; 2 4; 2 4; 2 4; 4 4; 4 4; 2 2; 2 2];
zid_y=[2 2; 4 4; 4 4; 2 2; 4 4; 2 2; 2 2; 4 4];
zid_z=[4 4; 4 4; 2 2; 2 2; 2 4; 2 4; 2 4; 2 4];
c = ones(8,2); % za graf
[A,B]=size(zid_x);

% Inicijalizacija populacije
populacija=randi(6,[broj_clanova,duljina_gena]);   % postavljanje
populacije
elitni=populacija(1,:); % početni elitni
x_koord=ones(broj_clanova,1)*start_x;
y_koord=ones(broj_clanova,1)*start_y;
z_koord=ones(broj_clanova,1)*start_z;
x_koord_r=ones(broj_clanova,1)*start_x;
y_koord_r=ones(broj_clanova,1)*start_y;
z_koord_r=ones(broj_clanova,1)*start_z;
x_koord_d=ones(broj_clanova,1)*start_x;
y_koord_d=ones(broj_clanova,1)*start_y;
z_koord_d=ones(broj_clanova,1)*start_z;

%   Algoritam za linearno gibanje po koordinatama - osnovni 3D - Glava
petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje ako se postigne cilj bez
sudara
        broj1=broj1+1;
        if broj1==2
            break
        end
    end

    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord,z_koord] =
koordVekt3D(x_koord,y_koord,z_koord,populacija);

    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2+(cilj_z-z_koord(i,duljina_gena+1))^2);
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - za
udaljenost
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
```

```
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija1;
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:duljina_gena
                for k=1:A
                    if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4

funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1) && z_koord(i,j+1)==z_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
            end
        end
    end

    if max(fitnes_uk)==100 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
```

```
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk);
    for i=1:broj_clanova
        if fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end


    % Vjerojatnost izbora člana populacije
    vjerojatnost=fitnes_uk/sum(fitnes_uk);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelj(i,:)=populacija(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelj(u:t,:); %parovi gena

        if rand<=pc
            p1=par_trenutni(1,:);
            p2=par_trenutni(2,:);
            qq=randi([1,duljina_gena-1]);

            P11=[p1(1:qq) p2(qq+1:duljina_gena)];
            P22=[p2(1:qq) p1(qq+1:duljina_gena)];
            par_trenutni_krizani=[P11;P22];
        else
            par_trenutni_krizani=par_trenutni; % ako nema križanja
        end
        nova_pop(u:t,:)=par_trenutni_krizani;
    end

    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                nova_pop(i,j)=randi(6);
            end
        end
    end

    % Slaganje nove populacije
    populacija=nova_pop;
    populacija(1,:)=elitni;

    % Graf
    figure(1)
    subplot(2,2,[1,3])
```

```
plot3(start_x,start_y,start_z,'ko','MarkerSize',20,'MarkerFaceColor','m'),h
old on
    str=char('START');
    text(0.5,0.4,0.9,str,'FontSize',15)
    plot3(cilj_x,cilj_y,cilj_z,'ko','MarkerSize',20,'MarkerFaceColor','g')
    str=char('CILJ');
    text(6.4,6,0.9,str,'FontSize',15)
    surf(zid_x,zid_y,zid_z,c)
    plot3(x_koord(1,:),y_koord(1,:),z_koord(1,:),'Linewidth',2)

plot3(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),z_koord(1,duljina
_gena+1),'bo','MarkerSize',10,'MarkerFaceColor','b')
    title('3D putanja dobivena osnovnim algoritmom')
    axis([0 6 0 6 0 6]),hold off
    subplot(2,2,2)
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
    %-----------------------------------------------------
end

%   Algoritam za linearno gibanje po koordinatama - fitness sharing 3D -
Glavna petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje za rješenje bez sudara
        broj1=broj1+1;
        if broj1==2
            break
        end
    end

    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord,z_koord] =
koordVekt3D(x_koord,y_koord,z_koord,populacija);

    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2+(cilj_z-z_koord(i,duljina_gena+1))^2);
```

```
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - ovisi o
udaljenosti
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija1;
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:duljina_gena
                for k=1:A
                    if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4

funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1) && z_koord(i,j+1)==z_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
            end
        end
    end
```

```
    if max(fitnes_uk)==100 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk); % fitnes4 - za elitizam -
ukupno
    for i=1:broj_clanova
        if udaljenost(i,1)<=min(fitnes3) && fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end


    % Vjerojatnost izbora člana populacije
     for i=1:broj_clanova
        for j=1:broj_clanova
            d=abs(fitnes_uk(i,:)-fitnes_uk(j,:)); % d je udaljenost između
dva člana
            if d<10 % sigma vrijednost je postavljena na 10
                sh(i,j)=1-(d/10); % ako je d  manji od sigme
            else
                sh(i,j)=0;
            end
        end
    end
    fitnes_uk_sh=fitnes_uk./sum(sh,2);
    vjerojatnost_sh=fitnes_uk_sh/sum(fitnes_uk_sh);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost_sh(j);
            if SUM>r
                roditelj(i,:)=populacija(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelj(u:t,:); %parovi gena
        if rand<=pc
            p1=par_trenutni(1,:);
            p2=par_trenutni(2,:);
            qq=randi([1,duljina_gena-1]);
            P11=[p1(1:qq) p2(qq+1:duljina_gena)];
            P22=[p2(1:qq) p1(qq+1:duljina_gena)];
            par_trenutni_krizani=[P11;P22];
        else
            par_trenutni_krizani=par_trenutni; % ako nema križanja
        end
        nova_pop(u:t,:)=par_trenutni_krizani;
    end
```

```
    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                nova_pop(i,j)=randi(6);
            end
        end
    end

    % Slaganje nove populacije
    populacija=nova_pop;
    populacija(1,:)=elitni;

    % Graf
    figure(1)
    subplot(2,2,[1,3])

plot3(start_x,start_y,start_z,'ko','MarkerSize',20,'MarkerFaceColor','m'),h
old on
    plot3(cilj_x,cilj_y,cilj_z,'ko','MarkerSize',20,'MarkerFaceColor','g')
    surf(zid_x,zid_y,zid_z,c)
    plot3(x_koord(1,:),y_koord(1,:),z_koord(1,:),'Linewidth',2)

plot3(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),z_koord(1,duljina
_gena+1),'bo','MarkerSize',10,'MarkerFaceColor','b')
    title('3D putanja dobivena uz dodatak fitness sharing-a')
    axis([0 6 0 6 0 6]),hold off
    subplot(2,2,2)
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
end

%   Algoritam za linearno gibanje po koordinatama - crowding 3D - Glavna
petlja
for trenutna_generacija=1:broj_generacija
    if  max(fitnes_uk)==100 % Prekidanje petlje za rješenje bez sudara
        broj1=broj1+1;
        if broj1==2
```

```
            break
        end
    end

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=randi(broj_clanova+1-i);   % iz populacije se nasumično izabiru
roditelji
        roditelji(i,:)=populacija(r,:);
        populacija(r,:)=[];
    end

    % Križanje
    for u=1:2:broj_clanova
        t=u+1;
        par_trenutni=roditelji(u:t,:); %parovi gena
        p1=par_trenutni(1,:);
        p2=par_trenutni(2,:);
        qq=randi([1,duljina_gena-1]);
        P11=[p1(1:qq) p2(qq+1:duljina_gena)];
        P22=[p2(1:qq) p1(qq+1:duljina_gena)];
        par_trenutni_krizani=[P11;P22];
        djeca(u:t,:)=par_trenutni_krizani;
    end

    % Mutacija
    for i=1:broj_clanova
        for j=1:duljina_gena
            if rand<=pm
                djeca(i,j)=randi(6);
            end
        end
    end

    % Postavljanje koordinata putanje u vektore
    [x_koord_r,y_koord_r,z_koord_r] =
koordVekt3D(x_koord_r,y_koord_r,z_koord_r,roditelji);
    [x_koord_d,y_koord_d,z_koord_d] =
koordVekt3D(x_koord_d,y_koord_d,z_koord_d,djeca);

    %Fitnes
    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova  % za roditelje
        udaljenost_r(i,:)=sqrt((cilj_x-
x_koord_r(i,duljina_gena+1))^2+(cilj_y-
y_koord_r(i,duljina_gena+1))^2+(cilj_z-z_koord_r(i,duljina_gena+1))^2);
    end
    fitnes1_r=(nagrada1+udaljenost_r)./(1+udaljenost_r); % Fitnes 1 - ovisi
o udaljenosti

    for i=1:broj_clanova % za djecu
        udaljenost_d(i,:)=sqrt((cilj_x-
x_koord_d(i,duljina_gena+1))^2+(cilj_y-
y_koord_d(i,duljina_gena+1))^2+(cilj_z-z_koord_d(i,duljina_gena+1))^2); %
za roditelje
    end
    fitnes1_d=(nagrada1+udaljenost_d)./(1+udaljenost_d); % Fitnes 1 - ovisi
o udaljenosti

    % Kažnjavanje udaranja u zid
    funkcija1_r=zeros(broj_clanova,duljina_gena); % za roditelje
```

```
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord_r(i,j+1)>=2 && x_koord_r(i,j+1)<=4 &&
y_koord_r(i,j+1)>=2 && y_koord_r(i,j+1)<=4 && z_koord_r(i,j+1)>=2 &&
z_koord_r(i,j+1)<=4
                    funkcija1_r(i,j)=1;
                end
            end
        end
    end
    funkcija2_r=sum(funkcija1_r,2);
    fitnes2_r=nagrada2-nagrada2*funkcija2_r./duljina_gena;  % Fitnes 2 -
ovisi o udaranju u zid
    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3_r=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2_r(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord_r(i,j+1)>=2 && x_koord_r(i,j+1)<=4 &&
y_koord_r(i,j+1)>=2 && y_koord_r(i,j+1)<=4 && z_koord_r(i,j+1)>=2 &&
z_koord_r(i,j+1)<=4

funkcija3_r(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3_r(i,j)>0
                    break
                end
            end
            funkcija4_r=sum(funkcija3_r,2);
            fitnes2_r(i,:)=fitnes2_r(i,:)+funkcija4_r(i,:);
        end
    end
    fitnes_uk_r=fitnes1_r+fitnes2_r; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord_r(i,j+1)==x_koord_r(i,j-1) &&
y_koord_r(i,j+1)==y_koord_r(i,j-1) && z_koord_r(i,j+1)==z_koord_r(i,j-1)
                fitnes_uk_r(i,:)=fitnes_uk_r(i,:)-kzv;
                break
            end
        end
    end
    funkcija1_d=zeros(broj_clanova,duljina_gena); % za djecu
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord_d(i,j+1)>=2 && x_koord_d(i,j+1)<=4 &&
y_koord_d(i,j+1)>=2 && y_koord_d(i,j+1)<=4 && z_koord_d(i,j+1)>=2 &&
z_koord_d(i,j+1)<=4
                    funkcija1_d(i,j)=1;
                end
            end
        end
    end
```

```
    funkcija2_d=sum(funkcija1_d,2);
    fitnes2_d=nagrada2-nagrada2*funkcija2_d./duljina_gena;  % Fitnes 2 -
ovisi o udaranju u zid
    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3_d=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2_d(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord_d(i,j+1)>=2 && x_koord_d(i,j+1)<=4 &&
y_koord_d(i,j+1)>=2 && y_koord_d(i,j+1)<=4 && z_koord_d(i,j+1)>=2 &&
z_koord_d(i,j+1)<=4

funkcija3_d(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3_d(i,j)>0
                    break
                end
            end
            funkcija4_d=sum(funkcija3_d,2);
            fitnes2_d(i,:)=fitnes2_d(i,:)+funkcija4_d(i,:);
        end
    end
    fitnes_uk_d=fitnes1_d+fitnes2_d; % Ukupni fitnes
    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord_d(i,j+1)==x_koord_d(i,j-1) &&
y_koord_d(i,j+1)==y_koord_d(i,j-1) && z_koord_d(i,j+1)==z_koord_d(i,j-1)
                fitnes_uk_d(i,:)=fitnes_uk_d(i,:)-kzv;
                break
            end
        end
    end

    %Usporedba
    rod_za_usp=bi2de(roditelji,10,'left-msb'); % kombinaciji gena se
dodjeljuje vrijednost koja se koristi za usporedbu
    dijete_za_usp=bi2de(djeca,10,'left-msb');

    for u=1:2:broj_clanova
        t=u+1;
        d1=abs(rod_za_usp(u)-dijete_za_usp(u)); % trazi se udaljenost
između djece i roditelja
        d2=abs(rod_za_usp(t)-dijete_za_usp(t));
        d3=abs(rod_za_usp(u)-dijete_za_usp(t));
        d4=abs(rod_za_usp(t)-dijete_za_usp(u));
        if (d1+d2)<=(d3+d4) % u ovom slučaju uspoređuju se r1-d1 i r2-d2
            if fitnes_uk_r(u)>=fitnes_uk_d(u)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(u,:);
            end
            if fitnes_uk_r(t)>=fitnes_uk_d(t)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(t,:);
            end
```

```
        else % u ovom slučaju uspoređuju se r1-d2 i r2-d1
            if fitnes_uk_r(u)>=fitnes_uk_d(t)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(t,:);
            end
            if fitnes_uk_r(t)>=fitnes_uk_d(u)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(u,:);
            end
        end
    end

    % Ubacivanje elitnog iz prošle generacije - samo zbog grafa
    populacija(1,:)=elitni;

    % Testiranje nove populacije
    % Postavljanje koordinata putanje u vektore
    [x_koord,y_koord,z_koord] =
koordVekt3D(x_koord,y_koord,z_koord,populacija);

    % Računanje udaljenosti krajnje pozicije robota od cilja
    for i=1:broj_clanova
        udaljenost(i,:)=sqrt((cilj_x-x_koord(i,duljina_gena+1))^2+(cilj_y-
y_koord(i,duljina_gena+1))^2+(cilj_z-z_koord(i,duljina_gena+1))^2);
    end
    fitnes1=(nagrada1+udaljenost)./(1+udaljenost); % Fitnes 1 - ovisi o
udaljenosti
    if min(udaljenost)==0 && broj2==0 % za graf
        za_graf1=trenutna_generacija;
        broj2=broj2+1;
    end

    % Kažnjavanje udaranja u zid
    funkcija1=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        for j=1:duljina_gena
            for k=1:A
                if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4
                    funkcija1(i,j)=1;
                end
            end
        end
    end
    funkcija1;
    funkcija2=sum(funkcija1,2);
    fitnes2=nagrada2-nagrada2*funkcija2./duljina_gena;

    % ovo je samo za slučaj u kojem dolazi do kolizije pri čemu se daje
dodatna nagrada ako je do sudara došlo u kasnijem koraku
    funkcija3=zeros(broj_clanova,duljina_gena);
    for i=1:broj_clanova
        if fitnes2(i,:)<nagrada2
            for j=1:(duljina_gena-5)
                for k=1:A
                    if x_koord(i,j+1)>=2 && x_koord(i,j+1)<=4 &&
y_koord(i,j+1)>=2 && y_koord(i,j+1)<=4 && z_koord(i,j+1)>=2 &&
z_koord(i,j+1)<=4
```

```
funkcija3(i,j)=(nagrada2/duljina_gena)*(j/duljina_gena);
                        break
                    end
                end
                if funkcija3(i,j)>0
                    break
                end
            end
            funkcija4=sum(funkcija3,2);
            fitnes2(i,:)=fitnes2(i,:)+funkcija4(i,:);
        end
    end
    fitnes_uk=fitnes1+fitnes2; % Ukupni fitnes

    %kazna za vraćanje
    for i=1:broj_clanova
        for j=2:duljina_gena
            if x_koord(i,j+1)==x_koord(i,j-1) &&
y_koord(i,j+1)==y_koord(i,j-1) && z_koord(i,j+1)==z_koord(i,j-1)
                fitnes_uk(i,:)=fitnes_uk(i,:)-kzv;
                break
            end
        end
    end

    if max(fitnes_uk)==100 % za graf
        za_graf2=trenutna_generacija;
        broj3=broj3+1;
    end

    % Elitizam
    fitnes3(trenutna_generacija)=min(udaljenost);
    fitnes4(trenutna_generacija)=max(fitnes_uk);
    for i=1:broj_clanova
        if udaljenost(i,1)<=min(fitnes3) && fitnes_uk(i)>=max(fitnes4)
            elitni=populacija(i,:);
            break
        end
    end

    % Graf
    figure(2)
    subplot(2,2,[1,3])

plot3(start_x,start_y,start_z,'ko','MarkerSize',20,'MarkerFaceColor','m'),h
old on
    plot3(cilj_x,cilj_y,cilj_z,'ko','MarkerSize',20,'MarkerFaceColor','g')
    surf(zid_x,zid_y,zid_z,c)
    plot3(x_koord(1,:),y_koord(1,:),z_koord(1,:),'Linewidth',2)

plot3(x_koord(1,duljina_gena+1),y_koord(1,duljina_gena+1),z_koord(1,duljina
_gena+1),'bo','MarkerSize',10,'MarkerFaceColor','b')
    title('3D putanja dobivena uz dodatak crowding-a')
    axis([0 6 0 6 0 6]),hold off
    grid on
    subplot(2,2,2)
    x1=linspace(0,trenutna_generacija,trenutna_generacija);
    y1=fitnes4;
    plot(x1,y1,'Linewidth',3), grid on
    xlabel('Broj generacija');
```

```
    title('Fitnes najboljeg člana populacije');
    axis([0 trenutna_generacija 0 100.5]);
    subplot(2,2,4)
    cla
    str=char('Fitnes:',num2str(fitnes4(trenutna_generacija)));
    text(0.1,5,0.9,str,'FontSize',14)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.1,3.4,0.9,str,'FontSize',14)
    if min(udaljenost)==0
        str=char('Generacija u kojoj je prvi put dostignut
cilj:',num2str(za_graf1));
        text(0.1,1.8,0.9,str,'FontSize',14)
    end
    if max(fitnes_uk)==100
        str=char('Generacija u kojoj je dostignut cilj bez
sudara:',num2str(za_graf2));
        text(0.1,0.2,0.9,str,'FontSize',14)
    end
    axis([0 5 0 5]),axis off
    pause(0.05);
end

%   Funkcija za postavljanje koordinata putanja u vektore za 3D prostor
function [x_koord,y_koord,z_koord] =
koordVekt3D(x_koord,y_koord,z_koord,populacija)
%   koordVekt3D
    for i=1:size(populacija,1)
        for j=1:size(populacija,2)
            x_koord(i,j+1)=x_koord(i,j);
            y_koord(i,j+1)=y_koord(i,j);
            z_koord(i,j+1)=z_koord(i,j);
            if populacija(i,j)==1
                x_koord(i,j+1)=x_koord(i,j+1)+1; % za 1 robot ide + po x
osi
            elseif populacija(i,j)==2
                x_koord(i,j+1)=x_koord(i,j+1)-1; % za 2 robot ide - po x
osi
            elseif populacija(i,j)==3
                y_koord(i,j+1)=y_koord(i,j+1)+1; % za 3 robot ide + po y
osi
            elseif populacija(i,j)==4
                y_koord(i,j+1)=y_koord(i,j+1)-1; % za 4 robot ide - po y
osi
            elseif populacija(i,j)==5
                z_koord(i,j+1)=z_koord(i,j+1)+1; % za 5 robot ide + po z
osi
            elseif populacija(i,j)==6
                z_koord(i,j+1)=z_koord(i,j+1)-1; % za 6 robot ide - po z
osi
            end
        end
    end
end

%   Inicijalizacija algoritma za gibanje po polinomu - 2D
global zid_x zid_y

broj_clanova=50;  %Broj clanova u populaciji
broj_dimenzija=7; % Broj dimenzija i ujedno broj gena
broj_generacija=500; % Ograničenje trajanja simulacije
pc=0.7; % Vjerojatnost križanja
```

```
pm=0.5; % Vjerojatnost mutacije
beta=0.5; % Utjecaj mutacije - ne vise od 0.5
A1=5; % Najveća apsolutna vrijednost koeficijenta polinoma

if mod (broj_clanova,2)~=0
    disp('Greška, prekid programa')
end

% Zidovi
zid_x=[   0    1;   0   1;  0.3 0.3; 0.7 0.7;   0 0.7;    0    0.000001;
1   1];
zid_y=[-0.2 -0.2; 1.2 1.2; -0.2 0.4; 0.2 0.8; 0.8 0.8;  0.2         1.2; -
0.2 0.8];
for i=1:size(zid_x,1)
    zid(i,:)=polyfit(zid_x(i,:),zid_y(i,:),1); % pretvara koordinate zida u
polinom
end

DG=0; % X koordinate Početne točke
GG=1; % X koordinate krajnje točke
diskr=(GG-DG)/100; % Diskretizacija u x smijeru
xx=[DG:diskr:GG]; % Sve točke u x smijeru
lx=length(xx); % Broj tocaka u x smijeru
SUM=0;
fmax=0;

% INICIJALIZACIJA POPULACIJE
roditelji=zeros(broj_clanova,broj_dimenzija);
djeca=zeros(broj_clanova,broj_dimenzija);
for i=1:broj_dimenzija
    populacija(:,i)=-A1+rand(broj_clanova,1)*2*A1; % početna populacija
    xxx(i,:)=xx.^(i); % x^1 x^2 x^3 .....
end
elitni=populacija(1,:); % početni elitni

%   Algoritam za gibanje po polinomu - osnovni 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Promjena populacije - ako algoritam zapne na pocetku
    if trenutna_generacija>1 && min(kazna)~=0 &&
mod(trenutna_generacija,50)==0  % ako se unutar 50 generacija ne pronađe
clan s kaznom 0...
        for i=1:broj_dimenzija
            populacija(:,i)=-A1+rand(broj_clanova,1)*2*A1; % ...onda se
populacija odbacuje i zamjenjuje
        end
    end

    % Testiranje fitnesa populacije
    kazna=provjeraSudara2D(populacija,zid);
    fitnes=fitnesFunkcija2D(populacija,xx,xxx,kazna);

    % ELITIZAM
    [fmax,index]=max(fitnes);
    elitni=populacija(index,:);

    % Vjerojatnost izbora člana populacije
    vjerojatnost=fitnes/sum(fitnes);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
```

```
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelji(i,:)=populacija(j,:);
                SUM=0;
                break;
            end
        end
    end

    % Križanje i mutacija
    djeca = varijacija2D(roditelji,pc,pm,beta);

    % Slaganje nove populacije
    populacija=djeca;
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
    populacija(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija(:,1:namjesteni_clan-1),2)-
sum(populacija(:,namjesteni_clan+1:end),2);

    % ELITIZAM
    elit=randi([1,broj_clanova]);
    populacija(elit,:)=elitni;

    % Za graf
    pop_fun=populacija*xxx; % za graf
    pop_fun(:,1)=0;  % y koordinate Početne točke
    pop_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_fun=elitni*xxx; % za graf
    elit_fun(:,1)=0;  % y koordinate Početne točke
    elit_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_kazna=provjeraSudara2D(elitni,zid);
    elit_fitnes=fitnesFunkcija2D(elitni,xx,xxx,elit_kazna);
    elit_duljina=sum(sqrt(diff(elit_fun).^2+diff(xx).^2));

    % Graf
    figure(1)
    subplot(1,3,1)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,pop_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'),hold off,axis off
    ylabel('f(x)')
    title('Sve putanje')
    subplot(1,3,2)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,elit_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'),hold off,axis off
    ylabel('f(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
```

```
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
end

%   Algoritam za gibanje po polinomu - fitness sharing 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Promjena populacije - ako algoritam zapne na pocetku
    if trenutna_generacija>1 && min(kazna)~=0 &&
mod(trenutna_generacija,50)==0  % ako se unutar 50 generacija ne pronađe
clan s kaznom 0...
        for i=1:broj_dimenzija
            populacija(:,i)=-A1+rand(broj_clanova,1)*2*A1; % ...onda se
populacija odbacuje i zamjenjuje
        end
    end

    % Testiranje fitnesa populacije
    kazna=provjeraSudara2D(populacija,zid);
    fitnes=fitnesFunkcija2D(populacija,xx,xxx,kazna);

    % ELITIZAM
    [fmax,index]=max(fitnes);
    elitni=populacija(index,:);

    % Vjerojatnost izbora člana populacije
    % Fitnes sharing
    for i=1:broj_clanova
        for j=1:broj_clanova
            d=abs(fitnes(i,:)-fitnes(j,:)); % d je udaljenost između dva
člana
            if d<10 % sigma vrijednost je postavljena na 10
                sh(i,j)=1-(d/10); % ako je d manji od sigme
            else
                sh(i,j)=0;
            end
        end
    end
    fitnes_sh=fitnes./sum(sh,2);
    vjerojatnost=fitnes_sh/sum(fitnes_sh);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelji(i,:)=populacija(j,:);
                SUM=0;
                break;
            end
        end
    end
```

```
    % Križanje i mutacija
    djeca = varijacija2D(roditelji,pc,pm,beta);

    % Slaganje nove populacije
    populacija=djeca;
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
    populacija(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija(:,1:namjesteni_clan-1),2)-
sum(populacija(:,namjesteni_clan+1:end),2);
    elit=randi([1,broj_clanova]);
    populacija(elit,:)=elitni;

    % Za graf
    pop_fun=populacija*xxx; % za graf
    pop_fun(:,1)=0;  % y koordinate Početne točke
    pop_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_fun=elitni*xxx; % za graf
    elit_fun(:,1)=0;  % y koordinate Početne točke
    elit_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_kazna=provjeraSudara2D(elitni,zid);
    elit_fitnes=fitnesFunkcija2D(elitni,xx,xxx,elit_kazna);
    elit_duljina=sum(sqrt(diff(elit_fun).^2+diff(xx).^2));

    % Graf
    figure(1)
    subplot(1,3,1)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,pop_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'),hold off,axis off
    ylabel('f(x)')
    title('Sve putanje')
    subplot(1,3,2)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,elit_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'),hold off,axis off
    ylabel('f(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
```

```
end

%   Algoritam za gibanje po polinomu - crowding 2D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Promjena populacije - ako algoritam zapne
    if trenutna_generacija>1 && min(kazna)~=0 &&
mod(trenutna_generacija,50)==0  % ako se unutar 50 generacija ne pronađe
clan s kaznom 0...
        for i=1:broj_dimenzija
            populacija(1:(broj_clanova/2),i)=-
A1+rand(broj_clanova/2,1)*2*A1; % ...onda su pola populacije odbacuje i
zamjenjuje
        end
    end

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=randi(broj_clanova+1-i);   % iz populacije se nasumično izabiru
roditelji
        roditelji(i,:)=populacija(r,:);
        populacija(r,:)=[];
    end

    % Križanje i mutacija
    djeca = varijacija2D(roditelji,pc,pm,beta);

    % Fitnes
    kazna=provjeraSudara2D(roditelji,zid); % Za roditelje
    fitnes_r=fitnesFunkcija2D(roditelji,xx,xxx,kazna); % Za roditelje
    kazna=provjeraSudara2D(djeca,zid); % Za djecu
    fitnes_d=fitnesFunkcija2D(djeca,xx,xxx,kazna); % Za djecu

    % Usporedba roditelja i djece
    for u=1:2:broj_clanova
        t=u+1;
        d1=(abs(roditelji(u,:)-djeca(u,:)));
        d2=(abs(roditelji(t,:)-djeca(t,:)));
        d3=(abs(roditelji(u,:)-djeca(t,:)));
        d4=(abs(roditelji(t,:)-djeca(u,:)));
        if mean(d1+d2)<=mean(d3+d4) % uspoređuju se r1-d1 i r2-d2
            if fitnes_r(u)>=fitnes_d(u)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(u,:);
            end
            if fitnes_r(t)>=fitnes_d(t)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(t,:);
            end
        else % u ovom slučaju uspoređuju se r1-d2 i r2-d1
            if fitnes_r(u)>=fitnes_d(t)
                populacija(u,:)=roditelji(u,:);
            else
                populacija(u,:)=djeca(t,:);
            end
            if fitnes_r(t)>=fitnes_d(u)
                populacija(t,:)=roditelji(t,:);
            else
                populacija(t,:)=djeca(u,:);
            end
```

```
        end
    end

    % Podešavanje polinoma (da završi u željenoj tocki)
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
    populacija(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija(:,1:namjesteni_clan-1),2)-
sum(populacija(:,namjesteni_clan+1:end),2);

    % Testiranje fitnesa nove populacije
    kazna=provjeraSudara2D(populacija,zid);
    fitnes=fitnesFunkcija2D(populacija,xx,xxx,kazna);
    [fmax(trenutna_generacija),index]=max(fitnes);
    if trenutna_generacija>1 && fmax(trenutna_generacija)>=max(fmax)
        elitni=populacija(index,:);
    end

    % Za graf
    pop_fun=populacija*xxx; % za graf
    pop_fun(:,1)=0;  % y koordinate Početne točke
    pop_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_fun=elitni*xxx; % za graf
    elit_fun(:,1)=0;  % y koordinate Početne točke
    elit_fun(:,size(xxx,2))=1; % y koordinate krajnje točke
    elit_kazna=provjeraSudara2D(elitni,zid);
    elit_fitnes=fitnesFunkcija2D(elitni,xx,xxx,elit_kazna);
    elit_duljina=sum(sqrt(diff(elit_fun).^2+diff(xx).^2));

    % Graf
    figure(1)
    subplot(1,3,1)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,pop_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'),hold off
    ylabel('f(x)')
    title('Sve putanje')
    subplot(1,3,2)
    plot(0,0,'ko','MarkerSize',15,'MarkerFaceColor','m'),hold on
    plot(1,1,'ko','MarkerSize',15,'MarkerFaceColor','g')
    for i=1:size(zid_x,1)
        plot(zid_x(i,:),zid_y(i,:),'k','Linewidth',3)
    end
    plot(xx,elit_fun,'Linewidth',2)
    axis([0 1 -0.2 1.2]);
    xlabel('x'), hold off,axis off
    ylabel('f(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
```

```
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
end

%   Funkcija za Križanje roditelja i mutaciju potomaka za 2D algoritme
function djeca = varijacija2D(roditelji,pc,pm,beta)
    % Križanje
    for u=1:2:size(roditelji,1)
        t=u+1;
        par_trenutni=roditelji(u:t,:); %parovi gena
        for i=1:size(roditelji,2)
            if rand<=pc
                alfa=rand;
                par_trenutni_krizani(1,i)=alfa*par_trenutni(1,i)+(1-
alfa)*par_trenutni(2,i);
                par_trenutni_krizani(2,i)=alfa*par_trenutni(2,i)+(1-
alfa)*par_trenutni(1,i);
            else
                par_trenutni_krizani(:,i)=par_trenutni(:,i); % ako nema
križanja kopiraju se roditelji
            end
        end
        djeca(u:t,:)=par_trenutni_krizani;
    end
    % Mutacija
    for i=1:size(roditelji,1)
        for j=1:size(roditelji,2)
            if rand<=pm
                djeca(i,j)=djeca(i,j)*(1-beta+rand*2*beta);
            end
        end
    end
end

%   Funkcija za provjeru sudara za zidovima 2D
function kazna = provjeraSudara2D(populacija,zid)
% provjeraSudara2D
    global zid_x zid_y
    polinom_putanje=fliplr(populacija); %Okreće se redoslijed polinoma x
x^2 x^3... ----> ...x^3 x^2 x

polinom_putanje(:,size(polinom_putanje,2)+1)=zeros(size(polinom_putanje,1),
1); % stavlja nulu na mjesto 0-tog koeficijenta u polinomu
    polinom_zida=zeros(size(zid,1),size(polinom_putanje,2));
    polinom_zida(:,end-1:end)=zid; % pretvara polinom zida u red polinoma
putanje robota
    kazna=zeros(size(polinom_putanje,1),1);
    for i=1:size(polinom_putanje,1)
        sudar=0;
        for j=1:size(polinom_zida,1)
            if abs(zid(j,1))>10000
                for k=1:size(populacija,2)
                    X(k,:)=zid_x(j,1)^k;
                end
                Y=populacija(i,:)*X;
                if zid_y(j,1)<Y && Y<zid_y(j,2)
                    sudar=sudar+90;
                    break;
                end
```

```
            end
            razlika=polinom_putanje(i,:)-polinom_zida(j,:);
            sjeciste=roots(razlika);
            for k=1:size(sjeciste,1)
                if isreal(sjeciste(k,1)) && sjeciste(k,1)>zid_x(j,1) &&
sjeciste(k,1)<zid_x(j,2)
                    sudar=sudar+100;
                end
                if sudar>0
                    break;
                end
            end
            if sudar>0
                break;
            end
        end
        kazna(i)=sudar;
    end
end

%   Funkcija za Računanje fitnesa populacije za 2D problem
function fitnes = fitnesFunkcija2D(populacija,xx,xxx,kazna)
    pop_fun=populacija*xxx;
    pop_fun(:,1)=0;  % y koordinate Početne točke
    pop_fun(:,size(xxx,2))=1;   % y koordinate krajnje točke
    for i=1:size(populacija,1)
        duljina(i,:)=sum(sqrt(diff(pop_fun(i,:)).^2+diff(xx).^2));
    end
    fit=1000./(1+kazna+duljina);
    fitnes=fit;
end

%   Inicijalizacija algoritma za gibanje po polinomu - 3D
global zid_x zid_y zid_z

broj_clanova=50;  %Broj clanova u populaciji
broj_dimenzija=7; % Broj dimenzija i ujedno broj gena
broj_generacija=500; % Ograničenje trajanja simulacije
pc=0.7; % Vjerojatnost križanja
pm=0.5; % Vjerojatnost mutacije
beta=0.5; % Utjecaj mutacije - ne vise od 0.5

if mod (broj_clanova,2)~=0
    disp('Greška, prekid programa')
end

% Zidovi
zid_x=[  -1    2; 0.2  0.8; 0.2 0.5]; % pod, zid i stup
zid_y=[  -1    2; 0.8  0.2; 0.5 0.2];
zid_z=[-0.1 -0.1;-0.1 0.15;-0.1 0.3];

for i=1:size(zid_x,1)
    zid_1(i,:)=polyfit(zid_x(i,:),zid_y(i,:),1); % pretvara koordinate zida
u polinom za y
    zid_2(i,:)=polyfit(zid_x(i,:),zid_z(i,:),1); % pretvara koordinate zida
u polinom za z
end
c = ones(size(zid_x,1),2); % za graf

DG=0; % X koordinate Početne točke
GG=1; % X koordinate krajnje točke
```

```
diskr=(GG-DG)/100; % Diskretizacija u x smijeru
xx=[DG:diskr:GG]; % Sve točke u x smijeru
lx=length(xx); % Broj tocaka u x smijeru
A1=5; % Najveća apsolutna vrijednost koeficijenta polinoma

% INICIJALIZACIJA
roditelji_y=zeros(broj_clanova,broj_dimenzija);
roditelji_z=zeros(broj_clanova,broj_dimenzija);
djeca_y=zeros(broj_clanova,broj_dimenzija);
djeca_z=zeros(broj_clanova,broj_dimenzija);
SUM=0;
fmax=0; % treba resetirati

for i=1:broj_dimenzija
    populacija_y(:,i)=-A1+rand(broj_clanova,1)*2*A1; % početna populacija -
y
    populacija_z(:,i)=-A1+rand(broj_clanova,1)*2*A1; % početna populacija -
z
    xxx(i,:)=xx.^(i); %  x^1 x^2 x^3 .....
end
elitni_y=populacija_y(1,:); % početni elitni
elitni_z=populacija_z(1,:);

%   Algoritam za gibanje po polinomu - osnovni 3D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Testiranje fitnesa populacije
    kazna=provjeraSudara3D(populacija_y,populacija_z,zid_1,zid_2);
    fitnes=fitnesFunkcija3D(populacija_y,populacija_z,xx,xxx,kazna);

    % Elitizam
    [fmax,index]=max(fitnes);
    elitni_y=populacija_y(index,:);
    elitni_z=populacija_z(index,:);

    % Vjerojatnost izbora člana populacije
    vjerojatnost=fitnes/sum(fitnes);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelji_y(i,:)=populacija_y(j,:);
                roditelji_z(i,:)=populacija_z(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje i mutacija
    [djeca_y,djeca_z] = varijacija3D(roditelji_y,roditelji_z,pc,pm,beta);

    % Slaganje nove populacije
    populacija_y=djeca_y;
    populacija_z=djeca_z;
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
```

```
    populacija_y(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija_y(:,1:namjesteni_clan-1),2)-
sum(populacija_y(:,namjesteni_clan+1:end),2);
    populacija_z(:,namjesteni_clan)=-sum(populacija_z(:,1:namjesteni_clan-
1),2)-sum(populacija_z(:,namjesteni_clan+1:end),2); % z završava u z=0 pa
zato nema ones

    % Elitizam
    elit=randi([1,broj_clanova]);
    populacija_y(elit,:)=elitni_y;
    populacija_z(elit,:)=elitni_z;

    % Za graf
    pop_fun_y=populacija_y*xxx; % za graf
    pop_fun_y(:,1)=0;  % y koordinate Početne točke
    pop_fun_y(:,size(xxx,2))=1;
    pop_fun_z=populacija_z*xxx; % za graf
    pop_fun_z(:,1)=0;  % z koordinate Početne točke
    pop_fun_z(:,size(xxx,2))=0;
    elit_fun_y=elitni_y*xxx;
    elit_fun_y(:,1)=0;
    elit_fun_y(:,size(xxx,2))=1;
    elit_fun_z=elitni_z*xxx;
    elit_fun_z(:,1)=0;
    elit_fun_z(:,size(xxx,2))=0;
    elit_kazna=provjeraSudara3D(elitni_y,elitni_z,zid_1,zid_2);
    elit_fitnes=fitnesFunkcija3D(elitni_y,elitni_z,xx,xxx,elit_kazna);

elit_duljina=sum(sqrt(diff(elit_fun_y).^2+diff(elit_fun_z).^2+diff(xx).^2))
;

    % Graf
    figure(1)
    subplot(1,3,1)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,pop_fun_y,pop_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Putanje')
    subplot(1,3,2)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
```

```
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,elit_fun_y,elit_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
end

%   Algoritam za gibanje po polinomu - fitness sharing 3D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Testiranje fitnesa populacije
    kazna=provjeraSudara3D(populacija_y,populacija_z,zid_1,zid_2);
    fitnes=fitnesFunkcija3D(populacija_y,populacija_z,xx,xxx,kazna);

    % Elitizam
    [fmax,index]=max(fitnes);
    elitni_y=populacija_y(index,:);
    elitni_z=populacija_z(index,:);

    % Vjerojatnost izbora člana populacije
    % Fitnes sharing
    for i=1:broj_clanova
        for j=1:broj_clanova
            d=abs(fitnes(i,:)-fitnes(j,:)); % d je udaljenost između dva
člana
            if d<10 % sigma vrijednost je postavljena na 10
                sh(i,j)=1-(d/10); % ako je d manji od sigme
            else
                sh(i,j)=0;
            end
        end
    end
    fitnes_sh=fitnes./sum(sh,2);
    vjerojatnost=fitnes_sh/sum(fitnes_sh);

    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=rand;
        for j=1:broj_clanova
```

```
            SUM=SUM+vjerojatnost(j);
            if SUM>r
                roditelji_y(i,:)=populacija_y(j,:);
                roditelji_z(i,:)=populacija_z(j,:);
                SUM=0;
                break
            end
        end
    end

    % Križanje i mutacija
    [djeca_y,djeca_z] = varijacija3D(roditelji_y,roditelji_z,pc,pm,beta);

    % Slaganje nove populacije
    populacija_y=djeca_y;
    populacija_z=djeca_z;
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
    populacija_y(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija_y(:,1:namjesteni_clan-1),2)-
sum(populacija_y(:,namjesteni_clan+1:end),2);
    populacija_z(:,namjesteni_clan)=-sum(populacija_z(:,1:namjesteni_clan-
1),2)-sum(populacija_z(:,namjesteni_clan+1:end),2);

    % Elitizam
    elit=randi([1,broj_clanova]);
    populacija_y(elit,:)=elitni_y;
    populacija_z(elit,:)=elitni_z;

    % Za graf
    pop_fun_y=populacija_y*xxx; % za graf
    pop_fun_y(:,1)=0;  % y koordinate Početne točke
    pop_fun_y(:,size(xxx,2))=1;
    pop_fun_z=populacija_z*xxx; % za graf
    pop_fun_z(:,1)=0;  % z koordinate Početne točke
    pop_fun_z(:,size(xxx,2))=0;
    elit_fun_y=elitni_y*xxx;
    elit_fun_y(:,1)=0;
    elit_fun_y(:,size(xxx,2))=1;
    elit_fun_z=elitni_z*xxx;
    elit_fun_z(:,1)=0;
    elit_fun_z(:,size(xxx,2))=0;
    elit_kazna=provjeraSudara3D(elitni_y,elitni_z,zid_1,zid_2);
    elit_fitnes=fitnesFunkcija3D(elitni_y,elitni_z,xx,xxx,elit_kazna);

elit_duljina=sum(sqrt(diff(elit_fun_y).^2+diff(elit_fun_z).^2+diff(xx).^2))
;

    % Graf
    figure(1)
    subplot(1,3,1)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
```

```
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,pop_fun_y,pop_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Putanje')
    subplot(1,3,2)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,elit_fun_y,elit_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
end

%   Algoritam za gibanje po polinomu - crowding 3D - Glavna petlja
for trenutna_generacija=1:broj_generacija
    % Generiranje parova roditelja
    for i=1:broj_clanova
        r=randi(broj_clanova+1-i);   % iz populacije se nasumično izabiru
roditelji
        roditelji_y(i,:)=populacija_y(r,:);
        populacija_y(r,:)=[];
        roditelji_z(i,:)=populacija_z(r,:);
        populacija_z(r,:)=[];
    end

    % Križanje i mutacija
    [djeca_y,djeca_z] = varijacija3D(roditelji_y,roditelji_z,pc,pm,beta);
```

```
% Fitnes
kazna=provjeraSudara3D(roditelji_y,roditelji_z,zid_1,zid_2);
fitnes_r=fitnesFunkcija3D(roditelji_y,roditelji_z,xx,xxx,kazna);
kazna=provjeraSudara3D(djeca_y,djeca_z,zid_1,zid_2);
fitnes_d=fitnesFunkcija3D(djeca_y,djeca_z,xx,xxx,kazna);

% Usporedba roditelja i djece
for u=1:2:broj_clanova
    t=u+1;
    d1_y=(abs(roditelji_y(u,:)-djeca_y(u,:)));
    d2_y=(abs(roditelji_y(t,:)-djeca_y(t,:)));
    d3_y=(abs(roditelji_y(u,:)-djeca_y(t,:)));
    d4_y=(abs(roditelji_y(t,:)-djeca_y(u,:)));
    if mean(d1_y+d2_y)<=mean(d3_y+d4_y) % uspoređuju se r1-d1 i r2-d2
        if fitnes_r(u)>=fitnes_d(u)
            populacija_y(u,:)=roditelji_y(u,:);
        else
            populacija_y(u,:)=djeca_y(u,:);
        end
        if fitnes_r(t)>=fitnes_d(t)
            populacija_y(t,:)=roditelji_y(t,:);
        else
            populacija_y(t,:)=djeca_y(t,:);
        end
    else % u ovom slučaju uspoređuju se r1-d2 i r2-d1
        if fitnes_r(u)>=fitnes_d(t)
            populacija_y(u,:)=roditelji_y(u,:);
        else
            populacija_y(u,:)=djeca_y(t,:);
        end
        if fitnes_r(t)>=fitnes_d(u)
            populacija_y(t,:)=roditelji_y(t,:);
        else
            populacija_y(t,:)=djeca_y(u,:);
        end
    end
    d1_z=(abs(roditelji_z(u,:)-djeca_z(u,:)));
    d2_z=(abs(roditelji_z(t,:)-djeca_z(t,:)));
    d3_z=(abs(roditelji_z(u,:)-djeca_z(t,:)));
    d4_z=(abs(roditelji_z(t,:)-djeca_z(u,:)));
    if mean(d1_z+d2_z)<=mean(d3_z+d4_z) % uspoređuju se r1-d1 i r2-d2
        if fitnes_r(u)>=fitnes_d(u)
            populacija_z(u,:)=roditelji_z(u,:);
        else
            populacija_z(u,:)=djeca_z(u,:);
        end
        if fitnes_r(t)>=fitnes_d(t)
            populacija_z(t,:)=roditelji_z(t,:);
        else
            populacija_z(t,:)=djeca_z(t,:);
        end
    else % u ovom slučaju uspoređuju se r1-d2 i r2-d1
        if fitnes_r(u)>=fitnes_d(t)
            populacija_z(u,:)=roditelji_z(u,:);
        else
            populacija_z(u,:)=djeca_z(t,:);
        end
        if fitnes_r(t)>=fitnes_d(u)
            populacija_z(t,:)=roditelji_z(t,:);
        else
```

```
                populacija_z(t,:)=djeca_z(u,:);
            end
        end
    end

    % Podešavanje polinoma u željenu krajnju tocku
    namjesteni_clan=randi([2,broj_dimenzija-1],1); % 1 < namjesteni_clan <
broj_dimenzija
    populacija_y(:,namjesteni_clan)=ones(broj_clanova,1)-
sum(populacija_y(:,1:namjesteni_clan-1),2)-
sum(populacija_y(:,namjesteni_clan+1:end),2);
    populacija_z(:,namjesteni_clan)=-sum(populacija_z(:,1:namjesteni_clan-
1),2)-sum(populacija_z(:,namjesteni_clan+1:end),2);

    % Testiranje fitnesa nove populacije
    kazna=provjeraSudara3D(populacija_y,populacija_z,zid_1,zid_2);
    fitnes=fitnesFunkcija3D(populacija_y,populacija_z,xx,xxx,kazna);
    [fmax(trenutna_generacija),index]=max(fitnes);
    if trenutna_generacija>1 && fmax(trenutna_generacija)>=max(fmax)
        elitni_y=populacija_y(index,:);
        elitni_z=populacija_z(index,:);
    end

    % Za graf
    pop_fun_y=populacija_y*xxx; % za graf
    pop_fun_y(:,1)=0;  % y koordinate Početne točke
    pop_fun_y(:,size(xxx,2))=1;
    pop_fun_z=populacija_z*xxx; % za graf
    pop_fun_z(:,1)=0;  % z koordinate Početne točke
    pop_fun_z(:,size(xxx,2))=0;
    elit_fun_y=elitni_y*xxx;
    elit_fun_y(:,1)=0;
    elit_fun_y(:,size(xxx,2))=1;
    elit_fun_z=elitni_z*xxx;
    elit_fun_z(:,1)=0;
    elit_fun_z(:,size(xxx,2))=0;
    elit_kazna=provjeraSudara3D(elitni_y,elitni_z,zid_1,zid_2);
    elit_fitnes=fitnesFunkcija3D(elitni_y,elitni_z,xx,xxx,elit_kazna);

elit_duljina=sum(sqrt(diff(elit_fun_y).^2+diff(elit_fun_z).^2+diff(xx).^2))
;

    % Graf
    figure(1)
    subplot(1,3,1)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,pop_fun_y,pop_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
```

```
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Putanje')
    subplot(1,3,2)
    plot3(0,0,0,'ko','MarkerSize',10,'MarkerFaceColor','m'),hold on
    plot3(1,1,0,'ko','MarkerSize',10,'MarkerFaceColor','g')
    for j=1:size(zid_x,1)
        if(zid_z(j,1)~=zid_z(j,2))
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
        else
            p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
        end
        p.FaceAlpha=0.5;
    end
    plot3(xx,elit_fun_y,elit_fun_z,'Linewidth',2)
    axis([-1 2 -1 2 -1 2]);
    xlabel('x'),hold off
    ylabel('f1(x)')
    zlabel('f2(x)')
    title('Najbolja putanja')
    subplot(1,3,3)
    cla
    str=char('Duljina:',num2str(elit_duljina));
    text(0.2,0.9,str,'FontSize',15)
    str=char('Kazna:',num2str(elit_kazna));
    text(0.2,0.7,str,'FontSize',15)
    str=char('Fitnes:',num2str(elit_fitnes));
    text(0.2,0.5,str,'FontSize',15)
    str=char('Trenutna generacija:',num2str(trenutna_generacija));
    text(0.2,0.3,str,'FontSize',15)
    axis([0 1 0 1]);axis off
    pause(.05);
end

%   varijacija2D je funkcija koja obavlja Križanje roditelja i mutaciju
function [djeca_y,djeca_z] =
varijacija3D(roditelji_y,roditelji_z,pc,pm,beta)
% Križanje
    for u=1:2:size(roditelji_y,1)
        t=u+1;
        par_trenutni_y=roditelji_y(u:t,:); %parovi gena
        par_trenutni_z=roditelji_z(u:t,:);
        for i=1:size(roditelji_y,2)
            if rand<=pc
                alfa=rand;
                par_trenutni_krizani_y(1,i)=alfa*par_trenutni_y(1,i)+(1-
alfa)*par_trenutni_y(2,i);
                par_trenutni_krizani_y(2,i)=alfa*par_trenutni_y(2,i)+(1-
alfa)*par_trenutni_y(1,i);
                par_trenutni_krizani_z(1,i)=alfa*par_trenutni_z(1,i)+(1-
alfa)*par_trenutni_z(2,i);
                par_trenutni_krizani_z(2,i)=alfa*par_trenutni_z(2,i)+(1-
alfa)*par_trenutni_z(1,i);
            else
                par_trenutni_krizani_y(:,i)=par_trenutni_y(:,i); % ako nema
križanja kopiraju se roditelji
```

```
                par_trenutni_krizani_z(:,i)=par_trenutni_z(:,i);
            end
        end
        djeca_y(u:t,:)=par_trenutni_krizani_y;
        djeca_z(u:t,:)=par_trenutni_krizani_z;
    end
    % Mutacija
    for i=1:size(roditelji_y,1)
        for j=1:size(roditelji_y,2)
            if rand<=pm
                djeca_y(i,j)=djeca_y(i,j)*(1-beta+rand*2*beta);
            end
            if rand<=pm
                djeca_z(i,j)=djeca_z(i,j)*(1-beta+rand*2*beta);
            end
        end
    end
end

%   Funkcija za provjeru sudara za zidovima 3D
function kazna=provjeraSudara3D(populacija_y,populacija_z,zid_1,zid_2)
%   provjeraSudara3D
    global zid_x zid_y zid_z
    polinom_putanje_y=fliplr(populacija_y); %Okreće se redoslijed polinoma
x x^2 x^3... ----> ...x^3 x^2 x
    polinom_putanje_z=fliplr(populacija_z);

polinom_putanje_y(:,size(polinom_putanje_y,2)+1)=zeros(size(polinom_putanje
_y,1),1); % stavlja nulu na mjesto 0-tog koeficijenta u polinomu

polinom_putanje_z(:,size(polinom_putanje_z,2)+1)=zeros(size(polinom_putanje
_z,1),1);
    polinom_zida_1=zeros(size(zid_1,1),size(polinom_putanje_y,2)); %
pretvara polinom zida u red polinoma putanje robota
    polinom_zida_2=zeros(size(zid_2,1),size(polinom_putanje_z,2));
    polinom_zida_1(:,end-1:end)=zid_1;
    polinom_zida_2(:,end-1:end)=zid_2;
    kazna=zeros(size(polinom_putanje_y,1),1);
    for i=1:size(polinom_putanje_y,1)
        sudar=0;
        for j=1:size(polinom_zida_1,1)
            if abs(zid_1(j,1))>10000
                for k=1:size(populacija_y,2)
                    X(k,:)=zid_x(j,1)^k;
                end
                Y=populacija_y(i,:)*X;
                Z=populacija_z(i,:)*X;
                if zid_y(j,1)<Y && Y<zid_y(j,2) && zid_z(j,1)<Z &&
Z<zid_z(j,2)
                    sudar=sudar+90;
                    break;
                end
            end
            razlika=polinom_putanje_y(i,:)-polinom_zida_1(j,:);
            sjeciste=roots(razlika);
            for k=1:size(sjeciste,1)
                if isreal(sjeciste(k,1))
                    for l=1:size(populacija_y,2)
                        X(l,:)=sjeciste(k,1)^l;
                    end
                    Z=populacija_z(i,:)*X;
```

```
                    if sjeciste(k,1)>zid_x(j,1) && sjeciste(k,1)<zid_x(j,2)
&& Z>zid_z(j,1) && Z<zid_z(j,2)
                        sudar=sudar+100;
                        break;
                    end
                end
            end
            if sudar>0
                break;
            end
            razlika=polinom_putanje_z(i,:)-polinom_zida_2(j,:);
            sjeciste=roots(razlika);
            for k=1:size(sjeciste,1)
                if isreal(sjeciste(k,1))
                    for l=1:size(populacija_y,2)
                        X(l,:)=sjeciste(k,1)^l;
                    end
                    Y=populacija_y(i,:)*X;
                    if sjeciste(k,1)>zid_x(j,1) && sjeciste(k,1)<zid_x(j,2)
&& Y>zid_y(j,1) && Y<zid_y(j,2)
                        sudar=sudar+100;
                        break;
                    end
                end
            end
            if sudar>0
                break;
            end
        end
        kazna(i)=sudar;
    end
end

%   Funkcija za Računanje fitnesa populacije za 3D problem
function fitnes=fitnesFunkcija3D(populacija_y,populacija_z,xx,xxx,kazna)
%   Fitnes funkcija 3D
    pop_fun_y=populacija_y*xxx;
    pop_fun_y(:,1)=0;  % y koordinate Početne točke
    pop_fun_y(:,size(xxx,2))=1;
    pop_fun_z=populacija_z*xxx;
    pop_fun_z(:,1)=0;  % z koordinate Početne točke
    pop_fun_z(:,size(xxx,2))=0;
    for i=1:size(populacija_y,1)

duljina(i,:)=sum(sqrt(diff(pop_fun_y(i,:)).^2+diff(pop_fun_z(i,:)).^2+diff(
xx).^2));
    end
    fit=1000./(1+kazna+duljina);
    fitnes=fit;
end

%   Inverzna kinematika robota
yy=elit_fun_y;
zz=elit_fun_z;
L1=0.3; % Duljina prvog članka
L2=0.5; % Duljina drugog članka
L3=0.6; % Duljina trećeg članka

figure(3);
subplot(2,2,4)
plot3(xx,yy,zz,'Color','b','LineWidth',2),hold on
```

```
xlim([-0.1 1.1]);
ylim([-0.1 1.1]);
zlim([-0.1 1.1]);axis vis3d;grid on;
for j=1:size(zid_x,1)
    if(zid_z(j,1)~=zid_z(j,2))
        p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,2) zid_y(j,2) zid_y(j,1)], [zid_z(j,1) zid_z(j,1)
zid_z(j,2) zid_z(j,2)], [0 0 0 0]);
    else
        p=patch( [zid_x(j,1) zid_x(j,2) zid_x(j,2) zid_x(j,1)] ,
[zid_y(j,1) zid_y(j,1) zid_y(j,2) zid_y(j,2)], [zid_z(j,1) zid_z(j,1)
zid_z(j,1) zid_z(j,1)], [0 0 0 0]);
    end
    p.FaceAlpha=0.5;
end
xlabel('x')
ylabel('y')
zlabel('z')
title('Krajnji položaj')
tocka1.x=1; tocka1.y=0; tocka1.z=-0.1;
tocka2.x=1; tocka2.y=0; tocka2.z=0.2;
tocka3.x=1; tocka3.y=0; tocka3.z=0.7;
tocka4.x=2; tocka4.y=0; tocka4.z=1.3;
l1= line([tocka1.x,
tocka2.x],[tocka1.y,tocka2.y],[tocka1.z,tocka2.z],'Color',[0.25,0.25,0.25],
'LineWidth',5);
l2 = line([tocka2.x,
tocka3.x],[tocka2.y,tocka3.y],[tocka2.z,tocka3.z],'Color',[0.31,0.31,0.31],
'LineWidth',5);
l3 = line([tocka3.x,
tocka4.x],[tocka3.y,tocka4.y],[tocka3.z,tocka4.z],'Color',[0.5,0.5,0.5],'Li
neWidth',5);
pause(0.5);
for i=1:size(xx,2)
    [tocka2,tocka3,tocka4] = inverznaKin(xx(i),yy(i),zz(i),L1,L2,L3);

set(l1,'ZData',[tocka1.z,tocka2.z],'YData',[tocka1.y,tocka2.y],'XData',[toc
ka1.x,tocka2.x]);

set(l2,'ZData',[tocka2.z,tocka3.z],'YData',[tocka2.y,tocka3.y],'XData',[toc
ka2.x,tocka3.x]);

set(l3,'ZData',[tocka3.z,tocka4.z],'YData',[tocka3.y,tocka4.y],'XData',[toc
ka3.x,tocka4.x]);
    hold on;
    plot3(tocka4.x,tocka4.y,tocka4.z,'xk');
    pause(0.1);
end

%   Funkcija za inverznu pa onda direktnu kinematiku robota
function [tocka2,tocka3,tocka4] = inverznaKin(x,y,z,L1,L2,L3)
% inverznaKin
    x=x-1;
    q1=atan2(y,x);
    r=sqrt(x^2+y^2);
    s=z-L1;

    D=(r^2+s^2-L2^2-L3^2)/(2*L2*L3);
    % Elbow UP
    q3=acos(D);
    q2=atan2(r,s)-atan2(L3*sin(q3),L2+L3*cos(q3));
```

```
    % Elbow DOWN
%     q3=-acos(D);
%     q2=atan2(r,s)-atan2(L3*sin(q3),L2+L3*cos(q3));

    T2 = rotacija(q1)*translacijaRotacija(q2,L1);
    T3 =
rotacija(q1)*translacijaRotacija(q2,L1)*translacijaRotacija(q3,L2);
    T4 =
rotacija(q1)*translacijaRotacija(q2,L1)*translacijaRotacija(q3,L2)*translac
ija(L3);
    tocka2.x= T2(1,4); tocka2.y= T2(2,4); tocka2.z= T2(3,4);
    tocka3.x= T3(1,4); tocka3.y= T3(2,4); tocka3.z= T3(3,4);
    tocka4.x= T4(1,4); tocka4.y= T4(2,4); tocka4.z= T4(3,4);
end
%   Funkcija za rotaciju kod kinematike robota
function A1 = rotacija(q1)
    A1=[cos(q1) -sin(q1) 0 1;...
        sin(q1)  cos(q1) 0 0;...
           0        0    1 0;...
           0        0    0 1];
end

%   Funkcija za translaciju kod kinematike robota
function A4= translacija(L3)
    A4=[1 0 0  0;...
        0 1 0  0;...
        0 0 1 L3;...
        0 0 0  1];
end

%   Funkcija za translaciju pa rotaciju kod kinematike robota
function A2 = translacijaRotacija(q2,L1)
    A2=[ cos(q2) 0 sin(q2)  0;...
            0    1    0     0;...
        -sin(q2) 0 cos(q2) L1;...
            0     0    0      1];
end
```