Fakulteta za elektrotehniko,
računalništvo in informatiko

Miha Ravber

UPORABA ŠAHOVSKEGA SISTEMA RANGIRANJA ZA PRIMERJAVO EVOLUCIJSKIH ALGORITMOV VEČKRITERIJSKE OPTIMIZACIJE

Magistrsko delo

Maribor, 2015

UPORABA ŠAHOVSKEGA SISTEMA RANGIRANJA ZA PRIMERJAVO EVOLUCIJSKIH ALGORITMOV VEČKRITERIJSKE OPTIMIZACIJE

Magistrsko delo

Študent:	Miha Ravber
Študijski program:	Računalništvo in informacijske tehnologije (MAG)
Mentor:	doc. dr. Matej Črepinšek

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

Številka: E5013409
Datum in kraj: 24. 02. 2015, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 46/2012)
izdajam

SKLEP O MAGISTRSKEM DELU

1. **Mihi Ravberju**, študentu študijskega programa 2. stopnje RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE, se dovoljuje izdelati magistrsko delo.

2. Tema magistrskega dela je pretežno s področja Inštituta za računalništvo.

MENTOR: doc. dr. Matej Črepinšek

3. Naslov magistrskega dela:
UPORABA ŠAHOVSKEGA SISTEMA RANGIRANJA ZA PRIMERJAVO EVOLUCIJSKIH ALGORITMOV VEČKRITERIJSKE OPTIMIZACIJE

4. Naslov magistrskega dela v angleškem jeziku:
A CHESS RATING SYSTEM FOR THE COMPARISON OF MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

5. Magistrsko delo je potrebno izdelati skladno z »Navodili za izdelavo magistrskega dela« in ga do 24. 02. 2016 v 2 vezanih in 1 v spiralo vezanem izvodu oddati v pristojni referat za študentske zadeve.

V skladu z Navodili o pripravi in oddaji e-diplom je potrebno magistrsko delo oddati v Digitalno knjižnico Univerze v Mariboru.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 15 dni.

Obvestiti:
1. kandidata
2. mentorja
3. odložiti v arhiv

UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO 6

Dekan:
red. prof. dr. Borut Žalik

ZAHVALA

Zahvaljujem se mentorju doc. dr. Mateju Črepinšku za pomoč in vodenje pri opravljanju magistrskega dela.
Rad bi se zahvalil tudi staršem, ki so mi omogočili študij.
Posebna zahvala gre Nataliji, ki mi je skozi študijska leta in pri pisanju magistrskega dela stala ob strani ter mi zmeraj pomagala.

Uporaba šahovskega sistema rangiranja za primerjavo evolucijskih algoritmov večkriterijske optimizacije

Ključne besede: Evolucijski algoritmi, večkriterijsko optimiranje, sistem rangiranja.

UDK: 004.421(043.2)

Povzetek

Magistrsko delo obravnava primerjavo evolucijskih algoritmov večkriterijske optimizacije z uporabo šahovskega rangiranja. Na začetku je opisano šahovsko rangiranje in osnovni pojmi večkriterijske optimizacije. Prikazana je nadgradnja orodja EARS (ang. Evolutionary Algorithms Rating System), ki omogoča ocenjevanje uspešnosti evolucijskih algoritmov za enokriterijsko optimizacijo. Predstavljena je implementacija primernih primerjalnih funkcij in nabora preizkusnih problemov. Prav tako so predstavljeni tudi nekateri bolj znani evolucijski algoritmi večkriterijske optimizacije, ki smo jih vključili v orodje EARS. Na koncu so prikazani rezultati in primerjava rezultatov orodja EARS z drugimi metodami.

A Chess Rating System for the Comparison of Multi-Objective Evolutionary Algorithms

Keywords: Evolutionary algorithms, multi-objective optimization, rating system.

UDK: 004.421(043.2)

Abstract

In this thesis the comparison of multi-objective evolutionary algorithms using chess ranking is presented. First, the chess ranking and the basic concepts of multi-objective optimization are described. Then the upgrade of EARS (Evolutionary Algorithms Rating System), which enables the assessment of the performance of evolutionary algorithms for single-objective optimization is presented. The implementation of appropriate comparator functions and a set of test problems is also presented. Some of more well-known evolutionary algorithms of multi-objective optimization which were included in EARS are shown. Finally, the results and the comparison of EARS results with other methods are outlined.

Kazalo

Slike

Seznam uporabljenih kratic in simbolov

BI	Boundary Intersection
CD	Crowding Dstance
CPU	Central Processing Unit
CSV	Comma Separated Values
DE	Differential Evolution
EA	Evolutionary Algorithm
EPS	Encapsulated PostScript
EP	External Population
ES	Evolutionary Strategy
GA	Genetic Algorithm
GDE	Generalized Differential Evolution
GD	Generational Distance
GP	Genetic Programming
IGD	Inverted Generational Distance
IP	Internal Population
MOEA	Multi-objective Evolutionary Algorithm
MOEA/D	Multi-objective Evolutionary Algorithm based on decomposition
MOP	Multi-objective Optimization Problem
NSGA	Elitist Non-Dominated Sorting Genetic Algorithm
OR	Operations Research
PAES	Pareto Archived Evolution Strategy
PF	Pareto Front
PS	Pareto Set
SPEA	Strength Pareto Evolutionary Algorithm

Poglavje 1

Uvod

V zadnjih letih je bilo predlaganih veliko evolucijskih algoritmov za večkriterijsko optimizacijo. Učinkovitost teh algoritmov je potrebno prikazati na umetnih ali dejanskih problemih s teoretično analizo ali empiričnim testiranjem. Da lahko algoritme primerjamo, morajo biti zadostno opisani in javno dostopni. Prav tako mora biti podano testno okolje. Vendar temu ni zmeraj tako. Literatura predlaganih algoritmov ni dostopna ali je nepopolna. Postavitev eksperimenta je le delno opisana. Tako je težko popolnoma ponoviti pogoje testiranja za pravično primerjavo algoritmov. Orodje EARS rešuje ta problem tako, da uporablja šahovsko rangiranje. Pri tem vsak algoritem predstavlja šahista in iskanje najboljše rešitve za določen problem predstavlja igro šaha. Dandanes je šahovsko rangiranje zelo zanesljivo za določitev ranga (moči) šahista. Orodje EARS dokazuje, da se lahko šahovsko rangiranje uporablja tudi za primerjavo algoritmov. Trenutne raziskave na področju uporabe šahovskega sistema rangiranja za primerjavo evolucijskih algoritmov poročajo samo o rezultatih primerjave evolucijskih algoritmov enokriterijske optimizacije, kar je tudi omejitev orodja EARS.

1.1 Cilji

Cilj magistrskega dela je ocenitev in primerjava evolucijskih algoritmov za večkriterijsko optimiranje. To bomo storili tako, da bomo nadgradili obstoječe orodje EARS, ki uporablja šahovsko rangiranje. Orodju bomo dodali ustrezne razširitve razredov, ki bodo omogočali večkriterijsko optimizacijo. V razširjenem orodju bomo implementi-

rali nekaj evolucijskih algoritmov za večkriterijsko optimizacijo in nabor preizkusnih večkriterijskih optimizacijskih problemov, ki jih bodo algoritmi reševali.

1.2 Struktura magistrskega dela

Magistrsko delo je razdeljeno na šest poglavij. Prvo poglavje je uvod, v drugem poglavju pa je predstavljeno večkriterijsko optimiranje. Opisane so tudi nekatere metrike uspešnosti in pristopi k večkriterijskem optimeranju. V tretjem poglavju so opisani evolucijski algoritmi, v četrtem je predstavljeno šahovsko rangiranje in orodje EARS ter njegova razširitev. V petem poglavju so opisani in prikazani testni scenariji in v šestem končamo z zaključkom.

Poglavje 2

Večkriterijsko optimiranje

Večkriterijsko optimiranje je področje, ki se ukvarja z matematično optimizacijo večkriterijskih optimizacijskih problemov (*Multi-objective Optimization Problem - MOP*). MOP je problem, ki ima dva ali več kriterijev, katere moramo optimizirati istočasno. Pomembno je omeniti, da so lahko na kriterijih uvedene omejitve. Poudariti je tudi potrebno, da so kriteriji MOP običajno med seboj v konfliktu. Če temu ni tako, potem obstaja samo ena rešitev za MOP. Splošno je lahko večkriterijski optimizacijski problem (MOP) formalno definiran kot:
Poišči vektor $\vec{X^*} = [X_1^*, X_2^*, ..., X_n^*]^T$, ki bo zadoščal m neenakostim:

$$g_i(x) \geq 0 \quad i = 1, 2, ..., m, \tag{2.1}$$

p enakostim

$$h_i(x) = 0 \quad i = 1, 2, ..., p, \tag{2.2}$$

in naj optimira vektorsko funkcijo

$$\vec{f}(x) = [f_1(x), f_w(x), ..., f_k(x)]^T \tag{2.3}$$

[2]

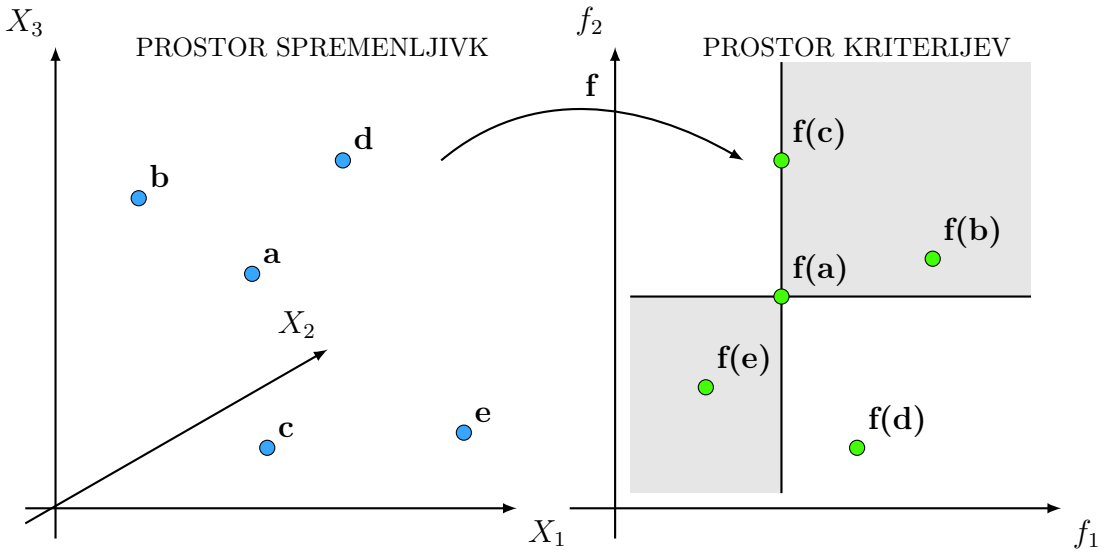
Vsak problem maksimizacije kriterijev, lahko enostavno prevedemo na problem minimizacije, tako bomo predpostavljali, da želimo vektorsko funkcijo $\vec{f}(x)$ minimizirati po vseh kriterijih. Ker je pri večkriterijskem optimiranju prostor kriterijev večdimenzionalen, si večina algoritmov za reševanje MOP pomaga s konceptom dominantnosti.

Rešitev večkriterijskega optimizacijskega problema x dominira rešitev y ($x \prec y$), če sta izpolnjeni naslednji zahtevi:

1. Rešitev x ni slabša od rešitve y po vseh kriterijih ($f_i(x) \leq f_i(y)$ za vse $i = 1, ..., k$).
2. Rešitev x je boljša od rešitve y po vsaj enem kriteriju ($f_j(x) < f_j(y)$ za vsaj en $j \in \{1, ..., k\}$).

Rešitev večkriterijskega optimizacijskega problema x šibko dominira rešitev y ($x \preceq y$), če je izpolnjena naslednja zahteva:

1. Rešitev x ni slabša od rešitve y po vseh kriterijih ($f_i(x) \leq f_i(y)$ za vse $i = 1, ..., k$).

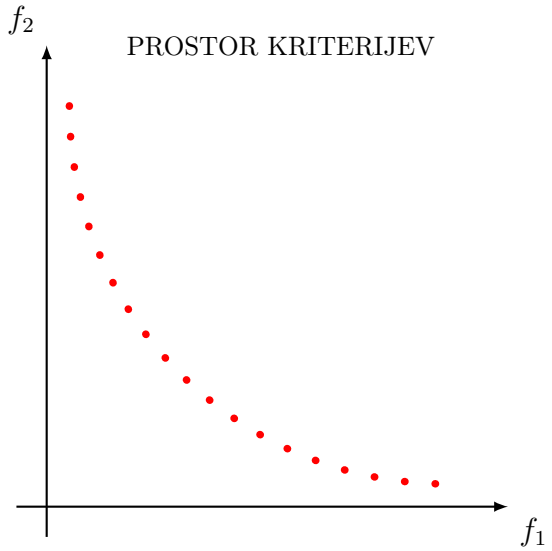


Slika 2.1: Prikaz koncepta dominantnosti na primeru večkriterijske funkcije f [3].

Slika 2.1 prikazuje primer večkriterijske funkcije f, ki preslika tridimenzionalen prostor spremenljivk v dvodimenzionalen prostor kriterijev. Ker želimo funkcijo f minimizirati, lahko iz slike (desno) razberemo, da za rešitev a velja, da dominira rešitvi b in c, je neprimerljiva z rešitvijo d in jo dominira rešitev e [3].

Pri večkriterijski optimizaciji se pojem "optimum" spremeni, saj iščemo dobre kompromise MOP, namesto ene same rešitve, kakor to delamo pri globalni optimizaciji.

Pojem "optimum", ki je najpogosteje sprejet, je bil prvotno uveden leta 1881 od Francisa Ysidra Edgewortha in kasneje, leta 1896 posplošen od Vilfreda Pareta [1], zato se imenuje Edgeworth-Pareto optimum, ali preprosto, Pareto optimum. Ta definicija pravi, da je rešitev za MOP Pareto optimalna, če ne obstaja nobena druga izvedljiva rešitev, ki bi zmanjšala en kriterij, brez da bi istočasno povečala vsaj en drug kriterij. Takšna rešitev torej ni dominirana od katere druge. Uporaba tega koncepta nam skoraj vedno da nabor rešitev in ne enotno rešitev. Nabor teh rešitev se imenuje Pareto optimalna fronta [2]. Primer Pareto optimalne fronte, lahko vidimo na sliki 2.2. Na sliki 2.1 predstavljata rešitvi e in d množico nedominiranih rešitev.



Slika 2.2: Primer konveksne Pareto fronte.

Kot primer večkriterijskega optimiranja si lahko predstavljamo, da želimo čim bolj varen avto, za čim manj denarja. Tako sta varnost in cena nasprotujoča si kriterija, saj najvarnejši avto ne bo hkrati tudi najcenejši. Vendar to ne pomeni, da je najdražji avto najvarnejši ali najcenejši avto najmanj varen. Različno varni avtomobili lahko imajo različne cene. Na Pareto fronti bodo najcenejši avtomobili za določeno varnost. Če na primer izberemo avto s srednjo varnostjo na Pareto fronti, ne bomo našli cenejšega avta z isto varnostjo. Tako lahko iz Pareto fronte izberemo takšen avto, ki ima dober kompromis med varnostjo in ceno.

2.1 Metrike uspešnosti

Glavni cilj pri večkriterijskem optimiranju je najti Pareto optimalno fronto za problem. Vendar v praksi lahko večkriterijsko optimiranje cilja na to, da bi našel reprezentativno podmnožico. Ker ta podmnožica običajno vsebuje slabše rešitve kot so v Pareto optimalni fronti, ali pa ne vsebuje vseh možnih rešitev (med katerimi lahko obstaja nešteto število za neštete probleme), obstaja potreba po merjenju kakovosti teh Pareto front. Številne metrike, znane tudi kot indikatorji so bile izoblikovane za ta namen [4].
Merjenje kakovosti te aproksimacije je bistveno težje kot pri enokriterijskem optimizacijskem problemu. V primeru, da imamo en kriterij lahko razliko med dobljenimi rešitvami nedvoumno izmerimo (in primerjamo z globalnim optimumom, če je znan) in ta meritev se lahko uporablja kot metrika uspešnosti. Za večkriterijsko optimiranje ne obstaja nobeno splošno sprejeto merilo uspešnosti.
Metrike za večkriterijske evolucijske algoritme (*Multi-objective Evolutionary Algorithm, MOEA*) na splošno odražajo kardinalno vrednost, povezano s številom točk rešitve, realno vrednost za bližino do Pareto fronte ali drugim nizom točk ali realno vrednost za merjenje lastnosti generirane Pareto fronte (raznolikost točk, uniformnost točk, itd.). Nekatere metrike so preproste za razumet in izračunat, druge morajo biti normalizirane in pomanjšane (linearno, nelinearno), da lahko ustvarimo primerljive vrednosti [5, 6].
Vsaka metrika uspešnosti razširi razred *QualityIndicator* in vrne uspešnost dobljene Pareto fronte. Za vsak problem, ki smo implementirali algoritem hrani datoteko, v kateri je shranjena optimalna Pareto fronta. Tako lahko zelo dobro ocenimo uspešnost posameznega MOEA. Algoritmu nastavimo metriko uspešnosti tako, da kličemo metodo *setQualityIndicator*, ki kot parameter sprejme objekt metrike. Če želimo izpisati vse metrike uspešnosti, v posameznem algoritmu nastavimo vrednost spremenljivke *display_data* na *true*. Ker je od metrike odvisno ali je manjša vrednost boljša ali slabša, smo razredu *QualityIndicator* dodali metodo *isMin*. Tako bomo za vsako metriko vedeli ali je rešitev potrebno minimizirati ali maksimizirati.

2.1.1 Generacijska razdalja

Generacijsko razdaljo (*Generational Distance, GD*) sta predstavila Van Veldhuizen in Lamont leta 2000. GD meri povprečno razdaljo elementov v Pareto fronti, ki jih

je našel algoritem do tistih v pravi Pareto fronti problema. Ta metrika je definirana kot:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2.4}$$

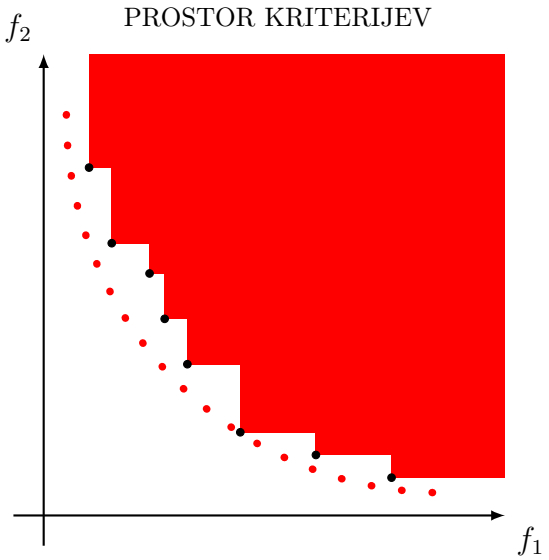
kjer je n število nedominiranih vektorjev, ki jih je našel algoritem in d_i evklidska razdalja (merjena v prostoru kriterijev) med vsakimi najdenimi člani in njihovim najbližjim članom v pravi Pareto fronti. Če je vrednost GD enaka nič, potem so vsi generirani elementi na pravi Pareto fronti problema. Torej, vse druge vrednosti kažejo, kako "daleč" smo od globalne Pareto fronte problema [7].

2.1.2 Obrnjena generacijska razdalja

Obrnjena generacijska razdalja (*Inverted Generational Distance, IGD*) meri povprečno razdaljo Pareto optimalne fronte do najdenega niza nedominiranih vektorjev. Tako se za vsakega člana Pareto optimalne fronte (aproksimacija) izmeri najkrajša razdalja do najdenega niza. Na koncu se izračuna povprečje teh razdalji [8].

2.1.3 Indikator HV

Indikator HV (*hypervolume*) ali S-metrika je postala široko uporabljena v zadnjih letih. HV je n-dimenzionalni prostor "zadržan" z n-dimenzionalnim nizom točk. Pri večkriterijskem optimiranju so lahko vrednosti n-dimenzionalnih kriterijev obravnavne kot točke. Torej HV niza je skupna velikost prostora, ki je dominirana od rešitev v nizu (glej sliko 2.3). HV povzema merilo razpršenosti rešitev vzdolž Pareto fronte, kot tudi razdaljo niza do Pareto optimalne fronte v enotno enostavno vrednost. Metrika ima nekaj koristnih matematičnih lastnosti. Bila je prva enočlenska metrika, ki je odkrila, da niz rešitev ni slabši kot drugi niz za vse pare rešitev. Poleg tega je maksimirana samo, če niz rešitev vsebuje vse Pareto optimalne točke.



Slika 2.3: Primer HV v dveh dimenzijah. Točke so zaznamovane s črnimi krogi. HV niza je prostornina prostora, ki je zajeta z vsemi točkami (obarvano z rdečo).

HV niza se meri glede na referenčno točko, običajno anti-optimalna točka ali "najslabša možna" točka za prostor. Izbira referenčne točke je pomembna, saj lahko vpliva na izid, ki izhaja iz primerjave dveh rešitev. Zaželeni so algoritmi z večjimi vrednostmi HV [4].

2.1.4 ε-indikator

ε -indikator daje faktor, za koliko je aproksimacijski niz slabši od drugega, glede na vse kriterije. V praksi se lahko izračuna po naslednji enačbi:

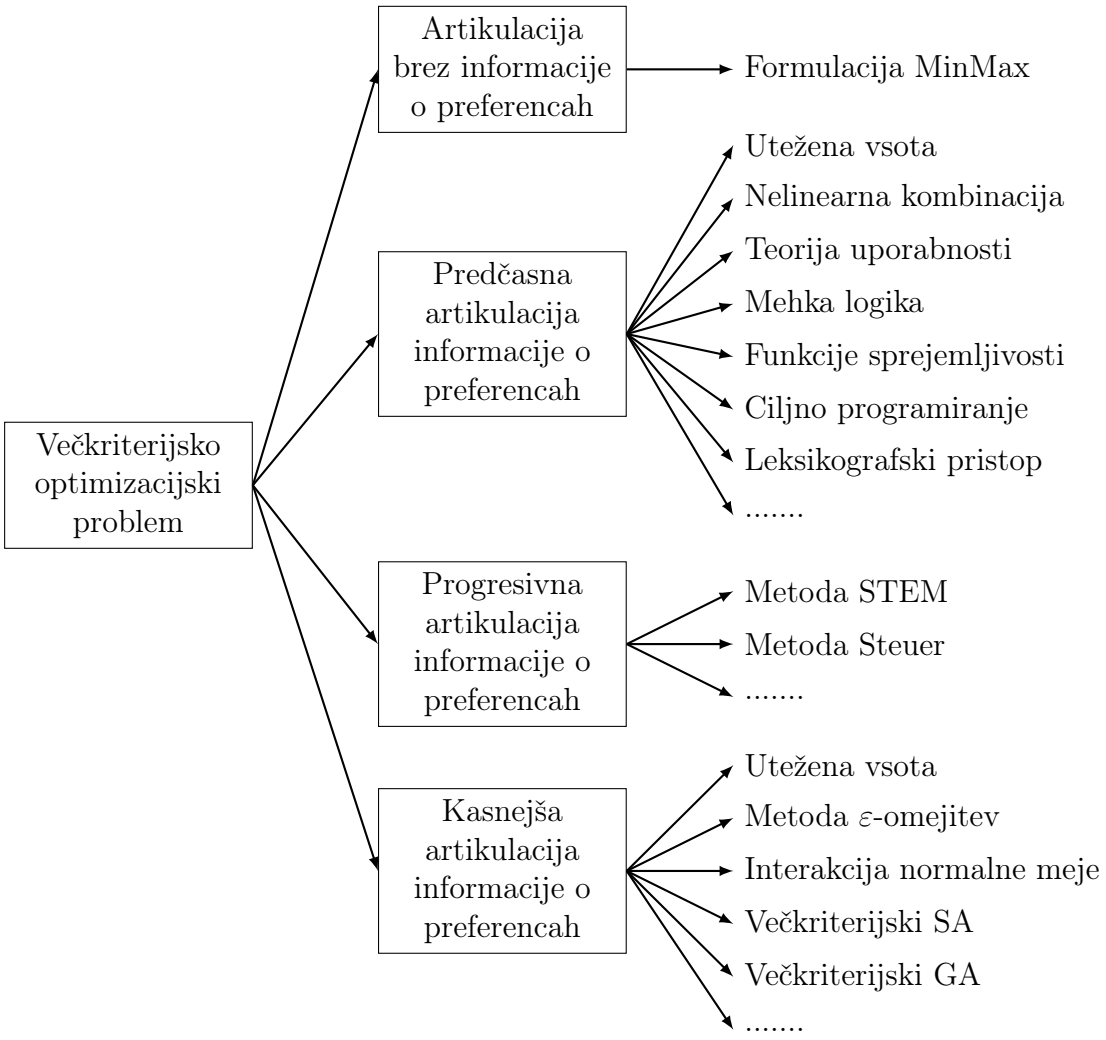
$$I_\varepsilon(A,B) = min\left\{\varepsilon \in \mathbb{R} | \forall b \in B \exists a \in A : a \succ_\varepsilon b\right\}, \tag{2.5}$$

kjer sta $A, B \subseteq X$. ε -indikator, $I_\varepsilon(A,B)$ je definiran kot minimum $\varepsilon \in \mathbb{R}$ tako, da je katerakoli rešitev $b \in B$ ε-dominirana z vsaj eno rešitvijo $a \in A$.

Ko je $I_\varepsilon(A,B) < 1$, so vse rešitve v B dominirane z rešitvijo v A. Če je $I_\varepsilon(A,B) = 1$ in $I_\varepsilon(B,A) = 1$, potem predstavljata A in B isto aproksimacijo Pareto fronte. Če je $I_\varepsilon(A,B) > 1$ in $I_\varepsilon(B,A) > 1$, potem sta A in B neprimerljiva (oba vsebujeta rešitve, ki niso dominirane iz drugega niza) [6].

2.2 Pristopi k večkriterijskem optimiranju

Večina optimizacijskih problemov je glede na njihovo naravo večkriterijskih. Obstaja veliko metod, ki so na voljo za reševanje tovrstnih problemov. Na splošno je MOP mogoče obravnavati na štiri različne načine. Način obravnave je odvisen od tega, kdaj odločevalec artikulira njegove preference za različne kriterije: nikoli, pred, med ali po dejanskem postopku optimizacije. Te različne metode so prikazane na sliki 2.4.



Slika 2.4: Klasifikacija nekaterih metod za večkriterijsko optimizacijo [9].

Ta klasifikacija je daleč od popolne, vendar daje dobro ogrodje za razpravljanje najpogostejših metod, primernih za optimizacijo. V magistrskem delu se bomo posvetili algoritmom, ki spadajo v kasnejšo artikulacijo informacije o preferencah.

2.2.1 Artikulacija brez informacije o preferencah

Te vrste metod ne uporabljajo nobenih podatkov o preferenci. Primera sta formulacija Min-Max in metoda globalnega kriterija.

2.2.2 Predčasna artikulacija informacije o preferencah

Najpogostejši način izvajanja večkriterijskega optimiranja je s predčasno artikulacijo preferenc odločevalca. Te metode potrebujejo odločevalca, da definira relativno pomembnost kriterijev MOP pred iskanjem (odloči → išči). To se običajno izraža v utežeh v povezavi s skupno vsoto kriterijev. Torej, preden se optimizacija dejansko izvede, se kriteriji združijo v eno samo merilo kakovosti. V realnem svetu znanstvenih in tehničnih problemov je iskanje ene rešitve, ki je v interesu odločevalca, netrivialna naloga [9].

Metoda utežene vsote

Najbolj enostaven in morda najbolj pogosto uporabljen pristop je metoda utežene vsote (*Weighted-sum approach*). Metoda optimizira pozitivno uteženo vsoto kriterijev. To pomeni, da je problem večkriterijskega optimiranja preoblikovan v skalarni optimizacijski problem:

$$min \sum_{i=1}^{k} w_i f_i(x), kjer \sum_{i=1}^{k} w_i = 1 \tag{2.6}$$

Preferenca odločevalca je upoštevana z izbiro različnih uteži w za različne kriterije. Ker so kriterijske funkcije običajno različnih magnitud, jih je morda potrebno najprej normalizirati. Čeprav je formulacija enostavna, je postopek nekoliko ad-hoc, saj ni jasne povezave med utežmi in dobljeno rešitvijo. Kako določiti vrednosti uteži glede na preference odločevalca, je tudi procedura ad-hoc. Druga pomanjkljivost te metode je, da ni mogoče poiskati rešitve na ne-konveksnem delu Pareto fronte [9].

Leksikografska metoda

Pri leksikografskih metodah odločevalec določi po katerem vrstnem redu se bodo kriteriji optimirali. Odločevalec tako razvrsti kriterije po pomembnosti. Tako kot je

v slovarju A pred B, odločevalec določi, da je kriterij i pred kriterijem j. Optimalno rešitev dobimo z minimizacijo kriterijskih funkcij. Pričnemo z najbolj pomembno in nadaljujemo po vrstnem redu pomembnosti. Pomanjkljivost leksikografske metode je, da morda niso upoštevani vsi kriteriji. Metoda se zdi najbolj primerna, kadar je pomembnost kriterijev (v primerjavi z ostalimi) dobro poznana [9].

2.2.3 Progresivna artikulacija informacije o preferencah

Ta razred metod se običajno imenuje interaktivne metode. Sklicujejo se na progresivno informacijo o preferencah odločevalca, medtem ko preiskujejo prostor rešitev (odloči $\leftrightarrow$ išči). Interaktivne metode so zelo pogoste na področju OR (*Operations Research*).

Te metode delajo v skladu s hipotezo, da odločevalec ne more predčasno navesti informacije o preferencah zaradi kompleksnosti problema. Vendar je odločevalec zmožen podati nekaj informacij o preferencah, ko je iskanje v teku. Odločevalec se uči o problemu ko se sooča z različnimi možnimi rešitvami problema. Prednosti teh metod so:

- predčasna informacija o preferencah ni potrebna,
- potrebna je samo informacija o lokalnih preferencah,
- je učni proces, kjer odločevalec pridobi znanje o problemu,
- ker odločevalec sodeluje pri iskanju, je bolj verjetno, da bo sprejel končno rešitev.

Slabosti so:

- rešitve so odvisne od tega, kako dobro lahko odločevalec artikulira svoje preference,
- veliko truda je potrebno od odločevalca med celotnim procesom iskanja,
- rešitev je odvisna od preferenc enega odločevalca. Če odločevalec spremeni svoje preference, ali če je prišlo do spremembe odločevalca, je postopek potrebno ponoviti,
- računska zahtevnost je večja kot v prejšnjih metodah [9].

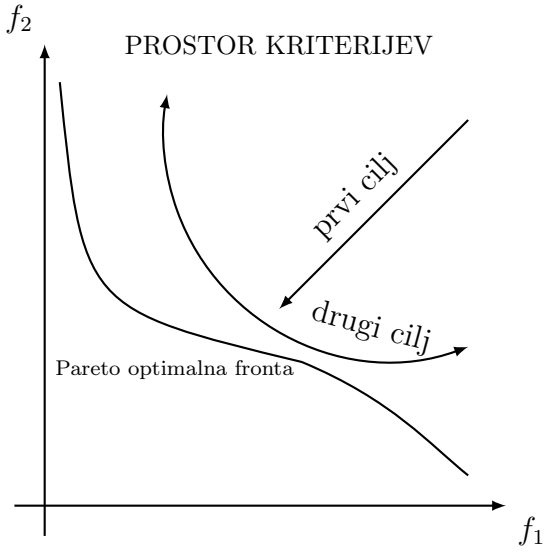
2.2.4 Kasnejša artikulacija informacije o preferencah

Te metode ne zahtevajo predhodne informacije o preferencah od odločevalca. Nekatere metode, vključene v to kategorijo, spadajo med najstarejše predlagane pristope za večkriterijsko optimiranje [6]. Obstajajo številne metode, ki najprej preiščejo prostor rešitev za nabor Pareto optimalnih rešitev in jih predložijo odločevalcu. Velika prednost s to vrsto metod je, da je rešitev neodvisna od preferenc odločevalca. Analizo je potrebno izvesti samo enkrat, saj se Pareto niz ne spremeni, dokler opis problema ostane nespremenjen. Vendar nekatere od teh metod trpijo pod veliko računsko obremenitvijo [9].

Pri reševanju problema s kasnejšo artikulacijo informacije o preferencah želimo, da večkriterijska optimizacijska metoda najde čim več Pareto optimalnih rešitev. Zaželeno je, da so dobljene rešitve čim bolj enakomerno porazdeljene po prostoru kriterijev, saj bomo izmed teh rešitev s pomočjo informacije o pomembnosti kriterijev izbrali najboljšo. Nalogo večkriterijskega optimiranja tako lahko pretvorimo v nalogo iskanja množice nedominiranih rešitev, za katere velja (slika 2.5):

- da so čim bliže Pareto optimlani fronti in
- da so enakomerno razporejene vzdolž Pareto optimalne fronte.

Ta dva cilja pa si pogosto nasprotujeta [3].



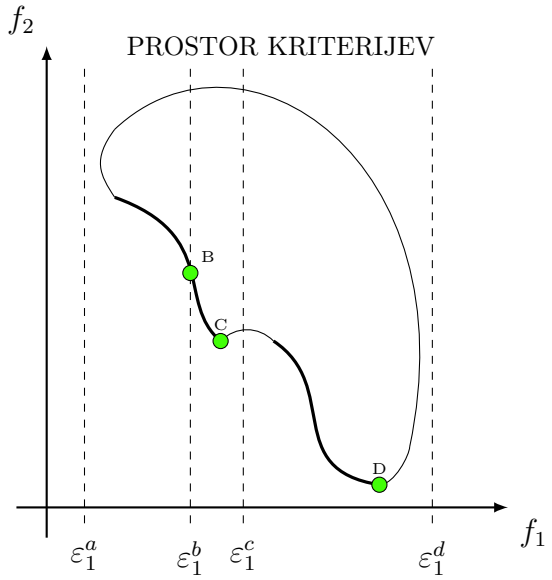
Slika 2.5: Cilja večkriterijskega optimiranja [3].

Metoda ε-omejitev

Pri metodi ε-omejitev (ε-*constraint*) je izbran en kriterij i, ki bo optimiran, ostali kriteriji so preoblikovani v omejitve. Ker pri metodi potrebujemo izhodiščno vrednost ε, bi jo lahko delno umestili tudi med pristope predčasne artikulacije preferenc. Preoblikovana optimizacijska naloga se tako glasi:

$$\begin{aligned} &min\left\{f_i\left(x\right)\right\} \\ &f_j\left(x\right) \leq \varepsilon_j, j \neq i, j = 1...k \end{aligned} \tag{2.7}$$

Delovanje metode ε-omejitev je prikazano na sliki 2.6. V našem primeru optimiramo $f_2\left(x\right)$, pri pogoju $f_1\left(x\right) \leq \varepsilon_1$. Privzemimo najprej, da je $\varepsilon_1 = \varepsilon_1^b$. Prostor kriterijev je zdaj razdeljen na dve območji - območje $f_1\left(x\right) \leq \varepsilon_1^b$ in območje $f_1\left(x\right) > \varepsilon_1^b$. Dopustno je samo levo območje, zato iščemo minimum f_2 na tem območju. Optimum je dosežen v točki B. Podobno velja tudi za primer, ko je $\varepsilon_1 = \varepsilon_1^c$. Tokrat je optimalna rešitev točka C. Opazimo lahko, da oblika Pareto optimalne fronte ne vpliva na delovanje metode. Če izberemo $\varepsilon_1 = \varepsilon_1^a$, problem nima dopustnih rešitev. Če pa izberemo $\varepsilon_1 = \varepsilon_1^d$, je dopusten cel prostor in rešitev problema je točka D [3].



Slika 2.6: Metoda ε-omejitev [3].

S postopnim spreminjanjem vrednosti ε_j lahko vzorčimo različne točke na Pareto fronti. Pomembno se je zavedati dejstva, da je potrebna predhodna analiza za iden-

tifikacijo ustreznih izhodiščnih vrednosti ε_j. Ustrezne vrednosti ε_j so običajno pridobljene z enokriterijskim optimiranjem vseh kriterijskih funkcij s pomočjo tehnik matematičnega programiranja [6].

Poglavje 3

Evolucijski algoritmi

Evolucijski algoritem (EA) je generični izraz za skupino stohastičnih optimizacijskih metod, ki temeljijo na populaciji. Le-te so bile navdihnjene po teoriji evolucije z naravnim izborom, ki je vodilni proces za nastanek kompleksnih in dobro prilagojenih organskih struktur. Povedano na kratko in zelo poenostavljeno, je evolucija rezultat medsebojnega ustvarjanja nove genetske informacije ter njenega ocenjevanja in izbora. Na posameznika populacije vplivajo drugi posamezniki populacije (npr. konkurenca za hrano, plenilci in parjenje), kot tudi okolje (npr. oskrba hrane in podnebje). Bolje kot se posameznik obnese v skladu s temi pogoji, večja je možnost za posameznika, da bo živel dlje časa in ustvaril potomce, ki podedujejo (porazdeljeno) genetsko informacijo starša. Tekom evolucije to vodi do penetracije populacije z genetsko informacijo posameznikov, ki imajo nadpovprečno sposobnost (*fitness*) [10].

EA so sestavljeni iz prevladujočih metod, kot so evolucijske strategije (ES), genetski algoritmi (GA), in genetsko programiranje (GP). Prve metode, ki se lahko štejejo kot EA so bile predlagane že leta 1950, vendar bolj pomembeno leto je 1975, ko je Holland objavil svojo priredbo v knjigi Natural and Artificial Systems [11] in ko je De Jong zaključil svojo doktorsko disertacijo [12]. V zadnjih desetletjih so EA postali popularni in pomembni. Psevdokod evolucijskega algoritma (Algoritem 1) je preprost.

Algoritem 1 Evolucijski_algoritem

```
1: t = 0;
2: ustvari populacijo P(t);
3: ovrednoti P(t);
4: repeat
5:     P′(t) = križanje P(t);
6:     P″(t) = mutacija P′(t);
7:     ovrednoti P″(t);
8:     P(t+1) = selekcija P″(t);
9:     t = t + 1;
10: until zaustavitveni_pogoj(t)
```

V prvem koraku se populacija inicializira. Ko ni predhodne informacije, so vrednosti odločitvenih spremenljivk populacije običajno izbrane iz nekega predhodno opredeljenega razpona vrednosti v skladu z enotno naključno porazdelitvijo. Če obstaja nekaj predznanja o lokaciji optimuma, potem je lahko inicializacija na primer izvedena tako, da temelji na Gaussovi porazdelitvi s središčem na domnevni lokaciji optimuma. Po inicializaciji se populacija razvija iterativno; vsaka ponovitev se imenuje generacija. Običajen kriterij za prekinitev iterativnega procesa je neka vnaprej določena zgornja meja za število generacij. Kriterij za prekinitev lahko temelji tudi na razvoju populacije v prejšnjih generacijah ali na nekem znanju o vrednosti optimuma. Prvi korak v iteracijski zanki je izbor članov populacije (pogosto imenovanih starši) za reprodukcijo. Selekcija se izvede na podlagi kakovosti (*fitness*) vsakega člana. Tako mora obstajati način, da se oceni *fitness* vrednost posamezne kombinacije odločitvenih spremenljivk. Običajno imajo boljši člani glede na *fitness* prednost pri selekciji, vendar obstajajo nekateri EA, kot je na primer DE (*Defirential Evolution*), ki izberejo vse člane za reprodukcijo. Reprodukcija je pogosto razdeljena na operatorja rekombinacije (pogosto imenovano tudi križanje) in mutacije. Rekombinacija kombinira dva ali več izbranih članov/staršev za reprodukcijo enega ali več otrok. Ideja rekombinacije je, da bi z združevanjem staršev ustvarili otroka, ki združuje dobre lastnosti od staršev, in je boljši od vseh staršev. Seveda (kakor v naravi), lahko rekombinacija ustvari otroka, ki je slabši od staršev. Mutacija pomeni rahla sprememba enega samega člana, njen namen je ustvariti variabilnost v populaciji in zmanjšati tveganje za stagnacijo. Zadnji korak v iteracijski zanki je selekcija članov za naslednjo generacijo iz članov prejšnje generacije in reproduciranih članov. Kakor prej, ima selekcija po navadi raje boljše člane. Če selekcija zagotovi, da je najboljši doslej najden član izbran za naslednjo generacijo, potem je selekcija elitistična in EA je elitistična optimizacijska metoda. Stopnja, do katere ima selekcija raje boljše člane, je znana

kot selekcijski pritisk in čeprav bi morala selekcija imeti raje boljše člane, selekcijski pritisk ne sme biti previsok. Prevelik selekcijski pritisk lahko privede do izgube raznolikosti v populaciji in prezgodnje konvergence. Prezgodnja konvergenca pomeni, da se je populacija približala sub-optimalni rešitvi in iskanje ne napreduje več. Raznolikost je izgubljena, saj so člani populacije preveč blizu drug drugemu v iskalnem prostoru [5].

3.1 Večkriterijski evolucijski algoritmi

Večkriterijski evolucijski algoritmi so postali priljubljeni na različnih domenah uporabe. Glavni vzrok za to je uspeh teh algoritmov pri reševanju različnih problemov iz realnega sveta. Algoritmi so ustvarili bodisi konkurenčne ali boljše rezultate, kot so jih ustvarile druge metode iskanja. Uporaba je klasificirana v štiri velike skupine: inženiring, industrija, znanost in razno. Uporaba za inženiring je daleč najbolj priljubljena v literaturi, morda zaradi dejstva, da imajo ti problemi dobro raziskane matematične modele. Povečan interes uporabe MOEA v toliko različnih disciplinah je predvsem zaradi večkriterijske narave problemov iz realnega sveta [6, 2].

Poglavje 4

Šahovsko rangiranje

Šah je strateška igra dveh igralcev, ki ima tri možne izide: zmaga, poraz ali neodločeno. Igralci šaha se udeležujejo turnirjev, kjer igrajo več iger, z več nasprotniki. Šahovska zveza spremlja igralce in jih ocenjuje na podlagi izidov posameznih iger. Igralci pridobijo točke z zmagami na organiziranih turnirjih. Na podlagi točk se določi ocena igralca (rang).
Različne šahovske zveze imajo svoja pravila in formule za posodabljanje ranga svojih igralcev. V večini primerov velja, da so igralci z večjim rangom boljši. Prav tako je običajno, da se igralcu rang zmanjša, če izgubi proti igralcu z manjšim rangom in poveča, če zmaga proti igralcu z večjim rangom.
Eden najpogostejših sistemov rangiranja je sistem Elo, ki ga je razvil Arpad Elo. Čeprav je sistem rangiranja Elo znan po svoji preprostosti in široki uporabi, ima nekaj pomanjkljivosti. Iz tega razloga je Glickman v letu 1995 uvedel nov sistem šahovskega rangiranja z imenom sistem Glicko. V letu 2012 je Glickman predstavil izboljšano različico sistema rangiranja, to je sistem Glicko-2 [13].

4.1 Sistem Elo

V sistemu Elo se za vsakega igralca izračuna celoštevilski rang, ki se zmanjšuje ali zvišuje glede na izid igre dveh rangiranih igralcev. Ko dva igralca tekmujeta, sistem rangiranja predvideva, da igralec z višjim rangom zmaga bolj pogosto. Vsaka organizacija, ki je adaptirala sistem Elo, ga je implementirala na svoj način in nobena od njih ne sledi natančno izvorni predlogi sistema.
Za vse implementacije sistema velja, da se rang ne more določiti absolutno, temveč

je odvisen od ranga nasprotnika in števila zmag, porazov in remijev, ki jih je igralec odigral. Med igro dveh igralcev z oceno R_A in R_B se pričakovani rezultat igre za igralca A izračuna po naslednji formuli:

$$E = \frac{1}{1 + 10^{-(R_A - R_B)/400}}, \tag{4.1}$$

Na primer, da ima igralec A rang 1500 in igralec B rang 1700, potem je pričakovani rezultat za A približno 0,24.
Novi rang igralca A se izračuna po enačbi:

$$R'_A = R_A + K(S_A - E_A) \tag{4.2}$$

kjer je R'_A rang igralca po turnirju, R_A je rang igralca pred turnirjem, S_A je celoten rang igralca v turnirju, E_A je celotni pričakovan rezultat, ocenjen iz ranga igralca pred turnirjem in ranga nasprotnika pred turnirjem, K je faktor, ki določa koliko poudarka damo na uspešnost igralca glede na njegov rang pred turnirjem. Za slabše igralce nastavimo število $K = 32$ in $K = 16$ za mojstre.
Sistem Elo ima nekaj pomanjkljivosti, kot so pravilna nastavitev vrednosti števila K, nenatančen distribucijski model ali nezanesljiv rang. Glavni problem pri nezanesljivem rangu je možnost, da igralec, ki zmaga igro izgubi točke, ali izgubi igro, vendar dobi točke. Drugi problem pri nezanesljivem rangu se pojavi v igri, ko imata igralca enak rang, a eden igralec ni igral več let, medtem ko drugi igra neprekinjeno. Zmagovalec bi pridobil enako število točk, kot bi jih poraženec izgubil, kar ni pravilno. Igralec, ki ni igral več let, bi moral imeti bolj nezanesljiv rang in igralec, ki igra neprekinjeno, bi moral imeti bolj zanesljiv rang. Torej, če prvi igralec zmaga, se pričakuje, da se bo njegov rang bolj povišal, kot se bo rang drugega igralca zmanjšal. Poleg tega nam rang igralca ne razkrije njegovega vedenja o igranju ali zanesljivosti moči. [14]

4.2 Sistem Glicko

Zaradi pomanjkljivosti sistema Elo je leta 1995 Mark Glickman [15] predstavil nov sistem šahovskega rangiranja z imenom sistem Glicko. V sistemu Glicko vsak igralec dobi svoj rang R in odklon ranga RD (*Rating Deviaton*). Odklon ranga označuje kako zanesljiv je rang igralca. Če ima igralec majhen RD, je njegov rang zanesljiv in igra precej pogosto. Če je RD vrednost igralca visoka, je njegov rang nezanesljiv.

Ker rang ni odvisen od enega samega števila, se uporablja interval. Tako lahko s 95% verjetnostjo rečemo, da se igralčev rang R nahaja v intervalu $[R - 2RD, R + 2RD]$. Pred začetkom ocenjevalnega obdobja (turnir) za vsakega igralca določimo rang R in odklon ranga RD. Glickman je priporočal, da se rang R nastavi na 1500, odklon ranga RD na 350. Novi odklon ranga se izračuna po enačbi:

$$RD' = min\left(\sqrt{RD^2 + c^2 t}, 350\right), \tag{4.3}$$

kjer t označuje število turnirjev, ki je preteklo odkar je igralec nazadnje igral in c označuje konstanto, ki regulira povečanje nezanesljivosti s časom. Če se je igralec udeležil zadnjega turnirja, potem je t enak 1. Konstanto c določi vsaka organizacija posebej in določa, koliko ocenjevalnih obdobij mora preteči preden postane rang tipičnega igralca tako nezanesljiv kot rang neocenjenega igralca.
Sistem Glicko-2 temelji na sistemu Glicko, vendar uporablja drugačno merilo. Zato je potrebno rang R in odklon ranga RD pretvoriti iz sistema Glicko v Glicko-2. K tem podatkom dodamo še volatilnost σ, ki označuje stopnjo pričakovanega nihanja ranga igralca. Nizka vrednost σ pomeni, da je igralčeva ocena pričakovana in stabilna. Visoka vrednost σ pove, da se je rang igralca nepričakovano zvišal ali zmanjšal, medtem ko je pred tem bil na isti ravni. Sistem Glicko-2 uvaja novo konstantno τ, ki igra podobno vlogo kot t in odraža volatilnost skozi čas. Glickman je priporočal izbiro iz intervala (0,3, 1,2). Kakor pri sistemu Glicko, moramo pred začetkom ocenjevalnega obdobja za vsakega igralca izračunati R in RD. Pretvorjena vrednost za rang R je označena z μ in pretvorjena vrednost za odklon ranga je označena z ϕ. Pretvorba je izvedena po enačbah:

$$\mu = \frac{R - 1500}{173,7178} \; in \; \phi = \frac{RD}{173,7178} \tag{4.4}$$

Če se igralec ni udeležil turnirja, potem se nov odklon izračuna po enačbi:

$$\phi' = \sqrt{\phi^2 + \sigma^2} \tag{4.5}$$

in se nato pretvori nazaj v sistem Glicko:

$$RD' = 173,7178\phi' \tag{4.6}$$

Vsak igralec turnirja igra k iger, eno igro proti vsakemu od k nasprotnikov, katerih rangi so $R_1, R_2, \ldots, R_k$ in odkloni rangov $RD_1, RD_2, \ldots, RD_k$. Doseženi rezultati proti nasprotnikom so označeni s $S_1, S_2, \ldots, S_k$, kjer je S_i enak 1 za zmago, 0 za poraz in 0,5 za izenačenje. Kadar igra igralec proti enemu nasprotniku večkrat, se to obravnava kot več iger proti nasprotnikom, ki imajo enako oceno in odklon. Ocenjena varianca ν igralčevega ranga, ki bazira samo na izidih iger se izračuna po enačbi:

$$\nu = \left(\sum_{i=1}^{k} g\left(\phi_i\right)^2 E\left(\mu, \mu_i, \phi_i\right) \left(1 - E\left(\mu, \mu_i, \phi_i\right)\right) \right)^{-1} \tag{4.7}$$

kjer $g\left(\phi_i\right)$ in $E\left(\mu, \mu_i, \phi_i\right)$ izračunamo po naslednjih enačbah:

$$g\left(\phi_i\right) = \frac{1}{\sqrt{q + 3\phi^2/\pi^2}} \tag{4.8}$$

$$E\left(\mu, \mu_i, \phi_i\right) = \frac{1}{1 + 10^{-g(\phi_i)(\mu - \mu_i)}} \tag{4.9}$$

Izračunu gravitacijskega faktorja $g\left(\phi_i\right)$ in pričakovanega rezultata $E\left(\mu, \mu_i, \phi_i\right)$ sledi izračun izboljšave ranga Δ:

$$\Delta = \nu \sum_{i=1}^{k} g\left(\phi_i\right) \left(S_i - E\left(\mu, \mu_i, \phi_i\right)\right) \tag{4.10}$$

Za izračun nove volatilnosti σ' uporabimo algoritem Illinois, ki je različica metode napačnega pravila ali $regula falsi$. Funkcija s katero to dosežemo je:

$$f\left(x\right) = \frac{e^x \left(\Delta^2 - \phi^2 - \nu - e^x\right)}{2\left(\phi^2 + \nu + e^x\right)^2} - \frac{x - a}{\tau^2}, \tag{4.11}$$

kjer je $a = ln(\sigma^2)$ in natančnost je $\varepsilon = 0,000001$. Nato izračunamo vmesno vrednost odklona:

$$\phi^* = \sqrt{\phi^2 + \sigma^2} \tag{4.12}$$

Posodobimo odklon ranga ϕ' in rang μ':

$$\phi' = \frac{1}{\sqrt{\frac{1}{\phi^{*2}} + \frac{1}{\nu}}} \tag{4.13}$$

$$\mu' = \mu + \phi'^2 \sum_{i=1}^{k} g(\phi_i)(S_i - E(\mu, \mu_i, \phi_i)) \tag{4.14}$$

Na koncu dobljeni vrednosti pretvorimo iz sistema Glicko-2 v sistem Glicko:

$$R' = 173,7178\mu' + 1500 \; in \; RD' = 173,7178\phi' \tag{4.15}$$

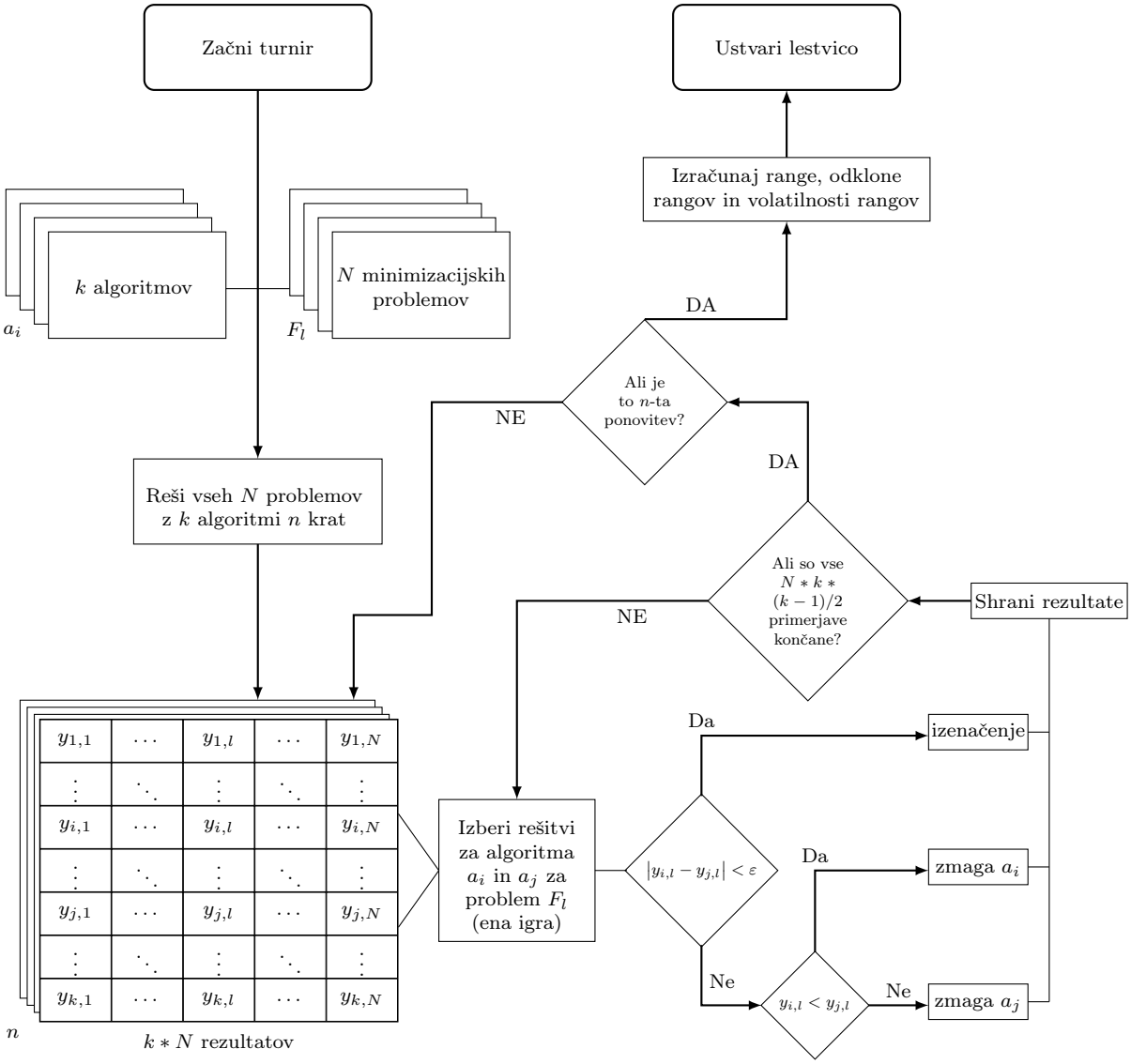
[13]

4.3 Orodje EARS

EARS (*Evolutionary Algorithms Rating System*) je odprtokodno orodje [16], razvito v programskem jeziku Java. Cilj orodja je zagotavljati pošteno in enostavno primerjavo med evolucijskimi algoritmi. EARS prejeme kot vhod testne probleme in evolucijske algoritme, ki vključujejo druge parametre, potrebne za natančen opis problemov in algoritmov. Merilo uspešnosti (*benchmark*) je opredeljeno z nizom optimizacijskih problemov / funkcij, ki so implementirane v Javi in s številom ovrednotenj *fitness* ocene, ki služi kot zaustavitveni pogoj vseh sodelujočih algoritmov. Algoritem mora biti implementiran tako, da deduje iz abstraktnega razreda *Algorithm* in mora povoziti metodo $run(BenchmarkProblem)$. Parametre turnirja (število neodvisnih zagonov n in mejo izenačenja ε) je treba določiti, ko se turnir začne. Za vsak algoritem shranimo rezultate in nato izračunamo statistiko in range na podlagi šahovskega sistema Glicko-2. Na koncu turnirja se prikaže lestvica rangov in različna statistična poročila. Algoritmi se lahko primerjajo samo v *benchmarku*, ki je ustvarjen v EARS, in morajo tekmovati z vsemi drugi tekmovalci *benchmarka* na vseh optimizacijskih problemih znotraj *benchmarka*.

Na sliki 4.1 je prikazan diagram izvedbe eksperimenta v orodju EARS. Eksperiment se izvede v obliki turnirja, ki je sestavljen iz k algoritmov $a_1, a_2, \ldots, a_k$, N optimizacijskih problemov in je izveden v n neodvisnih zagonih. Vsak algoritem vrne najboljšo rešitev za vsak optimizacijski problem v vsaki ponovitvi ($K * N * n$ rešitev). Te re-

šitve se nato primerjajo med seboj. Sklop $\{a_i, a_j\}_{l,m}$ je mišljen kot ena primerjava ali ena igra algoritmov a_i in a_j na optimizacijskem problemu F_l pri zagonu m, kjer $i, j \in \{1, \ldots, k\}, i \neq j, l \in \{1, \ldots, N\}$ in $m \in \{1, \ldots, n\}$. Rešitvi y_i in y_j od algoritmov a_i in a_j za isti problem F_l pri zagonu m se primerjata in tisti algoritem, katerega rešitev je najbližja optimumu problema F_l, zmaga. Če je razlika med rešitvama manjša od vnaprej določene meje ε, je rezultat igre izenačen. Turnir je torej sestavljen iz $(k * (k-1)/2) * N * n$ iger. Ko je turnir zaključen, se rezultati zberejo v obliki zmag, porazov in izenačenj. Posodobijo se rangi, odkloni rangov in volatilnosti rangov, kakor predlaga sistem Glicko-2. Vsi zbrani podatki se shranijo za naslednji turnir in so prikazani na lestvici [13].



Slika 4.1: Diagram izvedbe eksperimenta v orodju EARS [13].

4.4 Razširitve

Za vključitev komponent, potrebnih za večkriterijsko optimiranje, smo morali razširiti obstoječe razrede v orodju EARS. Tako so nove komponente kompatibilne s sistemom rangiranja algoritmov. Z obstoječim sistemom bomo lahko rangirali vse algoritme za večkriterijsko optimiranje. Kakor pri enokriterijski optimizaciji, bodo novi algoritmi imeli omejeno število ovrednotenj. Vsa koda, ki smo jo vključili in prilagodili okolju EARS, izhaja iz ogrodja jMetal [17].

4.5 Algoritmi

Vsak algoritem razširi razred *Algorithm*. Glavna metoda tega razreda je *run*, ki prejme razred *Task* in po izvajanju vrne najboljšega posameznika v populaciji. Algoritmi imajo nastavljeno privzeto velikost populacije. Spremenimo jo tako, da damo želeno vrednost v konstruktor. Razred *Individual* smo razširili dvakrat. Prvi razred, ki razširi *Individual*, je *MOIndividual* in predstavlja eno rešitev v prostoru kriterijev. Razred ima dodatne spremenljivke, ki jih potrebujejo nekateri algoritmi za delovanje. Ker je prostor kriterijev večdimenzionalen, je v razširjenem razredu spremenljivka *eval* polje. Drugi razred, ki razširi *Individual*, je *MOParetoIndividual*. *MOParetoIndividual* ima seznam *MOIndividual* in predstavlja eno rešitev algoritma. Seznam razreda je torej aproksimacija Pareto fronte. Nadgradili smo tudi razred *Task*. *Task* je neke vrste vmesnik med algoritmom in problemom. Hrani *Problem* in preverja število ovrednotenj. Če algoritem prekorači maksimalno število, se izvajanje zaključi. V razred smo dodali dve novi metodi, prilagojeni za večkriterisjko optimiranje. Prva je *eval*, ki ovrednoti posameznika *MOIndividual*, druga pa vrne na novo zgeneriranega posameznika in se imenuje *getRandomMOIndividual*.

4.5.1 NSGA-II

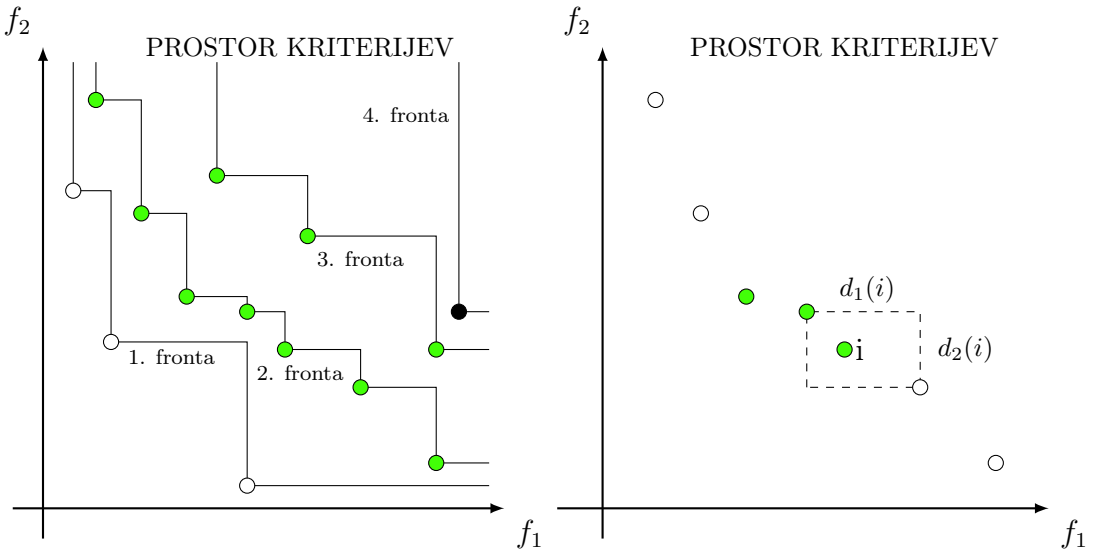
NSGA-II (*Elitist Non-Dominated Sorting Genetic Algorithm*) je popularen genetski algoritem za večkriterijsko optimiranje Deba in sodelavcev iz leta 2000 [18], ki se je v tistem času odrezal najbolje v primerjalnih študijah od vseh evolucijskih algoritmov za večkriterijsko optimiranje. Algoritem je nadgradnja predhodnika NSGA [19], ki je prvi uporabljal nedominirano sortiranje (angl. non-dominated sorting) in metriko nakopičenosti (*crowding distance metric*, CD) za izločanje manj obetavnejših rešitev.

Psevdokod algoritma je prikazan na sliki 2. NSGA-II deluje podobno kot enostavni genetski algoritem za enokriterijsko optimiranje. Prilagoditev za večkriterijsko optimiranje je vidna le pri selekciji. Selekcija in rekombinacija (križanje in mutacija) predstavljata glavni korak algoritma (12. točka na sliki 2). Najprej združimo staro populacijo staršev in potomcev ($P_t - 1\ in\ Q_t - 1$) v skupno populacijo ($R_t - 1$) z dvakratno velikostjo pop_size. V naslednjem koraku izvedemo nedominirano sortiranje, s katerim osebke sortiramo po frontah. V novo populacijo staršev P_t vstavljamo fronte od najboljše do najslabše. Ko je nova populacija tako polna, da vanjo ne moremo vstaviti celotne fronte, jo nadalje sortiramo z metriko nakopičenosti. Tako zapolnimo novo populacijo z najmanj nakopičenimi osebki. Iz dobljene populacije staršev P_t generiramo novo populacijo potomcev z uporabo turnirske selekcije, križanja in mutacije. Turnirska selekcija deluje tako, da izmed dveh naključnih staršev izbere tistega, ki je boljši, glede na nedominirano sortiranje. Če sta starša izenačena (ležita na isti fronti), v turnirju zmaga tisti, ki ima boljšo vrednost metrike nakopičenosti. Po selekciji izbrane starše še križamo in mutiramo ter ponavljamo, dokler ne zapolnimo nove populacije potomcev Q_t. Vse osebke iz populacije Q_t še ovrednotimo.

Algoritem 2 NSGA-II [3]

input: $pop\_size = 100$;

1: naključno generiraj in ovrednoti začetno populacijo staršev P_0;
2: pripravi prazno začetno populacijo potomcev Q_0;
3: t = 0;
4: **repeat**
5: t = t + 1;
6: združi stari populaciji staršev in potomcev: $R_{t-1} = P_{t-1} \cup Q_{t-1}$;
7: Izvedi nedominirano sortiranje na R_{t-1} in določi fronte $F_i, i = 1, 2, \ldots$;
8: pripravi novo prazno populacijo staršev P_t;
9: v populacijo P_t daj prvih i front, ki še gredo cele v njo;
10: fronto F_{i+1}, ki ne gre več cela v populacijo P_t sortiraj z uporabo metrike nakopičenosti;
11: populacijo P_t dopolni z osebki iz F_{i+1}, ki so najmanj nakopičeni;
12: populacijo potomcev Q_t generiraj iz populacije staršev P_t z uporabo turnirske selekcije, križanja in mutacije;
13: ovrednosti osebke iz populacije potomcev Q_t;
14: **until** zaustavitveni_pogoj(t)



Slika 4.2: Nedominirano sortiranje po frontah in računanje metrike nakopičenosti [3].

Na sliki 4.2 je levo prikazana populacija s sedmimi osebki (združena populacija staršev in potomcev, zato vsebuje štirinajst osebkov), ki je nedominirano sortirana po frontah. Ker ima prva fronta tri osebke, jo lahko v celoti vstavimo v novo populacijo. Druge fronte ne moremo vstaviti v celoti, zato moramo iz nje izbrati štiri najboljše osebke glede na metriko nakopičenosti (slika 4.2 desno). Metrika nakopičenosti najbolje oceni skrajne osebke v fronti (da imajo rešitve čim večji razpon). Ostale osebke v fronti ocenimo z njihovo razdaljo do najbližjih sosedov. Če optimiramo k kriterijev, potem za vsak kriterij $j = 1, \ldots, k$ osebke najprej sortiramo po naraščajočih vrednostih f_j. V naslednjem koraku za vsak vmesni osebek i izračunamo razdaljo $d_j(i)$ med najbližjima sosednjima osebkoma p in q (za katera je $f_j(p) \leq f_j(i) \leq f_j(q)$) kot:

$$d_j(i) = \frac{f_j(q) - f_j(p)}{f_j^{max} - f_j^{min}}, \tag{4.16}$$

kjer sta f_j^{max} in f_j^{min} maksimalna in minimalna vrednost j-tega kriterija. Skrajnima dvema osebkoma (glede na kriterij j) določimo najvišjo možno razdaljo. Metrika nakopičenosti za osebek i je vsota teh razdalj po vseh kriterijih:

$$c(i) = \sum_{j=1}^{k} d_j(i) \tag{4.17}$$

Na sliki 4.2 (desno) je metrika nakopičenosti za osebek i enaka polovici obsega pravokotnika, ki ga določata sosednja osebka. V novo populacijo gredo vsi osebki, ki so na

sliki označeni z belo barvo (trije osebki s prve fronte in štirje osebki z druge fronte, ki imajo največjo vrednost metrike nakopičenosti) [3].

4.5.2 SPEA2

SPEA2 (*Strength Pareto Evolutionary Algorithm*) Zitzlerja in sodelavcev iz leta 2001 [20] je, kakor NSGA-II, popularen algoritem za večkriterijsko optimiranje. SPEA2 je primerljiv pri doseganju prvega cilja večkriterijskega optimiranja (konvergenca k Pareto optimalni fronti) in boljši pri doseganju drugega cilja (enakomerna razporejenost rešitev). Algoritem uporablja osnovno populacijo (namesto populacije staršev) in arhiv konstantne velikosti, v katerega shranjuje najboljše dobljene rešitve. Osebke ovrednoti z merjenjem moči (*strength*), grobe uspešnosti (*raw fitness*) in gostote (*density*) in ne s frontami, kakor NSGA-II. Moč osebka je vsota osebkov iz populacije in arhiva, ki jih dominira (slika 4.3 levo). Groba uspešnost osebka je vsota moči vseh osebkov iz populacije in arhiva, ki ta osebek dominirajo (slika 4.3 desno). Ker imajo nedominirani osebki grobo uspešnost enako in jih je skozi proces evolucije zmeraj več, je potrebna še dodatna informacija o razporejenosti rešitev. Zato za vse osebke i z enako grobo uspešnostjo $R(i)$ izračunamo še gostoto $D(i)$:

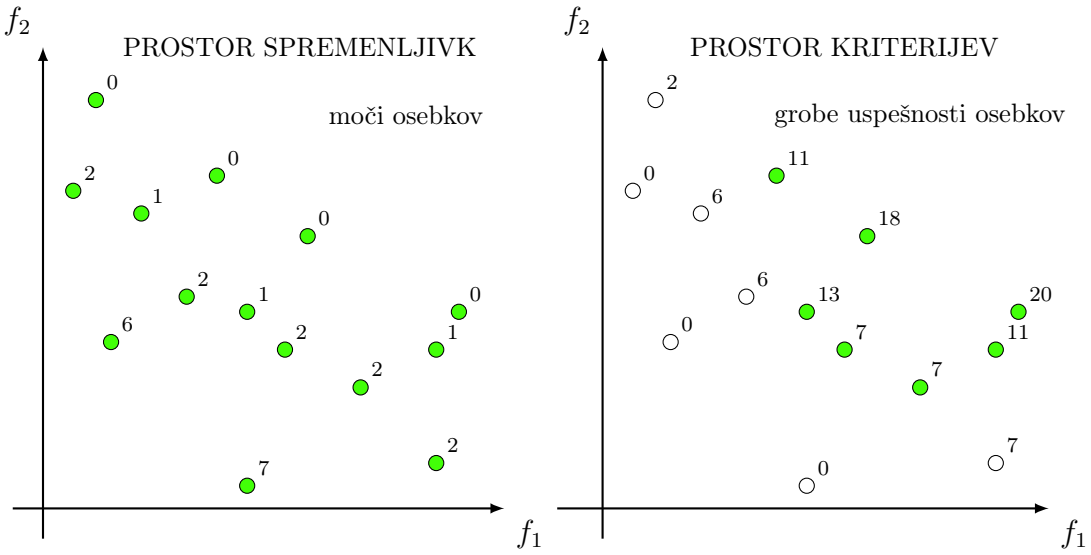
$$D\left(i\right) = \frac{1}{\sigma_i^k + 2}, \tag{4.18}$$

kjer je σ_i^k razdalja do k-tega osebku najbližjega soseda (k je konstanta, odvisna od velikosti populacije in arhiva). Uspešnost osebka $F(i)$ izračunamo kot vsoto grobe uspešnosti in gostote:

$$F\left(i\right) = R\left(i\right) + D\left(i\right). \tag{4.19}$$

Čeprav je F imenovana uspešnost, je to količina, ki jo želimo minimizirati.
Na sliki 4.3 desno je primer, kjer želimo med vsemi izbrati sedem (velikost arhiva) najboljših osebkov (označeni z belo barvo). Najboljših šest izberemo tako, da vzamemo tiste, ki imajo najmanjšo grobo uspešnost. Pri izbiri sedmega osebka se odločamo med tremi osebki, ki imajo enako uspešnost in sicer 7. Med temi vzamemo tistega, ki ima najmanjšo gostoto.
Glavni del algoritma SPEA2 je prikazan na sliki 3 pod točko 4. Najprej nedominirane osebke iz stare populacije in arhiva prepišemo v nov arhiv A_t. Če je število nedominiranih osebkov preveliko, odstranimo tiste, ki so blizu drugim osebkom. Če arhiv ni

poln, ga dopolnimo z najboljšimi osebki iz populacije in starega arhiva. Dokler zaustavitveni kriterij ni izpolnjen, generiramo novo populacijo P_t iz arhiva A_t z uporabo turnirske selekcije, križanja in mutacije (podobno kot pri NSGA-II). Na koncu vse osebke iz populacije in arhiva ovrednotimo glede na moč, grobo uspešnost in gostoto [3].



Slika 4.3: Računanje moči in grobih uspešnosti osebkov v algoritmu SPEA2 (vrednosti moči in grobe uspešnosti lahko vidimo zgoraj desno poleg osebka) [3].

Algoritem 3 SPEA2 [3]

input: $pop\_size = 100$;
$archive\_size = 100$;

1: naključno generiraj in ovrednoti začetno populacijo staršev P_0;
2: pripravi prazen začetni arhiv A_0;
3: t = 0;
4: **while** true **do**
5: t = t + 1;
6: v nov arhiv A_t prepiši vse nedominirane osebke iz P_{t-1} in A_{t-1};
7: če je nedominiranih osebkov več kot je velikost novega arhiva, izloči osebke, ki so blizu drugim osebkom;
8: sicer pa, če nedominirani osebki ne zapolnijo arhiva, arhiv dopolni z najboljšimi dominirani osebki iz P_{t-1} in A_{t-1};
9: **if** zaustavitveni kriterij izpolnjen **then** končaj
10: **else** populacijo P_t generiraj iz arhiva A_t z uporabo turnirske selekcije, križanja in mutacije;
11: **end if**
12: ovrednoti vse osebke iz populacije P_t in arhiva A_t;
13: **end while**

4.5.3 PAES

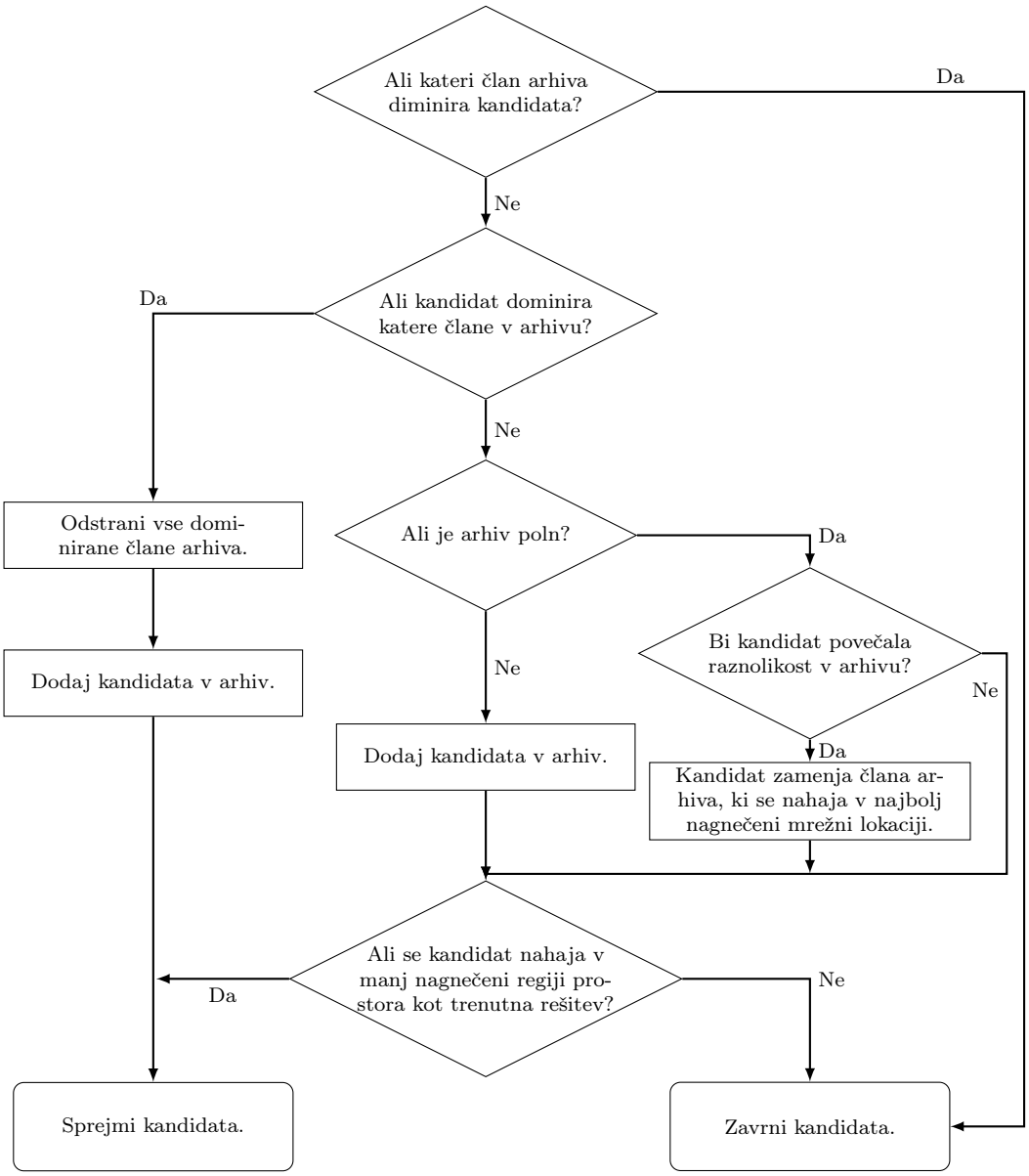
Algoritem PAES (*Pareto Archived Evolution Strategy*) Knowlesa in Corneta iz leta 1999 [21] je bil razvit z dvema ciljema. Prvi je, da mora biti algoritem strogo omejen na lokalno iskanje. Tako uporablja mutacijo (edini operator) za majhne spremembe in se premika od sedanje rešitve do bližnjega soseda. Tako se precej razlikuje od ostalih popularnih algoritmov za večkriterijsko optimiranje. Drugi cilj je, da mora algoritem vse nedominirane rešitve obravnavati kot, da imajo enako vrednost. PAES ohranja arhiv predhodno odkritih nedominiranih rešitev. Ta arhiv se nato uporablja za oceno dejanske dominiranosti dveh rešitev.

Struktura algoritma PAES je prikazana na psevdokodi 4. Algoritem prične z inicializacijo enega samega kromosoma (trenutna rešitev), ki je nato ocenjen z večkriterijsko funkcijo. Nato se ustvari kopija, nad katero je izvedena mutacija. Ta mutirana kopija je ovrednotena in predstavlja novega kandidata za rešitev. Kandidata in trenutno rešitev je treba nato primerjati. Izbrana je tista rešitev, ki dominira drugo. V primeru, da nobena rešitev ne dominira, se kandidat primerja z referenčno populacijo že arhiviranih nedominiranih rešitev. Če primerjava z arhivom ni uspešna, ima prednost rešitev, ki je v najmanj natrpani regiji prostora.

Algoritem 4 PAES [21]

input: $archive\_size = 100$;
$bisections = 5$;
generiraj naključno trenutno rešitve C;
oceni C in dodaj v arhiv A;
t = 0;
repeat
    mutiraj C, da dobiš novega kandidata C';
    ovrednoti C';
    if $C \succ C'$ **then**
        zavrni C';
    else if $C' \succ C$ **then**
        zamenjaj C z C' in dodaj C' v A;
    else if $\exists_{C'' \in A} (C'' \succ C')$ **then**
        zavrni C';
    else
        izvedi test(C,C',A), da določiš trenutno rešitev in ali dodaš C' v A
    end if
    t = t + 1;
until zaustavitveni_pogoj(t)

Arhiv služi dvema ločenima namenoma. Prvič, shranjuje in posodablja vse ustvarjene nendominirane rešitve (glede na merilo raznolikosti). Drugič, med izvajanjem se uporablja kot pripomoček za natančno izbiro med kandidatom in sedanjim vektorjem rešitev, tako da deluje kot približek trenutni nedominirani fronti. Slednje tudi zagotavlja selekcijski pritisk, ki neprestano usmerja proces k boljšim rešitvam.
Proces arhiviranja je prikazan na sliki 4.4. Vsak ustvarjen kandidat rešitve, ki ni dominiran od starša (trenutna rešitev), se primerja z vsakim članom niza primerjav. Kandidati, ki dominirajo niz primerjav, so zmeraj sprejeti in arhivirani. Kandidati, ki jih dominira niz primerjav, so zmeraj zavrnjeni, medtem ko tisti, ki niso dominirani so sprejeti in/ali arhivirani, glede na stopnjo gneče v mrežni lokaciji.
Za spremljanje stopnje gneče v različnih regijah prostora rešitev se uporablja d-dimenzionalna mreža, kjer je d število kriterijev problema. Za vsako generirano rešitev je mrežna lokacija določena z uporabo rekurzivne porazdelitve in zabeležena z drevesnim kodiranjem. Ohranja se tudi zemljevid omrežja, ki za vsako omrežno lokacijo označuje, koliko in katere rešitve v arhivu se tam nahajajo. Ko se kandidat rešitve pridruži arhivu, zamenja eno arhivirano rešitev z največjim številom lokacij mreže, kolikor je lastno število lokacij mreže manjše. Ta pristop se uporablja tudi za izbiro med trenutnimi in kandidati rešitev, ko kandidat ni dominiran in ne dominira nobenega člana v arhivu. V tem primeru je izbrana rešitev z manjšim številom lokacij mreže [21].



Slika 4.4: Logika arhiviranja in sprejemanja [21]

4.5.4 PESA-II

Algoritem PESA-II (*Pareto Envelope-based Selection Algorithm*) Corneta in sodelavcev iz leta 2001 [22] ima poleg standardnih parametrov, kot sta verjetnost za križanje in mutacijo, še dva parametra za velikost populacije in enega za strategijo izrivanja hiper mreže (*Hyper-grid crowding strategy*). Visokonivojski opis algoritma PESA-II je prikazan na sliki 5. Parametra za populacijo sta P_i (velikost notranje populacije,

IP) in P_E (največja velikost arhiva ali zunanje populacije, EP).

Algoritem 5 PESA-II [23]

input: $pop\_size = 100$;
$archive\_size = 100$;
$bisections = 5$;
$CR = 0.9$;
$F = 1$;
$DI = 20$; \\ distribucijski indeks
1: naključno generiraj IP velikosti P_i;
2: ovrednoti vsaki kromosom začetne "notranje" populacije (IP);
3: inicializiraj "zunanjo" populacijo (EP) velikosti P_e na prazni niz;
4: **repeat**
5: vključi nedominirane člane IP v EP;
6: odstrani trenutno vsebino IP;
7: **repeat**
8: z verjetnostjo PC izberi dva starša iz EP;
9: ustvari enega otroka s križanjem;
10: mutiraj otroka ustvarjenega v prejšnjem koraku;
11: z verjetnostjo $(1 - PC)$ izberi enega starša;
12: mutiraj izbranega starša, da ustvariš otroka;
13: **until** IP ni poln
14: **until** zaustavitveni_pogoj
15: **return** EP

Pri koraku "vključitev v arhiv" (korak 5), se trenutni nabor novih rešitev IP vključi v arhiv eden po eden. Kandidat lahko vstopi v arhiv, če je nedominiran znotraj IP, in če ni dominiran od katerega koli obstoječega člana arhiva. Ko je kandidat vnešen v arhiv, bodo člani arhiva, ki jih dominira (če obstajajo), odstranjeni. Če zaradi dodanega kandidata postane arhiv preveč poln (njegova velikost začasno postane $P_e + 1$), potem je odstranjen trenutni član EP (kakor pri algoritmu PAES). Selekcija uporabljena v koraku 13 je selekcija, temelječa na regiji (*Region based selection*). V izboru, ki temelji na regiji, je enota izbora hiperkocka in ne posameznik. Selektivna uspešnost je izpeljana za hiperkocko. Hiperkocka je izbrana z uporabo standardne metode za selekcijo in posameznik, ki bo šel skozi genetske operatorje, je naključno izbran iz te hiperkocke [22].

4.5.5 MOEA/D

Algoritem MOEA/D (*Multi-objective Evolutionary Algorithm based on decomposition*) Zhanga in sodelavcev iz leta 2009 [24] temelji na dekompoziciji. Algoritem razdeli MOP v številne skalarne optimizacijske podprobleme in jih optimizira istočasno. Vsak podproblem je optimiziran samo z uporabo informacije od svojih številnih sosednjih podproblemov. Obstaja več pristopov za pretvorbo problema aproksimacije Pareto fronte v številne skalarne optimizacijske probleme, kot so pristop utežene vsote, pristop Tchebycheff in pristop presek mej (*Boundary Intersection - BI*). Algoritem, ki so ga uporabili na tekmovanju CEC 09 uporablja pristop Tchebycheff.

Pristop Tchebycheff

Pri tem pristopu so skalarni optimizacijski problemi v obliki:

$$min\ g(x \mid \lambda, z^*) = max\ \{\lambda_i \mid f_i(x) - z_i^*\ |\} \tag{4.20}$$

kjer $i \in \{1, \ldots, m\}$, m je število kriterijev, $z^* = (z_1^*, \ldots, z_m^*)^T$ referenčna točka, npr. $z_i^* = min\ \{f_i(x)\}$. Za vsako Pareto optimalno točko x^* obstaja utežen vektor λ tako, da je x^* optimalna rešitev enačbe 4.20 in vsaka optimalna rešitev enačbe 4.20 je Pareto optimalna rešitev. Zato je možno pridobiti različne Pareto optimalne rešitve z reševanjem niza enokriterijskih optimizacijskih problemov, opredeljenih s pristopom Tchebycheff z različnimi vektorji uteži [25].

MOEA/D z dinamičnim dodeljevanjem virov

Naj bo $\lambda^1, \ldots, \lambda^N$ niz enakomerno razporejenih vektorjev uteži in z^* referenčna točka. Problem aproksimacije Pareto Fronte je možno razčleniti na N skalarnih optimizacijskih podproblemov, kriterijska funkcija j-tega podproblema je:

$$g(x \mid \lambda^j, z^*) = max\{\lambda_i^j \mid f_i(x) - z_i^*\ |\} \tag{4.21}$$

kjer je $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$, $j = \{1, \ldots, N\}$ vektor uteži t.j. $\lambda_i^j \geq 0$ za vsak $i = \{1, \ldots, m\}$ in $\sum_{i=1}^m \lambda_i^j = 1$. MOEA/D minimizira vse N kriterijske funkcije istočasno v enem zagonu. Sosedske relacije med temi enokriterijskimi podproblemi so definirane

na osnovi razdalje med njihovimi vektorji uteži. Vsak podproblem je optimiziran z uporabo informacije, predvsem iz sosednjih podproblemov. MOEA/D z dinamičnim dodeljevanjem virov (MOEA/D-DRA) v tej različici opredeli in izračuna koristnost π_i za vsak podproblem i. Računski trud je porazdeljen na te podprobleme glede na njihovo korist.
Med iskanjem MOEA/D-DRA s pristopom Tchebycheff ohranja:

- populacijo N točk $x^1, \ldots, x^N$, kjer je x^i trenutna rešitev za i-ti podproblem;
- $FV^1, \ldots, FV^N$, kjer je FV^i F-vrednost od x^i, npr. $FV^i = F(x^i)$ za vsak $i = 1, \ldots, N$;
- $z = (z_1, \ldots, z_m)^T$, kjer je z_i najboljša (najnižja) vrednost za kriterij f_i do sedaj;
- $\pi_1, \ldots, \pi_N$: kjer je π^i korist podproblema i.
- gen : trenutna številka generacije.

Algoritem deluje na naslednji način:
Vhod:

- MOP;
- zaustavitveni pogoj;
- N: število podproblemov, obravnavanih v MOEA/D;
- enakomerna porazdelitev N utežnih vektorejv: $\lambda^1, \ldots, \lambda^N$;
- T: število utežnih vektorjev v soseščini vsakega utežnega vektorja.

Izhod: $\{x^1, \ldots, x^N\}$ in $\{F(x^1), \ldots, F(x^N)\}$

Algoritem 6 MOEA/D [24]

input: $pop\_size = 300$;
$T = 20$; \\ velikost soseščine
$delta = 0.9$; \\ verjetnost, da bo starš izbran iz soseščine
$nr = 2$; \\ maksimalno število posodobljenih rešitev

izračunaj evklidsko razdaljo med katerima koli dvema vektorjema uteži in nato poišči T najbližjih vektorjev uteži za vsak vektor uteži.
za vsak $i = 1, \ldots, N$, nastavi $B(i) = \{i_1, \ldots, i_T\}$, kjer so $\lambda^{i_1}, \ldots, \lambda^{i_T}$ T najbljižjih utežnih vektorjev do λ^i;
generiraj začetno populacijo $x^1, \ldots, x^N$;
inicializiraj $z = (z_1, \ldots, z_m)^T$ tako, da nastaviš $z_i = min\{f_i(x^1), f_i(x^2), \ldots, f_i(x^N)\}$;
$gen = 0$;
$\pi^i = 1$ za vse $i = 1, \ldots, N$;
repeat
▷ **Izbira podproblemov za iskanje**
    ustvari začetno I z indeksi podproblemov, katerih kriteriji so MOP posamezni cilji f_i;
    Z uporabo 10-turnirske selekcije, ki temelji na π^i, izberi ostalih $\left[\frac{N}{5}\right] - m$ indeksov in jih dodaj v I;
    for each $i \in I$ **do**
        Nastavi

$$P = \begin{cases} B(i) & \text{če } rand < \delta \\ \{1, \ldots, N\} & \text{če } rand \geq \delta \end{cases}$$

        ▷ **Reprodukcija**
        generiraj rešitev $\overline{y}$ iz x^{r_1}, x^{r_2} in x^{r_3} z DE operatorjem ter izvedi mutacijo na $\overline{y}$, da dobiš novo rešitev y;
        ▷ **Popravi**
        če je element y izven meje od Ω, je njegova vrednost obnovljena tako, da je nastavljena na vrednost znotraj meja;
        ▷ **Posodobitev** z
        za vsak $j = 1, \ldots, m$, če je $z_j > f_j(y)$, potem nastavi $z_j = f_j(y)$;
        ▷ **Posodobitev rešitev**
        $c = 0$
        while c != n_r && P ni prazen **do**
            izberi naključni indeks j iz P;
            if $g(y \mid \lambda^j, z) \leq g(x^j \mid \lambda^j, z)$ **then**
                $x^j = y$;
                $FV^j = F(y)$;
                $c = c + 1$;
            end if
            odstrani j iz P;
        end while
    end for

26: $gen = gen + 1$;
27: **if** gen večkratnik števila 50 **then**
28: izračunaj Δ^i, relativno zmanjšanje kriterija za vsak podproblem i v zadnjih 50-ih generacijah, posodobi

$$\pi^i = \begin{cases} 1 & \text{če } \Delta^i > 0,001; \\ (0,95 + 0,05\frac{\Delta^i}{0,001})\pi^i & \text{drugače.} \end{cases}$$

29: **end if**
30: **until** zaustavitveni_pogoj

Vsi koraki algoritma so prikazani na psevdokodi 6. V 10-turnirski selekciji (korak 9), je izbran indeks z največjo vrednostjo π^i izmed 10 enakomerno naključno izbranih indeksov in dodan v I. Selekcija se ponovi $\left[\frac{N}{5}\right] - m$ krat.
V koraku 29, je relativno zmanjšanje definirano kot:

$$\frac{\text{stara vrednost funkcije - nova vrednost funkcije}}{\text{stara vrednost funkcije}}$$

Če je Δ^i manjši od 0,001, se vrednost π^i zmanjša.
V DE operatorju, uporabljenem v koraku 12, je vsak element $\overline{y}_k$ v $\overline{y} = (\overline{y}_1, \ldots, \overline{y}_n)^T$ generiran na sledeči način:

$$\overline{y}_k = \begin{cases} x_k^{r_1} + F \times (x_k^{r_2} - x_k^{r_3}) \text{ z verjetnostjo CR} \\ x_k^{r_1}, \qquad \text{z verjetnostjo 1 - CR} \end{cases} \tag{4.22}$$

kjer sta CR in F kontrolna parametra.
Operator mutacije v koraku 12 generira $y = (y_1, \ldots, y_n^T)$ iz $\overline{y}$ na sledeči način:

$$y_k = \begin{cases} \overline{y}_k + \sigma_k \times (b_k - a_k), \text{ z verjetnostjo } p_m, \\ \overline{y}_k, \qquad \text{z verjetnostjo } 1 - p_m, \end{cases} \tag{4.23}$$

z

$$\sigma_k = \begin{cases} (2 \times rand)^{\frac{1}{\eta+1}} - 1 & \text{če } rand < 0,5, \\ 1 - (2 - 2 \times rand)^{\frac{1}{\eta+1}} & \text{če } rand \geq 0,5 \end{cases}$$

kjer je $rand$ enakomerno naključno število iz itervala [0,1]. Indeks distribucije η in mutacija p_m sta kontrolna parametra a_k in b_k sta zgornja in spodnja meja k-te odločilne spremenljivke [24].

4.5.6 GDE3

Algoritem GDE3 (*Generalized Differential Evolution*) Kukkonena in sodelavcev iz leta 2005 [26] je razširitev DE (*Differential Evolution*) za globalno optimiranje s poljubnim številom kriterijev in omejitev. Prva različica GDE je razširjena DE za omejeno večkriterijsko optimiranje in ima spremenjeno samo pravilo selekcije osnovne DE. Osnovna ideja pravila selekcije v GDE je, da je preizkusni vektor izbran za zamenjavo starega vektorja v naslednji generaciji, če šibko dominira stari vektor. Druga različica, GDE2, je uvedla operacijo za ohranitev raznolikost v GDE. Ponovno je bilo spremenjeno samo pravilo selekcije osnovne DE. Selekcija se izvaja na podlagi izrivanja (*crowding*) v prostoru kriterijev, ko sta preizkusni in stari vektor dopustna in neprimerljiva gleda na Pareto dominanco [1]. Implementacija je bila izvedena z uporabo razdalje izrivanja algoritma NSGA-II. Izrivanje je izmerjeno med celotno populacijo. Tretja različica je GDE3, ki ima poleg modificirane selekcije še drugi del osnovne DE: sedaj se shranita oba vektorja, če primerjamo izvedljive in neprimerljive rešitve. Zato je lahko ob koncu generacije velikost populacije večja, kot je bila prvotno. Če je temu tako, se populacija nato zmanjša nazaj na prvotno velikost s selekcijo, ki temelji na podobnem pristopu, kot je uporabljen v NSGA-II. Člani populacije so razvrščeni glede na cilje za kasnejšo optimizacijo. Najslabši člani populacije so odstranjeni glede na ne-dominanco in izrivanje, da se velikost populacije zmanjša na originalno velikost. Ne-dominanca je primarni, in izrivanje sekundarni kriterij za razvrščanje (kakor v NSGA-II). Celoten algoritem GDE3 je prikazan na psevdokodi 7. Deli, ki so novi v primerjavi s prejšnjimi različicami GDE, so v okvirjih [26].

[1] Rešitvi x_1 in x_2 sta neprimerljivi, če x_1 ne šibko dominira x_2 in obratno.

Algoritem 7 GDE3 [26]

input: $pop\_size = 100$;
$CR = 0.5$;
$F = 0.5$;

naključno generiraj in ovrednoti začetno populacijo;
G = 0;
m = 0;
repeat
  for $i = 0, NP$ **do**
    naključno izberi r_1, r_2, r_3, ki so različni med seboj in od i;
    for $j = 0, D$ **do**
      $j_{rand} \in \{1, 2, \ldots, D\}$;
      if $j_{rand} < CR \,||\, j == j_{rand}$ **then**
        $u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G})$;
      else
        $u_{j,i,G} = x_{j,i,G}$;
      end if
    end for
    if $\vec{u}_{i,G} \preceq_c \vec{x}_{i,G}$ **then**
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
    end if
    if $\forall j : g_j(\vec{u}_{i,G}) \leq 0$ && $\vec{x}_{i,G+1} == \vec{x}_{i,G}$ && $\vec{x}_{i,G} \not\prec \vec{u}_{i,G}$ **then**
      m = m + 1;
      $\vec{x}_{NP+m,G+1} = \vec{u}_{i,G}$
    end if
  end for
  while m > 0 **do**
    Izberi $\vec{x} \in \{\vec{x}_{1,G+1}, \vec{x}_{2,G+1}, \ldots, \vec{x}_{NP+m,G+1}\}$:

$$\begin{cases} \forall i \ \vec{x} \not\prec_c \vec{x}_{i,G+1} \\ \wedge \\ \forall (\vec{x}_{i,G+1} : \vec{x}_{i,G+1} \not\prec_c \vec{x}) \ \ CD(\vec{x}) \leq CD(\vec{x}_{i,G+1}) \end{cases}$$

    Odstrani $\vec{x}$
    m = m - 1;
  end while
  G = G + 1;
until $G < G_{max}$

4.6 Testni problemi

Zaradi narave večkriterijskih evolucijskih algoritmov se njihovo vedenje in zmogljivost večinoma preučuje eksperimentalno. V zadnjih letih je bilo predlaganih več zbirk testov neprekinjenih MOP, ki so odigrali ključno vlogo pri razvoju in študiju večkriterijskih evolucijskih algoritmov. V EARS smo vključili vseh deset neomejenih problemov iz konference za evolucijsko računanje CEC 2009 (*IEEE Congress on Evolutionary Computation*). Vsak kriterij razširi razred *Problem*. Ker pri večkriterijskem optimiranju problem sestavlja več kriterijev, smo jih hranili v seznamu razreda *MOProblem*, ki tudi razširi razred *Problem*.

4.6.1 Neomejen problem 1

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2 \tag{4.24}$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2 \tag{4.25}$$

kjer je $J_1 = \{j \mid j$ je liho in $2 \leq j \leq n\}$ in $J_2 = \{j \mid j$ je sodo in $2 \leq j \leq n\}$.
Prostor iskanja je $[0,1] \times [-1,1]^{n-1}$.

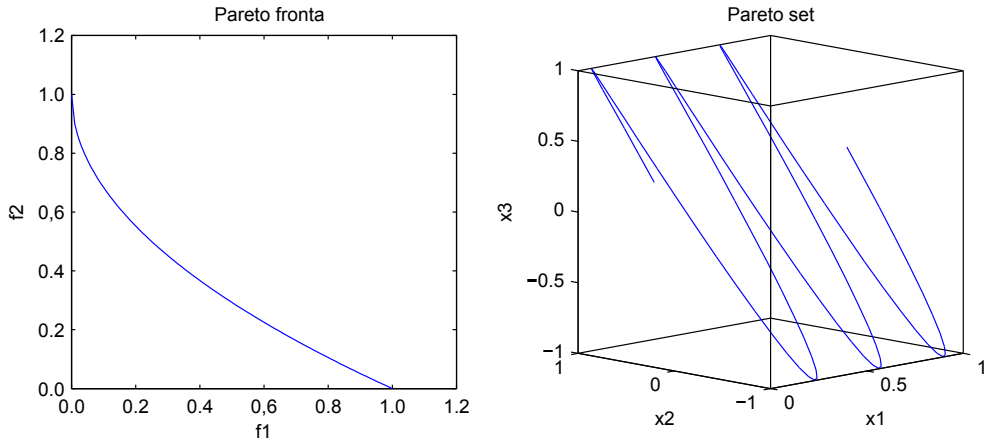
Njegova PF (*Pareto Front*) je

$$f_2 = 1 - \sqrt{f_1},\ 0 \leq f_1 \leq 1. \tag{4.26}$$

Njegov PS (*Pareto Set*) je

$$x_j = \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n,\ 0 \leq x_1 \leq 1. \tag{4.27}$$

$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.5.



Slika 4.5: Prikaz PF in PS za neomejen problem 1 [27].

4.6.2 Neomejen problem 2

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} y_j^2 \tag{4.28}$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} y_j^2 \tag{4.29}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1 in

$$y_j = \begin{cases} x_j - [0,3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0,6x_1] \cos(6\pi x_1 + \frac{j\pi}{n}) \; j \in J_1 \\ x_j - [0,3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0,6x_1] \sin(6\pi x_1 + \frac{j\pi}{n}) \; j \in J_2 \end{cases}$$

Prostor iskanja je $[0,1] \times [-1,1]^{n-1}$.

Njegova PF je

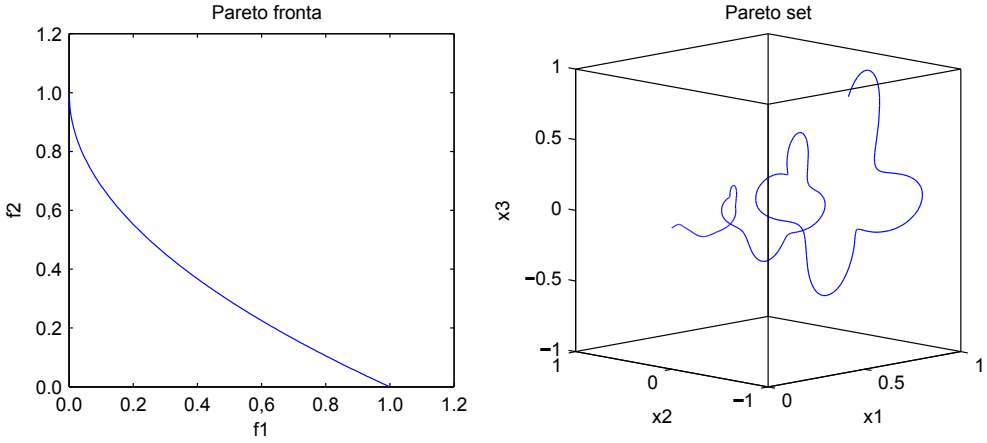
$$f_2 = 1 - \sqrt{f_1}, \; 0 \leq f_1 \leq 1. \tag{4.30}$$

Njegov PS je

$$y_j = \begin{cases} \{0,3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0,6x_1\} \cos(6\pi x_1 + \frac{j\pi}{n}) \; j \in J_1 \\ \{0,3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0,6x_1\} \sin(6\pi x_1 + \frac{j\pi}{n}) \; j \in J_2 \end{cases} \quad 0 \leq x_1 \leq 1$$

(4.31)

$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.6.



Slika 4.6: Prikaz PF in PS za neomejen problem 2 [27].

4.6.3 Neomejen problem 3

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + \frac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2) \tag{4.32}$$

$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2) \tag{4.33}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1 in

$$y_j = x_j - x_1^{0,5(1,0+\frac{3(j-2)}{n-2})}, j = 2, \ldots, n.$$

Prostor iskanja je $[0, 1]^n$.

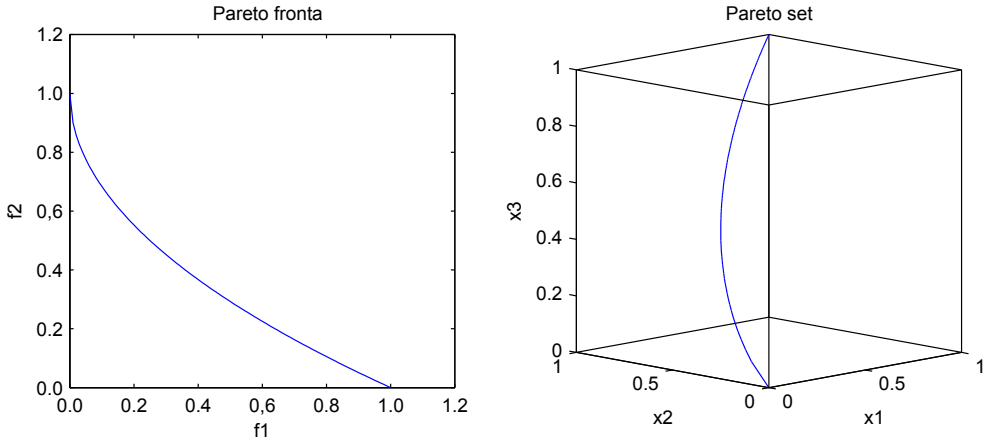
Njegova PF je

$$f_2 = 1 - \sqrt{f_1},\ 0 \leq f_1 \leq 1. \tag{4.34}$$

Njegov PS je

$$x_j = x_1^{0,5(1,0+\frac{3(j-2)}{n-2})}, j = 2, \ldots, n \; 0 \leq x_1 \leq 1. \tag{4.35}$$

$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.7.



Slika 4.7: Prikaz PF in PS za neomejen problem 3 [27].

4.6.4 Neomejen problem 4

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} h(y_j) \tag{4.36}$$

$$f_2 = 1 - x_1^2 + \frac{2}{|J_2|} \sum_{j \in J_2} h(y_j) \tag{4.37}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1,

$$y_i = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n$$

in

$$h(t) = \frac{|t|}{1 + e^{2|t|}}.$$

Prostor iskanja je $[0,1] \times [-2,2]^{n-1}$.

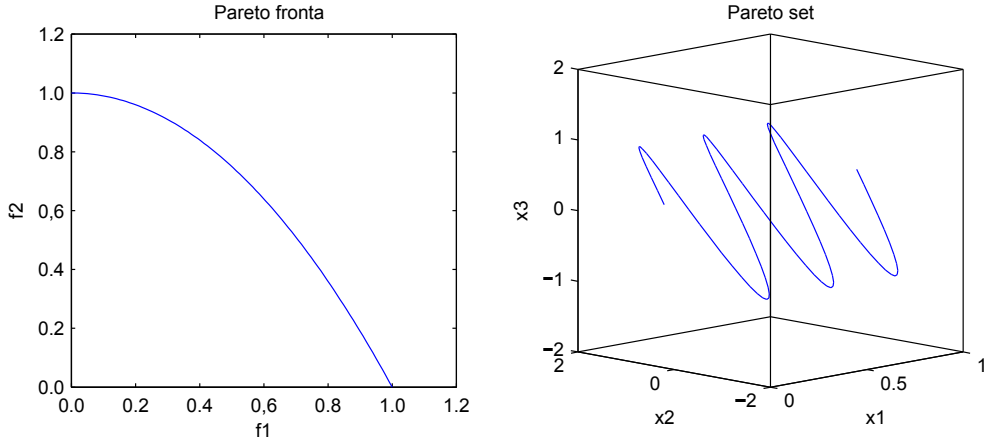
Njegova PF je

$$f_2 = 1 - f_2^2,\ 0 \leq f_1 \leq 1. \tag{4.38}$$

Njegov PS je

$$x_j = \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n.\ 0 \leq x_1 \leq 1. \tag{4.39}$$

$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.8.



Slika 4.8: Prikaz PF in PS za neomejen problem 4 [27].

4.6.5 Neomejen problem 5

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + (\frac{1}{2N} + \varepsilon)\,|\sin(2N\pi x_1)| + \frac{2}{|J_1|}\sum_{j \in J_1} h(y_j) \tag{4.40}$$

$$f_2 = 1 - x_1 + (\frac{1}{2N} + \varepsilon)\,|\sin(2N\pi x_1)| + \frac{2}{|J_2|}\sum_{j \in J_2} h(y_j) \tag{4.41}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1. N je celo število, $\varepsilon > 0$,

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n$$

in

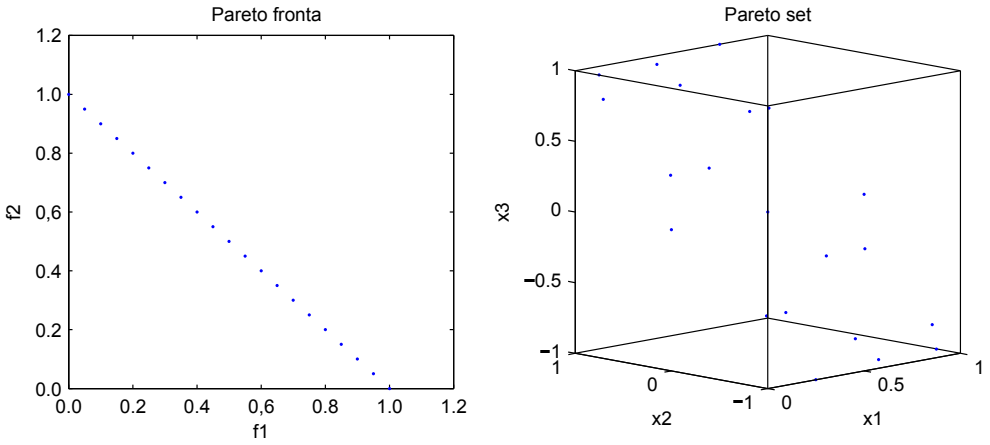
$$h(t) = 2t^2 - \cos(4\pi t) + 1.$$

Prostor iskanja je $[0,1] \times [-1,1]^{n-1}$.

Njegova PF ima $2N + 1$ Pareto optimalnih rešitev:

$$(\frac{i}{2N}, 1 - \frac{i}{2N}) \tag{4.42}$$

za $i = 0, 1, \ldots, 2N$.
$N = 10, \varepsilon = 0,1$ in $n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.9.



Slika 4.9: Prikaz PF in PS za neomejen problem 5 [27].

4.6.6 Neomejen problem 6

Kriterija, ki je treba minimizirati, sta:

$$f_1 = x_1 + max\{0, 2(\frac{1}{2N} + \varepsilon)\sin(2N\pi x_1)\} + \frac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2) \tag{4.43}$$

$$f_2 = 1-x_1+max\{0, 2(\frac{1}{2N}+\varepsilon)\sin(2N\pi x_1)\}+\frac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2-2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}})+2) \tag{4.44}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1 in

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n.$$

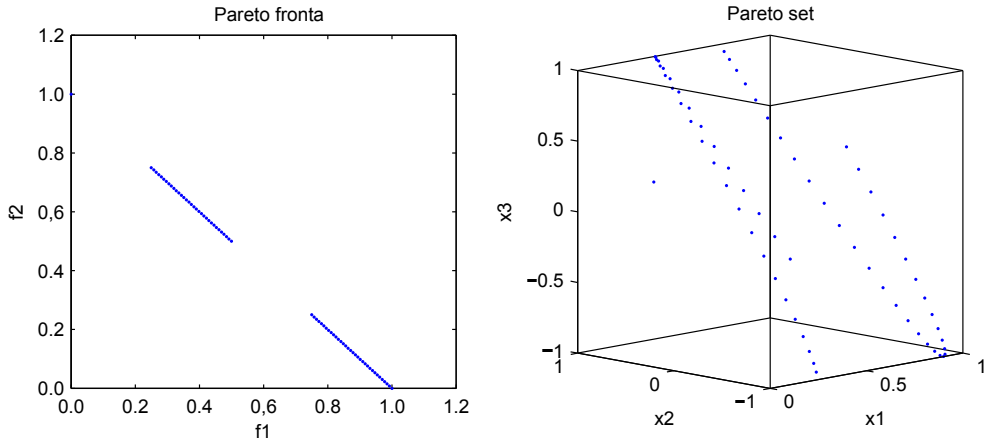
Prostor iskanja je $[0,1] \times [-1,1]^{n-1}$.

Njegova PF je sestavljena iz

- ene izolirane točke (0,1) in
- N nepovezanih delov:

$$f_2 = 1 - f_1, f_1 \in \bigcup_{i=1}^{N}[\frac{2i-1}{2N}, \frac{2i}{2N}]. \tag{4.45}$$

$N = 2, \varepsilon = 0,1$ in $n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.10.



Slika 4.10: Prikaz PF in PS za neomejen problem 6 [27].

4.6.7 Neomejen problem 7

Kriterija, ki je treba minimizirati, sta:

$$f_1 = \sqrt[5]{x_1} + \frac{2}{|J_1|} \sum_{j \in J_1} y_j^2 \tag{4.46}$$

$$f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} y_j^2 \tag{4.47}$$

kjer sta J_1 in J_2 enaka kot pri problemu 1 in

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n.$$

Prostor iskanja je $[0, 1] \times [-1, 1]^{n-1}$.

Njegova PF je

$$f_2 = 1 - f_1,\ 0 \leq f_1 \leq 1. \tag{4.48}$$

Njegov PS je

$$x_j = \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n,\ 0 \leq x_1 \leq 1. \tag{4.49}$$

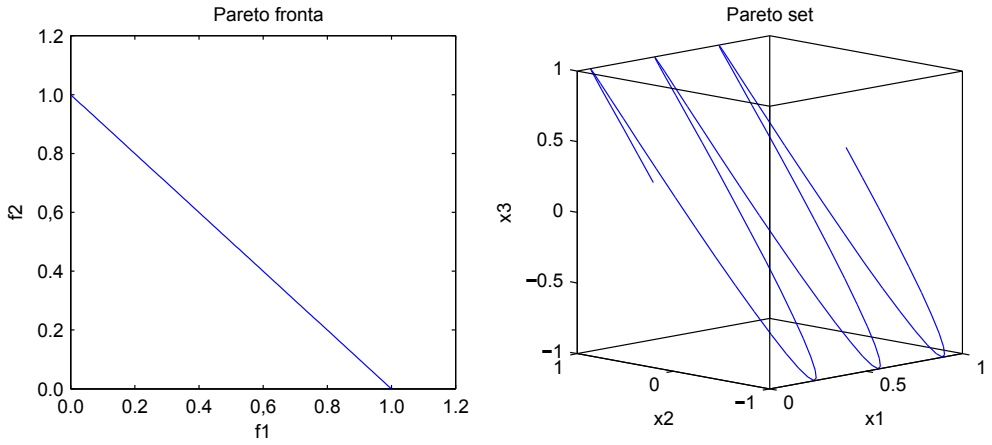
$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.11.

4.6.8 Neomejen problem 8

Kriterije, ki jih je treba minimizirati so:

$$f_1 = \cos(0,5x_1\pi)\cos(0,5x_2\pi) + \frac{2}{|J_1|} \sum_{j \in J_1} (x_j - 2x_2 \sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.50}$$

$$f_2 = \cos(0,5x_1\pi)\sin(0,5x_2\pi) + \frac{2}{|J_2|} \sum_{j \in J_2} (x_j - 2x_2 \sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.51}$$



Slika 4.11: Prikaz PF in PS za neomejen problem 7 [27].

$$f_3 = \sin(0,5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.52}$$

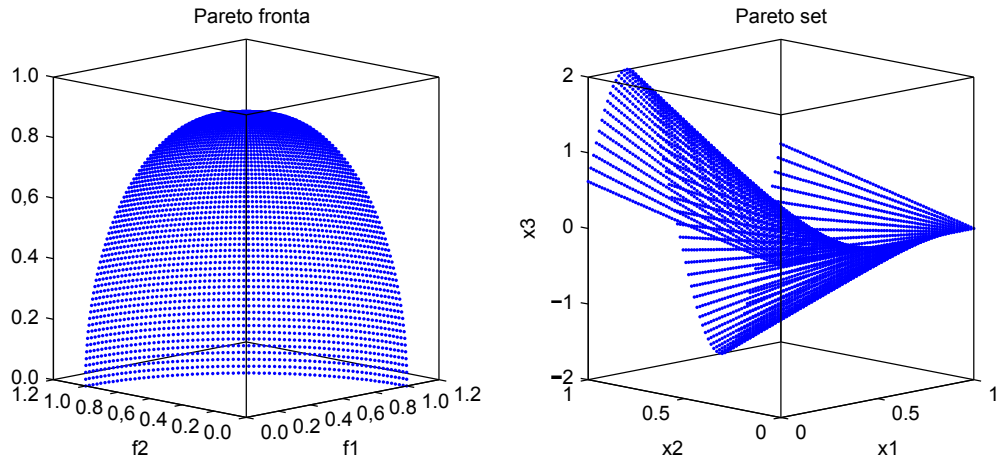
kjer so $J_1 = \{j \mid 3 \leq j \leq n \text{ in } j-1 \text{ je večkratnik števila } 3\ \}$,
$J_2 = \{j \mid 3 \leq j \leq n \text{ in } j-2 \text{ je večkratnik števila } 3\ \}$ in
$J_3 = \{j \mid 3 \leq j \leq n \text{ in } j \text{ je večkratnik števila } 3\ \}$.
Prostor iskanja je $[0,1]^2 \times [-2,2]^{n-2}$.

Njegova PF je $f_1^2 + f_2^2 + f_3^3 = 1, 0 \leq f_1, f_2, f_3 \leq 1$.
Njegov PS je $x_j = 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}), j = 3, \ldots, n$.
$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.12.

4.6.9 Neomejen problem 9

Kriterije, ki jih je treba minimizirati so:

$$f_1 = 0,5[max\left\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\right\} + 2x_1]x_2 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.53}$$



Slika 4.12: Prikaz PF in PS za neomejen problem 8 [27].

$$f_2 = 0,5[max\left\{0,(1+\varepsilon)(1-4(2x_1-1)^2)\right\} - 2x_1 + 2]x_2 + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.54}$$

$$f_3 = 1 - x^2 + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2 \tag{4.55}$$

kjer so J_1, J_2 in J_3 enaki kot pri problemu 8 in

$$\varepsilon = 0,1$$

ε lahko zavzema katerekoli druge pozitivne vrednosti.
Prostor iskanja je $[0,1]^2 \times [-2,2]^{n-2}$.

PF ima dva dela. Prvi del je

$$0 \le f_3 \le 1,$$
$$0 \le f_1 \le \frac{1}{4}(1-f_3),$$
$$f_2 = 1 - f_1 - f_3;$$

in drugi del je

$$0 \le f_3 \le 1,$$

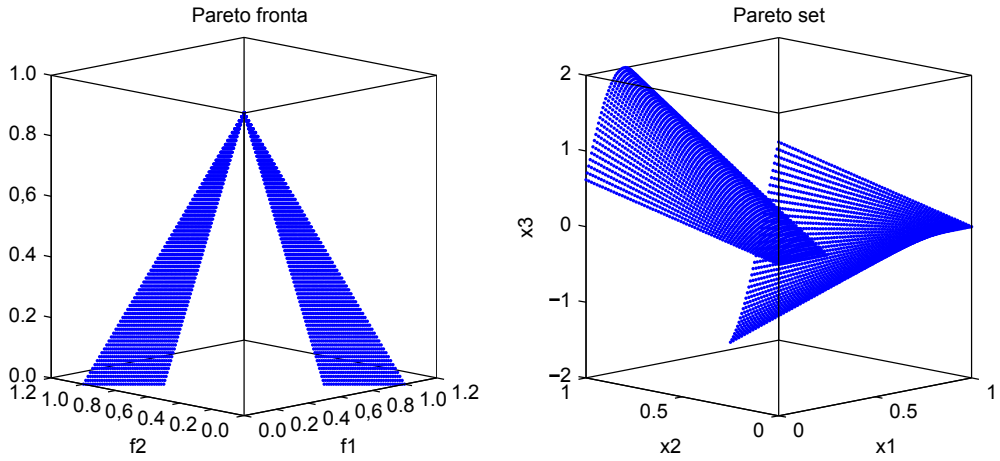
$$\frac{3}{4}(1 - f_3) \leq f_1 \leq 1,$$
$$f_2 = 1 - f_1 - f_3.$$

PS ima tudi dva prekinjena dela:

$$x_1 \in [0; 0, 25] \cup [0, 75; 1], 0 \leq x_2 \leq 1,$$

$$x_j = 2x_2 \sin(2\pi x_1 + \frac{j\pi}{n}), j = 3, \ldots, n.$$

$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.13.



Slika 4.13: Prikaz PF in PS za neomejen problem 9 [27].

4.6.10 Neomejen problem 10

Kriterije, ki jih je treba minimizirati so:

$$f_1 = \cos(0, 5x_1\pi)\cos(0, 5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1] \tag{4.56}$$

$$f_2 = \cos(0, 5x_1\pi)\sin(0, 5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1] \tag{4.57}$$

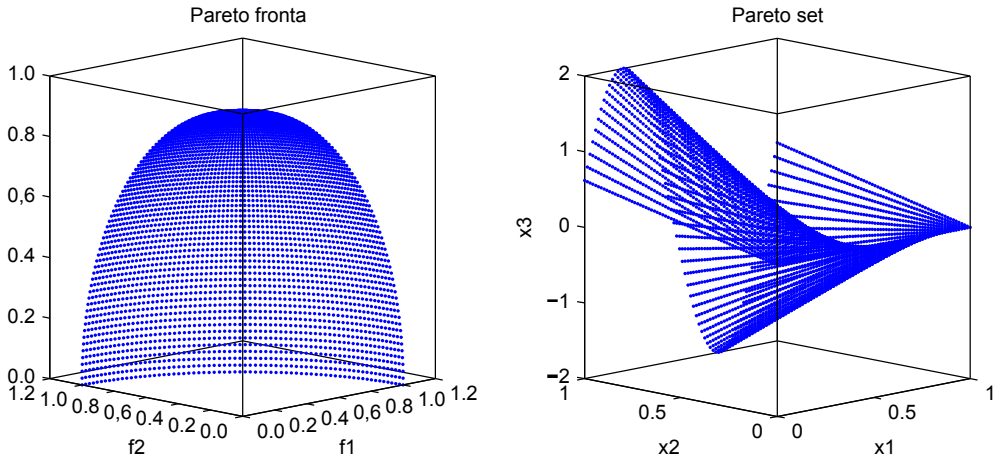
$$f_3 = \sin(0,5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1] \tag{4.58}$$

kjer so J_1, J_2 in J_3 enaki kot pri problemu 8 in

$$y_j = x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}), j = 3, \ldots, n.$$

Prostor iskanja je $[0,1]^2 \times [-2,2]^{n-2}$.

Njegova PF je $f_1^2 + f_2^2 + f_3^3 = 1, 0 \leq f_1, f_2, f_3 \leq 1$.
Njegov PS je $x_j = 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}), j = 3, \ldots, n$.
$n = 30$ v tekmovanju CEC 09.
Njuna PF in PS sta prikazana na sliki 4.14.



Slika 4.14: Prikaz PF in PS za neomejen problem 10 [27].

Poglavje 5

Eksperiment

Za primerjavo rezultatov smo pripravili tri scenarije. V prvem scenariju smo vse algoritme testirali na neomejenem problemu 4. Maksimalno število ovrednotenj smo nastavili na 300.000, kakor je bilo podano v navodilih za tekmovanje CEC 2009 [27]. Pri testiranju smo beležili čas procesiranja na CPU (*Central Processing Unit*). Ko je algoritem dosegel maksimalno število ovrednotenj, je vrnil Pareto fronto. To Pareto fronto je aplikacija shranila v CSV (*Comma Separated Values*) datoteko. Da smo lahko rezultate vizualno prikazali, smo v program dodali knjižnico JavaPlot. S knjižnico lahko izrisujemo gnuplot grafe z uporabo Java ukazov. Pareto fronto lahko izrisujemo med izvajanjem ali jo shranimo kot sliko v formatu EPS (*Encapsulated PostScript*). Na podlagi pridobljene Pareto fronte smo izračunali metrike uspešnosti.
V drugem scenariju smo primerjali EARS s klasično statistično metodo Z-test. Za Z-test smo izračunali povprečje in standardni odklon metrike IGD petdesetih zagonov na neomejenem problemu 4.
V tretjem scenariju smo primerjali *benchmark* orodja EARS s Friedmanovim testom [28, 29]. Za Friredmanov test rezultat algoritma predstavlja povprečna IGD vrednost dobljene Pareto fornte petdesetih zagonov. V *benchmarku* so vsi problemi iz tekmovanja CEC 2009. Začetni rang R smo nastavi na 1500 in odklon ranga RD na 350.

5.1 Scenarij 1

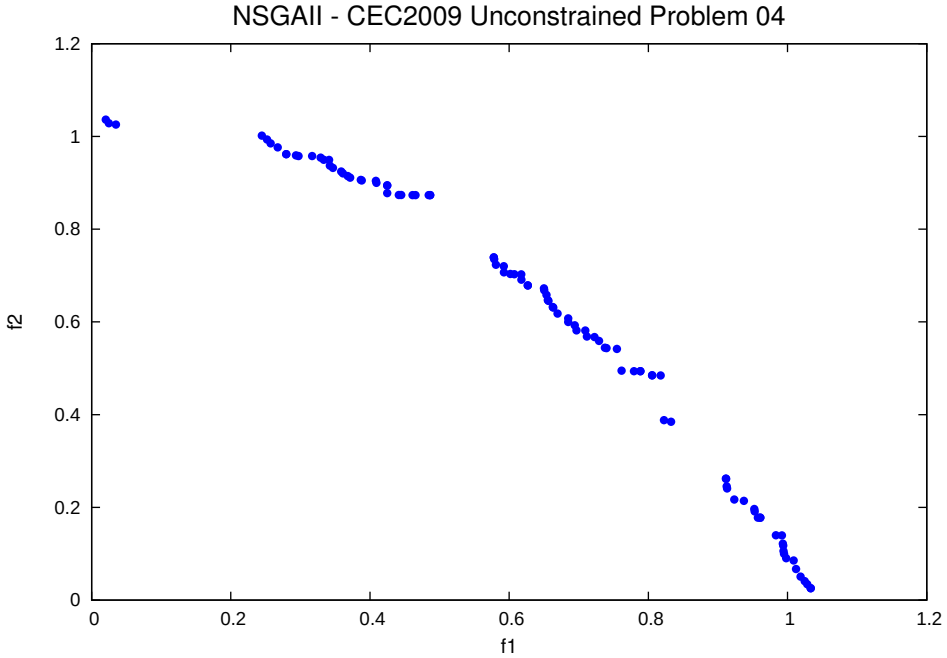
Cilj tega scenarija je, da se preverijo uspešnosti implementiranih algoritmov. Za vsak algoritem je podana tabela z okvirnim procesorskim časom in metrikami uspešnosti.

Izrisana je tudi slika Pareto fornte, ki jo je vrnil algoritem. S primerjavo podatkov in slik z rezultati iz tekmovanja CEC, smo lahko videl, če je delovanje algoritma pravilno.

5.1.1 Testni rezultati algoritma NSGA-II

Čas procesiranja (ms):	5335
HV:	0,2548260155
IGD:	0,0017655498
GD:	0,0049511474
epsilon:	0,1005874192

Algoritem NSGA-II se je odrezal zelo dobro. Iz slike 5.1 je razvidno, da so rešitve zelo blizu Pareto optimalne fronte in dobro razporejene. Tako sta zadoščena oba cilja večkriterijskega optimiranja.

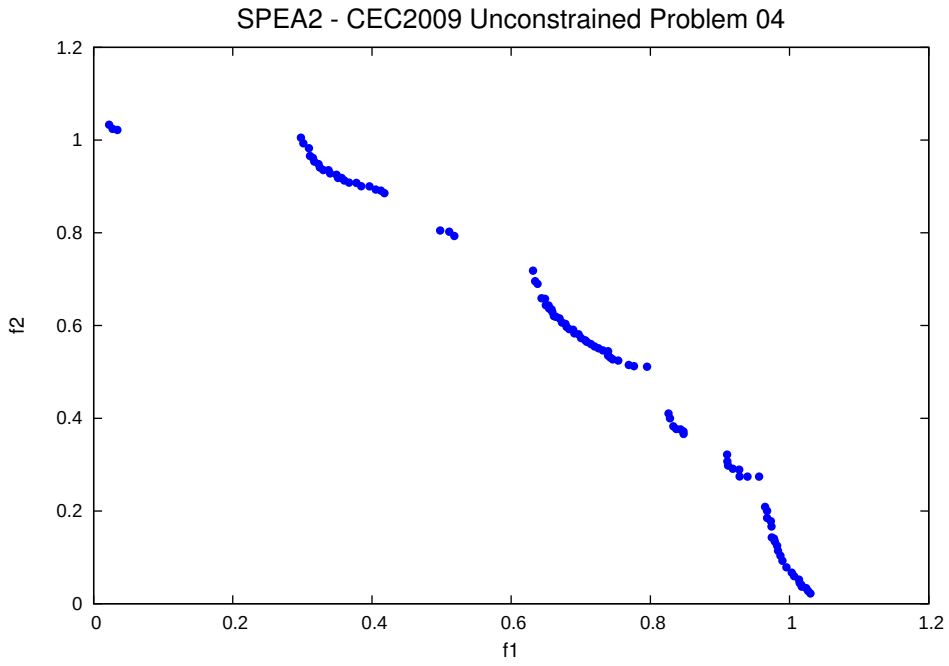


Slika 5.1: Pareto fronta, ki jo je vrnil algoritem NSGA-II za neomejen problem 4.

5.1.2 Testni rezultati algoritma SPEA2

Čas procesiranja (ms):	17022
HV:	0,2527031593
IGD:	0,0019777028
GD:	0,0048221085
epsilon:	0,0877500019

Algoritma SPEA2 se je izvajal največ časa. Kljub temu je rešitev, ki jo je vrnil, zelo dobra. Na sliki 5.2 lahko vidimo samo en večji odsek (med f1 = 0,01-0,3), ki ni zapolnjen.

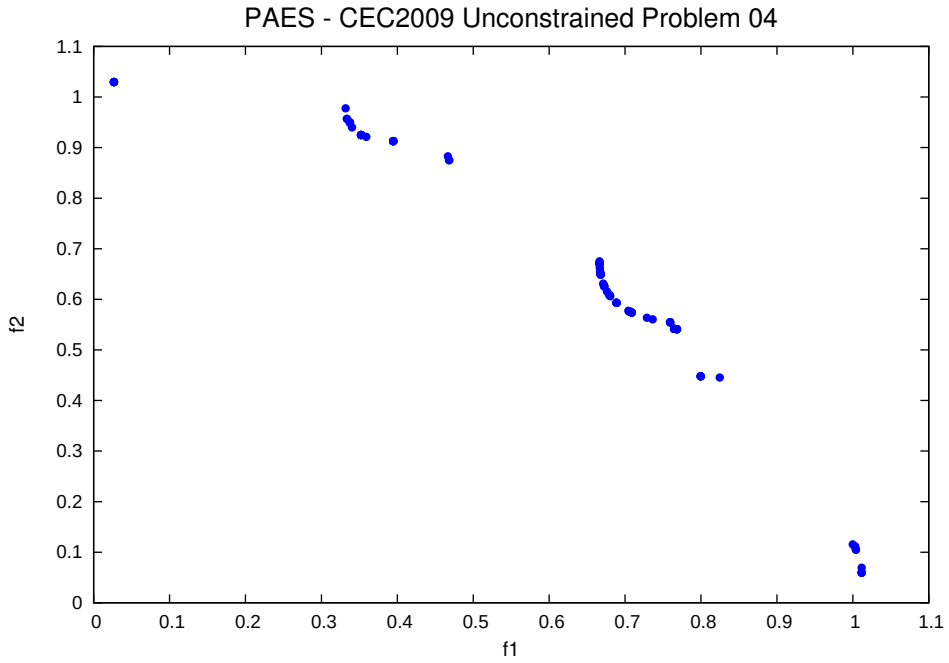


Slika 5.2: Pareto fronta, ki jo je vrnil algoritem SPEA2 za neomejen problem 4.

5.1.3 Testni rezultati algoritma PAES

Čas procesiranja (ms):	2999
HV:	0,2037883606
IGD:	0,0029265979
GD:	0,0056351506
epsilon:	0,1560209367

Algoritem PAES se je odrezal najslabše, kar je bilo tudi za pričakovati od algoritma, ki uporablja evolucijske strategije. Pri teh je značilno, da obtičijo v lokalnem optimumu. Na sliki 5.3 lahko vidimo, da je aproksimacija Pareto fronte zelo nepopolna. Posledično ima zato tudi slabe vrednosti metrik uspešnosti.

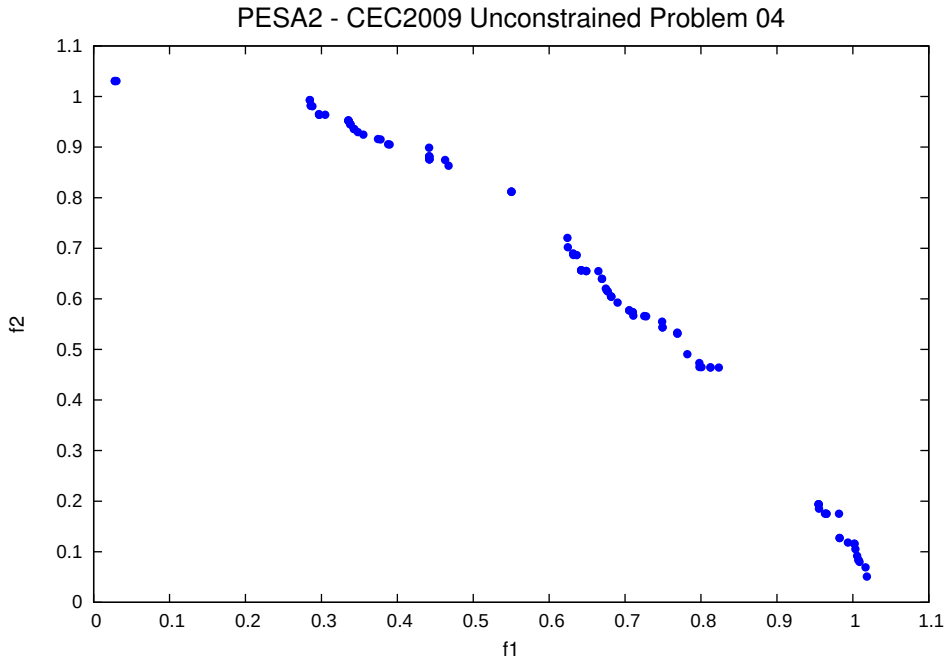


Slika 5.3: Pareto fronta, ki jo je vrnil algoritem PAES za neomejen problem 4.

5.1.4 Testni rezultati algoritma PESA-II

Čas procesiranja (ms):	16849
HV:	0,2323865219
IGD:	0,0023079135
GD:	0,0053466829
epsilon:	0,1350703052

Kot vidimo na sliki 5.4, se je algoritem PESA-II zelo približal Pareto optimalni fronti. Kljub temu rešitve niso dovolj enakomerno porazdeljene po prostoru kriterijev, kar je tudi vidno v metrikah. Algoritem ima prav tako zelo velik čas procesiranja.

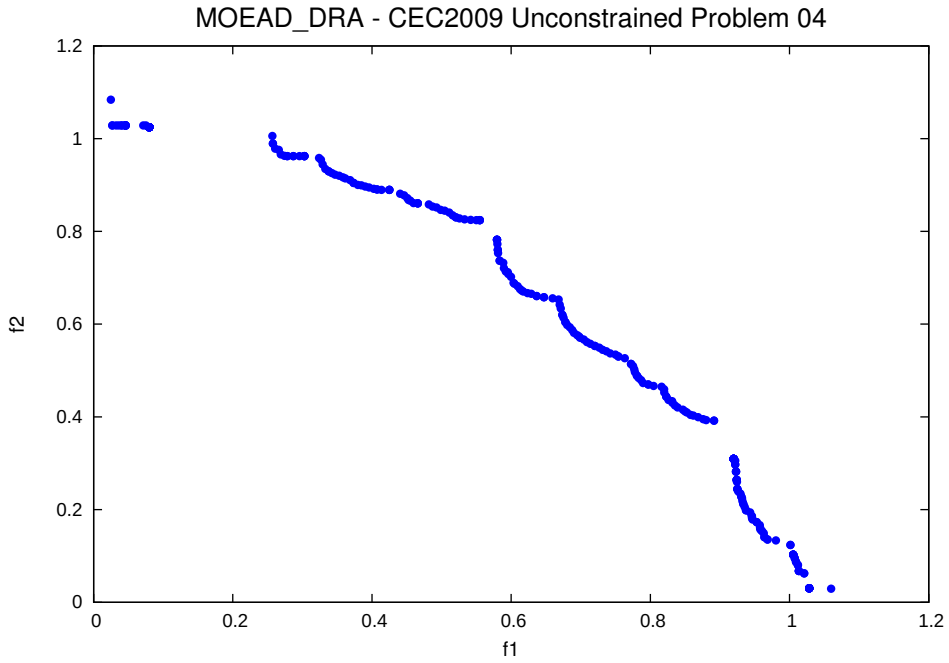


Slika 5.4: Pareto fronta, ki jo je vrnil algoritem PESA-II za neomejen problem 4.

5.1.5 Testni rezultati algoritma MOEA/D

Čas procesiranja (ms):	4257
HV:	0,2589058589
IGD:	0,0016808934
GD:	0,0029698478
epsilon:	0,0857959711

Na sliki 5.5 lahko vidimo, da se Pareto fronta, ki jo je vrnil algoritem, najbolje prilega Pareto optimalni fronti. Algoritem je v vseh metrikah najboljši, torej ima najmanjši IGD,GD in epsilon ter največjo HV. Algoritem se za svojo kompleksnost izvaja zelo hitro.

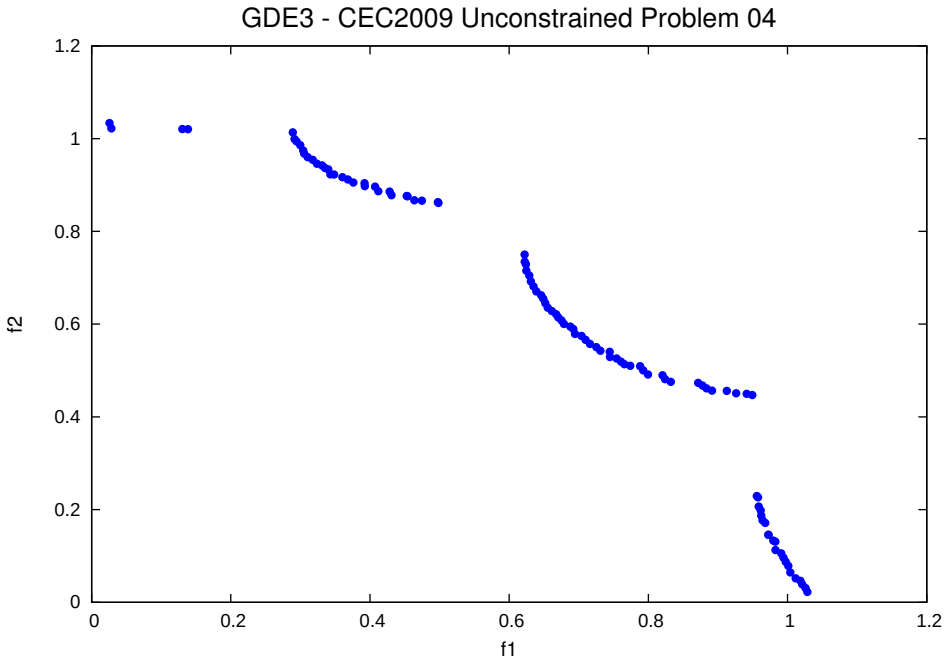


Slika 5.5: Pareto fronta, ki jo je vrnil algoritem MOEA/D za neomejen problem 4.

5.1.6 Testni rezultati algoritma GDE3

Čas procesiranja (ms):	3173
HV:	0,2315032908
IGD:	0,0021443372
GD:	0,0065366351
epsilon:	0,1290340458

Rezultati metrik algoritma GDE3 so slabi, ker je kot rešitev vrnil dosti točk, ki so zelo oddaljene od Pareto optimalne fronte. Na sliki 5.6 lahko vidimo, da se te točke nahajajo na odsekih, kjer je aproksimacija Pareto fronte prekinjena.



Slika 5.6: Pareto fronta, ki jo je vrnil algoritem GDE3 za neomejen problem 4.

5.1.7 Primerjava rezultatov

Iz rezultatov v tabeli 5.1 lahko vidimo, da vsi algoritmi izbirajo samo dominantne rešitve in lahko tako sklepamo, da so pravilno implementirani. Najbolje se odrezal algoritem MOEA/D, saj ima najboljše rezultate v vseh metrikah, vendar je bil najhitrejši algoritem PAES. Pri primerjavi metrike HV ne smemo pozabiti, da so večje vrednosti boljše. Za referenco smo izračunali HV optimalne Pareto fronte, ki znaša 0,332832999. Uspešnost algoritmov je v okvirju pričakovanih vrednosti.

Tabela 5.1: Primerjava rezultatov algoritmov za neomejen problem 4.

	NSGA-II	SPEA2	PAES	PESA-II	MOEA/D	GDE3
HV	0,254826016	0,252703159	0,203788361	0,232386522	**0,258905859**	0,231503291
IGD	0,00176555	0,001977703	0,002926598	0,002307914	**0,001680893**	0,002144337
GD	0,004951147	0,004822109	0,005635151	0,005346683	**0,002969848**	0,006536635
ε	0,100587419	0,087750002	0,156020937	0,135070305	**0,085795971**	0,129034046
CPU (ms)	5335	17022	**2999**	16849	4257	3173

5.2 Scenarij 2

V tem scenariju smo orodje EARS primerjali z Z-testom. Z-test je statističen test, kjer se uporablja normalna porazdelitev in imamo večjo število vzorcev ($n \geq 30$). S

testom lahko primerjamo dve povprečji in ugotovimo, če sta signifikantno različni. Ničelna hipoteza H_0 navaja, da sta povprečni vrednosti metrike IGD enaki. V tabeli 5.2 so zapisane povprečne vrednosti in standardni odkloni metrike IGD petdesetih zagonov algoritmov na neomejenem problemu 4. Za vsak algoritem smo tudi izračunali standardni odklon. Iz povprečne vrednosti metrike IGD lahko vidimo, da ima najboljši rezultat algoritem MOEA/D. Ko imamo vse podatke. lahko izračunamo vrednost z po enačbi:

$$z = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{5.1}$$

Tabela 5.2: Povprečne IGD vrednosti in standardni odklon algoritmov za neomejen problem 4.

	MOEA/D	NSGA-II	SPEA2	GDE3	PESA-II	PAES
$\bar{x}$	**1,5534e-03**	1,8474e-03	1,9745e-03	2,3666e-03	2,1511e-03	3,6046e-03
σ	1,551306007e-4	1,356444264e-4	0,001634940	2,244179556e-4	9,459884213e-5	1,423136030e-4

V tabeli 5.3 so izračunane z vrednosti za vse pare algoritmov. Iz z vrednosti smo nato izračunali vrednost p, ki je zapisana v tabeli 5.4. Kjer je vrednost p manjša od $\alpha = 0,05$, je ničelna hipoteza zavrnjena. Vidimo, da je v večini primerov vrednost p manjša od 0,0001, kar pomeni, da je med rezultatoma signifikantna razlika. Večje vrednosti p se pojavijo pri primerjavi katerega koli algoritma z algoritmom SPEA2, razen pri primerjavi algoritma PAES z algoritmom SPEA2.

Tabela 5.3: Izračunane z vrednosti.

	NSGA-II	SPEA2	GDE3	PESA-II	PAES
MOEA/D	-10,0883	-1,8131	-21,0771	-23,2604	-68,8969
NSGA-II		-0,5478	-14,0005	-12,9857	-63,2000
SPEA2			-1,6801	-0,7625	-7,0236
GDE3				6,2569	-32,9422
PESA-II					-60,1441

Tabela 5.4: Izračunane p vrednosti.

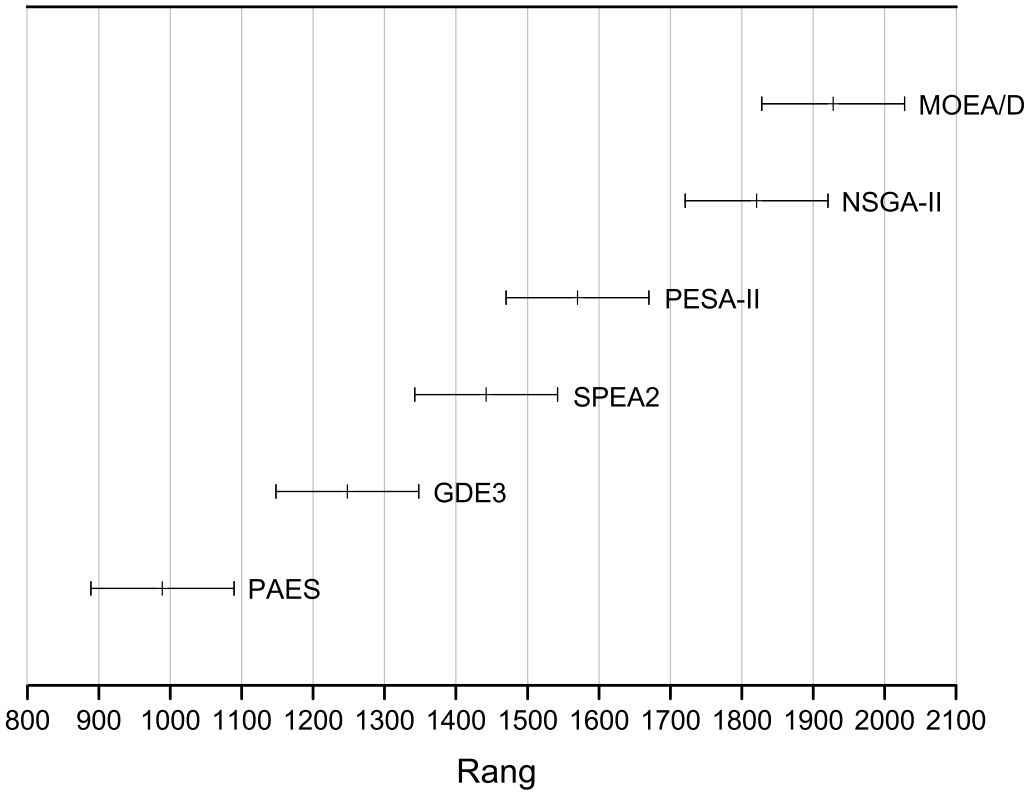
	NSGA-II	SPEA2	GDE3	PESA-II	PAES
MOEA/D	<0,0001	**0,0698**	<0,0001	<0,0001	<0,0001
NSGA-II		**0,5838**	<0,0001	<0,0001	<0,0001
SPEA2			**0,0929**	**0,4458**	<0,0001
GDE3				<0,0001	<0,0001
PESA-II					<0,0001

V tabeli 5.5 so prikazani rangi vseh algoritmov, ki jih je vrnilo orodje EARS za problem UP04. Izračunali smo tudi intervale rangov, kjer je RD za vsak rang enak

50. Na sliki 5.7, lahko vidimo izrisane intervale rangov vseh algoritmov. Kjer se rangi prekrivajo, ne moremo trditi, da je algoritem signifikantno boljši. Tako je algoritem MOEA/D prvi in signifikantno boljši od vseh, razen od algoritma NSGA-II. Iz slike lahko vidimo, da so vsi algoritmi signifikantno boljši od algoritma PAES.

Tabela 5.5: Rangi algoritmov, ki jih je vrnil EARS za UP04.

	Algoritem	Rang	RD	± 2RD (95%)
1.	MOEA/D	1928,3	50	[1828, 2028]
2.	NSGA-II	1821,2	50	[1721, 1921]
3.	PESA-II	1570	50	[1470, 1670]
4.	SPEA2	1442	50	[1342, 154]
5.	GDE3	1248,8	50	[1148, 1348]
6.	PAES	989,4	50	[889, 1089]



Slika 5.7: Prikaz intervala rangov, ki jih je vrnil EARS za UP04.

Iz rezultatov lahko vidimo, da smo z orodjem EARS odkrili več signifikantnih razlik med algoritmi kakor z Z-testom. Ker imamo tudi interval ranga, lahko lažje vidimo,

kako signifikantna je razlika. Iz rezultatov lahko sklepamo, da se orodje EARS lahko primerja s klasičnimi statističnimi metodami.

5.3 Scenarij 3

V tem scenariju smo rezultate EARS primerjali s Friedmanovim testom, ki je neparametrična alternativa testu ANOVA. Friedmanov test rangira algoritme z uporabo nabora podatkov. Povprečen rang algoritma je $R_j = \frac{1}{N}\sum_i^N r_{i,j}$, kjer je N število problemov, k število algoritmov in $r_{i,j}$ rang j-tega algoritma na i-tem problemu. Ničelna hipoteza navaja, da opazovani algoritmi vračajo enak rezultat in imajo zato enak rang. Friedmanova statistika (enačba 5.2) je porazdeljena po χ^2 in ima $(k-1)$ stopenj prostosti.

$$\chi_F^2 = \frac{12N}{k\,(k+1)}\left(\sum_j R_j^2 - \frac{k\,(k+1)^2}{4}\right) \tag{5.2}$$

Friedmanov test smo opravili na celotnem *benchmarku* in smo za določanje ranga algoritma uporabili povprečno vrednost IGD petdesetih zagonov. V tabeli 5.5 lahko v zadnji vrstici vidimo povprečne range, potrebne za Friedmanov test. Manjši je rang, boljši je algoritem. Pri vsakem rezultatu smo izračunali še standardni odklon. Rang posameznega algoritma na problemu je podan v oklepajih. Pri določanju ranga smo upoštevali mejo ε = 1e-07, ki smo jo tudi nastavili za *benchmark*. Na prvem mestu je algoritem MOEA/D. Drugo mesto si delita algoritma NSGA-II in SPEA2, nato sledijo algoritmi GDE3, PESA-II in na zadnjem mestu PAES. Friedmanova statistika, porazdeljena po χ^2, s 5 stopnjami prostosti je enaka $\chi_F^2 = 28,91428571$. Vrednost p, izračunana s Fridmanovim testom, je enaka 2.41028506e-5. Ker je p manjša od signifikantne stopnje $\alpha = 0,05$, je hipoteza, da ni razlike med rangi teh 6 algoritmov, zavrnjena.

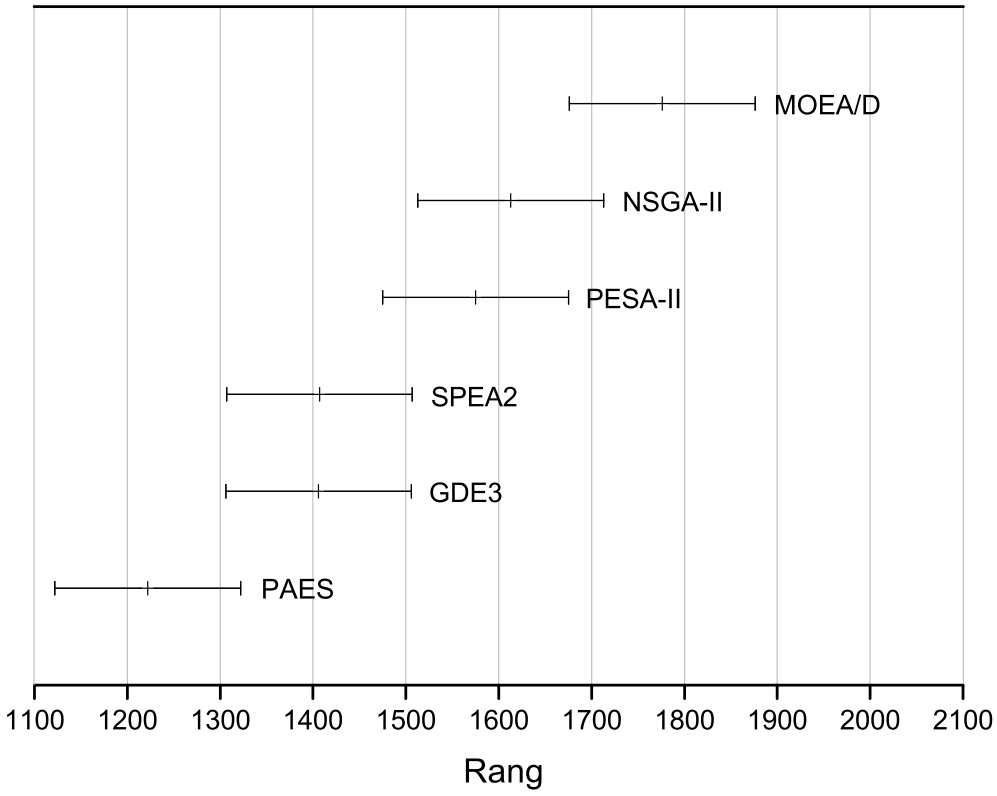
Tabela 5.6: Rezultati Friedmanovega test.

	MOEA/D	NSGA-II	SPEA2	GDE3	PESA-II	PAES
UP01	3,6339e-04 (1)	3,1698e-03 (3)	3,6678e-03 (5)	2,9111e-03 (2)	3,3749e-03 (4)	7,6104e-03 (6)
σ	7,8756e-05	1,4395e-04	1,4111e-04	1,2676e-04	1,5536e-04	3,2283e-03
UP02	4,7436e-04 (1)	1,5299e-03 (3)	1,6734e-03 (4)	1,2412e-03 (2)	2,2880e-03 (5)	2,6003e-03 (6)
σ	1,1523e-04	1,5476e-04	2,1087e-04	3,4328e-04	4,1227e-04	9,3085e-04
UP03	3,3849e-03 (1)	6,5537e-03 (2)	8,7084e-03 (4)	8,2643e-03 (3)	1,1143e-02 (5)	1,1188e-02 (6)
σ	5,7613e-04	1,6427e-03	1,6567e-03	9,5483e-04	1,0635e-03	1,1715e-03
UP04	1,5534e-03 (1)	1,8474e-03 (2)	1,9745e-03 (3)	2,3666e-03 (5)	2,1511e-03 (4)	3,6046e-03 (6)
σ	8,4038e-05	1,3073e-04	1,7531e-04	1,3838e-04	1,9798e-04	2,3735e-03
UP05	6,3433e-02 (4)	5,1838e-02 (2)	5,4766e-02 (3)	4,1920e-02 (1)	7,2924e-02 (5)	1,0906e-01 (6)
σ	1,6773e-02	1,0408e-02	1,4289e-02	7,0722e-03	2,6463e-02	2,7594e-02
UP06	1,0993e-02 (1)	1,2795e-02 (3)	1,2438e-02 (2)	1,3381e-02 (4)	1,4545e-02 (5)	1,8521e-02 (6)
σ	9,0289e-04	2,1352e-03	1,4380e-03	1,5756e-04	3,3672e-03	4,6382e-03
UP07	3,9169e-04 (1)	1,4280e-02 (3)	1,4612e-02 (4)	1,3885e-02 (2)	1,5106e-02 (5)	1,7713e-02 (6)
σ	2,0747e-04	3,6323e-04	2,0100e-04	1,9521e-04	1,2229e-03	4,1569e-03
UP08	7,6604e-04 (1)	2,9843e-03 (3)	2,2395e-03 (2)	3,5855e-03 (4)	6,7356e-03 (6)	4,6772e-03 (5)
σ	9,6427e-05	4,2892e-04	1,2023e-04	3,0406e-04	4,6172e-04	7,2174e-04
UP09	1,9646e-03 (1)	4,5302e-03 (4)	3,5565e-03 (2)	5,5081e-03 (5)	6,3824e-03 (6)	3,7810e-03 (3)
σ	5,3591e-04	2,9341e-04	3,8890e-04	9,3387e-04	2,7106e-04	8,3360e-04
UP10	4,7394e-03 (2)	5,2188e-03 (5)	4,6731e-03 (1)	6,2014e-03 (6)	4,8559e-03 (4)	4,8547e-03 (3)
σ	1,2025e-04	5,4292e-04	1,9932e-04	2,0704e-03	1,5904e-04	7,5552e-04
$\bar{x}$	**1,4**	**3**	**3**	**3,4**	**4,9**	**5,3**

V tabeli 5.7 so prikazani rangi vseh algoritmov, ki jih je vrnilo orodje EARS za *benchmark* $RatingCEC2009$. Višji je rang, boljši je algoritem. Kakor v scenariju 2, smo tudi tukaj izračunali in izrisali interval ranga, kjer ima RD minimalno vrednost 50. Na sliki 5.8 lahko vidimo, da se intervali veliko bolj prekrivajo, kakor pri scenariju 2. To je posledica tega, da *benchmark* vključuje več problemov in smo lahko tako natančneje določili rang posameznega algoritma. Algoritem MOEA/D je še zmeraj signifikantno boljši od vseh algoritmov, razen od algoritma NSGA-II. Ne moremo več trditi, da so vsi algoritmi signifikantno boljši od algoritma PAES, saj se njegov interval ranga prekriva z intervalom algoritma GDE3 in PESA-II. V scenariju 2 je bil algoritem PESA-II signifikantno boljši od algoritma GDE3, v tem scenariju pa se njuna intervala rangov skoraj popolnoma prekrivata. Tako lahko vidimo, kako pomembno je algoritme testirati na več različnih problemih.

Tabela 5.7: Rangi algoritmov, ki jih je vrnil EARS za *benchmark* $RatingCEC2009$.

	Algoritem	Rang	RD	± 2RD (95%)
1.	MOEA/D	1776	50	[1676, 1876]
2.	NSGA-II	1613,3	50	[1513, 1713]
3.	SPEA2	1575,1	50	[1475, 1675]
4.	PESA-II	1407	50	[1307, 1507]
5.	GDE3	1406,2	50	[1306, 1506]
6.	PAES	1222,3	50	[1122, 1322]



Slika 5.8: Prikaz intervala rangov, ki jih je vrnil EARS za *benchmark* $RatingCEC2009$.

V tabeli 5.8 lahko vidimo primerjavo, kako sta EARS in Friedmanov test uvrstila algoritme. Vrstni red algoritmov je v obeh primerih enak, razen za algoritma GDE3 in PESA-II. Po Friedmanovem testu se je algoritem GDE3 uvrstil pred PESA-II, pri EARS pa je ravno obratno. Če primerjamo range teh algoritmov, ki jih je vrnil EARS, ugotovimo, da sta algoritma skoraj enako uspešna, kar ni videti iz rangov Friedmanovega testa. Zaradi majhnih vrednosti rangov Friedmanovega testa veliko težje primerjamo uspešnosti med algoritmi, medtem ko nam orodje EARS to omogoča.

Tabela 5.8: Primerjava rangov EARS in Friedmanovega testa.

	Friedmanov test	EARS
1.	MOEA/D (1,4)	MOEA/D (1776)
2.	NSGA-II (3)	NSGA-II (1613,3)
3.	SPEA2 (3)	SPEA2 (1575,1)
4.	GDE3 (3,4)	PESA-II (1407)
5.	PESA-II (4,9)	GDE3 (1406,2)
6.	PAES (5,3)	PAES (1222,3)

Poglavje 6

Zaključek

Magistrsko delo smo pričeli z namenom, da bi lahko ocenili in primerjali algoritme za večkriterijsko optimiranje z uporabo sistema za šahovsko rangiranje. Odločili smo se, da bomo nadgradili orodje EARS, da bo omogočalo ocenjevanje algoritmov za večkriterijsko optimizacijo in ne samo enokriterijsko optimizacijo. Da lahko orodje podpira primerjavo algoritmov za večkriterijsko optimizacijo, smo morali dodati nekatere funkcionalnosti in razširiti večino obstoječih razredov. Ko smo pripravili okolje, smo v orodju implementirali nekaj obstoječih popularnih algoritmov. Ker algoritmi vrnejo aproksimacijo Pareto fronte, smo vključili metrike uspešnosti, ki vrnejo neko oceno Pareto fronte, s katero smo lahko primerjali rezultate. Pri šahovskem rangiranju smo se omejili na metriko IGD, ki so jo uporabili tudi na tekmovanju CEC 2009. Iz tega tekmovanja smo implementirali tudi večkriterijske probleme. Prav tako smo morali za implementacijo problemov razširiti obstoječe razrede. EARS je zasnovan tako, da lahko vsak na novo implementiran algoritem rešuje obstoječe probleme in vsak na novo dodan problem lahko rešujejo obstoječi algoritmi. Za testiranje orodja smo pripravili več scenarijev, kjer lahko vidimo rešitve problemov in ocene posameznih algoritmov. Za konec smo pripravili *benchmark* z imenom $RatingCEC2009$, ki uporablja sistem šahovskega rangiranja, pri tem se algoritmi primerjajo z reševanjem vseh problemov, ki so bili uporabljeni na tekmovanju CEC 2009. *Benchmark* nam je vrnil lestvico algoritmov in njihov rang. S pomočjo rangov lahko primerjamo uspešnost algoritmov na podanih problemih. Uporabnik orodja lahko po želji doda ali odstrani algoritem ali problem iz *benchmarka* in ga ponovno požene. Tako lahko enostavno ustvarimo testno okolje z različnimi algoritmi in problemi. V eksperimentalnem delu smo pokazali, da se lahko EARS primerja s standardnim statističnim testom. Z rangi, ki jih je vrnilo orodje EARS, smo lahko uspešno ocenili implemen-

tirane algoritme in odkrili signifikantne razlike med njimi. Za najboljši algoritem se je izkazal MOEA/D, ki pa ni signifikantno boljši od algoritma NSGA-II. Rezultati eksperimenta so bili v okviru pričakovanj, tako lahko trdimo, da so algoritmi uspešno implementirani.

Z nadgradnjo orodja EARS smo dosegli zastavljene cilje in z rezultati pokazali, da lahko z uporabo šahovskega rangiranja primerjamo algoritme za večkriterijsko optimizacijo. Ker je orodje prosto dostopno, lahko vsak uporabnik implementira svoj algoritem in ga primerja z algoritmi v orodju. Vsi algoritmi imajo enako število ovrednotenj in rešujejo iste probleme, kar pomeni, da je ocenjevanje tudi pošteno. Pri uporabi orodja moramo upoštevati, da večina metrik, vključno z IGD, potrebuje optimalno Pareto fronto za izračun uspešnosti. Ker imajo problemi iz tekmovanja CEC 2009 podane optimalne Pareto fronte, nismo imeli težav pri izračunu. V primeru, da Pareto fronte nimamo podane, se tej težavi lahko izognemo tako, da uporabimo implementirano metriko HV, ki ne potrebuje optimalne Pareto fronte. Pri primerjavi algoritmov moramo vedeti, da je rang algoritma odvisen od *benchmarka* in od algoritmov, ki so sodelovali.

Za prihodnost imamo željo, da bi orodje EARS postalo standardna platforma za primerjavo evolucijskih algoritmov, kjer bi lahko raziskovalci objektivno ocenili svoje evolucijske algoritme na različnih *benchmarkih*. Rezultate bi javno objavili, kjer bi jih lahko primerjali z ostalimi algoritmi. Orodje EARS je najbolj primerno za primerjavo evolucijskih algoritmov, ker je učinkovito, hitro in objektivno, saj je bilo razvito s strani tretje osebe, ki nima lastnih interesov. Na novo razviti algoritmi bi lahko hitro pridobili rang in s tem bi tudi prikazali njihovo učinkovitost in moč ter prihranili na času. Želja je tudi, da bi EARS uporabljali recenzenti, ki morajo pogosto ocenjevati na novo razvite algoritme.

Literatura

[1] V. Pareto, Cours D′Economie Politique, *Droz*, 1896.

[2] C. A. Coello Coello, Evolutionary multi-objective optimization: a historical view of the field, *Computational Intelligence Magazine (IEEE)*, 1(1): 28–36, 2006.

[3] Robič Tea, Filipič Bogdan, Večkriterijsko optimiranje z genetskimi algoritmi in diferencialno evolucijo, *Delovno poročilo IJS-DP 9065*, 2005.

[4] L. *Bradstreet, The hypervolume indicator for multi-objective optimisation: calculation and use*, doktorska disertacija, 2011.

[5] S. Kukkonen, *Generalized Differential Evolution For Global Multi-objective Optimization With Constraints*, 2012.

[6] C. A. Coello Coello, G. B. Lamont, D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, New York, ZDA, 2007.

[7] C. A. Coello Coello, N. Cruz Cortés, Solving Multiobjective Optimization Problems using an Artificial Immune System, *Genetic Programming and Evolvable Machines*, 6(2): 163–190, 2005.

[8] I. Radziukyniene, A. Žilinskas, Evolutionary Methods for Multi-Objective Portfolio Optimization, *Proceedings of the World Congress on Engineering*, 2(1): 1155–1159, 2008.

[9] J. Andersson, *A Survey of Multiobjective Optimization in Engineering Design*, 2000.

[10] T. Bäck, U. Hammel, H. P. Schwefel, Evolutionary computation: comments on the history and current state, *IEEE Transactions on Evolutionary Computation*, 1(1): 3 –17, 1997.

[11] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, ZDA, 1975.

[12] De Jong, K. A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, doktorska disertacija, 1975.

[13] N. Veček, M. Mernik, M. Črepinšek, A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms, *Information Sciences*, 277(1): 656—679, 2014.

[14] M. E. Glickman, A. C. Jones, Rating the chess rating system, *Chnace*, 12(2): 21—28, 1999.

[15] M. E. Glickman, A comprehensive guide to chess ratings, *American Chess Journal*, 3(1): 59—102, 1995.

[16] M. Črepinšek, *Evolutionary Algorithms Rating System* (GitHub). Dostopno na naslovu: `https://github.com/matejxxx/EARS`, 2015.

[17] A.J. Nebro, J.J. Durillo, *jMetal: A Java framework for multi-objective optimization* (Sourceforge). Dostopno na naslovu: `http://jmetal.sourceforge.net`, 2015.

[18] K. Deb, S. Agrawal, A. Pratab in T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *Parallel Problem Solving from Nature (PPSN VI)*, 849–858, 2000.

[19] N. Srinivas in K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation*, 2(3): 221–248, 1994.

[20] E. Zitzler, M. Laumanns, L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, 95–100, 2001.

[21] J. Knowles, D. Corne, The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation, *Proceedings of the Congress of Evolutionary Computation*, 98–105, 1999.

[22] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 283–290, 2001.

[23] D. W. Corone, J. D. Knowles, M. J. Oates, The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization, *Conference on Parallel Problem Solving from Nature (PPSN VI)*, 839–848, 2000.

[24] Q. Zhang, W. Liu, H. Li, The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances, *IEEE Congress on Evolutionary Computation (CEC' 2009)*, 203–208, 2009.

[25] K. Miettinen, Nonlinear Multiobjective Optimization, *Kluwer Academic Publishers*, 1999.

[26] S. Kukkonen and J. Lampinen, GDE3: The third Evolution Step of Generalized Differential Evolution, *IEEE Congress on Evolutionary Computation (CEC' 2005)*, 443–450, 2005.

[27] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition, *Technical Report CES-487*, 2009.

[28] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *J. Am. Statist. Assoc.*,75–701, 1937.

[29] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, *Ann. Math. Statist.*, 86–92, 1940.

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

IZJAVA O AVTORSTVU

Spodaj podpisani/-a Miha Ravber

z vpisno številko E5013409

sem avtor/-ica magistrskega dela z naslovom:

Uporaba šahovskega sistema rangiranja za primerjavo evolucijskih algoritmov večkriterijske optimizacije

(naslov magistrskega dela)

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

doc. dr. Matej Črepinšek

in somentorstvom (naziv, ime in priimek)

- so elektronska oblika magistrskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela.
- soglašam z javno objavo elektronske oblike magistrskega dela v DKUM.

V Mariboru, dne 26.08.2015

Podpis avtorja/-ice:

Ravber

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

IZJAVA O USTREZNOSTI ZAKLJUČNEGA DELA

Podpisani mentor :

doc. dr. Matej Črepinšek

(ime in priimek mentorja)

in somentor (eden ali več, če obstajata):

_____

(ime in priimek somentorja)

Izjavljam (-va), da je študent

Ime in priimek: Miha Ravber

Vpisna številka: E5013409

Na programu: Računalništvo in informacijske tehnologije (MAG)

izdelal zaključno delo z naslovom:

Uporaba šahovskega sistema rangiranja za primerjavo evolucijskih algoritmov večkriterijske optimizacije

(naslov zaključnega dela v slovenskem in angleškem jeziku)

A Chess Rating System for the Comparison of Multi-Objective Evolutionary Algorithms

v skladu z odobreno temo zaključnega dela, Navodilih o pripravi zaključnih del in mojimi (najinimi oziroma našimi) navodili.

Preveril (-a, -i) in pregledal (-a, -i) sem (sva, smo) poročilo o plagiatorstvu.

Datum in kraj:
28. 8. 2015, Celje

Podpis mentorja:

Datum in kraj:

Podpis somentorja (če obstaja):

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek avtorja-ice: Miha Ravber

Vpisna številka: E5013409

Študijski program: Računalništvo in informacijske tehnologije (MAG)

Naslov zaključnega dela: Uporaba šahovskega sistema rangiranja za primerjavo evolucijskih algoritmov večkriterijske optimizacije

Mentor: doc. dr. Matej Črepinšek

Somentor:

Podpisani-a Miha Ravber izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Zaključno delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija zaključnega dela je istovetna z elektronsko verzijo elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, varstva industrijske lastnine ali tajnosti podatkov naročnika: _____ ne sme biti javno dostopno do _____(datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela).

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum zaključka študija, naslov zaključnega dela), na spletnih straneh in v publikacijah UM.

Datum in kraj: 26.08.2015, Maribor Podpis avtorja-ice: Ravber

Podpis mentorja: _____
(samo v primeru, če delo ne sme biti javno dostopno)

Podpis odgovorne osebe naročnika in žig:
(samo v primeru, če delo ne sme biti javno dostopno) _____