Rukovanje ograničenjima prostora u algoritmu diferencijalne evolucije

Radonjić, Dino

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

Sveučilišni diplomski studij Računarstvo

RUKOVANJE OGRANIČENJIMA PROSTORA U ALGORITMU DIFERENCIJALNE EVOLUCIJE

Diplomski rad

Dino Radonjić

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Dino Radonjić
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1321R, 07.10.2022.
JMBAG:	0165082414
Mentor:	doc. dr. sc. Dražen Bajer
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Bruno Zorić
Član Povjerenstva 1:	doc. dr. sc. Dražen Bajer
Član Povjerenstva 2:	dr. sc. Mario Dudjak
Naslov diplomskog rada:	Rukovanje ograničenjima prostora u algoritmu diferencijalne evolucije
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati algoritam diferencijalne evolucije kao vrstu evolucijskih algoritama s naglaskom na numeričku optimizaciju s ograničenjima prostora. Posebno se osvrnuti na mehanizme za rukovanje takvim ograničenjima koji se često ugrađuju u algoritam diferencijalne evolucije. Ugraditi barem četiri inačice algoritma koji se razlikuju u mehanizmu za rukovanje ograničenjima prostora. Eksperimentalno ispitati učinkovitost ugrađenih inačica algoritma na nekoliko standardnih testnih funkcija. Rezervirano za: Dino Radonjić

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	17.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	30.9.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Dino Radonjić
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1321R, 07.10.2022.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Rukovanje ograničenjima prostora u algoritmu diferencijalne evolucije**

izrađen pod vodstvom mentora doc. dr. sc. Dražen Bajer

i sumentora

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

Potpis pristupnika:

SADRŽAJ

1. UVOD

Kontinuirana optimizacija (engl. *continuous optimisation*) predstavlja ključan izazov u različitim inženjerskim zadacima, a u današnje vrijeme često se susreće i u strojnom učenju. Takvi problemi uključuju optimizaciju funkcije cilja (engl. *objective function*), gdje je cilj pronaći optimalna rješenja unutar zadanog prostora pretrage. Oblik funkcije cilja ponekad nije poznat unaprijed ili je toliko složen da nema jednostavnih derivacija, što je često slučaj u problemima simulacije. Za rješavanje takvih problema prvobitno su korištene klasične iterativne metode za numeričku optimizaciju. Ove metode obično nisu pogodne za funkcije cilja s većim brojem lokalnih ekstrema, te često imaju lošije performanse pri problemima s nelinearnim ili višedimenzionalnim prostorima pretrage. Evolucijski algoritmi (engl. *evolutionary algorithms*, EAs) ističu se kao učinkovite metode za rješavanje ovakvih složenih optimizacijskih problema, nudeći određene prednosti poput robusnosti i mogućnosti istraživanja složenih prostora rješenja. Učinkovitost EA znatno ovisi o prilagodbi operatora i parametara određenom optimizacijskom problemu, pa optimalna konfiguracija za pojedini problem nije uvijek očita.

Algoritam diferencijalne evolucije (engl. *differential evolution*, DE) posebno se istaknuo kao jedan od najučinkovitijih evolucijskih algoritama za probleme kontinuirane optimizacije, dok su drugi EA često prikladniji za diskretne optimizacijske probleme. Za razliku od uobičajenih EA, DE koristi jednostavniju mutaciju kroz razliku vektora trenutnih rješenja, te često zahtijeva manji broj parametara što smanjuje kompleksnost podešavanja algoritma. Njegovi operatori mutacije omogućuju generiranje novih rješenja kroz kombinaciju postojećih, što doprinosi učinkovitoj pretrazi prostora rješenja. Međutim, u većini praktičnih optimizacijskih problema, prostor pretrage je ograničen, a operator mutacije može generirati rješenja izvan dopuštenih granica prostora. Rješenja izvan dopuštenih granica smatraju se nevaljanima, te nisu prihvatljiva za daljnju optimizaciju. Takve situacije zahtijevaju primjenu mehanizama za rukovanje ograničenjima prostora (engl. *bound constraint handling*). U literaturi su predloženi različiti mehanizmi za rješavanje ovog problema, ali nije svaki mehanizam jednako prikladan za sve vrste problema. Iz tog razloga odabir odgovarajućeg mehanizma za rukovanje ograničenjima prostora ima može imati značajan utjecaj na performanse algoritma DE.

Drugo poglavlje rada sadrži detaljan opis algoritma DE i njegovih parametara. Također, istaknuta je važnost rukovanja ograničenjima prostora i detaljno su objašnjena četiri mehanizma. Na kraju je prikazana zastupljenost tih mehanizama u radovima iz literature. Treće poglavlje daje opis ostvarenog programskog rješenja, korištenog za eksperimentalnu analizu. Prikazan je dijagram

toka algoritma, te način rada i korištenja programskog rješenja. U četvrtom poglavlju opisana je provedena eksperimentalna analiza. U prvom dijelu opisani su eksperimenti i prikazane korištene postavke, dok su u drugom dijelu prikazani i komentirani rezultati svih provedenih eksperimenata.

2. ALGORITAM DIFERENCIJALNE EVOLUCIJE I MEHANIZMI RUKOVANJA OGRANIČENJIMA PROSTORA

Danas se u mnoštvo grana znanosti pronalaze problemi koji zahtijevaju neku vrstu optimizacije rješenja. To često podrazumijeva optimiziranje određenih svojstava ili izlaznih vrijednosti sustava kontinuiranim izborom prihvatljivih rješenja dok se ne pronađe optimalno rješenje. U ovakvim problemima, rješenja možemo prikazati vektorima čije su varijable obično vrijednosti podskupa realnih brojeva, a prihvatljiva rješenja su ona koja se nalaze unutar određenih granica prostora pretrage. Glavni pristup rješavanju takvih problema je dizajniranje funkcije cilja koja može modelirati ciljeve problema, uzimajući u obzir sva ograničenja [1].

Jedan od načina rješavanja optimizacijskog problema je evolucijski algoritam (EA) koji predstavlja stohastički proces traženja optimalnog rješenja za neki problem. EA se sastoji od nekoliko ključnih koraka: prikaz rješenja problema (obično vektori realnih brojeva), određivanje funkcije cilja, inicijalizacija populacije rješenja, primjena operatora reprodukcije te selekcije novih rješenja [2]. Diferencijalna evolucija (DE) je vrsta EA bazirana na populaciji rješenja, koju su razvili Storn i Price [1]. Glavno svojstvo DE, naspram drugih EA, je mutacija koja koristi razlike trenutnih rješenja kako bi stvorila deformacije pri generiranju novih rješenja. Mutacija, uz operator križanja, daje algoritmu DE jednostavan način pretrage i konvergencije rješenja prema optimalnom rješenju, odnosno rješenju funkcije cilja [3]. Diferencijalna evolucija je jednostavan, pouzdan i efikasan algoritam, pa se istaknula kao jedan od često korištenih alata za rješavanje problema optimizacije rješenja (vidi primjerice [4,5,6]).

U algoritmu DE, mutacija ima ključnu ulogu u pretraživanju prostora rješenja, ali je zato moguće i da generira rješenja koja izlaze van granica prostora pretrage. Budući da optimizacijski problemi često uključuju donje i gornje granice koje ograničavaju dopuštene vrijednosti rješenja, nužno je pravilno rukovati s takvim vrijednostima kako bi se osiguralo da rješenja ostanu unutar dopuštenih granica. U literaturi [7] se može pronaći mnoštvo različitih mehanizama za rukovanje ograničenjima prostora, pri čemu izbor konkretnog mehanizma može značajno utjecati na učinkovitost algoritma.

2.1. Struktura algoritma

Algoritam DE je paralelna metoda direktne pretrage koja koristi NP vektora parametara kao populaciju za svaku generaciju G [1]. Ključno je da vrijednost NP ostaje nepromijenjena tijekom izvođenja algoritma. Za svaki vektor iz populacije generira se novi kandidat vektor dodavanjem

razlike između dva člana populacije trećem članu. Ako generirani vektor daje nižu vrijednost funkcije cilja u odnosu na postojećeg vektora populacije, novogenerirani vektor zamijenit će postojeći vektor u sljedećoj generaciji populacije. Ovaj proces se ponavlja dok se ne postigne odgovarajući uvjet zaustavljanja algoritma.

Na početku algoritma potrebno je inicijalizirati populaciju rješenja nulte generacije. Populacija rješenja u algoritmu DE sastoji se od NP, D-dimenzionalnih vektora rješenja

$$P(x_i), \quad i = 0, 1, 2, \dots, NP - 1, \tag{2-1}$$

gdje svaki vektor predstavlja potencijalno rješenje problema. Početna populacija se inicijalizira nasumično, pri čemu se svaka komponenta vektora generira unutar granica pretrage, definiranih s donjom (L) i gornjom (U) granicom .

Nasumično generiranje početnih rješenja, kao i osiguranje da sva početna rješenja budu unutar definiranih granica pretrage garantira se formulom:

$$x_{j,i} = L + rand[0,1] * (U - L), \ \forall j \in \{0, 1, 2, \dots, D - 1\}. \tag{2-2}$$

Postoje i napredne metode za inicijalizaciju populacije koje mogu poboljšati performanse algoritma, no one nisu nužne za osnovno razumijevanje i primjenu DE-a [2].

Mutacija je ključni element diferencijalne evolucije i po njoj je algoritam dobio ime. Ona za svaki vektor iz populacije generira novi mutant vektor rješenja dodavanjem diferencijalnog vektora na ciljni vektor. Najčešći operator mutacije je rand/1 koji za svako rješenje iz trenutne populacije, odnosno ciljni vektor x_i, nasumično odabire bazni vektor x_{r1} i još dva različita vektora iz populacije x_{r2}, x_{r3} te stvara mutant vektor v_i prema formuli:

$$v_i = x_{r1} + F * (x_{r2} - x_{r3}), \tag{2-3}$$

gdje r1, r2, r3 predstavljaju indekse elemenata iz trenutne populacije. Oni su nasumično odabrani iz skupa {0, 1, …, NP – 1} te za njih mora vrijediti i ≠ r1 ≠ r2 ≠ r3. F predstavlja faktor skaliranja koji je obično realni broj između nula i jedan, koji kontrolira intenzitet promjene u mutant vektoru i utječe na raznolikost populacije.

Drugi često korišteni operatori mutacije uključuju best/1, gdje se za bazni vektor x_{r1} uzima najbolje rješenje iz trenutne populacije i rand/2, gdje se koriste četiri nasumično odabrana vektora za stvaranje mutant vektora. Operator best/1 obično vodi ka bržem konvergiranju, ali s rizikom od preuranjene konvergencije na neki od lokalnih minimuma funkcije cilja, dok rand/1 pruža bolje istraživanje prostora pretrage no može zahtijevati veći broj vrednovanja funkcije cilja [3].

Mutacija na ovaj način može generirati rješenja koja izlaze iz dozvoljenih granica prostora pretrage, osobito pri populacijama nižih generacija. Primjerice, ako su nasumično odabrana dva

vrlo različita vektora, njihovim oduzimanjem dobije se relativno velik vektor. S takvim rješenjima se mora rukovati kako bi ostala valjana, a to se obično postiže kroz primjenu nekog od mehanizama za rukovanje ograničenjima prostora.

Kako bi povećali raznolikost između rješenja u novim populacijama, odrađuje se korak križanja vektora. Križanje u DE-u kombinira mutant vektor v_i s vektorom x_i kako bi se stvorio novi probni vektor u_i [1]. Postoje dva najčešće korištena operatora križanja: binomno i eksponencijalno križanje. Kod binomnog križanja, koji je ujedno i najčešće korišten, svaka komponenta j vektora u_i se nasumično odabire iz mutant vektora v_i ili vektora x_i, pri čemu stopa križanja CR određuje vjerojatnost odabira komponente iz mutant vektora. Veća vrijednost CR znači da će veći broj komponenata biti odabrano iz mutant vektora u odnosu na ciljni vektor x_i. Ovakvo križanje može se prikazati formulom:

$$u_{j,i} = \begin{cases} v_{j,i} & ako\ je\ rand[0,1] < CR\ ili\ j = j_r, \quad j_r = rand[0, D-1] \\ x_{j,i} & inače \end{cases}, \tag{2-4}$$

kako bi se osiguralo da probni vektor u_i nikada nije identičan x_i, osim korištenja faktora križanja CR, uzima se barem jedna komponenta j_r koja će uvijek biti odabrana iz v_i [2].

Eksponencijalno križanje započinje s nasumično odabranim indeksom j_s koji označava početnu točku u vektoru. Za svaki indeks od početnog, komponente se redom prepisuju iz mutant vektora sve dok je nasumično generirani broj iz intervala [0, 1] manji od stope križanja CR. Prvi put kada broj bude veći od faktora križanja, sve daljnje komponente uzimaju se od ciljnog vektora x_i.

Kada su stvoreni kandidati rješenja nakon koraka mutacije i križanja, odrađuje se selekcija nove generacije populacije. Svaki vektor x_i iz trenutne generacije uspoređuje se s odgovarajućim probnim vektorom u_i prema vrednovanju funkcije cilja f(x) formulom:

$$y_i = \begin{cases} u_i & ako\ je\ f(u_i) < f(x_i) \\ x_i & inače \end{cases}, \tag{2-5}$$

gdje vektor y_i predstavlja vektor koji će biti izabran za iduću generaciju rješenja. U ovom slučaju gleda se koji vektor ima manju vrijednost funkcije cilja pošto se radi o problemu minimizacije, pa on zamjenjuje vektor x_i u novoj populaciji. Ova jednostavna selekcija osigurava da se populacija s vremenom poboljšava ili ostaje ista, ali nikad ne pogoršava.

Koraci mutacije, križanja i selekcije ponavljaju se za svaku generaciju populacije dok se ne ispuni zadani uvjet zaustavljanja algoritma. Uvjet zaustavljanja može biti unaprijed definirani broj generacija, postizanje željene vrijednosti funkcije cilja ili maksimalan broj vrednovanja funkcije

cilja. Slika 2.1 prikazuje pseudo-kod algoritma na visokoj razini u standardnoj rand/1/bin inačici kakav je korišten u nastavku rada.

```
postavi ulazne parametre algoritma NP, D, F, CR, maxEvals, L, U
generiraj početnu populaciju P tako da svaka varijabla u svakom rješenju bude nasumično
odabrana unutar granica [L, U] i svakom rješenju odredi vrijednost funkcije cilja
evals = NP
gen = 0
dok je evals < maxEvals činiti
  za i = 1 do NP činiti
    nasumično odaberi 3 različita rješenja x1,gen, x2,gen, x3,gen iz populacije, različita od xi,gen
    izračunaj probni vektor vi,gen = x1,gen + F(x2,gen - x3,gen)
    vi,gen = primjeniMehanizamZaRukovanjeOgraničenjimaProstora(vi,gen)
    za j = 1 do D
      neka je rand nasumično odabran broj u rasponu [0,1]
      ako je rand < CR onda
        ui,j,gen = vi,j,gen
      inače
        ui,j,gen = xi,j,gen
    ako je f(ui,gen) > f(xi,gen) onda
      xi,gen+1 = ui,gen
    inače
      xi,gen+1 = xi,gen
  gen = gen + 1
  evals = evals + NP
```

Slika 2.1. *Pseudo-kod algoritma diferencijalne evolucije*

2.2. Parametri algoritma

Pri definiranju algoritma DE potrebno je odabrati njegove komponente, kao što su operatori varijacije (mutacija i križanje) koji odgovaraju odabranoj reprezentaciji, mehanizmi selekcije, te početna populacija. Svaka od ovih komponenata može imati određene parametre. Vrijednosti tih parametara uvelike utječu na sposobnost algoritma da pronađe rješenje blizu optimuma, kao i na učinkovitost kojom će doći do takvog rješenja [9].

Standardni algoritam DE ovisi o tri ključna parametara koji mogu značajno utjecati na ponašanje i performanse algoritma: veličina populacije (NP), faktor skaliranja (F) i stopa križanja (CR). Kako ne postoji univerzalno dobra konfiguracija ovih parametara, parametre je potrebno prilagoditi svakom optimizacijskom problemu kako bi algoritam dao najbolje rezultate [8]. Veličina populacije predstavlja broj kandidata rješenja u svakoj generaciji, a pravilno podešavanje ovog parametra je ključno za balansiranje između istraživanja prostora i iskorištavanja dobrih rješenja.

Pri odabiru veličine populacije obično nema gornje granice, ali donja granica je često ograničena ovisno o odabranom operatoru mutacije. Prevelika populacija može usporiti konvergenciju, dok premala može dovesti do preranog zaustavljanja u lokalnim minimumima.

Faktor skaliranja je ključan za očuvanje raznolikosti populacije i time za izbjegavanje preranog zaustavljanja algoritma u lokalnim minimumima. Iako F može poprimiti bilo koju realnu vrijednost, postavljanje F na nulu nema smisla jer bi tada mutant bio jednak baznom vektoru, a prema [8] negativne vrijednosti nisu potrebne jer se razlike između vektora, u pravilu, dobivaju nasumičnim odabirom članova populacije. Nadalje, vrijednosti veće od jedan gotovo nikada nisu potrebne. To implicira da je učinkoviti raspon vrijednosti F unutar intervala ⟨0, 1]. Međutim, vrijednost faktora skaliranja blizu jedan mogu povećati vjerojatnost generiranja rješenja izvan granica pretraživačkog prostora, što zahtjeva primjenu mehanizama za rukovanje ograničenjima.

Stopa križanja (CR) određuje koliko će komponenti iz mutantnog vektora biti zadržano u novom rješenju. Visoke vrijednosti CR-a povećavaju raznolikost unutar populacije, dok niske vrijednosti mogu osigurati stabilniju, ali potencijalno sporiju konvergenciju. Pravilno podešavanje CR-a ključno je za održavanje balansa između istraživanja novih područja i finog ugađanja rješenja unutar trenutno istraženih regija. Kako stopa križanja predstavlja vjerojatnost odabira komponenti iz mutantnog vektora, efektivna vrijednost stope križanja je unutar intervala [0, 1]. Prema [9], trenutno je općenito prihvaćeno da stopa križanja ne bi trebala biti preniska, a vrijednosti ispod 0.6 koriste se vrlo rijetko.

Općenito, različiti optimizacijski problemi zahtijevaju različite postavke ovih parametara. U literaturi se često ističe potreba za prilagodljivim mehanizmima koji automatski podešavaju ove parametre tijekom izvođenja algoritma, ovisno o fazi pretrage i prirodi problema. Na primjer, faktor skaliranja može imati značajan utjecaj na to koliko je vjerojatno da se stvore rješenja izvan prostora pretrage, što je posebno važno za probleme sa strogo definiranim ograničenjima. Stoga, pravilna postavka i prilagodba ovih parametara tijekom izvođenja algoritma može značajno unaprijediti učinkovitost i pouzdanost diferencijalne evolucije.

2.3. Rukovanje ograničenjima prostora

Rukovanje ograničenjima prostora pretrage ključno je za uspješnu primjenu DE u optimizacijskim problemima. Kada se promatraju optimizacijski problemi s ograničenjima, najčešće donje i gornje granice, zahtijevaju se primjene mehanizama za upravljanje situacijama kada generirana rješenja izlaze izvan dozvoljenog prostora pretrage. Postoji nekoliko standardnih mehanizama za

rješavanje ovih problema, a svaki od njih ima svoje prednosti i nedostatke te različit utjecaj na ponašanje algoritma.

2.3.1. Projekcija

Kada vrijednost izađe iz dopuštenih granica, postavlja se na vrijednost najbliže granice prema formuli

$$w_i = \begin{cases} v_i & l_i \leq v_i \leq u_i \\ u_i & v_i > u_i \\ l_i & v_i < l_i \end{cases}, \tag{2-6}$$

gdje v_i predstavlja vrijednost komponente vektora, a l_i i u_i donju i gornju granicu prostora [10]. Prema [7], koristi se još i projekcija prema središnjoj točki i projekcija prema baznom vektoru. Projekcija prema središnjoj točki rezultira se postavljanjem nevaljanih vrijednosti prema središnjoj točki valjanog područja formulom

$$w_i = (1 - \alpha) * 0.5(l_i + u_i) + \alpha * v_i, \tag{2-7}$$

gdje je $\alpha \in [0, 1]$ najveća vrijednost za koju vrijedi da za svaki i = 1, … n

$$l_i \leq w_i \leq u_i. \tag{2-8}$$

Slično kao prethodna projekcija, projekcija prema baznom vektoru razlikuje se samo u smjeru prema kojem se projektira vrijednost izvan granica:

$$w_i = (1 - \alpha) * x_{r1} + \alpha * v_i, \tag{2-9}$$

gdje također vrijedi uvjet pod formulom (2-8).

Primjena mehanizma projekcije rezultira značajnim brojem rješenja smještenih na granicama dopuštenog područja što može uvelike smanjiti raznolikost populacije, no svejedno je projekcija jedan od češće korištenih mehanizama zbog svoje jednostavne implementacije [7]. Upravo zbog brojnih rješenja na granicama, projekcija kao mehanizam radi vrlo dobro kada je globalni minimum funkcije cilja blizu granica pretrage.

2.3.2. Ponovna inicijalizacija

Primjena mehanizma ponovne inicijalizacije podrazumijeva nasumično generiranje nove vrijednosti unutar granica prostora, kojom će vrijednost koja je izašla iz dopuštenih granica biti zamijenjena prema formuli

$$w_i = \begin{cases} v_i & l_i \leq v_i \leq u_i \\ \xi & v_i < l_i \; ili \; v_i > u_i \end{cases}. \tag{2-10}$$

ξ, u formuli (2-10) predstavlja nasumično generiranu vrijednost iz intervala $[l_i, u_i]$. Prema [7] ovaj mehanizam je jedan od onih koji pridonose najvećoj raznolikosti populacije. Upravo na taj način se izbjegava preuranjena konvergencija prema lokalnim minimumima i daje mogućnost algoritmu pretraživanje šireg prostora, ali često zahtijeva veći broj generacija populacije kako bi pronašao bolja rješenja.

2.3.3. Refleksija

Refleksija zrcali rješenja koje izlaze izvan granica natrag unutar prostora, na način da se reflektira od granice pretraživačkog prostora formulom

$$w_i = \begin{cases} v_i & l_i \leq v_i \leq u_i \\ 2u_i - v_i & v_i > u_i \\ 2l_i - v_i & v_i < l_i \end{cases}. \tag{2-11}$$

Mehanizam refleksije u nekim slučajevima može generirati vrijednost koja i dalje nije u dopuštenim granicama, pa se u tom slučaju ponavlja formula (2-11) dok se ne postigne valjana vrijednost. Prema [10], uz mehanizam projekcije, i refleksija daje bolje rezultate optimizacije kod funkcija cilja s globalnim minimumom blizu granica pretrage. Ovaj mehanizam značajno ovisi o udaljenosti vrijednosti koje su izašle izvan granica prostora. Ako je ta udaljenost mala, reflektirana vrijednost će se nalaziti blizu same granice pretraživačkog prostora. S druge strane, ako je udaljenost veća, refleksija će proizvesti raznolikija rješenja, što može pridonijeti većoj raznolikosti populacije. Ovaj mehanizam tako omogućuje dinamičnu prilagodbu, ovisno o stupnju odstupanja, što može povećati istraživački potencijal algoritma u širem prostoru rješenja.

2.3.4. Konzervativna metoda

Ukoliko je bilo koja vrijednost generiranog mutant vektora izašla izvan dopuštenih granica, vektor se odbacuje i za novu populaciju se uzima ciljni vektor x_i. Drugim riječima, algoritam neće pokušati ispraviti vrijednosti izvan granica (2-12), nego ih odbacuje [10].

$$w = \begin{cases} v & ako\ je\ l_i \leq v_i \leq u_i\ za\ svaki\ i = 0, \dots, n \\ x & inače \end{cases}. \tag{2-12}$$

Ovaj mehanizam osigurava stabilnost algoritma odbacivanjem neodgovarajućih rješenja, ali uvelike umanjuje raznolikost populacije i mogućnost pretraživanja prostora, te često dovodi do preranog zaustavljana algoritma ili konvergiranja u lokalne minimume.

2.4. Zastupljenost mehanizma rukovanja ograničenjima prostora u literaturi

Tablica 2.1. daje pregled zastupljenosti različitih mehanizama za rukovanje ograničenjima prostora pretrage u literaturi, pri čemu su naznačeni mehanizmi razmatrani u radu. Prikazano je devet mehanizama, pri čemu su neki mehanizmi zastupljeni u većem broju radova, dok drugi imaju znatno manju prisutnost. Najčešće spominjani mehanizmi uključuju projekciju, ponovnu inicijalizaciju, refleksiju i ponovno uzorkovanje, koji su prisutni u gotovo svim pregledanim radovima. Ovi mehanizmi pokazali su se vrlo kao vrlo robusni i generalno učinkoviti za širok spektar optimizacijskih problema. S druge strane, mehanizmi poput konzervatizma, projekcije prema baznom ili ciljnom vektoru, te omatanje također su zabilježeni, ali u manjem broju radova. Razlog tome je što su često učinkovitiji kod specifičnih problema optimizacije. U literaturi [1], Storn i Price već spominju kaznenu funkciju kao mehanizam za rukovanje ograničenjima, ali se ona s vremenom, zbog veće složenosti podešavanja, koristila sve rjeđe.

Također, značajan broj radova ([17, 18, 19]) spominje ograničenja prostora i potrebu za rukovanje ograničenjima u algoritmu DE, ali ne navodi korištene mehanizme. S druge strane, neki radovi vezani uz algoritam DE uopće ne spominju bilo kakvo rukovanje ograničenjima prostora, primjerice [20, 21]. Nedostatak informacija o korištenim mehanizmima rukovanja ograničenjima može otežati razumijevanje o tome kako je algoritam postigao određene rezultate, što posebno dolazi do izražaja kod problema s kompleksnim ili strogo definiranim ograničenjima prostora.

Tablica 2.1. *Zastupljenost mehanizama za rukovanje ograničenjima prostora u literaturi*

Mehanizam	Literatura u kojoj je spomenut
projekcija	[7], [10], [11], [12], [13], [14], [15]
ponovna inicijalizacija	[2], [7], [10], [11], [12], [13], [14], [16]
refleksija	[7], [10], [11], [12], [13], [14]
konzervatizam	[7], [10], [11], [13]
projekcija prema baznom vektoru	[7], [11], [13]
projekcija prema ciljnom vektoru	[7], [11], [13], [14]
omatanje	[7], [10], [11]
ponovno uzorkovanje	[2], [7], [10], [11], [13], [14]
kaznene funkcije	[1], [2], [7], [11], [15], [16]

3. OSTVARENO PROGRAMSKO RJEŠENJE

Radi točnije i jednostavnije provedbe eksperimentalne analize rukovanja ograničenjima prostora u algoritmu DE, izrađena je konzola aplikacija u programskom jeziku C#. Programsko rješenje omogućuje korisniku postavljanje svih parametara algoritma DE te uvjeta završetka algoritma koji je ovom slučaju postavljen na broj vrednovanja funkcije cilja. Nakon što se izvrše sva ponavljanja algoritma s postavljenim parametrima, rezultati izvođenja, kao i svih postavljeni parametri, zapisuju se u datoteku na računalu.

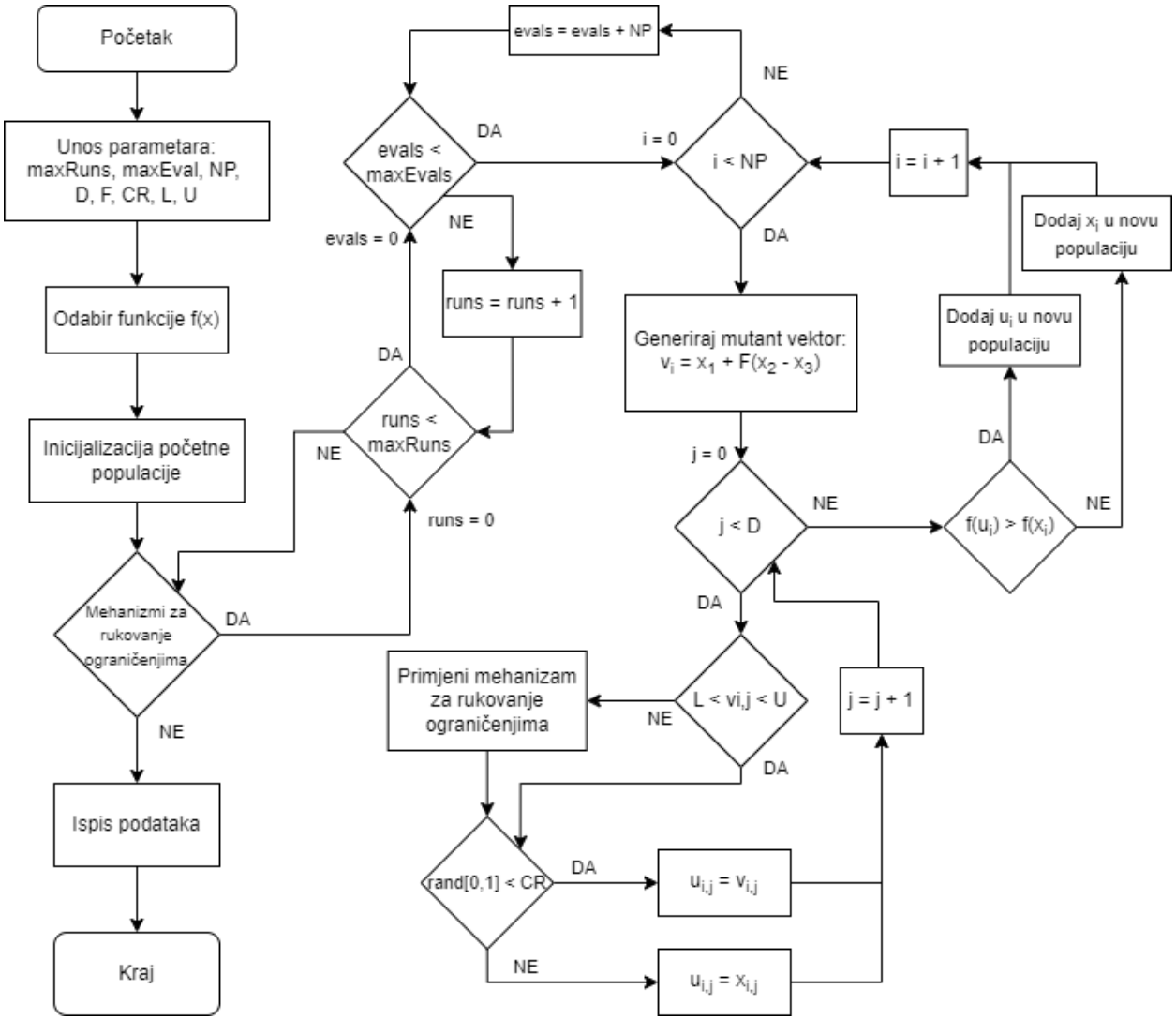
3.1. Način rada programskog rješenja

Pri pokretanja konzolne aplikacije, od korisnika se prvo traži unos sljedećih podataka:

- broj izvršavanja algoritma (maxRuns)
- maksimalan broj vrednovanja funkcije cilja (maxEvals)
- veličina populacije (NP)
- broj varijabli, odnosno dimenzija rješenja (D)
- faktor skaliranja (F)
- faktor križanja (CR)
- donja (L) i gornja (U) granica prostora

Nakon postavljanja svih parametara algoritma, korisnik izabire jednu od deset ponuđenih funkcija cilja. Pošto je nužno osigurati iste parametre i okoline izvršavanja algoritma za različite mehanizme za rukovanje ograničenjima prostora, korisnik ne odabire mehanizam za pojedino izvršavanje algoritma, već se algoritam, s jednom postavljenim parametrima, izvršava za sva četiri prijašnje navedena mehanizma. Algoritam se izvršava više puta (maxRuns) za svaki mehanizam, te se na kraju uzimaju srednje vrijednosti rezultata kako bi izbjegli nasumična odstupanja u rezultatima.

Kada algoritam završi s izvršavanjem instance za pojedini mehanizam, kreira se nova tekstualna datoteka s nazivom korištenog mehanizma u koju se spremaju korišteni parametri za izvršavanje algoritma te sva praćena statistika za svaku generaciju populacije. Takvih izlaznih datoteka biti će onoliko koliko je korišteno mehanizama za rukovanje ograničenjima – u ovom slučaju četiri.

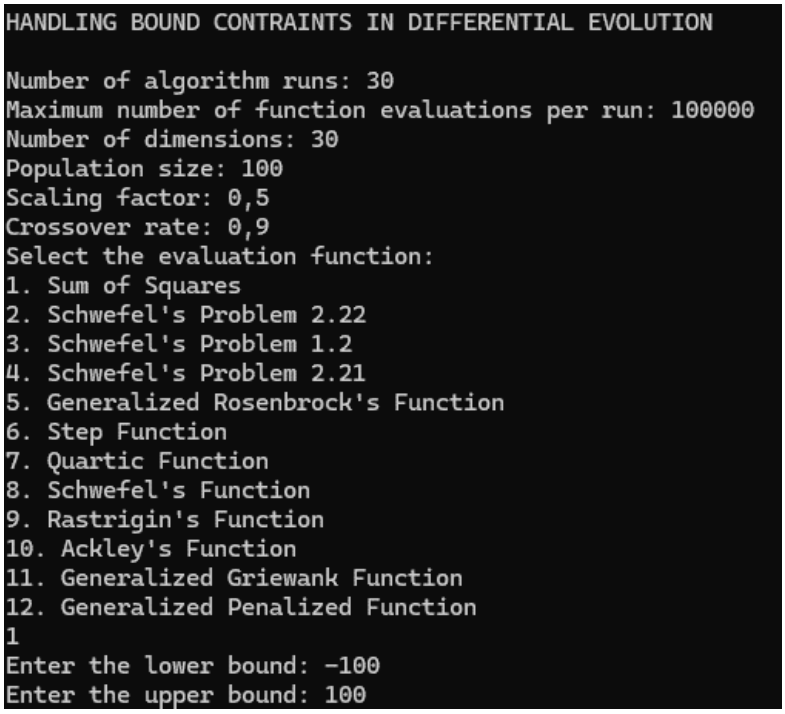


Slika 3.1. *Dijagram toka programskog rješenja*

Nakon postavljanja svih potrebnih parametara i odabira funkcije, kreira se inicijalna populacija slučajnih vektora u granicama [L, U] formulom (2-2). Ova inicijalna populacija biti će korištena za svako pokretanje algoritma te za svaki mehanizam. Na svaku varijablu mutant vektora primjenjuje se mehanizam za rukovanje ograničenjima, koji osigurava da su sve varijable unutar dopuštenih granica pretrage. Nakon toga mutant vektor se križa s ciljnim vektorom, ovisno o stopi križanja (CR), korištenjem formule (2-4). U koraku selekcije gleda se je li probni vektor bolji od ciljnog prema funkciji cilja, ako je dodaje se u novu populaciju, inače ciljni vektor x_i prelazi u novu populaciju. Proces se ponavlja sve dok algoritam ne ispuni uvjete zaustavljanja, nakon čega se rezultati pohranjuju i ispisuju. Uvjet zaustavljanja u ovom programskom rješenju je broj vrednovanja funkcije cilja, a ne broj generacija. Razlog tome je što želimo dobiti pravedniju

usporedbu različitih algoritama, a na ovaj način izbjegavamo ovisnost o generaciji te dajemo svim algoritmima jednak broj vrednovanja, neovisno o veličini populacije.

3.2. Prikaz i način uporabe programskog rješenja



Slika 3.2. *Primjer unošenja ulaznih parametara u programskom rješenju*

Ostvareno programsko rješenje pruža fleksibilan alat za optimizaciju funkcija unutar zadanih granica pretrage koristeći diferencijalnu evoluciju (DE). Korisnik prvo mora postaviti sve tražene parametre algoritma prema slici 3.2. Ukoliko je unesena vrijednost van dozvoljenih granica, od korisnika se zatraži novi unos. Kako bi se mogla napraviti analiza učinkovitosti pojedinog mehanizma, algoritam tijekom izvođenja prati ključne statističke podatke, uključujući najbolju vrijednost funkcije cilja po generaciji, broj primjena mehanizma po generaciji, broj odabranih rješenja nakon primjene mehanizma nad njima, kao i prosječnu udaljenost do granica pretrage. Na kraju svakog izvođenja, rezultati se automatski pohranjuju u datoteku, čime se omogućuje detaljna analiza dobivenih rezultata. U datoteci se, uz parametre izvođenja, bilježe svi parametri algoritma DE, kao i odabrana funkcija cilja i korišteni mehanizam za to izvođenje.

Rješenje je pogodno za istraživanje različitih problema optimizacije, usporedbu učinkovitosti različitih mehanizama za rukovanje ograničenjima, te analizu utjecaja parametara DE algoritma na rezultate optimizacije. Ova fleksibilnost i mogućnost prilagodbe čini programsko rješenje

korisnim alatom za znanstveno istraživanje u području optimizacije, kao i za rješavanje konkretnih inženjerskih problema.

```
ALGORITHM EXECUTION PARAMETERS:
Evaluation function: 1
Constraint handling mechanism: Conservatism
Number of algorithm runs: 30
Maximum function evaluations per run: 100000
Number of dimensions: 30
Population size: 100
Scaling factor: 0,5
Crossover rate: 0,9
Lower bound: -100
Upper bound: 100

GENERATION 0
    Best evaluation: 62990,0975936927
GENERATION 1
    Best evaluation: 62990,09759369266
    Worst evaluation: 137991,8309680591
    Average evaluation: 96760,77204566714
    Standard deviation of evaluations: 15095,945795292891
    Mechanism applications: 99
    Selected vectors after constraint handling: 0
    Average distance out of bounds: 24,96118532696101
```

Slika 3.3. *Primjer ispisa algoritma nakon prve generacije*

Ispis na slici 3.3 prikazuje rezultate izvođenja algoritma za zadanu funkciju cilja, pri čemu su parametri izvođenja detaljno navedeni na početku. Kao prvi dio ispisa, prikazani su parametri algoritma, uključujući odabranu funkciju cilja, korišteni mehanizam za rukovanje ograničenjima, broj ponavljanja algoritma i drugi.

Nakon toga slijedi ispis rezultata po generacijama, pri čemu su ovih sedam parametara po generaciji (najbolja, najlošija i srednja vrijednost, standardna devijacija, broj primjena mehanizma za rukovanje ograničenjima, broj odabranih vektora nakon primjene mehanizma, te prosječna udaljenost izvan granica) prikazane kao srednje vrijednosti (u ovom primjeru 30) pokretanja algoritma. To znači da su svi podaci agregirani i prosječni rezultati su prikazani za svaku generaciju, čime se daje bolji uvid u ponašanje algoritma tijekom optimizacije. Ovaj format ispisa omogućuje detaljno praćenje performansi algoritma tijekom optimizacije i daje korisne informacije o učinku odabranih parametara i mehanizama.

4. EKSPERIMENTALNA ANALIZA

Cilj eksperimentalne analize bio je ispitati ponašanje i učinkovitost algoritma DE u kontekstu različitih mehanizama za rukovanje ograničenjima prostora pretrage. Glavni zadatak bio je evaluirati kako različiti pristupi upravljanja ograničenjima utječu na konačne rezultate algoritma. Ova analiza trebala je pokazati u kojoj mjeri određeni mehanizmi pomažu u održavanju rješenja unutar zadanih granica, koliko učinkovito algoritam pronalazi optimalno rješenje i koliko je stabilan pri različitim parametrima pretrage. Inačica algoritma DE korištena za analizu sadržava rand/1 operator mutacije i binomno križanje.

U tablici 4.1 navedeno je dvanaest funkcija cilja, često korištenih za eksperimentalnu analizu optimizacijskih algoritama [22]. Prvih šest funkcija su unimodalne, što znači da imaju samo jedan globalni minimum (ili maksimum) i ne postoje drugi lokalni minimumi, dok su drugih šest funkcija višemodalne što znači da imaju više lokalnih minimuma (ili maksimuma), te jedan globalni minimum. f_{min} predstavlja vrijednost funkcije cilja u globalnom minimumu, a prikazane su vrijednosti za dvije dimenzionalnosti D (30 i 50). Višemodalne funkcije mogu znatno otežati optimizacijski proces upravo zbog mogućnosti konvergiranja u neki od lokalnih ekstrema. Većina funkcija imaju globalni minimum blizu nuli pa se on nalazi u centru prostora pretrage. Kako bi bolje prikazali učinkovitost različitih mehanizama za rukovanje ograničenjima potrebno je pomicati granice prostora pretrage, najčešće tako da ili gornja ili donja granica budu postavljenje na vrijednost blizu globalnog optimuma.

Tablica 4.1. *Odabrane funkcije cilja i njihove karakteristike*

Funkcija cilja	Granice prostora	Globalni minimum	f_{min}	
			D=30	D=50
$f_1 = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$\{0\}^D$	0	0
$f_2 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	$\{0\}^D$	0	0
$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	$\{0\}^D$	0	0
$f_4 = max_i\{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100, 100]^D$	$\{0\}^D$	0	0
$f_5 = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^D$	$\{1\}^D$	0	0
$f_6 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	$[-0.5, 0.5)^D$	0	0
$f_7 = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]^D$	$\{0\}^D$	0	0
$f_8 = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^D$	$\{420.9687\}^D$	-12569.5	-21048.4
$f_9 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	$\{0\}^D$	0	0
$f_{10} = -20exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}\right) - \exp\left(\frac{\sum_{i=1}^{D} cos2\pi x_i}{D}\right) + 20 + e$	$[-32, 32]^D$	$\{0\}^D$	0	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	$\{0\}^D$	0	0
$f_{12} = \frac{\pi}{n}\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < a \end{cases}$	$[-50, 50]^D$	$\{0\}^D$	0	0

4.1. Postavke eksperimenta

Zbog stohastičke prirode algoritma DE, za kvalitetno izvršavanje eksperimentalne analize, potrebno je odraditi više ponavljanja izvođenja algoritma s istim vrijednostima parametara da dobijemo stvaran uvid o učinkovitosti algoritma. Zbog toga je broj izvođenja algoritma (maxRuns) postavljen na 30 za sva pokretanja algoritma. Uvjet završetka pojedinog izvođenja algoritma je maksimalan broj vrednovanja funkcije cilja (maxEvals). Uvjet završetka mora biti jednak pri svim razmatranim postavkama kako bi usporedbe bile poštene, pa je maksimalan broj vrednovanja funkcije cilja uvijek postavljen na 100000.

Literatura [1] preporuča postavljanje veličine populacije (NP) na vrijednost unutar intervala [5D, 10D] gdje je D dimenzionalnost rješenja. To naglašava da bi NP trebao ovisiti o broju dimenzija, ali se NP često postavlja na vrijednost 100 ili 50, neovisno o broju dimenzija [8]. Odabranim funkcijama moguće je mijenjati dimenzionalnost, pa su napravljene analize za vrijednosti dimenzija 30 i 50. Kako faktor skaliranja (F) može znatno utjecati na ponašanje algoritma i stvaranja rješenja izvan prostora pretrage, tako može utjecati i na učinkovitost različitih mehanizama za rukovanje ograničenjima. Iz tog razloga, odabrano je tri vrijednosti za F i napravljena je usporedba kako utječe promjena F-a pri jednakoj konfiguraciji ostalih parametara. Prema [8] najčešće korištena vrijednost F-a je 0.5, a uz to još su odabrane vrijednosti 0.3 i 0.7. Stopa križanja (CR), za sva pokretanja algoritma, postavljena je na 0.9. Na taj način osigurava se veća raznolikost populacije i pretraživanje šireg prostora. Detaljan prikaz svih korištenih parametara prikazan je u tablici 4.2.

Tablica 4.2. *Korišteni parametri za eksperimentalnu analizu*

Parametar	Vrijednost parametra
maxRuns	30
maxEvals	100000
D	30, 50
NP	100
F	0.5, 0.3, 0.7
CR	0.9

Tijekom izvođenja algoritma, prikupljani su različiti podaci za svaku generaciju i svako izvođenje. Konkretno, za svaku generaciju prikupljeni su: najbolja, najlošija te srednja vrijednost vrijednosti funkcije cilja unutar populacije te standardna devijacija, koja prikazuje raspršenost rješenja u odnosu na prosjek. Uz to, pratila se i frekvencija primjene mehanizama za rukovanje

ograničenjima, broj vektora koji su odabrani nakon primjene mehanizama te prosječna udaljenost rješenja izvan granica dozvoljenog prostora. Svi prikupljeni podaci agregirani su tako da su, nakon 30 ponovljenih izvođenja algoritma, izračunate prosječne vrijednosti za svaku mjerenu kategoriju. Time je, bez obzira na stohastičku prirodu algoritma, omogućeno uspoređivanje različitih mehanizama za rukovanje ograničenjima i njihova analiza kroz više generacija i evaluacija.

4.2. Rezultati

Eksperimentalna analiza podijeljena je na tri dijela. U prvom poglavlju korištene su standardne vrijednosti parametara iz tablice 4.2 i promatrano je kako pojedini mehanizam utječe na brzinu konvergencije za jednaku inicijalnu populaciju. Drugo poglavlje opisuje kako promjena faktora skaliranja može utjecati na učinkovitost, odnosno točnost algoritma pri različitim mehanizmima. Na kraju, u trećem poglavlju korištene su iste vrijednosti parametara kao u prvom poglavlju, uz nekoliko različitih vrijednosti gornjih i donjih granica prostora pretrage kako bi se prikazalo ponašanje algoritma ako je optimum blizu granice prostora. Sva eksperimentalna mjerenja odrađena su za dvije dimenzionalnosti (30 i 50).

4.2.1. Utjecaj odabira mehanizma za rukovanje ograničenjima prostora

Cilj prvog izvođenja algoritma je prikazati kako koji mehanizam utječe na pronalazak globalnog minimuma pojedine funkcije cilja. Za sva izvođenja algoritma korišteni su isti parametri prema tablici 4.2 ($F = 0.5$), te je jednom generirana početna populacija korištena za sve mehanizme. Ostvareni rezultati izvršavanja algoritma za obje vrijednosti dimenzionalnosti prikazani su u tablici 4.3 i 4.4.

Tablica 4.3. *Prikaz rezultata izvršavanja algoritma za D = 30*

Funkcija cilja	Mehanizam	x_{avg}	x_{best}	x_{worst}	σ
f_1	projekcija	0.784 E-7	**1.120 E-8**	1.983 E-7	3.795 E-8
	reinicijalizacija	**0.362 E-7**	1.136 E-8	**1.003 E-7**	**2.455 E-8**
	refleksija	0.534 E-7	1.755 E-8	1.684 E-7	3.896 E-8
	konzervatizam	4.645 E-7	3.018 E-7	2.142 E-6	4.552 E-7
f_2	projekcija	6.358 E-4	2.108 E-4	1.310 E-3	3.599 E-4
	reinicijalizacija	**2.571 E-4**	**1.492 E-4**	**4.647 E-4**	**9.389 E-5**
	refleksija	3.039 E-4	1.733 E-4	8.164 E-4	1.504 E-4
	konzervatizam	8.600 E-4	2.608 E-4	1.488 E-3	2.852 E-4
f_3	projekcija	3.741 E+1	1.560 E+1	6.894 E+1	1.515 E+1
	reinicijalizacija	2.500 E+1	1.298 E+1	**6.437 E+1**	**1.386 E+1**
	refleksija	**2.487 E+1**	**1.196 E+1**	7.990 E+1	1.773 E+1
	konzervatizam	6.149 E+1	1.310 E+1	1.159 E+2	2.638 E+1
f_4	projekcija	4.301 E-1	7.878 E-2	3.703 E+0	9.123 E-1
	reinicijalizacija	**0.414 E-1**	**3.634 E-2**	**1.899 E+0**	**3.673 E-1**
	refleksija	5.754 E-1	6.453 E-2	2.968 E+0	6.205 E-1
	konzervatizam	5.021 E-1	1.012 E-1	2.155 E+0	4.578 E-1
f_5	projekcija	2.259 E+1	2.097 E+1	2.463 E+1	8.475 E-1
	reinicijalizacija	**2.214 E+1**	**2.123 E+1**	2.403 E+1	7.347 E-1
	refleksija	2.267 E+1	2.129 E+1	**2.354 E+1**	**5.623 E-1**
	konzervatizam	2.333 E+1	2.110 E+1	2.536 E+1	8.273 E-1
f_6	projekcija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	reinicijalizacija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	refleksija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	konzervatizam	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
f_7	projekcija	0.706 E-18	1.888 E-20	1.538 E-18	4.830 E-19
	reinicijalizacija	**0.276 E-18**	**1.198 E-20**	**2.742 E-18**	**5.536 E-19**
	refleksija	0.359 E-18	2.666 E-20	2.843 E-18	6.419 E-19
	konzervatizam	2.021 E-16	1.184 E-18	3.914 E-16	7.691 E-17
f_8	projekcija	-5.746 E+3	-7.026 E+3	-5.187 E+3	4.941 E+2
	reinicijalizacija	-5.173 E+3	-5.625 E+3	-4.519 E+3	2.641 E+2
	refleksija	**-8.945 E+3**	**-1.257 E+4**	**-6.351 E+3**	**2.003 E+3**
	konzervatizam	-3.851 E+3	-4.604 E+3	-3.254 E+3	3.370 E+2
f_9	projekcija	1.860 E+2	**1.460 E+2**	**2.003 E+2**	1.282 E+1
	reinicijalizacija	1.863 E+2	1.570 E+2	2.021 E+2	**1.025 E+1**
	refleksija	1.871 E+2	1.614 E+2	2.079 E+2	1.132 E+1
	konzervatizam	**1.859 E+2**	1.620 E+2	2.047 E+2	1.189 E+1
f_{10}	projekcija	9.065 E-5	3.628 E-5	1.406 E-4	2.467 E-5
	reinicijalizacija	**5.882 E-5**	3.227 E-5	**8.828 E-5**	**1.290 E-5**
	refleksija	6.592 E-5	**2.793 E-5**	1.194 E-4	1.881 E-5
	konzervatizam	2.705 E-4	1.062 E-4	5.038 E-4	8.363 E-5
f_{11}	projekcija	2.477 E-4	5.438 E-8	1.023 E-3	1.897 E-5
	reinicijalizacija	**0.138 E-6**	**2.659 E-8**	**4.632 E-5**	**8.981 E-6**
	refleksija	0.390 E-6	4.303 E-8	9.858 E-3	1.769 E-3
	konzervatizam	6.167 E-4	1.954 E-7	7.397 E-3	1.327 E-3
f_{12}	projekcija	2.181 E-4	7.786 E-9	6.541 E-3	1.174 E-3
	reinicijalizacija	0.777 E-7	**3.498 E-9**	2.459 E-7	5.674 E-8
	refleksija	**0.404 E-7**	7.643 E-9	**1.444 E-7**	**3.390 E-8**
	konzervatizam	5.050 E-7	4.557 E-8	1.421 E-6	4.002 E-7

Tablica 4.4. *Prikaz rezultata izvršavanja algoritma za D = 50*

Funkcija cilja	Mehanizam	x_{avg}	x_{best}	x_{worst}	σ
f_1	projekcija	1.240 E-3	**2.938 E-4**	3.488 E-3	5.681 E-4
	reinicijalizacija	**0.943 E-3**	3.019 E-4	**2.596 E-3**	**5.454 E-4**
	refleksija	1.187 E-3	2.995 E-4	2.625 E-3	6.370 E-4
	konzervatizam	2.069 E+4	7.188 E-2	1.182 E+5	2.367 E+4
f_2	projekcija	1.354 E-1	4.223 E-2	1.075 E+0	2.061 E-1
	reinicijalizacija	**3.641 E-2**	**1.965 E-2**	**6.542 E-2**	**9.574 E-3**
	refleksija	3.770 E-2	2.011 E-2	7.239 E-2	1.106 E-2
	konzervatizam	9.405 E+15	2.760 E+15	6.888 E+16	1.499 E+14
f_3	projekcija	6.905 E+3	4.530 E+3	1.620 E+4	2.317 E+3
	reinicijalizacija	**6.039 E+3**	**3.458 E+3**	1.122 E+4	1.716 E+3
	refleksija	6.129 E+3	4.198 E+3	**1.076 E+4**	**1.569 E+3**
	konzervatizam	3.844 E+4	1.261 E+4	2.616 E+5	1.083 E+5
f_4	projekcija	1.398 E+1	8.439 E+0	4.338 E+1	7.781 E+0
	reinicijalizacija	**5.974 E+0**	3.022 E+0	**1.344 E+1**	**2.304 E+0**
	refleksija	9.209 E+0	**2.177 E+0**	1.501 E+1	3.192 E+0
	konzervatizam	2.745 E+1	9.470 E+0	9.157 E+1	2.575 E+1
f_5	projekcija	5.201 E+1	4.115 E+1	1.171 E+2	2.484 E+1
	reinicijalizacija	5.364 E+1	4.578 E+1	**1.118 E+2**	1.828 E+1
	refleksija	**4.939 E+1**	**4.417 E+1**	1.159 E+2	**1.602 E+1**
	konzervatizam	2.236 E+7	6.489 E+2	1.207 E+8	2.632 E+3
f_6	projekcija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	reinicijalizacija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	refleksija	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
	konzervatizam	0.000 E+0	0.000 E+0	0.000 E+0	0.000 E+0
f_7	projekcija	7.888 E-11	1.036 E-11	3.135 E-10	7.792 E-11
	reinicijalizacija	**3.032 E-11**	2.650 E-12	**2.585 E-10**	**5.071 E-11**
	refleksija	6.332 E-11	**1.789 E-12**	3.024 E-10	7.650 E-11
	konzervatizam	2.711 E+1	9.657 E-7	2.404 E+2	5.979 E+1
f_8	projekcija	-7.661 E+3	-9.074 E+3	-6.585 E+3	**5.898 E+2**
	reinicijalizacija	-6.434 E+3	-7.055 E+3	-5.928 E+3	3.079 E+2
	refleksija	**-9.679 E+3**	**-1.536 E+4**	**-7.608 E+3**	1.539 E+3
	konzervatizam	-3.567 E+3	-4.708 E+3	-3.244 E+3	4.416 E+2
f_9	projekcija	3.789 E+2	3.472 E+2	**3.970 E+2**	**1.302 E+1**
	reinicijalizacija	**3.707 E+2**	**3.374 E+2**	4.047 E+2	1.596 E+1
	refleksija	3.755 E+2	3.425 E+2	4.059 E+2	1.392 E+1
	konzervatizam	4.532 E+2	3.469 E+2	7.196 E+2	8.696 E+1
f_{10}	projekcija	9.548 E-3	5.233 E-3	1.673 E-2	2.637 E-3
	reinicijalizacija	**6.735 E-3**	4.787 E-3	1.679 E-2	2.279 E-3
	refleksija	7.812 E-3	**4.330 E-3**	**1.138 E-2**	**1.789 E-3**
	konzervatizam	6.167 E+0	2.025 E-1	2.078 E+1	5.370 E+0
f_{11}	projekcija	1.724 E-3	6.167 E-4	3.613 E-3	**7.186 E-4**
	reinicijalizacija	**1.004 E-3**	**3.665 E-4**	**3.448 E-3**	7.915 E-4
	refleksija	1.401 E-3	4.083 E-4	3.577 E-3	7.356 E-4
	konzervatizam	9.017 E+0	2.479 E-1	2.287 E+2	5.329 E+1
f_{12}	projekcija	2.905 E-3	7.822 E-5	6.468 E-2	1.148 E-2
	reinicijalizacija	**3.652 E-4**	6.231 E-5	**4.538 E-3**	**1.375 E-3**
	refleksija	8.912 E-4	**5.236 E-5**	1.155 E-2	2.087 E-3
	konzervatizam	6.387 E+6	1.620 E-1	6.232 E+7	1.124 E+7

Tablice 4.3 i 4.4 prikazuju rezultate različitih mehanizama za rukovanje ograničenjima nakon završetka algoritma diferencijalne evolucije na dvanaest funkcija cilja (f_1–f_{12}). Podaci uključuju

prosječnu vrijednost najboljih rješenja (x_{avg}), najbolju postignutu vrijednost (x_{best}), najlošiju postignutu vrijednost (x_{worst}) te standardnu devijaciju (σ), koja prikazuje raspršenost rješenja. Ovi rezultati prikupljeni su nakon zadnje generacije algoritma i služe kao ključni pokazatelji uspješnosti svakog mehanizma.

Iz rezultata se može zaključiti da su ponovna inicijalizacija i refleksija u većini slučajeva najučinkovitiji mehanizmi, s nižim prosječnim vrijednostima i standardnim devijacijama, dok konzervatizam uglavnom postiže lošije rezultate, posebno na funkcijama s većim stupnjem složenosti. Za D=30, sve instance algoritma su većinom uspjele pronaći globalni minimum, uz iznimke na funkcijama f_8 i f_9. Ove funkcije obično predstavljaju veliki izazov za optimizacijske algoritme zbog većeg broja lokalnih minimuma. Lokalni minimumi često mogu algoritam odvesti u krivi smjer pretrage, pa im je potrebno više vremena za pronalazak globalnog minimuma ili se čak nikada ne izvuku iz lokalnog minimuma i ne pronađu globalni. Veći broj dimenzija rješenja dovodi do složenijeg problema za optimiziranje i rezultira lošijom učinkovitosti, što se može vidjeti u tablici 4.4.

Nadalje, kroz izvršavanje algoritma pratili su se i podaci o samoj primjeni mehanizama. Pošto se iz prethodne tablice može vidjeti da su skoro sve instance algoritma, neovisno o odabranoj funkciji i mehanizmu, uspjele pronaći globalni minimum, prikazom podataka o primjeni mehanizama može se bolje prikazati učinkovitost mehanizama. U tablici 4.5, stupac $n_{primjena}$ prikazuje ukupan broj primjena mehanizama na komponente mutant vektora kroz sve generacije populacije. Kako su konfiguracije parametara i početna populacija jednake za sve mehanizme, $n_{primjena}$ daje uvid u učinkovitost mehanizma da konvergira rješenja prema globalnom minimumu, odnosno unutar granica prostora pretrage, uz što manji broj primjena. Veći broj primjena obično označava da je mehanizam nakon primjene generirao vrijednosti rješenja koja su potencijalno bliže granicama prostora te su u narednim generacijama opet izašla van granica, zahtijevajući ponovnu primjenu mehanizma.

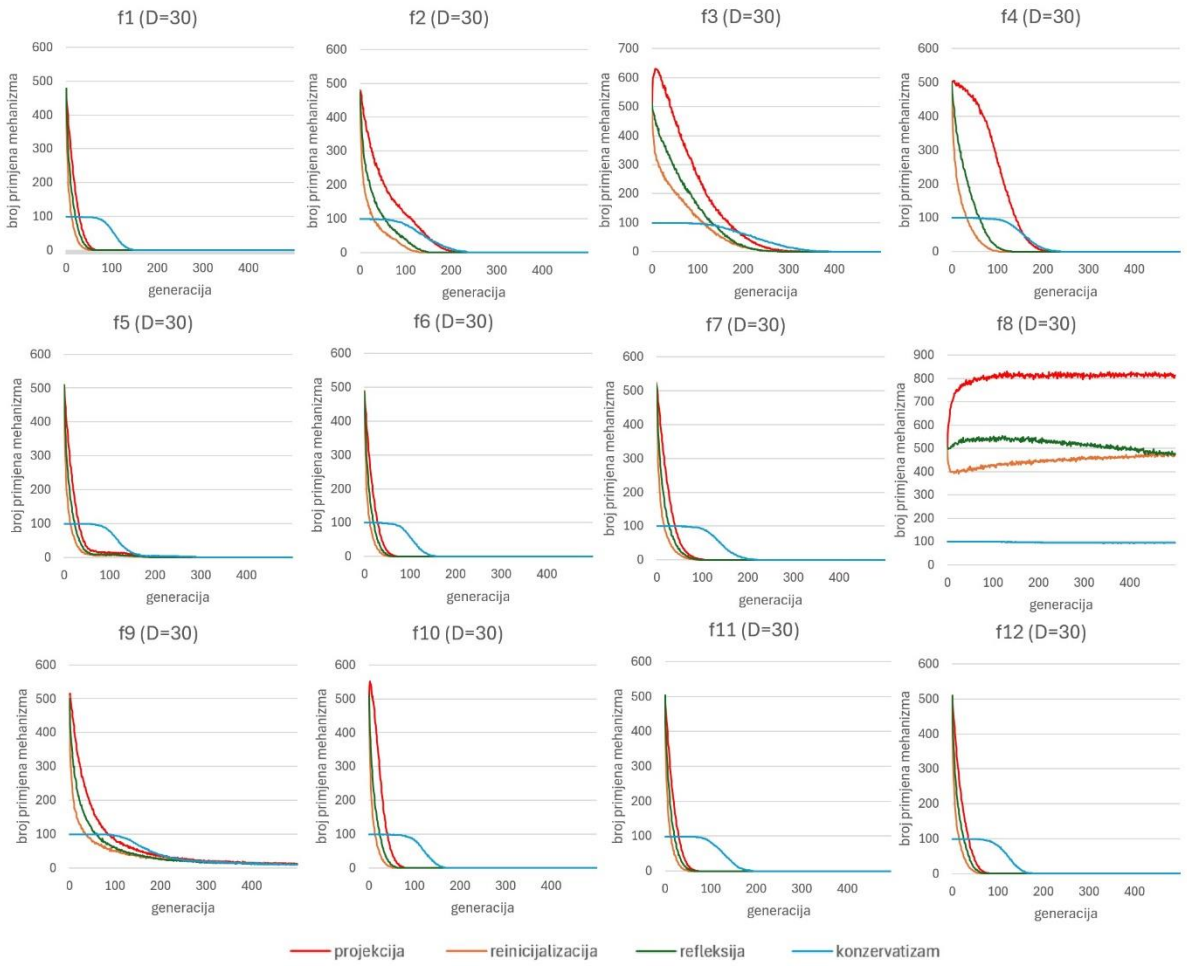
Iz rezultata je vidljivo da je mehanizam ponovne inicijalizacije uvijek imao najmanje primjena, dok su metoda konzervacije i projekcija imali najviše primjena. Metoda projekcije uvijek stvara rješenja na granicama prostora, pa je i pretpostavka da ima veći broj primjena pošto joj treba duži period za konvergenciju prema nuli. Kada se primjeni metoda konzervacije, vektor koji je prethodno izašao izvan granica prostora odabire se za novu generaciju. Time se povećava vjerojatnost da će i u sljedećim generacijama taj vektor ponovno izaći izvan dopuštenih granica, što objašnjava učestalu potrebu za primjenom mehanizma.

$n_{odabranih}$ prikazuje broj odabranih probnih vektora za iduću generaciju, nakon što je barem jedna komponenta rješenja izašla izvan granica prostora. Iako je mehanizam ponovne inicijalizacije uvijek imao najmanji broj primjena, kod funkcija s užim granicama prostora (f_8 i f_9) mehanizam projekcije češće je prouzrokovao odabir probnog vektora. Ukoliko se primjeni mehanizam konzervacije, nikad se neće uzeti probni vektor za iduću generaciju pa je $n_{odabranih}$ uvijek nula. Stupac g_{min} predstavlja zadnju generaciju populacije u kojoj je mehanizam bio primijenjen. Uočavamo da mehanizmi prestaju biti primjenjivani ranije u algoritmima s jednostavnijim funkcijama cilja (f_1), dok kod složenijih funkcija (f_3 i f_7) mehanizmi ostaju aktivni sve do kasnijih faza algoritma. Funkcija f_8 i f_9 pokazale su se kao vrlo zahtjeve zbog većeg broja lokalnih minimuma, te nijedan mehanizam pri odabranoj konfiguraciji nije uspio konvergirati rješenja prema nuli.

Može se zaključiti da je mehanizam ponovne inicijalizacije najučinkovitiji u smislu odabira probnih vektora, iako je rjeđe primjenjivan. Projekcija i refleksija također imaju značajan broj odabranih vektora, ali s više primjena u odnosu na ponovnu inicijalizaciju. Konzervatizam se pokazuje najmanje učinkovitim jer, unatoč učestaloj primjeni, treba mu veći broj generacija populacije kako bi se prestala generirati rješenja van granica prostora.

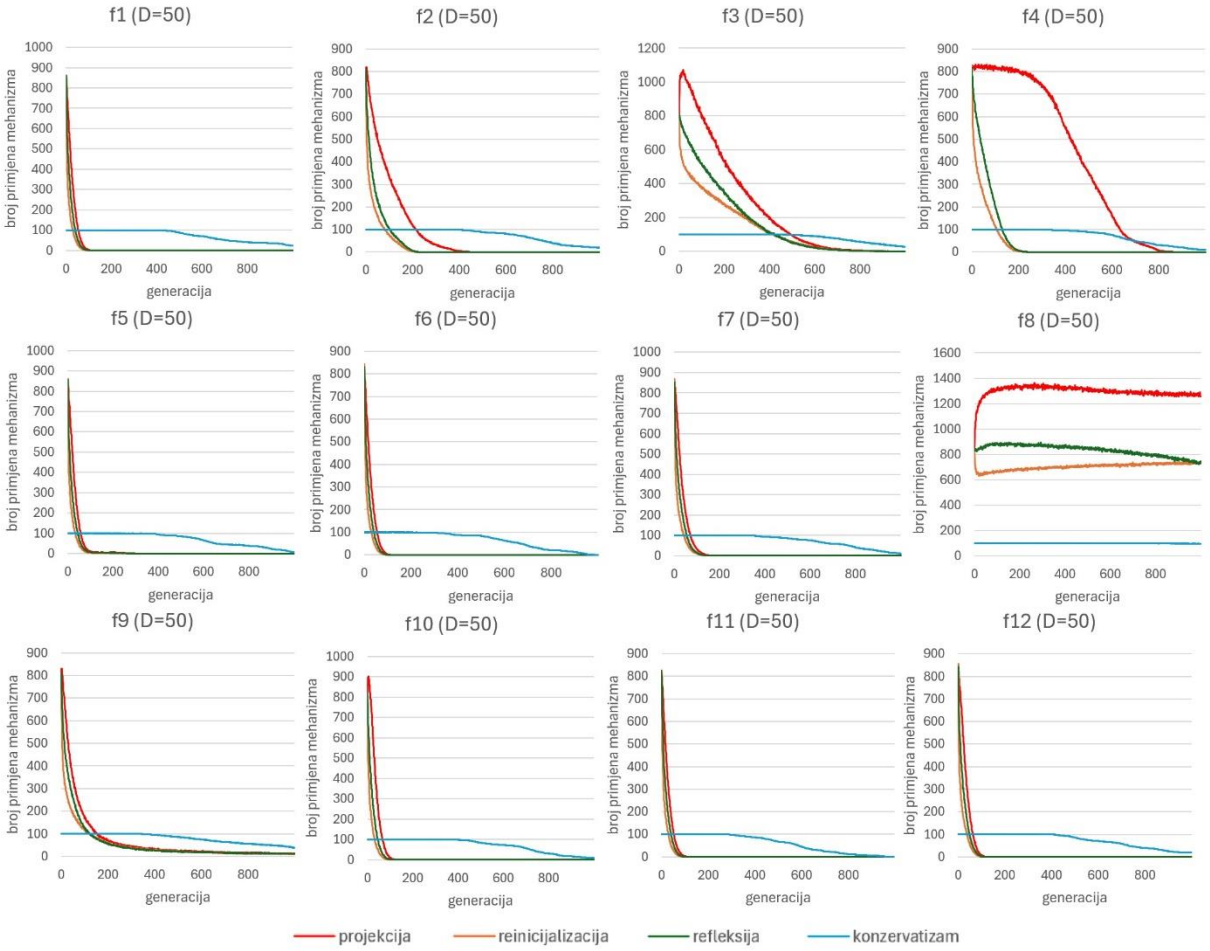
Tablica 4.5. *Prikaz podataka o primjeni mehanizama*

		$n_{primjena}$		$n_{odabranih}$		g_{min}	
Funkcija cilja	**Mehanizam**	**D = 30**	**D = 50**	**D = 30**	**D = 50**	**D = 30**	**D = 50**
f_1	projekcija	8719	23382	401	619	70	110
	reinicijalizacija	**3707**	**9714**	**471**	**722**	**55**	**92**
	refleksija	5878	14976	432	663	62	99
	konzervatizam	10348	74616	0	0	153	/
f_2	projekcija	28939	83335	**639**	**828**	222	438
	reinicijalizacija	**9358**	**23390**	559	781	**136**	**213**
	refleksija	14721	33224	555	748	154	223
	konzervatizam	14141	75515	0	0	252	/
f_3	projekcija	61668	254508	**788**	**919**	331	912
	reinicijalizacija	28327	131288	475	578	290	**877**
	refleksija	38276	163048	663	780	**285**	892
	konzervatizam	**22304**	**81411**	0	0	389	/
f_4	projekcija	52180	388366	0	0	217	860
	reinicijalizacija	**8951**	**32350**	**409**	**557**	**104**	**220**
	refleksija	16900	51777	356	473	132	238
	konzervatizam	15615	71028	0	0	240	/
f_5	projekcija	11497	26407	453	485	208	296
	reinicijalizacija	**5043**	**11320**	**482**	**679**	**177**	**268**
	refleksija	7597	16828	440	591	190	288
	konzervatizam	12213	70708	0	0	291	/
f_6	projekcija	8750	22861	404	603	72	112
	reinicijalizacija	**3736**	**9695**	**472**	**717**	**55**	**97**
	refleksija	5773	14804	426	634	62	102
	konzervatizam	10403	65531	0	0	158	987
f_7	projekcija	12867	30791	507	708	106	172
	reinicijalizacija	**5343**	**13501**	**558**	**808**	**89**	**150**
	refleksija	8273	20104	536	750	94	151
	konzervatizam	14001	73688	0	0	221	/
f_8	projekcija	806406	1296155	585	541	/	/
	reinicijalizacija	464465	701738	468	457	/	/
	refleksija	452930	831181	**1249**	**654**	/	/
	konzervatizam	**94492**	**99032**	0	0	/	/
f_9	projekcija	37040	72704	**528**	**678**	/	/
	reinicijalizacija	**23177**	**49408**	506	666	/	/
	refleksija	27039	56497	481	642	/	/
	konzervatizam	25230	79360	0	0	/	/
f_{10}	projekcija	14992	33584	**521**	674	83	127
	reinicijalizacija	**4226**	**10447**	489	**680**	**59**	**98**
	refleksija	6989	17127	459	616	67	105
	konzervatizam	11992	70864	0	0	172	/
f_{11}	projekcija	9202	22089	422	589	74	113
	reinicijalizacija	**3881**	**9453**	**493**	**697**	**55**	**88**
	refleksija	6099	14518	445	637	64	96
	konzervatizam	12713	58890	0	0	194	966
f_{12}	projekcija	10847	28348	275	420	82	125
	reinicijalizacija	**4541**	**11595**	**447**	**647**	**63**	**106**
	refleksija	7165	17727	393	563	70	112
	konzervatizam	11997	73398	0	0	176	/



Slika 4.1. *Grafovi ovisnosti primjene mehanizma o generaciji populacije (D=30)*

Grafovi na slici 4.1 prikazuju primjene različitih mehanizama tijekom izvođenja algoritma na dvanaest različitih funkcija cilja (f_1–f_{12}) uz vrijednost D = 30. Na x-osi svakog grafa nalazi se broj generacija, dok y-os prikazuje broj primjena pojedinih mehanizama u svakoj generaciji. Ovi grafovi omogućuju usporedbu učinkovitosti i frekvencije korištenja mehanizma, te pokazuju kako se njihova primjena smanjuje ili stabilizira kroz generacije. Iz grafova se može zaključiti da projekcija, ponovna inicijalizacija i refleksija na početku izvođenja većinom imaju sličan broj primjena, ali često vrlo brzo opadaju naspram mehanizma konzervacije. Kod mehanizma konzervacije većina mutant vektora koji proizlaze iz početnih generacija imaju barem jednu komponentu van granica prostora, pa se na skoro svaki mutant vektor primjenjuje mehanizam. Primjene obično krenu opadati oko generacije 100, dok kod drugih mehanizama primjene krenu opadati znatno ranije. Ponovno se može zaključiti kako je mehanizam ponovne inicijalizacije najbolji izbor od četiri ponuđena mehanizma, a uz njega vrlo učinkovit je i mehanizam refleksije.



Slika 4.2. *Grafovi ovisnosti primjene mehanizma o generaciji populacije (D=50)*

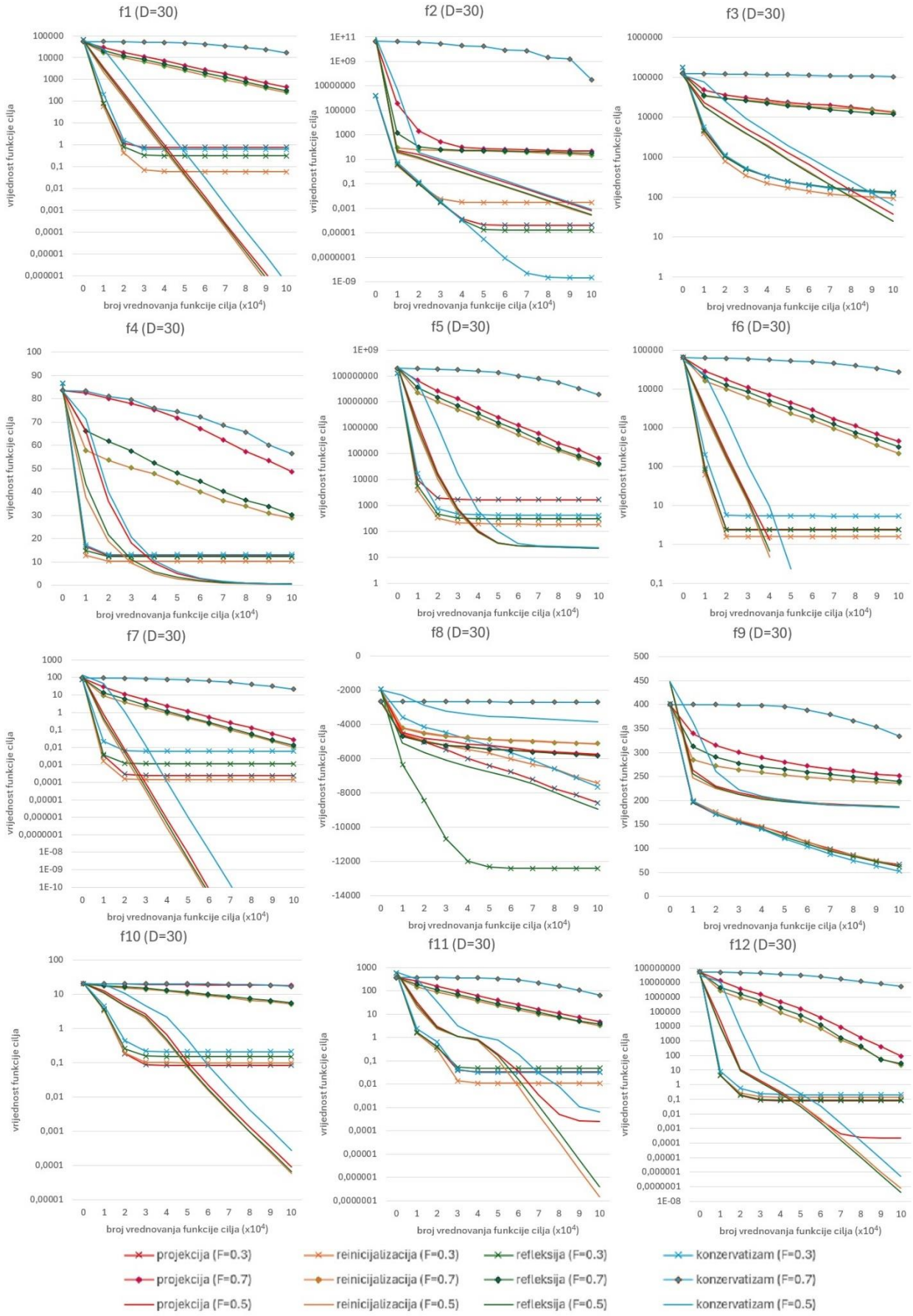
Uspoređujući grafove za D=30 i D=50, vidljivo je da povećanje dimenzionalnosti problema generalno dovodi do većeg broja primjena mehanizma za rukovanje ograničenjima. Za većinu funkcija, broj primjena mehanizma se i u ovom slučaju smanjuje s vremenom, što znači da populacija konvergira prema dozvoljenom prostoru pretrage. Pri vrijednosti D=50, svim mehanizmima je trebalo nešto duže da konvergiraju rješenja unutar granica, oko 200 generacija, dok je metoda konzervacije samo kod funkcije f_{11} uspjela konvergirati sva rješenja. Iz tog se može zaključiti da metodi konzervacije učinkovitost znatno više opada s povećanjem složenosti problema, nego što je to slučaj kod drugih mehanizama.

4.2.2. Utjecaj vrijednosti faktora skaliranja na učinkovitost mehanizma

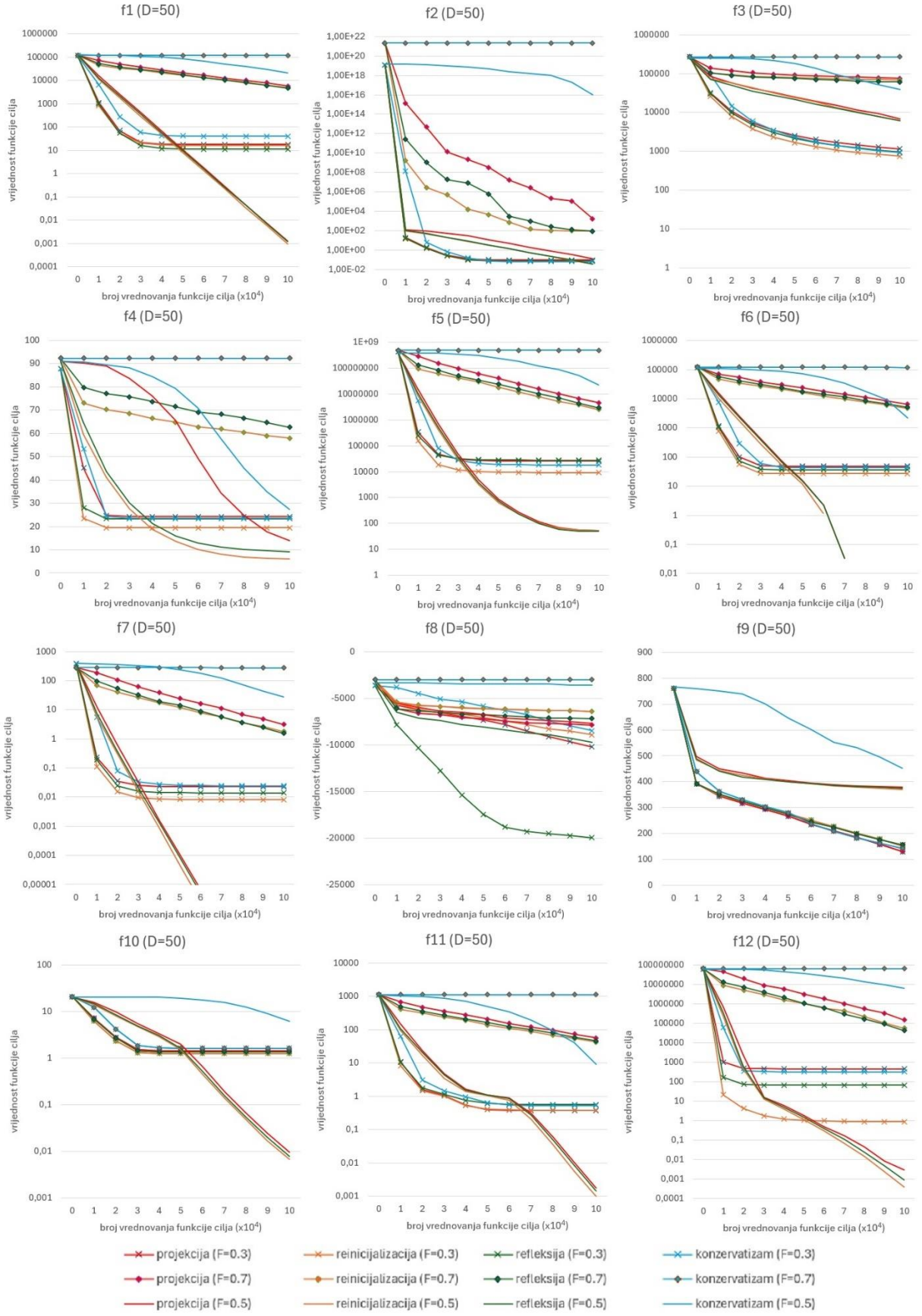
Na slici 4.3 i 4.4 prikazani su grafovi konvergencije najboljih rješenja za dvije vrijednosti D. Ovi grafovi prikazuju kako algoritmi napreduju prema boljim rješenjima tijekom izvršavanja. Na y-osi grafova prikazane su vrijednosti funkcije cilja, dok je na x-osi broj vrednovanja funkcije cilja. S

obzirom da je algoritmima cilj minimizirati funkciju cilja, niže vrijednosti na y-osi ukazuju na bolja rješenja. Za svaku funkciju prikazane su konvergencije najboljih rješenja za sva četiri mehanizma, te za tri odabrane vrijednosti faktora skaliranja.

Iz grafova se može vidjeti da su, za odabrane funkcije i korištene mehanizme, većinom učinkovitiji algoritmi s nižim vrijednostima F-a. Veće vrijednosti F-a su nešto korisnije na početku izvršavanja, kada je potrebno istražiti širi prostor rješenja, ali vrlo brzo se manje vrijednosti F-a pokazuju boljima preciznije pretrage. Kod funkcija f_1, f_7 i f_{10} nema znatnih razlika između instanci algoritma s D=30 ili D=50 međutim, vrijednost F=0.5 pokazala se kao najučinkovitija s konstantom konvergencijom prema globalnom minimumu. Za D=30, metoda konzervacije uspjela je pronaći globalni minimum (barem za F=0.5), no kod D=50 rješenja ima značajno veći broj varijabli pa je i veća mogućnost da je mutant vektor izvan granica prostora, uzimajući ciljni vektor za novu generaciju. Za funkcije f_2 i f_9, za obje vrijednosti D, vrijednost F=0.3 pokazala se kao najuspješnija. Razlog tome je što funkcije f_2 i f_9 imaju relativno uske granice prostora ([-10, 10] i [-5.12, 5.12]), pa niža vrijednost F-a smanjuje rizik od izlaska iz granica prostora. Kod funkcije f_8 većina mehanizama pri bilo kojoj vrijednosti F-a pokazivala je slične rezultate, s iznimkom za mehanizam refleksije pri F=0.3. Mehanizam refleksije s navedenim parametrima pokazao je iznimnu učinkovitost i jedini je uspio dokučiti globalni minimum (-12569.5 za funkciju f_8), a razlog je što ova funkcija ima jako puno lokalnih minimuma, pa se mehanizmom refleksije najbolje uspije uravnotežiti precizno istraživanje i stabilnu pretragu unutar granica prostora.



Slika 4.3. *Grafovi konvergencije najboljih rješenja za D=30 i za različite vrijednosti F*

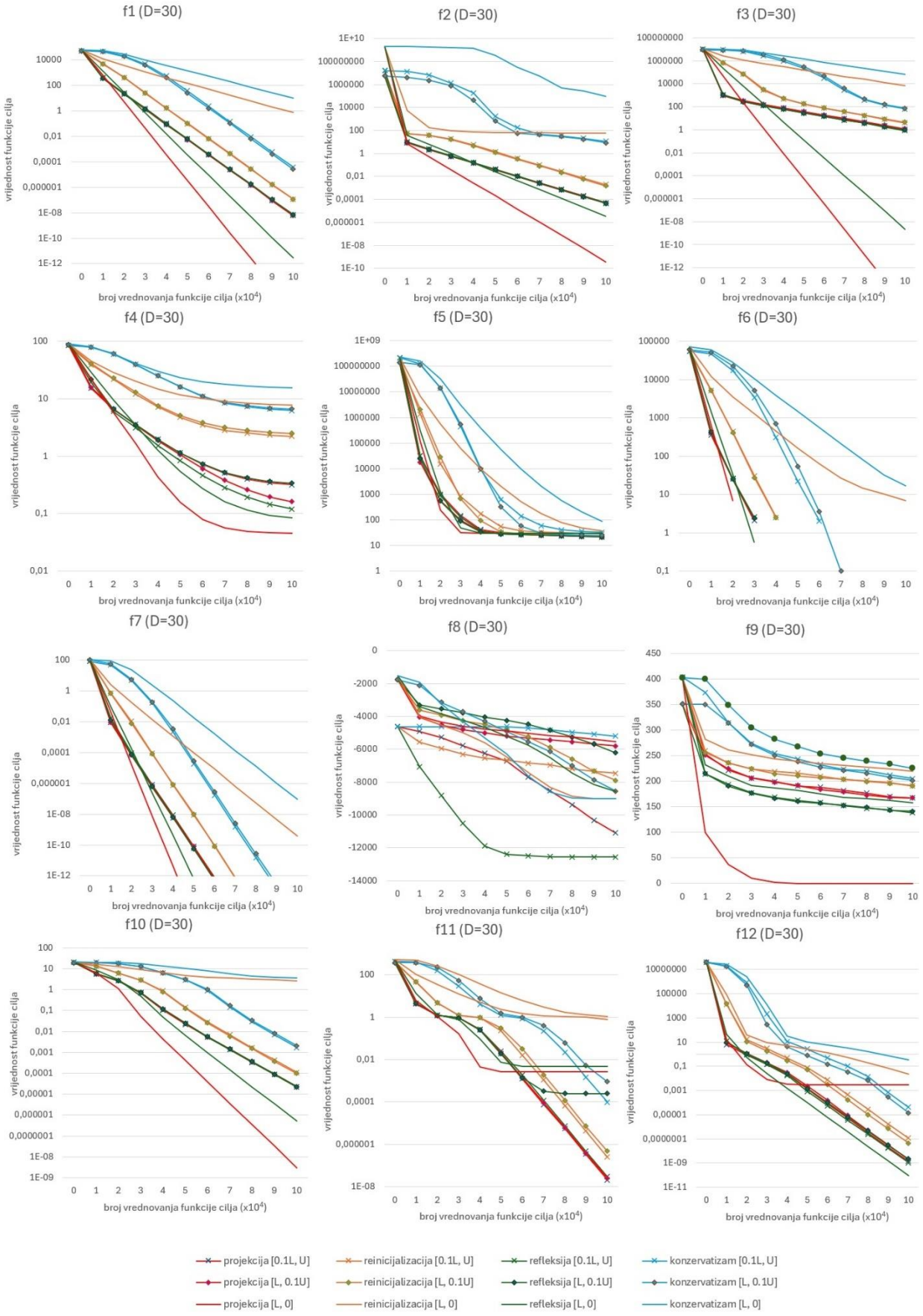


Slika 4.4. *Grafovi konvergencije najboljih rješenja za D=50 i za različite vrijednosti F*

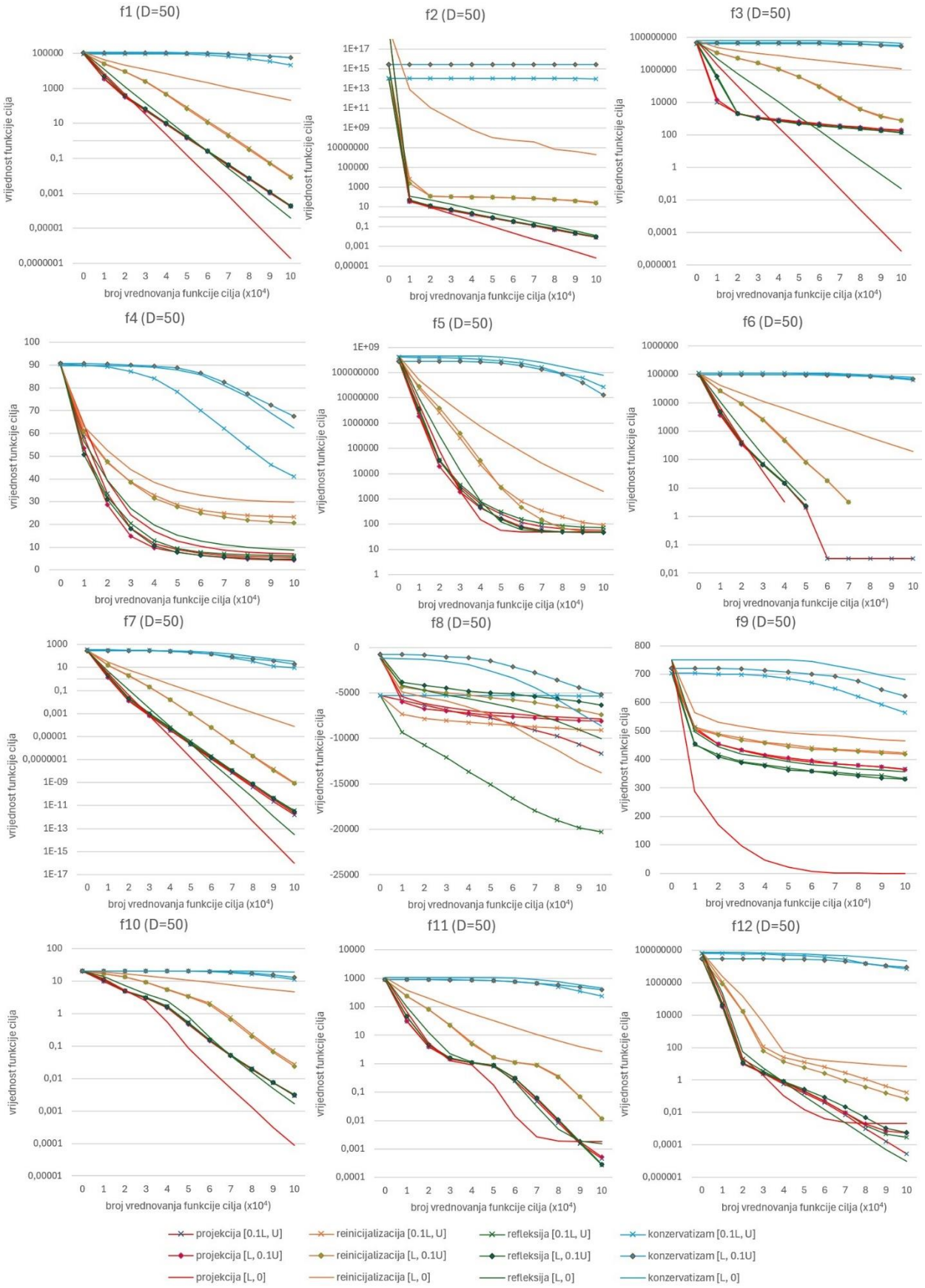
4.2.3. Utjecaj promjene granica prostora pretrage na učinkovitost mehanizma

Odabir različitih vrijednosti granica prostora pretrage izveden je kako bi se prikazao utjecaj promjena u veličini i asimetriji pretrage na učinkovitost mehanizama za rukovanje ograničenjima. Za ovu analizu odabrane su tri manipulacije prostora pretrage. Donja granica je smanjena na 10% svoje originalne vrijednosti ([0.1L, U]) kako bi se istražilo kako će reducirani negativni prostor utjecati na algoritam. S druge strane, granice [L, 0.1U] smanjuju gornju granicu na 10% svoje početne vrijednosti, čime se ograničava istraživanje u pozitivnom smjeru. Konačno, postavljanje granica na [L, 0] eliminira pozitivan dio prostora pretrage i ograničava algoritam na pretraživanje u negativnom dijelu. Ove manipulacije su odabrane kako bi se procijenila osjetljivost algoritma na različite veličine prostora pretrage te kako bi se utvrdilo kako varijacije u granicama utječu na generiranje i selekciju novih rješenja, kada globalni minimum više nije u središtu prostora pretrage.

Slike 4.5. i 4.6. prikazuju grafove konvergencija za dvije vrijednosti D, uz navedene promjene granica prostora pretrage. Vidljivo je da nema značajne razlike u ponašanju algoritma između grafova s D=30 i D=50, već su vrijednosti najboljih rješenja proporcionalne složenosti problema, što objašnjava niže vrijednosti na slici 4.6. Također, odabrane funkcije i mehanizmi se obično ponašaju jednako kada je prostor pretrage većinom u negativnom dijelu ([L, 0.1U]) ili većinom u pozitivnom dijelu ([0.1L, U]), što se može zaključiti iz čestih preklapanja grafova za ove dvije vrijednosti granica prostora. Na algoritme s mehanizmom ponovne inicijalizacije znatno utječe promjena granica prostora. Prema prijašnjim eksperimentima, ovaj mehanizam se pokazao kao najučinkovitiji, dok se u ovom eksperimentu pokazao kao jedan on najlošijih, uz metodu konzervacije. Mehanizam projekcije uvijek postavlja nevaljala rješenja na same granice prostora, a ovdje je ograničen prostor pretrage tako da je globalni minimum blizu granica. Upravo iz tog razloga se projekcija pokazala kao najučinkovitiji mehanizam u eksperimentu s gornjom granicom prostora pretrage postavljenom na nula ([L, 0]).



Slika 4.5. *Grafovi konvergencije najboljih rješenja za D=30 i za različite vrijednosti L i U*



Slika 4.6. *Grafovi konvergencije najboljih rješenja za D=50 i za različite vrijednosti L i U*

5. ZAKLJUČAK

Kao i odabir parametara algoritma, odabir mehanizma za rukovanje ograničenjima prostora predstavlja bitan korak u optimizacijskim problemima. Zadatak ovog rada bio je prikazati na koji način različiti mehanizmi utječu na učinkovitost algoritma DE, te je naveden razlog zašto je uopće nužno rukovati ograničenjima prostora. Opisana je struktura i parametri algoritma DE i način na koji rješenja mogu izaći iz granica prostora pretrage. Također, prikazani su često korišteni mehanizmi za rukovanje ograničenjima, te su detaljno opisana četiri korištena mehanizma. Za potrebe eksperimentalne analize, ostvareno je programsko rješenje koje nudi korisniku unos svih parametara algoritma, te odabir mehanizma i funkcije cilja.

Kroz eksperimentalnu analizu prikazano je kako odabrani mehanizmi utječu na performanse algoritma. Uspoređena su četiri mehanizma za rukovanje ograničenjima pri često korištenoj konfiguraciji parametra algoritma DE. Također, odrađene su analize za dvije vrijednosti dimenzionalnosti vektora rješenja kako bi se pokala učinkovitost mehanizama pri složenijim problemima. Metoda ponovne inicijalizacije se pokazala najučinkovitija, neovisno o odabranom faktoru skaliranja, dok je god globalni minimum funkcije bio u središtu prostora pretrage. Ukoliko su granice prostora pomaknute bliže minimumu, metoda projekcije daje bolju učinkovitost.

U budućem radu moglo bi se unaprijediti programsko rješenja tako da se korisniku ponudi odabir podataka koje algoritam prati tijekom izvršavanja. Isto tako, mogli bi biti prikazani još neki mehanizmi za rukovanje ograničenjima, kao i odrađene analize za još složenije funkcije cilja. Moguća je i usporedba rezultata pri promjeni nekih drugih parametara algoritma, primjerice veličine populacije ili faktora križanja. Također, bilo bi korisno vidjeti rezultate za neke druge operatore mutacije, pošto je upravo mutacija korak koji može izbaciti rješenja izvan granica prostora pretrage.

LITERATURA

[1] R. Storn and K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, prosinac 1997.

[2] A. P. Engelbrecht, Computational Intelligence: An Introduction, Second Edition, Wiley Publishing, Pretoria, rujan 2007.

[3] D. Bajer, Adaptive k-tournament mutation scheme for differential evolution, Applied Soft Computing, vol. 85, prosinac 2019.

[4] V.-H. Truong and S.-E. Kim, Reliability-based design optimization of nonlinear inelastic trusses using improved differential evolution algorithm, Advances in Engineering Software, vol. 121, pp. 59–74, srpanj 2018.

[5] N. H. Awad, M. Z. Ali, R. Mallipeddi, and P. N. Suganthan, An efficient differential evolution algorithm for stochastic opf based active–reactive power dispatch problem considering renewable generators, Applied Soft Computing, vol. 76, pp. 445–458, ožujak 2019.

[6] G. Strofylas, K. Porfyri, I. Nikolos, A. Delis, and M. Papageorgiou, Using synchronous and asynchronous parallel differential evolution for calibrating a second-order traffic flow model, Advances in Engineering Software, vol. 125, pp. 1–18, studeni 2018.

[7] R. Biedrzycki, J. Arabas, and D. Jagodzinski, Bound constraints handling in differential evolution: An experimental study, Swarm and Evolutionary Computation, vol. 50, p. 100453, studeni 2019.

[8] D. Bajer, Parameter control for differential evolution by storage of successful values at an individual level, Journal of Computational Science, vol. 68, p. 101985, travanj 2023.

[9] A. E. Eiben, R. Hinterding, and Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 124–141, srpanj 1999.

[10] J. Arabas, A. Szczepankiewicz, and T. Wroniak, Experimental comparison of methods to handle boundary constraints in differential evolution, Proceedings of the Parallel Problem Solving from Nature, PPSN XI, pp. 411–420, rujan 2010.

[11] R. Boks, A. Kononova and H. Wang, Quantifying the impact of boundary constraint handling methods on differential evolution, Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1199 – 1207, svibanj 2021.

[12] I. Trivedi, A. Gandomi, P. Jangir and N. Jangir, Study of Different Boundary Constraint Handling Schemes in Interior Search Algorithm, Artificial Intelligence and Evolutionary Computations in Engineering Systems, vol. 51., pp. 547–564, srpanj 2017.

[13] V. Kreischer, T. Magalhaes, H. Barbosa and E. Krempser, Evaluation of Bound Constraints Handling Methods in Differential Evolution using the CEC2017 Benchmark, XIII Brazilian Congress on Computational Intelligence, pp. 1-12, Rio de Janeiro, 2017.

[14] M. Mitran, A. Kononova, F. Caraffini and D. Zaharie, Patterns of Convergence and Bound Constraint Violation in Differential Evolution on SBOX-COST Benchmarking Suite, Proceedings of the Companion Conference on Genetic and Evolutionary Computation, svibanj 2023.

[15] E. Silva, H. Barbosa and A. Lemonge, An adaptive constraint handling technique for differential evolution in engineering optimization, International Conference on Engineering Optimization, Rio de Janeiro, 2008.

[16] J. Lampinen, A constraint handling approach for the differential evolution algorithm, Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1468-1473, Honolulu, 2002.

[17] S. Das and P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 4–31, veljača 2011.

[18] W.-j. Yu and J. Zhang, Adaptive differential evolution with optimization state estimation, Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 1285–1292, srpanj 2012.

[19] F. Penunuri, C. Cab, O. Carvente, M.A. Zambrano-Arjona, J.A. Tapia, A study of the Classical Differential Evolution control parameters, Swarm and Evolutionary Computation, vol. 26, pp. 86-96, siječanj 2016.

[20] F. Nerri and V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artificial Intelligence Review, vol. 33, pp. 61-106, veljača 2010.

[21] T. Eltaeib and A. Mahmood, Differential Evolution: A Survey and Analysis, Applied Sciences, vol. 8, no. 10, 2018.

[22] X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computing, vol. 3, no. 2, pp. 82–102, srpanj 1999.

SAŽETAK

U radu je opisan problem rukovanja ograničenjima prostora u diferencijalnoj evoluciji (DE), evolucijskom algoritmu koji se često koristi za optimizacijske probleme. Prikazana su četiri različita mehanizma za rukovanje ograničenjima prostora: projekcija, refleksija, ponovna inicijalizacija te konzervativna metoda. Dan je pregled osnovnih koraka DE i razlog zbog kojeg rješenja uopće izlaze iz granica prostora. Implementirana je standardna inačica algoritma DE i dan je opis i prikaz načina rada. Ostvareno programsko rješenje omogućuje izvođenje algoritma DE s prilagodljivim parametrima i odabirom različitih funkcija cilja. Odrađena je analiza učinkovitosti sva četiri mehanizma na nekoliko poznatih optimizacijskih problema. Kroz rezultate analize, pokazano je da odabir mehanizma za rukovanje ograničenjima, kao i odabir parametara algoritma, može znatno utjecati na učinkovitost algoritma.

Ključne riječi: diferencijalna evolucija, kontinuirana optimizacija, mutacija, rukovanje ograničenjima prostora

ABSTRACT

Handling bound constraints in differential evolution

The paper addresses the problem of handling bound constraints in Differential Evolution (DE), an evolutionary algorithm frequently used for optimization problems. Four different mechanisms for handling bound constraints are presented: projection, reflection, reinitialization, and the conservative method. An overview of the basic steps in DE is given, along with an explanation of why solutions can be out of bounds. A standard version of the DE algorithm has been implemented, and its use was demonstrated. The developed software solution allows for the execution of the DE algorithm with adjustable parameters and the choice of different objective functions. An analysis of the efficiency of all four constraint-handling mechanisms has been conducted on several well-known optimization problems. The results demonstrate that the choice of constraint-handling mechanism, as well as the selection of algorithm parameters, can significantly impact the algorithm's efficiency.

Keywords: differential evolution, continuous optimization, mutation, bound constraint handling

ŽIVOTOPIS

Dino Radonjić rođen je 5. prosinca 2000. godine u Osijeku. Osnovnu školu pohađao je u Donjem Miholjcu u Osnovnoj školi „August Harambašić“. 2015. godine upisuje srednju elektrotehničku školu u Srednjoj školi Valpovo koju završava 2019. godine. Iste godine, izravnim upisom, upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek, kojeg završava 2022. godine. Nakon toga upisuje prvu godinu diplomskog sveučilišnog studija računarstva, smjer Programsko inženjerstvo. U 2023. godini odrađuje stručnu praksu u tvrtki Mono Software, nakon koje nastavlja raditi kao razvojni programer u istoj tvrtki.