SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI SEMINAR

**VIŠEKRITERIJSKA OPTIMIZACIJA KORIŠTENJEM GENETSKIH ALGORITAMA**

Mia Primorac

Zagreb, svibanj 2013.

Sadržaj

[Uvod 1](#_Toc355556769)

[1. Definiranje problema višekriterijske optimizacije 2](#_Toc355556770)

[1.1. Jednokriterijska optimizacija 2](#_Toc355556771)

[1.2. Temeljne razlike višekriterijske u odnosu na jednokriterijsku optimizaciju 2](#_Toc355556772)

[1.3. Svođenje višekriterijske na jednokriterijsku optimizaciju 4](#_Toc355556773)

[1.3.1. Preslikavanje vektora dobrote u jednodimenzijski prostor 4](#_Toc355556774)

[1.3.2. Definiranje prioriteta među kriterijima 5](#_Toc355556775)

[1.4. Pareto optimalnost 5](#_Toc355556776)

[1.4.1. Relacija dominacije 6](#_Toc355556777)

[1.4.2. Globalni optimum višekriterijske optimizacije 6](#_Toc355556778)

[2. Genetski algoritmi višekriterijske optimizacije 8](#_Toc355556779)

[2.1. Klasični genetski algoritmi jednokriterijske optimizacije 8](#_Toc355556780)

[2.2. Nedominirano sortiranje 10](#_Toc355556781)

[2.2.1. Klasično nedominirano sortiranje 10](#_Toc355556782)

[2.2.2. Ubrzano nedominirano sortiranje 12](#_Toc355556783)

[2.3. Pregled poznatijih genetskih algoritama višekriterijske optimizacije 13](#_Toc355556784)

[2.3.1. VEGA 13](#_Toc355556785)

[2.3.2. MOGA 15](#_Toc355556786)

[2.3.3. NPGA 16](#_Toc355556787)

[2.3.4. SPEA 16](#_Toc355556788)

[2.3.5. NSGA 16](#_Toc355556789)

[2.3.6. NSGA-II 19](#_Toc355556790)

[2.3.7. PPES 21](#_Toc355556791)

[3. Implementacija i rezultati 22](#_Toc355556792)

[Zaključak 24](#_Toc355556793)

[Literatura 26](#_Toc355556794)

.

# Uvod

Mnogi su problemi iz stvarnog svijeta takvi da zahtijevaju optimizaciju istovremeno po više kriterija. U principu, višekriterijska je optimizacija poprilično različita od jednokriterijske optimizacije gdje je cilj pronaći jedan jedini globalni minimum ili maksimum. Razmatrajući problem višekriterijske optimizacije može se uočiti da u općenitom slučaju ne postoji jedinstveno rješenje koje je globalni minimum svih kriterija. Ako takvo rješenje postoji, to onda nije višekriterijski, već jednokriterijski problem. Dakle, uobičajeno je, kada se priča o višekriterijskoj optimizaciji, govoriti o optimalnom rješenju kao o skupu rješenja koja dominiraju nad ostalim rješenjima gledajući kolektivno sve kriterije, a ne svaki zasebno. Za ta rješenja poželjno je da su međusobno što različitija gledajući prostor rješenja jer je zbog prirode problema potrebno pronaći kvalitetna rješenja koja su na različite načine postigla tu kvalitetu te će korisnik koji ima problem sam odabrati kompromis koji mu odgovara. Kompromis je ključan jer sva bi rješenja iz optimalnog skupa rješenja bila lošija kada bi se usporedila, po samo jednom od kriterija, s nekim rješenjem dobivenim jednokriterijskom optimizacijom po tom istom kriteriju.

Genetski su se algoritmi od svog začetka prije više od tri desetljeća koristili za jednokriterijsku optimizaciju. Zbog toga što istovremeno rade sa čitavom populacijom rješenja, a ne samo jednim rješenjem, vrlo su prikladni i za problem višekriterisjke optimizacije jer, kao što je već napomenuto, kao optimalno rješenje traži se upravo skup rješenja. Stoga su vrlo brzo svi algoritmi višekriterijske optimizacije počeli koristiti razne evolucijske strategije.

Višekriterijsku optimizaciju ne može se razumjeti ako ne se razumije problem jednokriterisjke optimizacije, stoga je prvo razjašnjen taj pojam. Cilj ovog rada je razmotriti sličnosti i razlike jednokriterijske i višekriterijske optimizacije, definirati terminologiju te opisati par zanimljivih algoritama od kojih su neki više od povijesnog značaja kada se promatra kako je tekao razvoj višekriterijske optimizacije te kako je kasnije sve više algoritama poteklo u „smjeru nedominacije“ kada se ispostavilo da je on dobar, dok su drugi zanimljivi jer se često koriste, jednostavni su i/ili daju dobre rezultate. Na kraju će se predstaviti predstojeća implementacija za isprobavanje nekih od navedenih algoritama.

# Definiranje problema višekriterijske optimizacije

## Jednokriterijska optimizacija

Da bi pojam višekriterijske optimizacije bio u cjelosti razjašnjen, potrebno je prvo definirati jednokriterijsku optimizaciju. Kao što sam naziv govori, problem jednokriterijske optimizacije svodi se na pronalaženje najboljeg rješenja u ulaznom prostoru rješenja takvog da je vrijednost jedinstvenog kriterija tj. jedinstvene funkcije cilja *f* u tom rješenju optimalna u odnosu na vrijednost funkcije u svim ostalim rješenjima, s tim da traženo rješenje mora i zadovoljavati zadana ograničenja.

Neka je *X* n-dimenzijski prostor svih mogućih rješenja ***x*** oblika ***x*** *= {x1, x2,…, xn}, xi ϵ* , i neka su zadana ograničenja u obliku skupa nejednakosti *{gi(****x****) ≥ 0, i = 1, 2,…, I}*, skupa jednakosti *{hj(****x****) = 0, j = 1, 2,…, J}* te skupa ograničenja *{xkmin ≤ xk ≤ xkmax, k = 1, 2,…, n}* kojim se određuje minimalna i maksimalna dozvoljena vrijednost pojedine dimenzije rješenja. Funkcija cilja *f* definirana je kao preslikavanje *f : X →* . Ne smanjujući općenitost, u daljnjem će se tekstu pretpostaviti da se uvijek radi minimizacija ciljne funkcije.

Cilj jednokriterijske optimizacije je pronaći globalni optimum ***x****\** takav da je *f(****x****\*)* najmanja vrijednost koju funkcija *f* može poprimiti uz zadana ograničenja.
Dobrota rješenja definirana je kao skalar *z* pridružen svakom rješenju ***x****i*. Ako su ***x****1* i ***x****2* iz *X* i zadovoljavaju ograničenja te ako je *f(****x****1) < f(****x****2)*, tada za njima pridružene dobrote *z1* i *z2* nužno vrijedi da je *z1 > z2* te možemo reći da je rješenje ***x****1* bolje od rješenja ***x****2*. [3]

## Temeljne razlike višekriterijske u odnosu na jednokriterijsku optimizaciju

Zajedničke stvari problemima jednokriterijske i višekriterijske optimizacije su
n-dimenzijski ulazni prostor rješenja te prethodno navedena ograničenja nad rješenjima iz tog prostora. Temeljna razlika višekriterijske optimizacije u odnosu na jednokriterijsku jest to da više nemamo samo jedan kriterij (ciljnu funkciju *f*) po kojem vrednujemo rješenja, već M kriterija, odnosno M funkcija cilja {*fm(****x****), m = 1, 2,…,M}*. Ako neke od funkcija *fi* treba minimizirati, a ostale maksimizirati, to ne predstavlja problem s obzirom da ako funkciju *fj* treba minimizirati, a funkciju *fk* maksimizirati, kriterije možemo definirati tako da minimiziramo *fj*, ali *fk* ne maksimiziramo, već minimiziramo izmijenjenu funkciju cilja *–fk*. U prijevodu, problem minimizacije jednih, a maksimizacije drugih kriterija može se svesti na problem samo minimizacije ili samo maksimizacije svih M kriterija.

Glavni problem kod višekriterijske optimizacije leži u nečem drugom, a to je nemogućnost usporedbe kvalitete rješenja. Kod jednokriterijske optimizacije to je bilo jednostavno jer se svodilo na usporedbu dvaju skalara. Sada se rješenja iz ulaznog prostora rješenja (engl. *solution space*) ne preslikavaju više u skalarne vrijednosti u prostoru ciljnih funkcija (engl. *objective space*), već u M-dimenzijske vektore dobrote ***z(i)*** *= (f1(****x****(i)), f2(****x****(i)),…, f­M(****x****(i))) = (z1, z2…, zM)*, a njih se ne može uvijek jednoznačno usporediti. Na slici 1-1 nalazi se prikaz preslikavanja rješenja iz prostora rješenja u prostor ciljnih funkcija. Ako su rješenjima ***x****(1)* i ***x****(2)* pridružena dva vektora dobrote ***z****(1)* i ***z****(2)* takva da su sve komponente *z1,…,zM* vektora ***z****(1)* veće od odgovarajućih komponenti u vektoru ***z****(2)*, tada možemo zaključiti da je rješenje ***x****(1)* bolje od rješenja **x**(2). No, općenito govoreći, ako su neke komponente veće u jednom vektoru, a preostale u drugom vektoru, ne možemo reći da je neko rješenje bolje. Stoga je potrebno definirati način usporedbe vektora te će neki od njih biti opisani u nastavku.

Svi načini usporedbe vektora prema kvaliteti temelje se na jednom od tri principa od kojih prva dva problem svode na jednokriterijsku optimizaciju.. Takva rješenja otklanjaju navedeni problem s uspoređivanjem vektora, no često nisu pogodna za probleme u kojima kao konačno rješenje želimo ponuditi čitavu paletu potencijalno odgovarajućih rješenja i time korisniku omogućiti da sam odabere ono koje mu najviše odgovara. Da bi to bilo moguće, potreban je treći način razmišljanja koji ne svodi probleme na jednokriterijsku optimizaciju, a to omogućavaju relacija dominacije i pojam Pareto optimalnosti koji su srž višekriterijske optimizacije.



Slika 1‑1 Prostor rješenja i prostor ciljnih funkcija

## Svođenje višekriterijske na jednokriterijsku optimizaciju

Prednost ovakvog principa jest jednostavnost i mogućnost odabira između brojnih (genetskih) algoritama jednokriterijske optimizacije. No, problemi s ovakvim rješenjem su mnogobrojni, inače uopće ne bi bilo potrebe za višekriterijskom optimizacijom i algoritmima višekriterijske optimizacije.

### Preslikavanje vektora dobrote u jednodimenzijski prostor

Preslikavanjem vektora dobrote ***z*** u jednodimenzijski prostor on postaje skalar *z*, a problem višekriterijske optimizacije svodi se na problem jednokriterijske optimizacije. Nakon takve transformacije problem možemo riješiti nekim algoritmom za jednokriterijsku optimizaciju.

Prvi i najjednostavniji način preslikavanja jest pomoću linearne kombinacije pojedinih komponenata vektora***z***. Težinskim koeficijentima ωi regulira se koliki će utjecaj pojedini kriterij imati u konačnom iznosu dobrote z:

Mogu se definirati i razna nelinearna preslikavanja vektora u skalar pomoću nelinearne funkcije Ψ:

Koje god preslikavanje odabrali, sva ona imaju jednaka svojstva. Pozitivne strane ovakvog rješenja problema jesu jednostavnost te mogućnost korištenja mnogobrojnih algoritama za jednokriterijsku optimizaciju. Prvi problem je to da takvo preslikavanje nije injektivno, odnosno različiti vektori mogu se preslikati u iste skalarne vrijednosti. U slučaju da imamo dva kriterija, neće biti moguće razlikovati rješenje koje ima jedan kriterij izuzetno dobar, a drugi izuzetno loš, od rješenja koje ima prosječne vrijednosti za oba kriterija. Drugi problem je to da se optimizacijom uvijek dobije samo jedno konačno rješenje. Razmatrajući prirodu problema, to u većini slučajeva nije zadovoljavajuće. [3]

### Definiranje prioriteta među kriterijima

Kvalitetu rješenja iz ulaznog prostora rješenja možemo usporediti i tako da kriterije sortiramo po važnosti te vektore dobrote uspoređujemo komponentu po komponentu počevši od najznačajnije. Neka je ***x****(1)* rješenje kojemu je pridružen vektor ***z****(1) = {z1(1), z2(1),…, zM(1)}*, te neka je ***x****(2)* rješenje kojemu je pridružen vektor ***z****(2) = {z1(2), z2(2),…, zM(2)}* u prostoru ciljnih funkcija. Pri tome se na prvoj komponenti *z1(i)* nalazi vrijednost najvažnije funkcije cilja, a na M-toj komponenti *zM(i)* nalazi se vrijednost najmanje važne funkcije cilja. Ako se vektori podudaraju u svim komponentama, jednaki su. Ako se ***z****(1)* i ***z****(2)* podudaraju u prvih *j-1* komponenti, a u *j*-oj se komponenti razlikuju, tada je bolje ono rješenje ***x****(i)* čiji pridruženi vektor ***z****(i)* ima veću komponentu *zj(i)*, odnosno ako je *zj(1) > zj(2)* tada je bolje rješenje ***x****(1)*, a inače je bolje rješenje ***x****(2)*.[3]

Efektivno, rezultat ovakve usporedbe vektora jednak je kao kod preslikavanja u jednodimenzionalni prostor ciljnih funkcija i vrijede jednake prednosti i mane. Problem se indirektno svede na problem jednokriterijske optimizacije i nisu potrebni algoritmi za višekriterijsku optimizaciju. Optimizacija rezultira jednim konačnim rješenjem s kojim korisnik možda nije pretjerano zadovoljan. Također, ako su neki kriteriji ravnopravni to nije moguće definirati ovakvom usporedbom.

## Pareto optimalnost

Prethodno je obrazloženo zašto nije dobro višekriterijske probleme svesti na jednokriterijske, no ostalo je otvoreno pitanje kako procijeniti koje je rješenje (naj)kvalitetnije, a da se takvo što ne napravi.

### Relacija dominacije

Svaki optimizacijski algoritam mora moći procijeniti kvalitetu rješenja te na neki način rješenja sortirati po kvaliteti. Na putu ka konačnom rješenju problema potrebno je za početak definirati relaciju „dominantan nad“ za dva vektora.

Ako je ***x****(1)* rješenje kojemu je pridružen vektor dobrote ***z****(1) = {z1(1), z2(1),…, zM(1)}*, te ako je ***x****(2)* rješenje kojemu je pridružen vektor dobrote ***z****(2) = {z1(2), z2(2),…, zM(2)}* u prostoru ciljnih funkcija, tada je ***x(****1)* „dominantan nad“ ***x****(2)* ako i samo ako su sve komponente vektora ***z****(1)* veće ili jednake od odgovarajućih komponenti u vektoru ***z****(2)* te ako je barem jedna komponenta *zi(1)* strogo veća od odgovarajuće komponente *zi(2)* u ***z****(2)*. Od tri matematička svojstva relacija (refleksivnost, simetričnost, tranzitivnost), relacija „dominantan nad“ ima samo svojstvo tranzitivnosti jer vektor nije dominantan sam nad samim sobom pa nema refleksivnost, a ako je ***z****(1)* dominantan nad ***z****(2)* onda ***z****(2)* ne može biti dominantan nad ***z****(1)* pa ne vrijedi niti simetričnost. Tranzitivnost vrijedi jer ako je ***z****(1)* dominantan nad ***z****(2)* te ako je ***z****(2)* dominantan nad ***z****(3)*, onda je i ***z****(1)* dominantan nad ***z****(3)*.

Pomoću relacije dominacije kvaliteta rješenja može se mjeriti tako da se za sve moguće parove rješenja provjeri čiji je ciljni vektor dominantan. Broj rješenja koja dominiraju nad nekim rješenjem ***x*** naziva se rang rješenja. Stoga je očito da je od dva rješenja bolje ono koje ima manji rang. Ako se populacija sastoji od *N* rješenja, teoretski bi trebalo puta provjeriti relaciju dominacije nad dva vektora što je računski zahtjevno za malo veći broj *N*. [3]

### Globalni optimum višekriterijske optimizacije

Nakon što je nad svim mogućim parovima rješenja provjereno koje rješenje dominira, podskup rješenja nad kojim nitko ne dominira može se odvojiti od ostalih rješenja nad kojima barem jedno rješenje dominira. Takav podskup rješenja naziva se nedominirani skup. Globalni optimum u smislu višekriterijske optimizacije više nije definiran kao samo jedno rješenje kao kod jednokriterijske optimizacije, već kao nedominirani skup unutar skupa svih prihvatljivih rješenja. Takav globalni optimum naziva se globalni Pareto optimalni skup i konačni cilj algoritama višekriterijske optimizacije je pronaći upravo taj skup.

Ako takav skup ne postoji, ili ima suviše malo elemenata, poželjno je cijelu populaciju rješenja podijeliti u podskupove prema tome koliko su „blizu“ optimalnom skupu. Takvi podskupovi nazivaju se *k*-te Pareto fronte, s tim da je prva Pareto fronta upravo sam nedominirani skup, druga Pareto fronta je nedominirani skup unutar skupa rješenja bez prvog nedominiranog skupa, treća Pareto fronta je nedominirani skup unutar skupa rješenja bez prve dvije fronte itd. Postupak završava kada sva rješenja budu raspoređena u neku od fronti. Takvo sortiranje rješenja u podskupove naziva se nedominirano sortiranje i koristi se u nekoliko algoritama koji će biti opisani u nastavku.

Postoji problem s računalnom složenošću tako implementiranog nedominantnog sortiranja, koja je čak *O(MN3)* ako je M broj kriterija i N veličina populacije.

Na slici 1-2 prikazana su rješenja u prostoru ciljnih funkcija te optimalna Pareto fronta.



Slika 1‑2 Pareto fronta[8]

# Genetski algoritmi višekriterijske optimizacije

Pošto je optimalno rješenje problema višekriterijske optimizacije skup rješenja, prirodno je da će se takav problem rješavati algoritmima iz porodice genetskih algoritama koji već sami po sebi rade sa čitavom populacijom rješenja. Svi genetski algoritmi sastoje se od jednakih elemenata, stoga će prvo biti objašnjen jednostavan genetski algoritam za jednokriterijsku optimizaciju te će na tom općenitom primjeru biti objašnjeni osnovni pojmovi vezani uz genetske algoritme.

## Klasični genetski algoritmi jednokriterijske optimizacije

Genetski algoritmi općenito su heurističke metode optimiranja, odnosno to su načini pretrage prostora mogućih rješenja koji kombiniraju direktno i stohastičko traženje rješenja, obrađuju populaciju rješenja jednostavnim računskim operacijama nad rješenjima te uz pomoć evolucije traže globalni optimum zadane funkcije cilja. Promatrajući prirodu te proces evolucije živih bića, razvio se istoimeni proces pretraživanja prostora rješenja. Kao što se sve populacije živih bića postepeno prilagođavaju prirodi oko sebe svojim anatomskim i drugim obilježjima, tako se i u genetskim algoritmima rješenja iz populacije prilagođavaju tako da se iz generacije u generaciju približavaju optimumu funkcije cilja. Nositelj genetskog materijala u prirodi su molekule DNK (deoksiribonukleinske kiseline), dok su u genetskim algoritmima to rješenja prikazana nizom bitova, nizom brojeva ili nekom složenijom strukturom. Analogija dvaju evolucija, kao procesa u prirodi i kao genetskog algoritma, najbolje se očituje u operatorima selekcije, mutacije i križanja. Prirodna selekcija vrši se tako da uglavnom samo najprilagođenije jedinke preživljavaju i razmnožavaju se s drugim jedinkama te dio svog genetskog materijala prenose na potomstvo. Čin razmnožavanja svodi se na kombiniranje genetskog materijala jedinki koje su prošle prirodnu selekciju te stvaranje novih jedinki. No, da nema i treće komponente u evoluciji, svijet bi još uvijek izgledao kao prije par milijardi godina. Ta treća komponenta jest mutacija koja unosi novinu u genetski materijal i potencijalno čini jedinke još prilagođenijima od svojih roditelja. Analogno prirodnoj selekciji, operatori selekcije uglavnom biraju što bolje jedinke, ali ne isključivo samo takve da se ne suzi prostor pretraživanja, da se ne napravi preveliki selekcijski pritisak koji vodi ravno u lokalne optimume i da se izbjegne genetski drift, odnosno gubitak genetske raznolikosti. Neki od često korištenih operatora selekcije su proporcionalna selekcija (engl. *Roulette-wheel selection*), turnirska selekcija (engl. *Tournament selection*), selekcija linearnim rangiranjem (engl. *linear ranking selection*) i mnogi drugi. Operatori križanja na razne načine kombiniraju genetski materijal, a ako su rješenja prikazana nizom bitova najčešće se koristi uniformno križanje i križanje s *k* točaka prekida. Operator mutacije nad rješenjem prikazanim nizom bitova najčešće se izvodi tako da se s određenom vjerojatnošću invertira svaki bit rješenja. Potrebno je uočiti da je izbor operatora velik i da ne postoji univerzalni genetski algoritam koji je dobar za sve probleme. Najbolji prikaz rješenja i operatori koji se koriste ovise o tipu problema koji se rješava. No, niti svi operatori iz iste porodice (npr. operatori križanja rješenja prikazanih nizom bitova) nisu dobri za svaki problem, stoga je najbolje eksperimentirati s više njih prije konačne odluke. [2][3]

Prema vrsti selekcije genetske algoritme dijelimo na generacijske i eliminacijske. Generacijski genetski algoritam radi s dvije populacije u jednoj iteraciji, starom i novom, s time da se nova radi kao kopija nekih jedinki iz stare populacije. Eliminacijski genetski algoritam u jednoj iteraciji radi sa samo jednom populacijom iz koje se eliminiraju neke jedinke i nadoknađuju se novima. Pošto su svi algoritmi višekriterijske optimizacije koji će se u radu razmatrati iz porodice generacijskih algoritama, u nastavku je dan pseudokod takvog algoritma [3].

Generacijski genetski algoritam{
P = stvori\_početnu\_populaciju();

evaluiraj(P);

ponavljaj\_dok\_nije\_kraj{

nova\_populacija P' = *Ø*;

ponavljaj\_dok\_je (veličina(P') < veličina(P)){

odaberi R1 i R2 iz P;

{D1, D2} = križaj(R1, R2);

mutiraj(D1); mutiraj(D2);

dodaj D1 i D2 u P';

}

P = P'; //nova populacija postaje stara

}

ispisi\_najbolje\_rješenje();

}

Pseudokod 2.1. – generacijski genetski algoritam

U ovako definiranom algoritmu može se dogoditi da se u nekoj od generacija pojavi optimalno rješenje, ali ono bude križano s nekim lošijim i izgubi se. Uvođenje elitizma u genetski algoritam osigurava da se u svakoj generaciji još dodatno pamti trenutno najbolje rješenje, stoga je sigurno da do gubitka najboljeg rješenja neće doći. Problem koji nastaje uvođenjem elitizma jest velika složenost operacije sortiranja velikih populacija.

## Nedominirano sortiranje

Nedominirano sortiranje jest algoritam koji je sastavni dio mnogih algoritama za višekriterijsku optimizaciju, stoga će biti detaljnije opisan. Kao što je već rečeno, nedominirano sortiranje je postupak kojim se rješenja iz populacije rasporede u podskupove, takozvane fronte, prema tome koliko su blizu optimalnom nedominiranom skupu rješenja, odnosno skupu rješenja čiji je rang 0.

### Klasično nedominirano sortiranje

Za određivanje nedominiranog skupa u svakom koraku algoritma sortiranja trebao bi se koristiti nešto ubrzani algoritam koji ne određuje dominaciju nad svih parova rješenja, već se odluka da rješenje ***x*** ne pripada nedominiranom skupu *P'* donosi čim se naiđe na prvo rješenje koje nad ***x*** dominira.

Sam algoritam opisan je pseudokodom:

Nedominirano\_sortiranje\_1{

 P = dohvati\_populaciju\_rješenja();

 i = 1;

 ponavljaj\_dok\_je (veličina\_populacije(P) > 0){

 P' = nedominirani\_skup(P);

 P' pohrani kao i-tu frontu;

 P = P \ P'; //izbaci nedominirani skup

 i++;

 }

}

/\* traženje nedominiranog skupa \*/

nedominirani\_skup(Populacija P){

 P' = {};

 za svako rješenje x iz populacije P{

 zastavica dominiran = false;

 za svako rješenje y iz populacije P{

 ako (y dominantan nad x){

 dominiran = true;

 prekini\_petlju;

 }

}

Ako(dominantan = false){

 P' += x; //dodaj rješenje u nedominirani skup

}

 }

 vrati P';

}

Pseudokod 2.2 – klasično nedominirano sortiranje

Problem ovog algoritma jest njegova računska složenost. Samo funkcija nedominirani\_skup(P) ima složenost O(MN2) i ona se u najgorem slučaju može pozvati N puta, ako su sva rješenja ranga 1, a samo jedno rješenje je ranga 0 tj. ako je dominantnost ulančana tako da nad rješenjem u prvoj fronti nije nijedno rješenje dominantno, a nad svim ostalima je dominantno točno jedno rješenje. Složenost postupka je dakle O(MN3), što je puno previše ako uzmemo u obzir da genetski algoritmi obično rade sa populacijama reda veličine nekoliko stotina pa i tisuća jedinki.

### Ubrzano nedominirano sortiranje

Pošto je prethodno opisani algoritam jako spor, autor knjige [1] Kalyanmony Deb, predložio je nešto brži algoritam koji se koristi u svim genetskim algoritmima višekriterijske optimizacije čiji je on suautor, a i u drugima.

Nedominirano\_sortiranje\_2 {

za (rješenje i iz populacije P) {

 ni=0; Si=0;

 za(rješenje *ϵ* P, j!=i){

 ako(i dominira nad j){

 dodaj j u skup Si;

 }

 inače{

 ni = ni + 1;

 }

 }

 ako(ni == 0){

 zadrži\_rješenje\_u\_nedominiranoj\_fronti(i)

 }

 }

 }

 k = 1; //brojač fronti

 dok\_je(Pk != prazan\_skup){

 Q = 0; //sljedeća nedominirana fronta

 za (i *ϵ* Pk && j *ϵ* Si){

 nj = nj -1;

 ako(nj == 0){

 dodaj j u trenutnu frontu Q;

 }

 }

 k = k + 1;

 Pk = Q;

 }

}

##  Pregled poznatijih genetskih algoritama višekriterijske optimizacije

Nakon što je primijećeno da postoje problemi koji su po svojoj prirodi višekriterijski, počeli su se razvijati razni algoritmi koji su ih trebali riješiti. Najuspješnijima su se pokazali algoritmi iz porodice genetskih algoritama. U ovom poglavlju slijedi pregled nekih značajnijih algoritama od kojih se neki još uvijek intenzivno koriste, dok su drugi navedeni kao primjer lošeg pristupa problemu višekriterijske optimizacije.

### VEGA

Algoritam VEGA (Vector Evaluated Genetic Algorithm) objavljen je 1984. godine od autora Davida Schaffera. Problem višekriterijske optimizacije Schaffer je pokušao riješiti tako da je uzeo osnovni genetski algoritam (simple genetic algorithm, SGA) i promijenio jedino operator selekcije. Početna populacija od N' jedinki koja se optimizira prema M kriterija, evaluira se zasebno za svaki od M kriterija. Zatim se dijeli na M podpopulacija tako da u svakoj podpopulaciji bude N / M jedinki koje su odabrane iz čitave populacije proporcionalnom selekcijom na temelju nekog (jednog) kriterija. Nakon podjele u podpopulacije sve se jedinke iz podpopulacija okupljaju u populaciju veličine N nad kojom se dalje obavljaju klasični operatori križanja i mutacije.

Glavna prednost algoritma jest njegova jednostavnost, kako razumijevanja tako i implementacije. Međutim, algoritam ima pregršt mana.

Primijećeno je da rješenja dobivena ovim algoritmom nisu nedominirana u punom smislu te riječi, već je to ograničeno samo na trenutnu populaciju, stoga takva rješenja nisu i globalno nedominirana, već zasigurno postoje rješenja koja nad njima dominiraju zbog nespretnog načina selekcije. Naime, pošto se prednost daje onim rješenjima koja su u nekom kriteriju izuzetno dobra, a ostale se kriterije niti ne gleda, rješenja koja su prosječno dobra po svim kriterijima imat će puno manju šansu biti odabrana od onih koji su dobri u jednom kriteriju, a u ostalima teoretski mogu biti jako loša. Takav način nije u skladu s idejom višekriterijske optimizacije stoga se ne koristi često.
Autor algoritma naknadno je pokušao smanjiti pristranost svog algoritma dvjema heuristikama. Prva heuristika je nedominirana selekcija (*non-dominated selection heuristic*) koja kažnjava globalno dominirane jedinke tako da im vjerojatnost odabira u proporcionalnoj selekciji bude manja. Time nije puno postignuto jer su nedominirane jedinke postale prava rijetkost i imale su ogromnu vrijednost dobrote, a time je postignut velik selekcijski pritisak što značajno smanjuje prostor pretrage. Druga heuristika „*mate selection*“ odabire nasumičnu jedinku iz populacije, a pored nje odabire i njezin par, a to je ona jedinka koja je najviše udaljena od prve jedinke (gledajući euklidsku udaljenost). No, takva selekcija rezultirala je još lošijim jedinkama.

Zaključak je da je algoritam VEGA zanimljivo proučiti kao početak višekriterijske optimizacije, ali nikako nije algoritam koji bi danas vrijedilo koristiti. [4][6]



Slika 2‑1 Dijagram toka algoritma VEGA

### MOGA

Autori Fonesca i Fleming 1993. su godine objavili algoritam MOGA (Multi-Objective Genetic Algorithm). Oni su se vodili idejom Pareto-rangiranja (objašnjeno u poglavlju 1.4.1) koju je Goldberg predložio 1989. godine. Rang rješenja definira se kao broj jedniki koje dominiraju nad tim rješenjem. Vođeni tom idejom, autori su osmislili sljedeći način dodjele dobrote rješenjima:

1. Rješenja se podijele u Pareto-fronte i dodijele im se rangovi prema broju rješenja koja nad njima dominiraju.
2. Populacija se sortira po rangu.
3. Dobrota se dodjeljuje tako da rješenja s najboljim (najmanjim) rangom (0) imaju najveću vrijednost dobrote, a ona s najlošijim (najvećim) rangom imaju najmanju vrijednost dobrote. Interpolacija dobrote između početne i krajnje vrijednosti može se raditi npr. linearno.
4. Rješenjima s jednakim rangom izjednači se dobrota, ali tako da svatko od njih dobije prosječnu vrijednost.

Da bi se izbjegao preveliki selekcijski pritisak i prerana konvergencija, dodatno se koristi „*Niche formation*“ metoda za raspoređivanje rješenja po Pareto optimalnom području, no raspored se ne radi u prostoru rješenja, već u prostoru ciljnih funkcija i tu veliku ulogu igra parametar σshare. Kasnije će biti rečeno da poznati algoritam NSGA, za razilku od
MOGA-e, raspored dobrote radi u prostoru rješenja i to je jedan od razloga zašto je NSGA manje sklon preuranjenoj konvergenciji od MOGA-e. Ako na udaljenosti σshare od promatranog rješenja preslikanog u prostor ciljnih funkcija nema niti jedno drugo rješenje, bit će mu dodijeljena maksimalna dobrota. Što je više jedinki na okupu i što su one bliže, to su one manje dragocjene jer je njihova šansa da budu odabrane veća pa im se faktor dobrote smanjuje.

Prednosti ovakvog algoritma su njegova jednostavnost i dobra učinkovitost, svakako bolja nego kod VEGA-e, u smislu kvalitete dobivenih rješenja. Performanse algoritma uvelike ovise o parametru σshare. Algoritam nije pretjerano brz, pošto se selekcija obavlja nad čitavom populacijom.[6]

### NPGA

Algoritam NPGA (*Niched Pareto Genetic Algorithm*) zasnovan je na Pareto dominacijskom turniru (engl. Pareto domination tournament) i dijeljenju dobrote unutar klasa ekvivalencije. Vrlo je brz jer ne primjenjuje Pareto selekciju na čitavu populaciju (kao npr. MOGA), već se selekcija bazira na turnirskoj selekciji (prilagođenoj višekriterijskoj optimizaciji). Rješenja koja daje su vrlo dobre kvalitete, no zahtjevan je za implementaciju.[6]

### SPEA

Algoritam SPEA (*Strength Pareto Evolutionary Algorithm*) značajan je jer prvi predlaže korištenje elitizma i nedominacije zajedno te pokazuje njegov utjecaj na kvalitetu dobivenih rješenja. Kao i MOGA, vrlo je osjetljiv na parametre. Značajni parametri su σshare te tdom (veličina turnira). Ispostavilo se da algoritam u biti ne konvergira ka pravim Pareto optimalnim rješenjima ako Pareto optimalna fronta nije konveksna zbog načina na koji je dodijeljena dobrota u ovom algoritmu. [6]

### NSGA

Algoritam NSGA (Non-dominated Sorting Genetic Algortihm) kombinira na svoj način koncept nedominacije i nedominiranog sortiranja te dijeljenja dobrote. Od klasičnog genetskog algoritma razlikuje se samo u načinu na koji je izveden operator selekcije, dok križanje i mutacija ostaju jednaki. Prije same selekcije, jedinke populacije se nedominirano sortiraju kao što je opisano u poglavlju 2.2.2 (koristi se ubrzana inačica nedominiranog sortiranja). U prvoj iteraciji tog algoritma pronalaze se rješenja nad kojima nitko ne dominira i svima im se dodijeli neka strašno velika dobrota (npr. broj jedinki u populaciji), jednaka za sve. Kako bi se održala raznolikost populacije, nad tako evaluiranim jedinkama radi se podjela dobrote . Podjela dobrote, kao i u prethodnim algoritmima koji su je koristili, postiže se tako da se vrijednost dobrote umanji proporcionalno broju jedinaka oko nje. Tako evaluirane jedinke zanemaruju se u idućoj iteraciji, te se postupak ponavlja i za iduću frontu samo što je inicijalna količina dobrote dodijeljena rješenjima nužno manja od minimalne dobrote iz prethodne fronte (nakon dijeljenja). Postupak se nastavlja tako da se i u svakoj idućoj iteraciji zanemaruju jedinke koje su već evaluirane dok god sve jedinke ne budu evaluirane i razdijeljene po frontama.

Podjela dobrote u svakoj se fronti postiže izračunavanjem vrijednosti funkcije podjele između svaka dva rješenja iz iste fronte prema formuli:

Formula 2‑2 Funkcija podjele dobrote

Dakle ako je udaljenost između dvije jedinke manja od σshare, one su preblizu jer zajedno imaju preveliku šansu u proporcionalnoj selekciji naspram ostalih jedinki iz te fronte. Za parametar α autori algoritma uzeli su vrijednost 2. U navedenoj formuli *dij*je normalizirana udaljenost u prostoru rješenja, a ne u prostoru ciljnih funkcija i može se izračunati prema sljedećoj formuli:

Formula 2‑2 Udaljenost rješenja

Formula 2-2 pretpostavlja da je prostor rješenja *n*-dimenzijski te da su xkmax i xkmin maksimalna i minimalna vrijednost dimenzije *k* koju ima neka jedinka iz promatrane fronte. U daljnjem postupku računa se vrijednost *niche count* za svako rješenje prema formuli:

Formula 2‑2 Izračun parametra *niche count*

Konačna nova podijeljena vrijednost dobrote (*shared fitness*) dobije se tako da se za svako rješenje *xi* ona stara nasumce dodijeljena svima iz fronte podijeli s vrijednošću *nci*.

Dalje se algoritam ponaša kao klasični generacijski genetski algoritam uz proporcionalnu selekciju.

Dijagram toka algoritma prikazan je na slici 2-3.



Slika 2‑3 Dijagram toka algoritma NSGA

Glavne prednosti ovog algoritma su neosjetljivost na broj kriterija te podjela dobrote u prostoru rješenja, a ne u prostoru ciljnih funkcija (iako je to na prvi pogled neobično zbog prirode problema). Time se osigurava bolja distribucija jedinki i dozvoljavaju se višestruka jednako vrijedna rješenja u prostoru ciljnih funkcija (čije je „podrijetlo“ u prostoru rješenja vrlo različito). U nekim se izvorima tvrdi da je NSGA manje učinkovit od MOGA ako se promatra i računska složenost algoritma i kvaliteta rješenja koja se njime dobiju jer je NSGA poprilično spor. Drugi nedostatak je veća osjetljivost na parametar σshare. [1][4][6]

### NSGA-II

Algoritam NSGA-II značajan je po tome što (uspješno) uvodi elitizam u višekriterijsku optimizaciju i time osigurava da ne dolazi do gubitka najboljih rješenja.

Algoritam iz trenutne populacije *Pt*, koja sadrži *N* jedinki roditelja, stvara populaciju djece *Qt* koja je također veličine *N*. Unija te dvije populacije naziva se *Rt* i ona sadrži *2N* jedinki. Populacija *Rt* nedominirano se sortira i njezine su jedinke razdijeljene u ρ fronti *F1, F2, …, Fρ*. U novu populaciju *Pt+1*redom se kopiraju fronte dokle god stanu. Neka je fronta *Fμ* prva po redu fronta koje ne koja stane u potpunosti. Ona se dakle mora nekako razdijeliti na jedinke koje mogu i jedinke koje ne mogu ući u novu populaciju *Pt+1*.

Algoritam preferira onaj podskup jedinki koje su međusobno što više udaljene tako da pokrivenost fronte bude maksimalna. Takva rješenja pronalaze se postupkom sortiranja prema grupiranju (engl. *crowding sort*) i dodaju se u novu populaciju *Pt+1*. Grupirajućom turnirskom selekcijom (engl. *crowded tournament selection*) iz *Pt+1* se odabiru roditelji te se križanjem i mutacijom stvaraju djeca koja se dodaju u populaciju *Qt+1*. [3][5]

#### Sortiranje prema grupiranju (engl. *crowding sort*)

Sortiranje prema grupiranju sortira jedinke prema mjeri udaljenosti grupiranja (engl. *crowding distance*) *di* koja označava koliko su gusto rješenja nastanjena u okolici promatranog rješenja. Za svaki od *M* kriterija potrebno je izračunati kolika je udaljenost do prvog susjeda koji je po tom kriteriju manji od promatranog rješenja te kolika je udaljenost do prvog susjeda koji je po tom kriteriju veći. Ako su rješenja rubna, odnosno nemaju lijevog ili desnog susjeda, dodjeljuje ima se udaljenost *di* = **∞** zato što su takva rješenja dragocjena zbog tražene raznolikosti rješenja i potrebno je osigurati da se ne izgube. Ostalim rješenjima, koja imaju lijevog i desnog susjeda, *di* se povećava za normiranu razliku dobrota njegovih susjeda, prema formuli

gdje je *fm* kriterij za koji se računa udaljenost, a *fmax* i *fmin* maksimalna i minimalna vrijednost tog kriterija među rješenjima iz fronte *Fμ*.

Nakon što je svim rješenjima pridijeljena udaljenost *di*, može se iskoristiti bilo koji algoritam sortiranja da se dobije traženi broj jedinki koje imaju najveću grupirajuću udaljenost. Pošto se sortiranje izvodi onoliko puta koliko ima kriterija, preporuka je koristiti algoritam sortiranja što manje složenosti, npr. *merge sort* čija je složenost *N\*log(N)* pa bi ukupna složenost sortiranja prema grupiranju bila *M\*N\*log(N)*. [3][5]

#### Grupirajuća turnirska selekcija (engl. *crowded tournament selection*)

Operator grupirajuće turnirske selekcije (*<c*) uspoređuje dva rješenja i vraća pobjednika turnira. Pretpostavlja se da su svakom rješenju pridružene dvije vrijednosti:

* *ri* - nedominirani rang u populaciji, odnosno broj rješenja koja dominiraju nad tim rješenjem i
* *di* – udaljenost grupiranja izračunata u sortiranju prema grupiranju.

Pomoću tih vrijednosti definira se operator grupirajuće turnirske selekcije. Ako su natječu rješenje *i* te rješenje *j*, rješenje *i* bit će pobjednik turnira ako:

* rješenje *i* ima bolji rang od rješenja *j* (tj. *ri < rj*) ili
* rješenje *i* ima jednak rang kao rješenje *j* (tj. *ri = rj*) te rješenje *i* ima bolju udaljenost grupiranja od rješenja *j* (tj. *ri = rj* i *di > dj*).

Dakle pobjeđuje ono rješenje nad kojim dominira manje drugih rješenja, a ako su rješenja međusobno izjednačena po tom kriteriju pobjeđuje ono koje obitava u manje napučenom prostoru ciljnih funkcija. [3][5]

#### Primjene u znanosti

NSGA-II jedan je od najkorištenijih algoritama višekriterijske optimizacije u najraznolikijim granama znanosti. Samo neke od primjena su:

* optimizacija područne pokrivenosti izviđanja satelitskih konstelacija (*Optimization of Regional Coverage Reconnaissance Satellite Constellation by NSGA-II Algorithm*),
* konstrukcija strojnih postrojenja (*Pareto Optimization of Power System Reconstruction Using NSGA-II algorithm*),
* usmjeravanje datagrama kroz mrežu (*Flow-Based Path Selection for Internet Traffic Engineering with NSGA-II*) i dr.

### PPES

Algoritam PPES (*Predatory-prey evolution strategy*) zanimljiv je zbog svoje originalnosti i jednostavnosti. Algoritam se uvelike razlikuje od (skoro) svih do sada spomenutih algoritama jer ne koristi relaciju dominacije za dodjeljivanje dobrote rješenjima. Umjesto toga osmišljen je model predator-plijen pomoću kojeg se dodjeljuje dobrota. Skup rješenja predstavlja plijen *{****x****1,* ***x****2,…,* ***x****N}* koji se smješta u vrhove povezanog neusmjerenog grafa. Predatori su zapravo kriteriji, odnosno ciljne funkcije, i nasumice se smještaju u bilo koji vrh grafa. Predatori, kao i inače u prirodi, hvataju plijen i uvijek nastrada najslabiji iz skupine, stoga svaki predator pretražuje susjedne vrhove u potrazi za plijenom koji je najslabiji prema kriteriju kojeg predstavlja taj predator. Kada je plijen ***x****(i)* uhvaćen, briše se iz vrha i na njegovo mjesto dolazi novi plijen nastao križanjem i mutacijom iz dva slučajna susjeda koji su preživjeli. Predator zatim dalje napada nekog slučajno odabranog susjeda.

Opisana procedure može se ponavljati paralelno ili slijedno sa svih *M* predatora dok se ne prođe zadani broj iteracija.

Glavna prednost ovog algoritma je njegova jednostavnost. Slučajni obilazak vrhova povezanog grafa i zamjenu najgorih rješenja prema pojedinim kriterijima sa novim lako je implementirati. Još jedna prednost ovog algoritma jest to da ne preferira nedominirana rješenja direktno niti se koristi operator koji rješenja zadržava u tom području, no rješenja dobivena tim algoritmom su, kako autori tvrde, dobra i raznolika i za konveksne i za ne konveksne testne funkcije. Trik koji je upotrijebljen jest adaptivna vjerojatnost mutacije. U slučaju da imamo dva kriterija, za svako se rješenje zasebno računa donja granica vjerojatnosti mutacije prema formuli

gdje su *bestPrey1* i *bestPrey2* vjerojatnosti mutacije trenutno najboljeg rješenja prema prvom, odnosno prema drugom kriteriju. [6][7]

# Implementacija i rezultati

Kao pokaznu implementaciju planira se usporediti algoritme NSGA, NSGA-II te PPES. NSGA predstavnik je algoritama koji koriste nedominirano sortiranje te nemaju ugrađeni elitizam, dok NSGA-II ima ugrađen elitizam. S druge strane, zanimljivo će biti usporediti ta dva algoritma s algoritmom PPES koji je inspiriran sasvim drugačijom idejom (predator-plijen) te na testnim funkcijama vidjeti koji se algoritmi bolje ponašaju u kojim situacijama. Skup dvokriterijskih testnih funkcija prikazan je na slici i preuzet je iz [5].



Slika 3‑1 Testne funkcije

Očekuje se da će se NSGA-II najbolje ponašati u većini slučajeva, pošto će većina testnih funkcija biti iz članka u kojem je taj algoritam objavljen, no taj algoritam i inače pokazuje jako dobre rezultate zbog elitizma. S druge strane, bit će zanimljivo provjeriti tvrdnju autora PPES da taj algoritam unatoč svojoj iznimnoj jednostavnosti pokazuje zavidne rezultate.

# Zaključak

Postoje problemi koji su po svojoj prirodi višekriterijski i njihovim svođenjem na jednokriterijsku optimizaciju nikada nećemo postići tako dobre rezultate kao izravnom primjenom nekog pouzdanog algoritma višekriterijske optimizacije.

Napretku u području višekriterijske optimizacije velik je doprinos dao pojam dominacije te postupak rangiranja rješenja i nedominiranog sortiranja koje je 1989. godine predložio Goldberg. Pomoću njih definiran je globalni optimum višekriterijske optimizacije, tzv. Pareto optimalni skup, koji nije bilo moguće definirati isključivo pomoću pojmova koji se koriste u jednokriterijskoj optimizaciji. Važno je uočiti da je optimalno rješenje skup rješenja, a ne samo jedno rješenje, zato da sam korisnik, koji je postavio zahtjev za rješavanjem problema, odabere koje mu rješenje najviše odgovara. Iz toga se samo po sebi nameće da su idealni algoritmi za rješavanje problema višekriterijske optimizacije upravo genetski algoritmi koji ionako rade s populacijom rješenja.

Najuspješniji algoritmi višekriterijske optimizacije (MOGA, NSGA, NSGA-II,…) bazirani su upravo na osnovnom genetskom algoritmu te nedominiranom sortiranju. U ovom radu dan je kratak pregled na razne više i manje uspješne algoritme, te su dane prednosti i mane svakog od njih. U jednostavnoj VEGA-i pokušalo se na temelju provođenja više paralelnih jednokriterijskih optimizacija na podpopulacijama doći do zadovoljavajućeg rješenja, no ta se tehnika nije pokazala uspješnom u pronalaženju globalnog optimuma. MOGA je među prvim uspješnijim algoritmima koji je primijenio Goldbergovu ideju te ju je spojio s idejom podjele dobrote u prostoru ciljnih funkcija tako da se smanje šanse za izbor rješenjima koja su suviše blizu, no bio je suviše spor. NPGA je ubrzao proces selekcije prilagođenom turnirskom selekcijom što je kasnije iskorišteno i u drugim algoritmima. SPEA je značajan jer prvi kombinira ideju nedominiranog sortiranja s elitizmom. Nadalje, NSGA je na odličan način riješio podjelu dobrote nakon nedominiranog sortiranja te, unatoč tome što se podjela dobrote radi u prostoru rješenja te algoritam ne uključuje elitizam, daje vrlo dobre rezultate, no još je sporiji nego MOGA. NSGA-II najpopularniji je algoritam današnjice te se primjenjuje u svim mogućim granama znanosti. Njegovi su glavni aduti brzina i elitizam koji su prije bili nespojivi zbog skupoće samog sortiranja. No, NSGA-II je grupirajućom turnirskom selekcijom i sortiranjem prema grupiranju osigurao brzinu i raznolikost jedinki, a time i temeljitiju pretragu prostora rješenja. U konačnici, predstavljen je PPES kao algoritam potpuno različit od ostalih po svojoj jednostavnosti, koji daje relativno zadovoljavajuće rezultate.

Postoji još mnoštvo algoritama višekriterijske optimizacije, no izdvojeni su samo oni za koje se smatralo da su po nečemu bili značajni za proučavanje razvoja višekriterijske optimizacije kroz desetljeća ili su i dan danas često korišteni i vrijedi ih upotrijebiti za rješavanje vlastitih problema.

Ubuduće, planira se implementacija opisana u trećem poglavlju te prezentacija dobivenih rezultata. Višekriterijska optimizacija kao takva vrlo je široko područje računarske znanosti te ne samo da ima još mnogo činjenica dobivenih istraživanjima koje treba sistematizirati, proučiti i usporediti, nego ima još mnogo mjesta za napredak u istraživanju i razvoj novih, još boljih algoritama.

# Literatura

1. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms. Wiley, West Sussex*, *United Kingdom. 2009.*
2. M. Golub. *Vrednovanje uporabe genteskih algoritama za aproksimaciju vremenskih nizova. Magistarski rad.* Sveučilište u Zagrebu*. 1996*.
3. M. Čupić***.*** *Prirodom inspirirani optimizacijski algoritmi. Metaheuristike.* 2012***.***
4. N. Srinivas, K. Deb. *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*. *Journal of Evolutionary Computation, vol. 2, br. 3, 221-248*. *1994.*
5. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. *A Fast and Elitistic Multiobjective Genetic Algorithm: NSGA-II*. *IEEE Transactions on Evolutionary Computation, vol. 6, br. 2. 2002.*
6. A. Ghosh, S. Dehuri. *Evolutionary Algorithms for Multi-Criterion Optimization: A Survey. International Journal of Computing & Information Sciences, vol. 2, br. 1. 2004.*
7. M. Laumans, G. Rudolph, H. P. Schwefel. *A Spatial Predatory-Prey Approach to Multi-objective Optimization: A Preliminary Study. Proceedings of the Parallel problem solving from Nature. V. 241-249. 1998.*
8. Pareto fronta

http://www.noesissolutions.com/Noesis/sites/default/files/Pareto\_Front.png

04.05.2013.