SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2373

Radno okruženje za ispitivanje metaheuristika

Mirta Dvorničić

Zagreb, srpanj 2012.

Umjesto ove stranice umetnite izvornik Vašeg rada.

Da biste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

Problem pronalaska optimalnog rješenja mnogih optimizacijskih problema je složen te iziskuje previše resursa i vremena. Zbog neprimjenjivosti iscrpne pretrage ili analitičkog rješavanja u praksi se često koriste razne metode koje nude približna odnosno dovoljno dobra rješenja. Heuristike su algoritmi relativno niske računske složenosti koji nude takva rješenja no ne garantiraju njihovu optimalnost. Metaheuristike se mogu definirati kao općenite metode koje na visokoj razini nude predložak za oblikovanje heuristika koje rješavaju specifične optimizacijske probleme [18].

U središtu interesa ovog rada nalaze se algoritmi zasnovani na evolucijskom računanju koji se uobičajeno dijele u sljedeća područja:

– *evolucijski algoritmi*: genetski algoritmi, genetsko programiranje, evolucijsko programiranje, evolucijske strategije...

– *algoritmi zasnovani na inteligenciji roja*: algoritam mravlje kolonije, algoritam roja čestica, algoritam kolonije pčela, algoritam kukavičje pretrage...

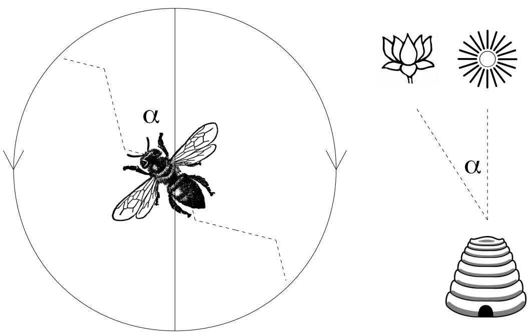
– *ostali algoritmi*: algoritmi umjetnih imunoloških sustava, diferencijska evolucija, harmonijsko pretraživanje...

Algoritmi zasnovani na evolucijskom računanju su populacijske metaheuristike inspirirane prirodom, osobito genetikom i evolucijom. Rade s populacijom odnosno skupom jedinki nad kojima se iterativno primjenjuju operatori odabira (engl. *selection*), križanja (engl. *crossover*) i mutacije (engl. *mutation*). Za potrebe optimizacije problem se modelira funkcijom cilja (engl. *objective function, cost function*) čija vrijednost određuje dobrotu (engl. *fitness*) jedinke. Time se problem optimizacije svodi na pronalazak željenog ekstrema funkcije cilja. Operatori odabira najčešće se oslanjaju upravo na dobrotu jedinke, dok operatori križanja i mutacije ovise o načinu zapisa rješenja problema odnosno genotipu (engl. *genotype*) jedinke.

U ovom radu razrađeno je nekoliko algoritama zasnovanih na evolucijskom računanju te njihova primjena na probleme s kontinuiranom domenom. Većinu algoritama moguće je uz manje promjene primijeniti i na probleme s diskretnom domenom.

2. Algoritam kolonije pčela

Algoritam kolonije pčela (engl. *Artificial Bee Colony, ABC*) predložio je D. Karaboga 2005. godine [12]. Algoritam je inspiriran inteligentnim ponašanjem pčela u prirodi tijekom pronalaženja hrane kao što je pčelinji ples prikazan na slici 2.1.



Slika 2.1: Pčelinji ples.

Sustav kolonije pčela obilježava specijalizacija, podjela rada i samoorganizacija. Takav sustav čine tri osnovne komponente: zaposlene pčele (engl. *employed bees*), nezaposlene pčele (engl. *unemployed bees*) te izvori hrane. Nezaposlene pčele čine pčele promatrači (engl. *onlooker bees*) i pčele izviđači (engl. *scout bees*). Zaposlene pčele i pčele promatrači iscrpljuju nektar svojih izvora hrane. Nakon što se određeni izvor hrane iscrpi pčela koja ga je iscrpila postaje izviđač te kreće u potragu za novim izvorom hrane.

U fazi inicijalizacije slučajnim odabirom unutar prostora pretraživanja izrazom 2.1 inicijaliziraju se izvori hrane koji predstavljaju potencijalna rješenja optimizacijskog problema pri čemu su l_i i u_i donja odnosno gornja granica prostora pretraživanja, a $\mathcal{U}$ uniformna distribucija.

$$x_{mi} = l_i + \mathcal{U}(0,1)(u_i - l_i) \tag{2.1}$$

Algoritam 1 ABC

faza inicijalizacije
ponavljaj
 faza zaposlenih pčela
 faza pčela promatrača
 faza pčela izviđača
dok nije ispunjen kriterij zaustavljanja

Tijekom faze zaposlenih pčela svaka pčela traži novi izvor hrane u susjedstvu vlastitog izvora hrane koji bi mogao imati više nektara odnosno veću dobrotu. Susjedni izvor hrane $\mathbf{v}_m$ pčele s izvorom hrane $\mathbf{x}_m$ određuje se izrazom 2.2 gdje je $\mathbf{x}_n$ slučajno odabrani izvor hrane različit od $\mathbf{x}_m$, ϕ slučajno odabrani broj iz intervala $[-1, 1]$, a i_{rand} slučajno odabrana dimenzija. Ako pronađeni susjedni izvor hrane ima veću dobrotu, trenutni izvor hrane se napušta te pčela prelazi na susjedni izvor hrane.

$$v_{mi} = \begin{cases} x_{mi} + \phi(x_{mi} - x_{ni}) & \text{ako je } i = i_{rand} \\ x_{mi} & \text{inače} \end{cases} \tag{2.2}$$

Tijekom faze pčela promatrača svaka pčela bira izvor hrane prema informacijama dobivenim od zaposlenih pčela. Vjerojatnost p_m da će biti odabran izvor hrane $\mathbf{x}_m$ najčešće se određuje proporcionalno dobroti izvora hrane prema izrazu 2.3. Nakon što je izvor hrane pronađen kao i u fazi zaposlenih pčela slučajnim se odabirom pronalazi novi izvor hrane izrazom 2.2 te se provodi pohlepna selekcija.

$$p_m = \frac{fitness(\mathbf{x}_m)}{\sum\limits_{i=1}^{n} fitness(\mathbf{x}_i)} \tag{2.3}$$

U fazi pčela izviđača svaka pčela čiji se izvor hrane nije poboljšao tijekom unaprijed određenog broja generacija $limit$ napušta svoj izvor te pronalazi novi izrazom 2.1.

Ovakvim mehanizmom algoritam zapravo provodi četiri vrste selekcije [13]: pčele promatrači provode globalnu selekciju te su usredotočene na istraživanje prostora, sve pčele provode lokalnu selekciju iskorištavanjem prostora u okolini trenutnih rješenja te pohlepnu selekciju odabirom izvora hrane s većom dobrotom dok pčele izviđači provode selekciju slučajnim odabirom. Najčešće je broj zaposlenih pčela jednak broju pčela promatrača te tijekom jedne iteracije samo jedna pčela postaje izviđač i napušta svoj izvor hrane.

3. Algoritam kukavičje pretrage

Algoritam kukavičje pretrage (engl. *Cuckoo Search, CS*) predložili su X. Yang i S. Deb 2009. godine [19]. Algoritam je inspiriran parazitiranjem gnijezda nekih vrsta kukavica te letom određenih vrsta ptica i vinske mušice prema Lévy distribuciji.

Svaka kukavica liježe jedno jaje te ga ostavlja u slučajno odabranom gnijezdu. Gnijezda s visokom kvalitetom jaja prenose se u sljedeću generaciju. Broj gnijezda je konstantan, a određen broj jaja koja su ostavile kukavice se otkriva pri čemu se napuštaju stara i grade nova gnijezda. Jaja u gnijezdu predstavljaju moguće rješenje optimizacijskog problema, dok nova jaja koja ostavlja ptica kukavica predstavljaju potencijalno bolja rješenja.

Algoritam 2 CS

---

inicijaliziraj gnijezda domaćine P
ponavljaj
  za sve $nest_i \in P$
    $cuckoo \leftarrow lévyFlight(nest_i)$
    slučajnim odabirom odaberi $nest_j$ iz P
    ako $fitness(nest_j) < fitness(cuckoo)$
      $nest_j \leftarrow cuckoo$
    napuštanje otkrivenih gnijezda
dok nije ispunjen kriterij zaustavljanja

Lévy let se provodi izrazom 3.1 gdje je $\alpha > 0$ veličina koraka (engl. *step size*) koja se postavlja ovisno o problemu, uobičajeno kao $\alpha = 0.01L$ ili $\alpha = 0.001L$ gdje je L širina prostora pretraživanja za određenu dimenziju. Za Lévy distribuciju se preporučuje korištenje Mantegna algoritma [20] danog izrazom 3.2 gdje su u i v normalno distribuirane varijable sa srednjom vrijednosti $\mu = 0$ i standardnim devijacijama danim izrazima 3.3 i 3.4 pri čemu je Γ gamma funkcija. Parametar distribucije β uzima se iz intervala $\langle 1, 3]$. Primjer Lévy leta prikazan je na slici 3.1.

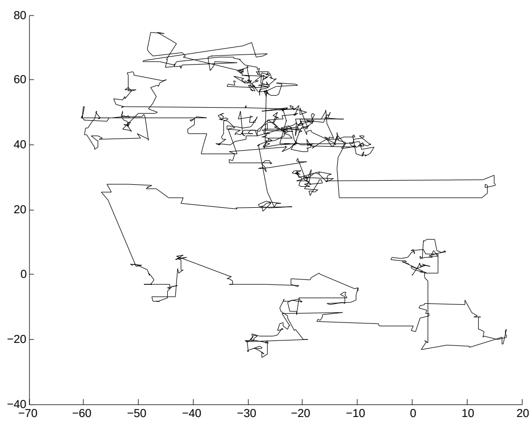
$$x_i(t+1) = x_i(t) + \alpha \cdot Lévy(\beta) \quad (3.1)$$

$$s = \frac{u}{|v|^{1/\beta}} \quad (3.2)$$

$$\sigma_u = \frac{\Gamma(1+\beta) \cdot sin(\beta \cdot \pi/2)}{\Gamma((1+\beta)/2) \cdot \beta \cdot 2^{(\beta-1)/2}} \quad (3.3)$$

$$\sigma_v = 1 \quad (3.4)$$

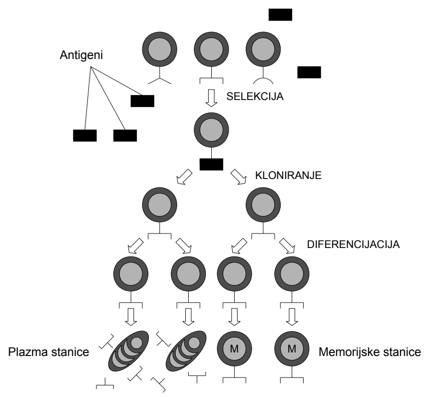
Nakon faze ptica kukavica, dio gnijezda određen parametrom p_a se napušta, te se slučajnim odabirom unutar prostora pretraživanja grade nova gnijezda.



Slika 3.1: Primjer leta prema Lévy distribuciji. 1000 koraka, $\alpha = 1$, $\beta = 1.5$.

4. Algoritam klonske selekcije

Algoritam klonske selekcije (engl. *Clonal Selection Algorithm, CLONALG*) predložili su L.N. de Castro and F.J. Von Zuben 2000. godine [4] kao jedan od algoritama umjetnih imunoloških sustava (engl. *Artificial Immune Systems, AIS*). Uz nekoliko modifikacija 2005. godine predložen je i algoritam klonske selekcije opt-IA [3]. Algoritam se temelji na teoriji klonske selekcije koja modelira odgovor prirodnog imunološkog sustava pri izlaganju infekciji.



Slika 4.1: Princip klonske selekcije.

Osnovni princip klonske selekcije prikazan je na slici 4.1. Nakon početnog izlaganja antigenu, B limfociti koju su prepoznali antigen ulaze u proces klonske ekspanzije. Tijekom tog procesa dolazi do povećanja prosječnog afiniteta prema antigenu odnosno sazrijevanja afiniteta uzrokovanog somatskom hipermutacijom i mehanizmom selekcije čime se postiže povećanje brzine i točnosti imunološkog odgovora pri svakom sljedećem izlaganju istom antigenu. U kontekstu algoritma, antitijela predstavljaju moguća rješenja optimizacijskog problema, antigen predstavlja optimizacijski problem, a afinitet predstavlja dobrotu.

Tijekom početne inicijalizacije antitijela se inicijaliziraju slučajnim odabirom unutar prostora pretraživanja.

Algoritam 3 CLONALG

inicijaliziraj populaciju antitijela P_{ab}
ponavljaj
  stvori populaciju klonova $P_{clo} \leftarrow kloniraj(P_{ab})$
  stvori populaciju sazrelih klonova $P_{hyp} \leftarrow hipermutiraj(P_{clo})$
  $P_{ab} \leftarrow selekcija(P_{ab} \cup P_{hyp})$
  provedi starenje populacije antitijela P_{ab}
dok nije ispunjen kriterij zaustavljanja

U fazi kloniranja stvara se populacija klonova, odnosno odabire se n antitijela koja se potom kloniraju jednoliko ili proporcionalno afinitetu. Kod jednolikog kloniranja broj klonova određen je parametrom β pri čemu se za svako antitijelo stvara $round(\beta \cdot N)$ klonova gdje je N veličina populacije.

Tijekom faze hipermutacije provodi se mutacija nad populacijom klonova čime nastaje sazrela populacija klonova. Neki od operatora mutacije su:

- *proporcionalna hipermutacija*: broj mutacija određen je s $\lfloor l \cdot \alpha \rfloor$ gdje je l dimenzija zapisa rješenja. Neki od predloženih izraza za stopu mutacije α su 4.1 i 4.2 gdje je f vrijednost dobrote normalizirana u $[0, 1]$, a ρ zadan parametar.
- *inverzno proporcionalna hipermutacija*: broj mutacija određen je izrazom 4.3 gdje je E^* najveća dobrota, a c stopa mutacije.
- *hipermakromutacija*: mutiraju se varijable na pozicijama $[i, j]$ gdje se indeksi i i j biraju slučajnim odabirom tako da vrijedi $i < j < l$.

$$\alpha = e^{-\rho * f} \tag{4.1}$$

$$\alpha = \frac{1}{\rho} e^{-f} \tag{4.2}$$

$$M(f(\mathbf{x})) = \left(1 - \frac{E^*}{f(\mathbf{x})}\right)(c \cdot l) + c \cdot l \tag{4.3}$$

Za izvedbu faze selekcije postoji nekoliko mogućnosti:

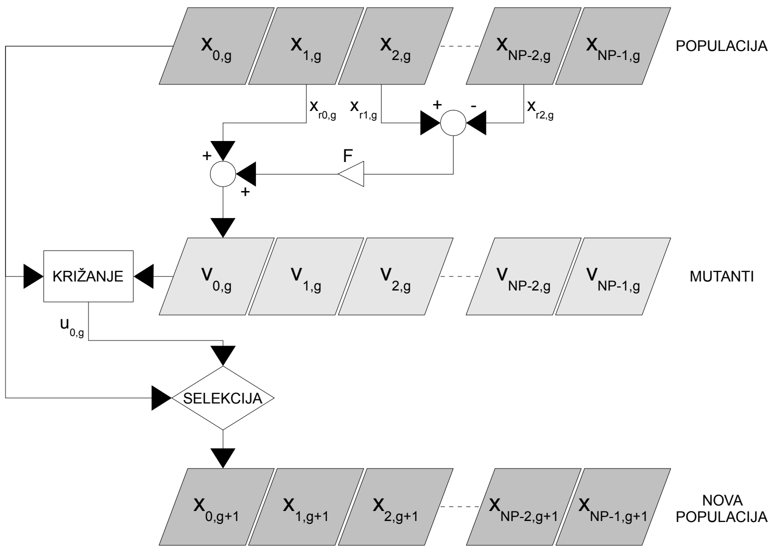
- svako antitijelo u sljedećoj se generaciji zamjenjuje najboljim od njegovih $\beta \cdot N$ hipermutiranih klonova ($\texttt{CLONALG}_1$).
- sljedeća generacija formira se od N najboljih hipermutiranih klonova ($\texttt{CLONALG}_2$).

Faza starenja uvodi se radi izbjegavanja preuranjene konvergencije. Parametar τ_B određuje najveći broj generacija tijekom kojeg antitijelo može ostati u populaciji. Tijekom klonske ekspanzije klon preuzima starost roditeljskog antitijela. Nakon faze hipermutacije klonovima koji su uspješno mutirali starost se postavlja na 0. Postoje različite vrste starenja, primjerice:

- *čisto statičko starenje* (engl. *static pure aging*): antitijelo se briše iz populacije nakon $\tau_B + 1$ generacija.
- *stohastičko starenje* (engl. *stochastic aging*): antitijelo se briše iz populacije s vjerojatnošću $P_{die}(\tau_B) = 1 - e^{-\ln(2)/\tau_B}$.

5. Diferencijska evolucija

Diferencijsku evoluciju (engl. *Differential Evolution, DE*) predložili su R. Storn i K. Price 1995. godine [17]. Algoritam diferencijske evolucije je u osnovi jednostavni matematički model složenog procesa evolucije prikazan na slici 5.1. Moguća rješenja optimizacijskog problema predstavljena su vektorima nad kojima se primjenjuju operatori mutacije, križanja i selekcije.



Slika 5.1: Osnovna ideja diferencijske evolucije.

Operator mutacije određen je izrazom 5.1. Vektor mutant nastaje tako da se osnovnom (engl. *base*) vektoru $\mathbf{x}_{r0}$ pribroji skalirani vektor razlike određen dvama slučajno odabranim vektorima iz populacije $\mathbf{x}_{r1}$ i $\mathbf{x}_{r2}$ pri čemu svi vektori moraju biti međusobno različiti te različiti od ciljnog (engl. *target*) vektora. Faktor skaliranja F služi za kontrolu diferencijske varijacije te se uobičajeno odabire iz intervala $[0, 1+]$.

$$\mathbf{v}_i = \mathbf{x}_{r0} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{5.1}$$

Operator križanja određen je izrazom 5.2. U križanju sudjeluju vektor mutant i ciljni vektor te se osigurava da barem jedna komponenta j_{rand} rezultantnog vektora bude naslijeđena od vektora mutanta. Rezultantni vektor naziva se probnim (engl. *trial*) vektorom. Faktor križanja CR uobičajeno se postavlja na 0.5.

$$u_{i,j} = \begin{cases} v_{i,j} & \text{ako je } \mathcal{U}(0,1) \leq CR \text{ ili } j = j_{rand} \\ x_{i,j} & \text{inače} \end{cases} \tag{5.2}$$

Operator selekcije je pohlepan. Probni vektor zamjenjuje ciljni vektor ako ima veću dobrotu, inače ciljni vektor zadržava svoje mjesto u populaciji u sljedećoj generaciji.

Algoritam 4 DE

```
inicijaliziraj populaciju P
ponavljaj
  za sve x ∈ P
    odaberi r0, r1, r2 iz P
    slučajnim odabirom odaberi dimenziju j_rand iz {1...D}
    za j ∈ {1...D}
      ako U(0,1) ≤ CR ili j = j_rand
        u_j := r0_j + F(r1_j − r2_j)
      inače
        u_j := x_j
    ako fitness(x) < fitness(u)
      x ← u
dok nije ispunjen uvjet zaustavljanja
```

Osim osnovnog operatora mutacije moguće je primijeniti i nekoliko drugih operatora mutacije odnosno strategija pretraživanja. Svaka strategija određena je odabirom osnovnog vektora te odabirom broja vektora razlike koji se skalirani dodaju osnovnom vektoru. Neke od često korištenih strategija određene su izrazima 5.3 - 5.7: DE/rand/1, DE/rand/2, DE/best/1, DE/best/2 i DE/rand-to-best/1.

$$\mathbf{v}_i = \mathbf{x}_{r0} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{5.3}$$

$$\mathbf{v}_i = \mathbf{x}_{r0} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2} + \mathbf{x}_{r3} - \mathbf{x}_{r4}) \tag{5.4}$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{5.5}$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2} + \mathbf{x}_{r3} - \mathbf{x}_{r4}) \tag{5.6}$$

$$\mathbf{v}_i = \mathbf{x}_i + F(\mathbf{x}_{best} - \mathbf{x}_i) + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \tag{5.7}$$

6. Algoritam roja čestica

Prvi algoritam zasnovan na roju čestica (engl. *Particle Swarm Optimizer, PSO*) predložili su 1995. godine James Kennedy i Russell C. Eberhart [14] [6]. PSO je populacijski algoritam zasnovan na principima inteligencije roja. U kontekstu PSO algoritma, roj (engl. *swarm*) predstavlja populaciju, a jedinka populacije je čestica (engl. *particle*) roja koja predstavlja moguće rješenje optimizacijskog problema vektorom položaja u višedimenzijskom prostoru. Čestici je pridružen i vektor brzine koji određuje smjer kretanja u prostoru. Svaka čestica pamti informaciju o vlastitom najboljem pronađenom položaju kao i o najboljem pronađenom položaju u susjedstvu.

Algoritam 5 PSO

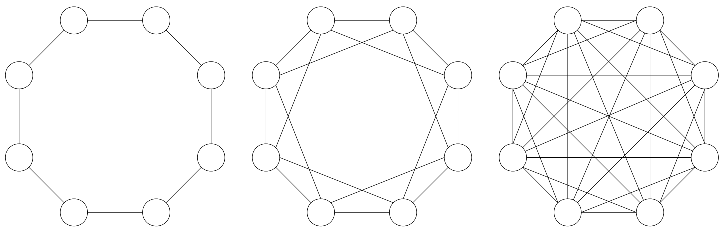
inicijaliziraj roj S
ponavljaj
  za sve $p \in S$
    ažuriraj brzinu $\mathbf{v}_p$
    ažuriraj položaj $\mathbf{x}_p$
    ako $fitness(\mathbf{x}_p) > fitness(\mathbf{x}_{pbest_p})$
      $\mathbf{x}_{pbest_p} \leftarrow \mathbf{x}_p$
      ako $fitness(\mathbf{x}_p) > fitness(\mathbf{x}_{lbest_p})$
        $\mathbf{x}_{lbest_p} \leftarrow \mathbf{x}_p$
dok nije ispunjen uvjet zaustavljanja

Ažuriranje brzine $\mathbf{v}$ i položaja $\mathbf{x}$ dano je izrazima 6.1 i 6.2 gdje je t redni broj iteracije, C_1 i C_2 su akceleracijski koeficijenti koji određuju utjecaj kognitivne i socijalne komponente, $\mathbf{r}_1$ i $\mathbf{r}_2$ su slučajno generirani vektori s komponentama iz intervala $[0, 1]$, $\mathbf{x}_{pbest}$ je najbolji pronađeni položaj čestice, a $\mathbf{x}_{lbest}$ je najbolji pronađeni položaj u susjedstvu čestice. Eliminacijom komponente određene faktorom C_1 dobiva se socijalni model ažuriranja brzine, a eliminacijom komponente određene faktorom C_2 dobiva se kognitivni model ažuriranja brzine.

$$\mathbf{v}(t) = \mathbf{v}(t-1) + C_1\mathbf{r}_1(\mathbf{x}_{pbest} - \mathbf{x}(t-1)) + C_2\mathbf{r}_2(\mathbf{x}_{lbest} - \mathbf{x}(t-1)) \tag{6.1}$$

$$\mathbf{x}(t) = \mathbf{x}(t-1) + \mathbf{v}(t) \tag{6.2}$$

Neke od mogućih topologija prikazane su na slici 6.1. Prstenasta (engl. *ring*) i Von Neumannova topologija predstavljaju $lbest$ inačicu ažuriranja brzine gdje svaka čestica ima četiri odnosno dvije susjedne čestice. Potpuno povezana (engl. *fully-connected*) topologija predstavlja $gbest$ inačicu ažuriranja brzine gdje je svakoj čestici dostupna informacija o najboljem pronađenom položaju u roju. Odabir topologije susjedstva utječe na brzinu širenja informacija unutar roja, a time i na brzinu konvergencije te sklonost stagnaciji. Često se koristi i vremenski promjenjiva topologija susjedstva radi uspostavljanja ravnoteže između brzine konvergencije i kvalitete rješenja.



Slika 6.1: Neke topologije susjedstva PSO algoritma.

Za poboljšanje osnovnog PSO algoritma uvodi se nekoliko novih pristupa[15] [5].

Faktor inercije (engl. *inertia weight*) ω zadaje se na intervalu $[0, 1\rangle$. Faktorom inercije množi se $\mathbf{v}(t-1)$ u izrazu 6.1. Iznos faktora inercije određuje tendenciju čestice da se nastavi kretati u približno jednakom smjeru kao i u prethodnoj iteraciji. Koristi se i vremenski promjenjiv faktor inercije definiran izrazom 6.3 s ciljem usmjeravanja pretrage od globalne prema lokalnoj ili obrnuto kako bi se uspostavila ravnoteža između istraživanja (engl. *exploration*) i iskorištavanja (engl. *exploitation*) prostora pretraživanja. Uobičajeno se postavlja $\omega_{min} = 0.4$ i $\omega_{max} = 0.9$ dok se broj iteracija kroz koji se mijenja faktor inercije ω_{tmax} određuje proporcionalno kvadratu veličine roja.

$$\omega(t) = \omega_{max} - \frac{t}{\omega_{tmax}}(\omega_{max} - \omega_{min}) \tag{6.3}$$

Faktor ograničenja (engl. *constriction factor*) χ određen je izrazom 6.4. Uobičajeno $\kappa = 1$ i $\varphi = C_1 + C_2 = 4.1$ što daje $\chi \approx 0.73$. Faktorom ograničenja množi se desna strana izraza 6.1 kako bi se spriječila pojava eksplozije roja (engl. *swarm explosion*). Do te pojave dolazi kada brzine čestica nekontrolirano porastu što rezultira divergencijom algoritma.

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{6.4}$$

Postoji i nekoliko alternativnih načina ažuriranja brzine i položaja.

Potpuno informirano (engl. *fully-informed*) ažuriranje brzine određeno izrazom 6.5 preuzeto je iz algoritma FIPS [5]. Svaka čestica p položaj ažurira prema najboljim pronađenim položajima svih čestica u susjedstvu N_p pri čemu se dodatno podrazumijeva da je svaka čestica susjed samoj sebi. Najčešće se postavlja $\varphi = 4$.

$$\mathbf{v}_p(t) = \omega(t)\mathbf{v}_p(t-1) + \sum_{n \in N_p} \frac{\varphi}{|N_p|}(\mathbf{x}_{pbest_n} - \mathbf{x}_p(t-1)) \tag{6.5}$$

Barebones PSO ažuriranje položaja čestice određeno je izrazom 6.6 gdje je $\sigma = |\mathbf{x}_{pbest} - \mathbf{x}_{gbest}|$. Ovakvim načinom ažuriranja brzine čestice algoritam se u početku usredotočuje na istraživanje, a kasnije na iskorištavanje prostora.

$$\mathbf{x}(t) = \mathcal{N}\left(\frac{\mathbf{x}_{pbest} + \mathbf{x}_{gbest}}{2}, \sigma\right) \tag{6.6}$$

7. Evolucijske strategije

Evolucijske strategije čine podskup optimizacijskih tehnika koje pripadaju evolucijskim algoritmima. Korijeni evolucijskih strategija sežu u 1960-e godine kada su P. Bienert, I. Rechenberg i H.-P. Schwefel razvili skup pravila za automatski razvoj shema optimalnih oblika tijela s minimalnim trenjem u aerodinamici koristeći dva jednostavna pravila [2]:

1. U jednoj iteraciji neznatno i slučajnim odabirom mijenjaj sve varijable.
2. Ako novi skup varijabli nije narušio dobrotu uređaja zadrži ga, inače se vrati na stari skup varijabli.

Ova pravila odgovaraju Darwinovoj teoriji evolucije pri čemu prvo pravilo modelira mutacije u prirodi, a drugo prirodni odabir odnosno preživljavanje najboljih jedinki. Od tada je razvijeno nekoliko inačica evolucijskih strategija koje se primjenjuju na širok skup optimizacijskih problema kontinuiranih ili diskretnih domena, s ili bez ograničenja.

Dvije osnovne inačice evolucijskih strategija su samo-adaptirajuće evolucijske strategije $(\mu/\rho, \lambda)$-ES i $(\mu/\rho + \lambda)$-ES gdje je μ veličina populacije, ρ broj jedinki koje sudjeluju u stvaranju potomstva, a λ broj potomaka. Sljedeća generacija se obavlja selekcijom iz skupa potomaka (engl. *comma-selection*) pri čemu mora vrijediti $\mu < \lambda$ ili selekcijom i iz skupa potomaka i iz populacije (engl. *plus-selection*). Osim vektora potencijalnog rješenja problema $\mathbf{x}$ i pripadne dobrote $f(\mathbf{x})$, svakoj jedinki je pridružen i vektor strategije $\mathbf{s}$.

Prilikom inicijalizacije populacije svakoj se jedinki slučajnim odabirom unutar prostora pretraživanja inicijaliziraju vektor rješenja i vektor strategije.

Za parametar $\rho = 1$ ne provodi se rekombinacija već se vektor rješenja i vektor strategije kopiraju od roditelja. Postoje dvije osnovne vrste rekombinacije:

– *diskretna rekombinacija* (engl. *discrete recombination*): svaka varijabla potomka slučajnim se odabirom uzima od jednog od ρ roditelja.

– *srednja rekombinacija* (engl. *intermediate recombination*): svaka varijabla potomka je aritmetička sredina pripadnih varijabla svih ρ roditelja.

Algoritam 6 $(\mu/\rho, \lambda)$-ES, $(\mu/\rho + \lambda)$-ES

inicijaliziraj populaciju P_μ
ponavljaj
  inicijaliziraj populaciju potomaka $P_\lambda := \emptyset$
  za $i \in \{1...\lambda\}$
    slučajnim odabirom odaberi ρ jedinki iz P_μ
    rekombinacijom ρ jedinki stvori potomak r_i
    mutiraj rješenje $\mathbf{x}_{r_i}$
    mutiraj strategiju $\mathbf{s}_{r_i}$
    evaluiraj r_i
    $P_\lambda \leftarrow P_\lambda \cup r_i$
  odaberi sljedeću populaciju P_μ iz P_λ (i P_μ)
dok nije ispunjen kriterij zaustavljanja

Mutacija rješenja provodi se izrazom 7.1, a mutacija strategije izrazom 7.2 gdje je parametar $\tau \sim \frac{1}{\sqrt{l}}$, uobičajeno $\tau = \frac{1}{\sqrt{2l}}$ pri čemu je l dimenzija zapisa rješenja.

$$x_i = x_i + s_i \cdot \mathcal{N}(0, 1) \tag{7.1}$$

$$s_i = s_i e^{\tau \cdot \mathcal{N}(0,1)} \tag{7.2}$$

Osim $(\mu/\rho, \lambda)$-ES i $(\mu/\rho+\lambda)$-ES često se koristi i hijerarhijski organizirana evolucijska strategija $(\mu'/\rho' + \lambda'(\mu/\rho, \lambda)^\gamma)$-ES. Kod te strategije se iz populacije veličine μ' bira ρ' jedinki koje stvaraju λ' potomaka. Stvorena populacija potomaka izolira se na γ generacija. U svakoj od γ generacija iz populacije veličine $\mu = \lambda'$ se bira ρ jedinki koje stvaraju λ potomaka od kojih samo μ najboljih prelazi u sljedeću generaciju. Nakon γ generacija μ' najboljih jedinki iz početne populacije i izolirane populacije prelazi u novu generaciju.

7.1. Adaptacija evolucijske strategije kovarijacijskom matricom

Adaptaciju evolucijske strategije kovarijacijskom matricom (engl. *Covariance Matrix Adaptation Evolution Strategy CMA-ES*) predložili su N. Hansen i A. Ostermeier 1996.

godine [9]. Danas CMA-ES predstavlja *state-of-the-art* algoritam u optimizaciji problema s kontinuiranom domenom $\mathbb{R}^n$ evolucijskim računanjem. Osnovnu razliku u odnosu na prethodne inačice evolucijskih strategija čini distribucija mutacije koja se generira prema kovarijacijskoj matrici. Algoritam koristi niz statističkih parametara za kontrolu kovarijacijske matrice i veličine koraka čime se distribucija mutacije prilagođava dobroti rješenja što može značajno ubrzati konvergenciju [10].

Algoritam 7 CMA-ES

odredi parametre $\lambda, \mu, w_{i=1..\lambda}, c_c, c_1, c_\mu, c_\sigma, d_\sigma$
inicijaliziraj $\mathbf{p}_\sigma = 0, \mathbf{p}_c = 0, \mathbf{C} = \mathbf{I}$
inicijaliziraj $\mathbf{m}$ i σ ovisno o problemu
ponavljaj
  generiraj populaciju prema multivarijatnoj normalnoj distribuciji $\mathcal{N}(\mathbf{m}, \mathbf{C})$
  selekcijom i rekombinacijom jedinki iz populacije odredi $\mathbf{m}$
  adaptiraj $\mathbf{C}$
  adaptiraj σ
dok nije ispunjen uvjet zaustavljanja

Podlogu algoritma čine dekompozicija pozitivno definitne matrice na svojstvene vrijednosti (engl. *eigendecomposition of a positive definite matrix*) i multivarijatna normalna distribucija (engl. *multivariate normal distribution*) [7].

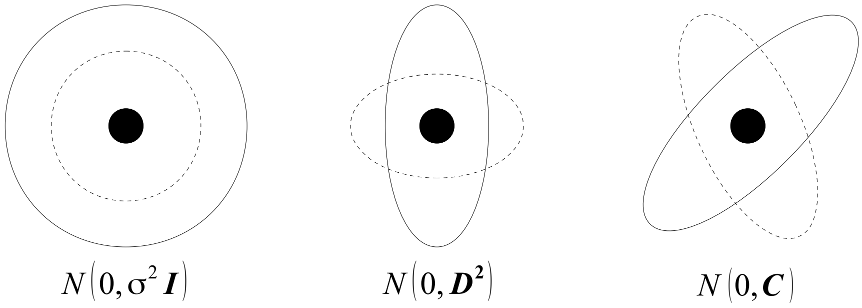
Dekompozicija pozitivno definitne matrice na svojstvene vrijednosti

Za simetričnu, pozitivno definitnu matricu $\mathbf{C} \in \mathbb{R}^{(n \times n)}$ i za svaki $\mathbf{x} \in \mathbb{R}^n \backslash \{0\}$ vrijedi $\mathbf{x}^T\mathbf{C}\mathbf{x} > 0$. Za dekompoziciju matrice $\mathbf{C}$ vrijedi $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T$. $\mathbf{B}$ je ortogonalna matrica $\mathbf{B}\mathbf{B}^T = \mathbf{B}^T\mathbf{B} = \mathbf{I}$ čiji su stupci linearno nezavisni svojstveni vektori matrice $\mathbf{C}$ jedinične duljine. $\mathbf{D}^2$ je dijagonalna matrica čiji su dijagonalni elementi svojstvene vrijednosti matrice $\mathbf{C}$ koji odgovaraju svojstvenim vektorima u stupcima matrice $\mathbf{B}$.

Multivarijatna normalna distribucija

Kovarijacijska matrica $\mathbf{C}$ može se geometrijski interpretirati hiperelipsoidom $\mathbf{x}^T\mathbf{C}\mathbf{x} = 1$ gdje je $\mathbf{x} \in \mathbb{R}^n$ čije su osi određene svojstvenim vektorima matrice $\mathbf{C}$, a duljine osi pripadnim svojstvenim vrijednostima matrice $\mathbf{C}$. Multivarijatna normalna distribucija $\mathcal{N}(\mathbf{m}, \mathbf{C})$ poprima zvonolik oblik gdje vrh zvona odgovara srednjoj vrijednosti $\mathbf{m}$. Primjer multivarijatne normalne distribucije dan je na slici 7.1.

Nova populacija od λ jedinki generira se multivarijatnom normalnom distribucijom prema izrazu 7.3.



Slika 7.1: Primjeri multivarijatne normalne distribucije.

$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}\left(0, \mathbf{C}^{(g)}\right) \quad \text{za } k = 1..\lambda \tag{7.3}$$

Nova srednja vrijednost računa se selekcijom i rekombinacijom prema izrazu 7.4. Izrazom je ostvarena selekcija odsijecanjem (engl. *truncation selection*) odabirom najboljih μ jedinki iz populacije λ jedinki, te težinska srednja rekombinacija (engl. *weighted intermediate recombination*). Dobar odabir w_i daje $\mu_{eff} \sim \lambda/4$. Jedna od uobičajenih postavki je $w_i \sim \mu - i + 1$ što daje $\mu_{eff} \sim \lambda/2$.

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(g+1)} \tag{7.4}$$

Adaptacija kovarijacijske matrice provodi se izrazima 7.5 i 7.6. Kombiniraju se informacije o korelaciji između pojedinih generacija određene rang-1 ažuriranjem (engl. *rank-one update*) važnije kod manjih populacija te informacije unutar trenutne generacije određene rang-μ ažuriranjem (engl. *rank-μ update*) važnije kod većih populacija.

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(g)} + c_1 \mathbf{p}_c^{(g+1)} \left(\mathbf{p}_c^{(g+1)}\right)^T + c_\mu \sum_{i=1}^{\mu} \mathbf{y}_{i:\lambda}^{(g+1)} \left(\mathbf{y}_{i:\lambda}^{(g+1)}\right)^T \tag{7.5}$$

$$\mathbf{y}_{i:\lambda}^{(g+1)} = \frac{\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{7.6}$$

Put evolucije (engl. *evolution path*) definira se kao niz uzastopnih koraka strategije kroz određen broj generacija. Put evolucije može biti izražen sumom uzastopnih koraka kao kumulacija (engl. *cumulation*). U praksi se koristi eksponencijalno glađenje te se put evolucije p_c određuje izrazom 7.7.

$$\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{7.7}$$

Veličina koraka σ određena izrazom 7.8 uvodi se za globalno skaliranje distribucije. Za kontrolu veličine koraka koristi se konjugirani evolucijski put p_σ određen izrazom 7.9.

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma^{(g+1)}||}{E||\mathcal{N}(0,\mathbf{I})||} - 1\right)\right) \tag{7.8}$$

$$\mathbf{p}_\sigma^{(g+1)} = (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{eff}}\mathbf{C}^{(g)^{-\frac{1}{2}}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{7.9}$$

Tablica 7.1: Oznake algoritma CMA-ES. [7]

Simbol	Značenje
$\lambda \geq 2$	veličina populacije
$\mu \leq \lambda$	broj roditelja
$w_i \quad i = 1..\mu$	rekombinacijske težine
$\mu_{eff} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}$	selekcijska masa s utjecajem na varijancu
$\mathbf{m}^{(g)} \in \mathbb{R}^{(n)}$	srednja vrijednost razdiobe pretraživanja
$\sigma^{(g)} \in \mathbb{R}^{+}$	veličina koraka
$\mathbf{C}^{(g)} \in \mathbb{R}^{(n \times n)}$	kovarijacijska matrica
$\mathbf{D}^{(g)} \in \mathbb{R}^{(n \times n)}$	dijagonalna matrica korijena svojstvenih vrijednosti **C**
$\mathbf{B}^{(g)} \in \mathbb{R}^{(n \times n)}$	ortogonalna matrica svojstvenih vektora **C** jedinične duljine
$c_c \leq 1$	stopa učenja za kumulaciju za kontrolu rang-1 ažuriranja **C**
$c_1 \leq 1 - c_\mu$	stopa učenja za rang-1 ažuriranje **C**
$c_\mu \leq 1 - c_1$	stopa učenja za rang-μ ažuriranje **C**
$c_\sigma \leq 1$	stopa učenja za kumulaciju za kontrolu ažuriranja σ
$d_\sigma \approx 1$	parametar prigušenja za ažuriranje σ

7.1.1. IPOP-CMA-ES

A. Auger i N. Hansen su 2005. godine predložili inačicu CMA-ES s ponovnim pokretanjem (engl. *restart*) i rastućom populacijom (engl. *increasing population, IPOP*) [1].

Svako ponovno pokretanje je nezavisno i događa se kad je ispunjen jedan od sljedećih kriterija:

– najveća vrijednost dobrote nije se mijenjala tijekom prethodnih $10 + \lceil 30n/\lambda \rceil$ generacija ili se sve vrijednosti dobrote trenutne generacije nalaze unutar intervala duljine `Tolfun=` 10^{-12} (`equalfunvalhist`)

– vrijednost standardne devijacije $\sigma^{(g)}$ te svih komponenata $\sigma^{(g)}\mathbf{p}_c^{(g)}$ manja je od `TolX`$= 10^{-12}\sigma^{(0)}$

– pribrajanje vektora 0.1-standardne devijacije u smjeru glavne osi $\mathbf{C}^{(g)}$ ne mijenja $\mathbf{m}^{(g)}$ odnosno $\mathbf{m}^{(g)} = \mathbf{m}^{(g)} + 0.1\sigma^{(g)}\sqrt{\lambda_i}\mathbf{v}_i$ gdje je $i = (g \mod n) + 1$, a λ_i i u_i su i-ta svojstvena vrijednost i i-ti svojstveni vektor matrice $\mathbf{C}^{(g)}$ (`noeffectaxis`)

– pribrajanje 0.2-standardne devijacije svakoj koordinati ne mijenja $\mathbf{m}^{(g)}$ odnosno $\mathbf{m}^{(g)} = \mathbf{m}^{(g)} + 0.2\sigma^{(g)}\mathbf{m}^{(g)}$ (`noeffectcoord`)

– broj uvjetovanosti $\kappa = \lambda_{max}/\lambda_{min}$ matrice $\mathbf{C}^{(g)}$ prelazi 10^{14} (`conditioncov`)

Porastom populacije nakon svakog pokretanja algoritma pretraživanje postaje sve više globalno, povećava se broj iteracija, ali često i dobrota najboljeg rješenja. Za faktor porasta populacije preporučuju se vrijednosti iz intervala $[1.5, 5]$, uobičajeno 2. Kao konačan kriterij zaustavljanja uzima se unaprijed određen broj iteracija ili vrijednost dobrote.

7.1.2. IPOP-aCMA-ES

N. Hansen i R. Ros su 2010. godine predložili inačicu algoritma IPOP-CMA-ES s negativnom težinskom adaptacijom kovarijacijske matrice [11]. Primjenjuje se devet nezavisnih pokretanja svaki s najviše $100 + 50(D+3)^2/\sqrt{\lambda}$ iteracija. Uvodi se parametar c^- takav da:

– smanjenje c^- s početne vrijednosti ne poboljšava algoritam

– množenje c^- faktorom 2 ne dovodi do neuspjeha algoritma

– broj uvjetovanosti matrice $\mathbf{C}^{(g)}$ ostaje manji od 10

Adaptacija kovarijacijske matrice određena je izrazom 7.10 uz $\mathbf{C}_\mu^+$ i $\mathbf{C}_\mu^-$ određene izrazima 7.11 i 7.12.

$$\begin{aligned}\mathbf{C}^{(g+1)} =& (1 - c_1 - c_\mu + c^-\alpha_{old}^-)\mathbf{C}^{(g)} + c_1\mathbf{p}_c^{(g+1)}\left(\mathbf{p}_c^{(g+1)}\right)^T \\ &+ (c_\mu + c^-(1-\alpha_{old}^-))\mathbf{C}_\mu^+ - c^-\mathbf{C}_\mu^-\end{aligned} \tag{7.10}$$

$$\mathbf{C}_\mu^+ = \sum_{i=1}^{\mu} w_i \frac{\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \frac{(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})^T}{\sigma^{(g)}} \tag{7.11}$$

$$\mathbf{C}_\mu^- = \sum_{i=0}^{\mu-1} w_{i+1} \mathbf{y}_{\lambda-i:\lambda} (\mathbf{y}_{\lambda-i:\lambda})^T$$
$$\text{gdje je} \quad \mathbf{y}_{\lambda-i:\lambda} = \frac{||\mathbf{C}^{(g)^{-\frac{1}{2}}} (\mathbf{x}_{\lambda-\mu+1+i:\lambda} - \mathbf{m}^{(g)})||}{||\mathbf{C}^{(g)^{-\frac{1}{2}}} (\mathbf{x}_{\lambda-i:\lambda} - \mathbf{m}^{(g)})||} \frac{\mathbf{x}_{\lambda-i:\lambda} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{7.12}$$

7.1.3. BIPOP-CMA-ES

N. Hansen je 2009. godine predložio inačicu algoritma CMA-ES s višestrukim pokretanjem (engl. *multistart*) i izmjenom dvaju režima (engl. *BI-Population, BI-POP*) kod ponovnog pokretanja čime se postiže izmjena globalne i lokalne pretrage [8].

Nakon prvog pokretanja s populacijom pretpostavljene veličine $\lambda_{def} = 4 + \lfloor 3 \ln D \rfloor$ ovisno o trenutnom budžetu evaluacija izmjenjuju se režim temeljen na rastućoj populaciji te režim temeljen na maloj populaciji promjenjive veličine. Prvo i zadnje pokretanje se uvijek odvijaju korištenjem prvog režima, a drugi režim se koristi samo ako je njegov budžet manji od zadnjeg budžeta prvog režima.

Kod režima temeljenog na rastućoj populaciji pri svakom ponovnom pokretanju veličina populacije množi se faktorom 2 do maksimalno devet pokretanja. Inicijalno, $\sigma^{(0)} = 2$ odnosno $1/5$ širine prostora pretraživanja. U budžet evaluacija pritom ne ulazi prvo pokretanje s pretpostavljenom veličinom populacije.

Kod režima temeljenog na maloj populaciji promjenjive veličine za veličinu populacije koristi se izraz 7.13 gdje je λ_l zadnja veličina populacije u prvom režimu, a $\mathcal{U}(0,1)$ označava nezavisne uniformno distribuirane brojeve iz intervala $[0,1]$. Time λ_l poprima vrijednosti iz intervala $[\lambda_{def}, \lambda_l/2]$. Inicijalno, $\sigma^{(0)} = 2 \cdot 10^{-2\mathcal{U}(0,1)^2}$. Maksimalan broj evaluacija funkcije postavlja se na polovicu zadnjeg budžeta prvog režima.

$$\lambda_s = \left\lfloor \lambda_{def} \left(\frac{1}{2} \frac{\lambda_l}{\lambda_{def}} \right)^{\mathcal{U}(0,1)^2} \right\rfloor \tag{7.13}$$

Ponovno pokretanje se događa se kad je ispunjen jedan od sljedećih kriterija:

- dosegnut je maksimalan broj iteracija `MaxIter`$= 100 + 50(D+3)^2/\sqrt{\lambda}$
- najveća vrijednost dobrote tijekom prethodnih $10 + \lceil 30n/\lambda \rceil$ generacija nije se promijenila za više od `TolHistFun`$= 10^{-12}$
- u više od $1/3$ zadnjih D generacija dobrote najboljeg i k-tog rješenja gdje je $k = 1 + \lceil 0.1 + \lambda/4 \rceil$ su jednake (`EqualFunVals`)
- sve komponente $\mathbf{p}_c^{(g)}$ i korijeni svih dijagonalnih elemenata $\mathbf{C}^{(g)}$ pomnoženi sa $\sigma^{(g)}/\sigma^{(0)}$ manji su od `TolX`$= 10^{-12}$

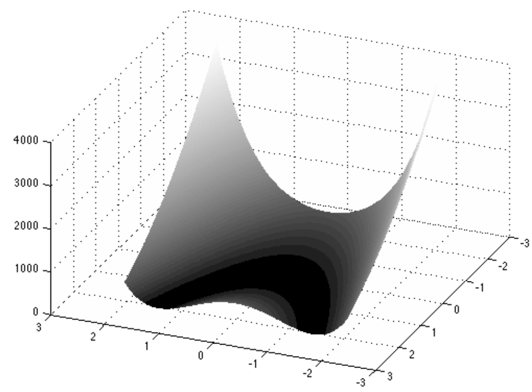
- $\sigma^{(g)}/\sigma^{(0)} >$ (`TolUpSigma`$= 10^{20})\sqrt{l^{(g)}}$ gdje je $l^{(g)}$ najveća svojstvena vrijednost $\mathbf{C}^{(g)}$
- medijan 20 najnovijih vrijednosti nije manji od medijana 20 najstarijih vrijednosti najveće dobrote i srednje vrijednosti dobrote tijekom zadnjih $\lceil 0.2g+120+30D/\lambda\rceil$ (`Stagnation`)
- broj uvjetovanosti $\kappa = \lambda_{max}/\lambda_{min}$ matrice $C^{(g)}$ prelazi 10^{14} (`ConditionCov`)
- pribrajanje vektora 0.1-standardne devijacije u smjeru glavne osi $\mathbf{C}^{(g)}$ ne mijenja $\mathbf{m}^{(g)}$ odnosno $\mathbf{m}^{(g)} = \mathbf{m}^{(g)}+0.1\sigma^{(g)}\sqrt{\lambda_i}\mathbf{v}_i$ gdje je $i = (g \mod n)+1$, a λ_i i u_i su i-ta svojstvena vrijednost i i-ti svojstveni vektor od $\mathbf{C}^{(g)}$ (`NoEffectAxis`)
- pribrajanje 0.2-standardne devijacije svakoj koordinati ne mijenja $\mathbf{m}^{(g)}$ odnosno $\mathbf{m}^{(g)} = \mathbf{m}^{(g)} + 0.2\sigma^{(g)}\mathbf{m}^{(g)}$ (`NoEffectCoor`)

Kao konačan kriterij zaustavljanja uz maksimalan broj pokretanja prvog režima uzima se određen broj iteracija te određena vrijednost dobrote.

Uvodi se i parametar h_σ koji ako vrijedi $||\mathbf{p}_\sigma^{(g+1)}|| < \sqrt{1-(1-c_\sigma)^{2(g+1)}}(1.4 + \frac{2}{D+1})E||\mathcal{N}(0,\mathbf{I})||$ poprima vrijednost 1, inače poprima vrijednost 0. Parametrom h_σ množi se drugi pribrojnik u izrazu 7.7, a faktor prvog pribrojnika u izrazu 7.5 mijenja se u $1 - c_1 - c_\mu + (1-h_\sigma)c_1c_c(2-c_c)$.

7.2. Primjer rada algoritma CMA-ES

Prikazan je rad algoritma CMA-ES na problemu pronalaska globalnog minimuma Rosenbrockove funkcije dvije varijable prikazane na slici 7.2 i opisane u poglavlju 8.2.



Slika 7.2: Rosenbrockova funkcija dvije varijable.

Određivanje parametara

1. korak: određivanje nepromjenjivih parametara

$n = 2$

$\lambda = 4 + \lfloor 3 \ln n \rfloor = 6$

$\mu = \lfloor \frac{\lambda}{2} \rfloor = 3$

$w_i = \frac{\ln(\frac{\lambda+1}{2}) - \ln i}{\sum_{j=1}^{\mu} (\ln(\frac{\lambda+1}{2}) - \ln j)} \Rightarrow \mathbf{w} = \begin{bmatrix} 0.637 & 0.285 & 0.078 \end{bmatrix}$

$\mu_{eff} = \frac{1}{||\mathbf{w}||} = 1.424$

$c_c = \frac{4+\mu_{eff}/n}{n+4+2\mu_{eff}/n} = 0.635$

$c_1 = \frac{2}{(n+1.3)^2+\mu_{eff}} = 0.162$

$c_\mu = \frac{2\mu_{eff}-2+1/\mu_{eff}}{(n+2)^2+\mu_{eff}} = 0.015$

$c_\sigma = \frac{\mu_{eff}+2}{n+\mu_{eff}+5} = 0.406$

$d_\sigma = 1 + c_\sigma + 2\max(0, \sqrt{\frac{\mu_{eff}-1}{n+1}} - 1) = 1.406$

$E||\mathcal{N}(0, \mathbf{I})|| = 1.254$

2. korak: inicijalizacija promjenjivih parametara neovisnih o problemu

$\mathbf{p}_\sigma^{(0)} = \begin{bmatrix} 0 & 0 \end{bmatrix}$

$\mathbf{p}_c^{(0)} = \begin{bmatrix} 0 & 0 \end{bmatrix}$

$\mathbf{C}^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mathbf{B}^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mathbf{D}^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mathbf{C}^{(0)^{-\frac{1}{2}}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

3. korak: inicijalizacija promjenjivih parametara ovisnih o problemu

$\mathbf{m}^{(0)} = \begin{bmatrix} -0.7 & -1.05 \end{bmatrix}$

$\sigma^{(0)} = 0.8192$

Prva iteracija

1. korak: generiranje populacije multivarijatnom normalnom distribucijom prema izrazu 7.3. Generirana populacija se sortira te je prikazana u tablici 7.2.

2. korak: adaptacija $\mathbf{m}$ prema izrazu 7.4. Računanjem težinske sredine genotipa jedinki 0, 1 i 2 iz tablice 7.2 dobije se $\mathbf{m}^{(1)} = \begin{bmatrix} -0.46 & -0.76 \end{bmatrix}$.

3. korak: adaptacija $\mathbf{C}$ prema izrazu 7.5. Iz izraza 7.7 dobije se $\mathbf{p}_c^{(1)} = \begin{bmatrix} 0.33 & 0.4 \end{bmatrix}$ čime se dobije $\mathbf{C}^{(1)} = \begin{bmatrix} 0.844 & 0.023 \\ 0.023 & 0.866 \end{bmatrix}$. Dekompozicijom $C^{(1)}$ na svojstvene vrijednosti

dobije se $\mathbf{B}^{(1)} = \begin{bmatrix} -0.534 & 0.845 \\ -0.845 & -0.534 \end{bmatrix}$, $\mathbf{D}^{(1)} = \begin{bmatrix} 0.94 & 0 \\ 0 & 0.91 \end{bmatrix}$ i $\mathbf{C}^{(1)^{-\frac{1}{2}}} = \begin{bmatrix} 1.089 & -0.015 \\ -0.015 & 1.075 \end{bmatrix}$.

4. korak: adaptacija σ prema izrazu 7.8. Iz izraza 7.9 dobije se $\mathbf{p}_\sigma^{(1)} = \begin{bmatrix} 0.28 & 0.34 \end{bmatrix}$ čime se dobije $\sigma^{(1)} = 0.68$.

Tablica 7.2: Populacija algoritma CMA-ES tijekom 1. iteracije.

Redni broj jedinke	Genotip	$f(x)$
0	$\begin{bmatrix} -0.373 & -0.151 \end{bmatrix}$	10.28
1	$\begin{bmatrix} -0.340 & -2.048 \end{bmatrix}$	469.8
2	$\begin{bmatrix} -1.625 & -1.013 \end{bmatrix}$	1341.1
3	$\begin{bmatrix} -1.524 & -1.369 \end{bmatrix}$	1369.9
4	$\begin{bmatrix} -1.450 & -2.048 \end{bmatrix}$	1727.5
5	$\begin{bmatrix} -1.650 & -1.733 \end{bmatrix}$	1991.9

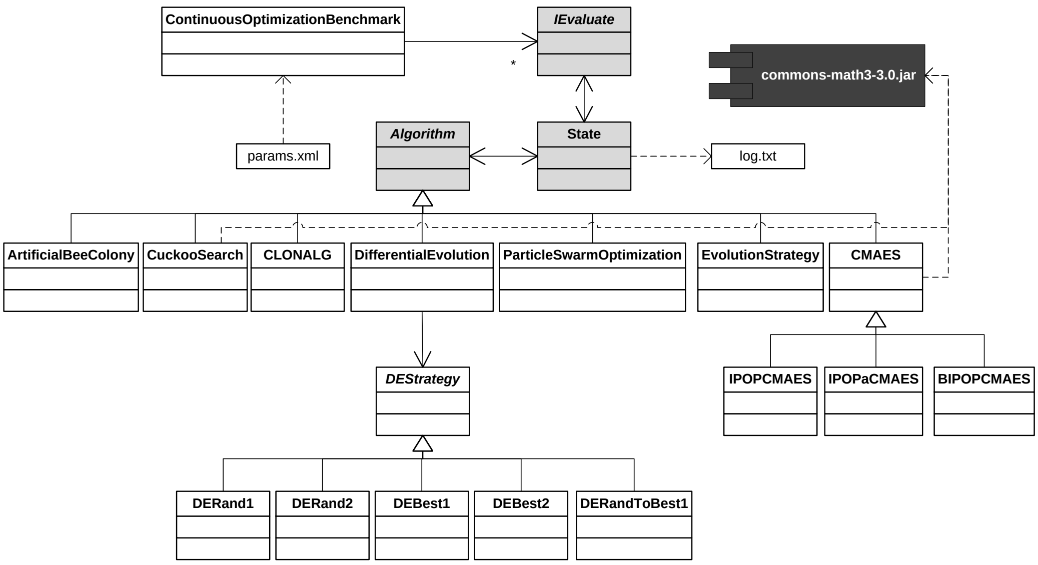
U sljedećim iteracijama ponavljaju se koraci od $1.$ do $4.$. U $49.$ iteraciji populacija se približila globalnom optimumu što je vidljivo u tablici 7.3. Nakon $49.$ iteracije populacija počinje konvergirati prema globalnom optimumu. Globalni optimum 0 za genotip $\begin{bmatrix} 1 & 1 \end{bmatrix}$ pronađen je u $224.$ iteraciji, a od $250.$ iteracije cijela se populacija nalazi u globalnom optimumu što dovodi do stagnacije algoritma.

Tablica 7.3: Populacija algoritma CMA-ES tijekom 49. iteracije.

Redni broj jedinke	Genotip	$f(x)$
0	$\begin{bmatrix} 0.975 & 0.949 \end{bmatrix}$	$1.08 \cdot 10^{-3}$
1	$\begin{bmatrix} 1.000 & 0.995 \end{bmatrix}$	$2.11 \cdot 10^{-3}$
2	$\begin{bmatrix} 1.002 & 0.998 \end{bmatrix}$	$2.80 \cdot 10^{-3}$
3	$\begin{bmatrix} 0.978 & 0.951 \end{bmatrix}$	$3.37 \cdot 10^{-3}$
4	$\begin{bmatrix} 0.958 & 0.914 \end{bmatrix}$	$3.96 \cdot 10^{-3}$
5	$\begin{bmatrix} 0.962 & 0.919 \end{bmatrix}$	$5.66 \cdot 10^{-3}$

8. Programsko ostvarenje

Svi algoritmi ostvareni su u okviru radnog okruženja za evolucijsko računanje (engl. *Evolutionary Computation Framework, ECF*). Pojednostavljena organizacija programskog ostvarenja prikazana je na slici 8.1. Svijetlo sivom bojom označen je dio radnog okruženja ECF, a tamno sivom bojom je označena biblioteka *Commons Math* dostupna na `http://commons.apache.org/math/`.



Slika 8.1: Organizacija programskog ostvarenja.

U razredu `ContinuousOptimizationBenchmark` ostvarene su ispitne funkcije. Svi razredi kojima su ostvarene ispitne funkcija implementiraju ECF sučelje `IEvaluate`. Temeljna metoda koju definiraju ti razredi je metoda *evaluate* koja računa vrijednost ispitne funkcije, odnosno definira objekt ECF razreda `MinFitness` jedinke ostvarene ECF razredom `Individual` s obzirom na to da se radi o problemu minimizacije funkcije. Iz metode *run* inicijalizira se stanje ostvareno ECF razredom `State` čijem se konstruktoru predaju argumenti komandne linije odnosno put do datoteke s parametrima algoritma. Stanje algoritma čine podaci o parametrima, algoritmu, populaciji, korištenim genotipovima i operatorima i slično. Tijekom izvođenja algoritma iz tih se podataka generira log datoteka. Stanje se inicijalizira parsiranjem datoteke s parametrima algoritma nakon čega se inicijalizira i pokreće konkretni algoritam. Svi algoritmi nasljeđuju apstraktni razred `Algorithm`. Također, svi algoritmi kao parametar primaju samo jedan genotip s vektorom brojeva s pomičnim zarezom ostvaren ECF razredom `FloatingPoint` te koriste ECF parametre *population.size* za veličinu populacije i *term.maxgen* za maksimalan broj generacija.

8.1. Algoritmi

Razredom `ArtificialBeeColony` ostvaren je algoritam kolonije pčela opisan u poglavlju 2. Jedini parametar algoritma je *limit* koji odgovara istoimenom opisanom parametru. Svaka jedinka ima i dva dodatna `FloatingPoint` genotipa za pohranu vjerojatnosti odabira jedinke u fazi pčela izviđača te za pohranu broja generacija tijekom kojeg se nije poboljšala vrijednost dobrote jedinke. Za slučajan odabir jedinke iz populacije kod odabira susjeda u fazi zaposlenih pčela i fazi pčela izviđača koristi se objekt ECF razreda `SelRandomOp` dok se za dohvat najbolje jedinke i njene dobrote koristi objekt ECF razreda `SelBestOp`.

Razredom `CuckooSearch` ostvaren je algoritam kukavičje pretrage opisan u poglavlju 3. Parametri algoritma *pa*, *alpha* i *beta* odgovaraju opisanim parametrima p_a, α i β. Za slučajan odabir jedinke iz populacije kod odabira gnijezda kukavice koristi se objekt ECF razreda `SelRandomOp` dok se za dohvat najbolje jedinke koristi objekt ECF razreda `SelBestOp`. Algoritam ovisi o biblioteci *Commons Math*.

Razredom `CLONALG` ostvaren je algoritam klonske selekcije opisan u poglavlju 4. Ostvarena je hipermutacija proporcionalna rangu slična inverzno proporcionalnoj hipermutaciji, selekcija $CLONALG_2$ te statičko starenje. Parametri algoritma *n*, *beta*, *c* i *tauB* odgovaraju opisanim parametrima n, β, c i τ_B. Svaka jedinka ima i dodatni `FloatingPoint` genotip za pohranu starosti jedinke.

Razredom `DifferentialEvolution` ostvarena je diferencijska evolucija opisana u poglavlju 5. Parametri algoritma *F* i *CR* odgovaraju istoimenim opisanim parametrima dok je parametrom *DEStrategy* definirano ime razreda kojim je ostvarena strategija pretraživanja. Strategije pretraživanja ostvarene su razredima `DERand1`, `DERand2`, `DEBest1`, `DEBest2` i `DERandToBest1` koji implementiraju sučelje `DEStrategy`. Za slučajan odabir jedinke iz populacije kod mutacije koristi se objekt ECF razreda `SelRandomOp` dok se za dohvat najbolje jedinke kod mutacije koristi objekt ECF razreda `SelBestOp`.

Razredom `ParticleSwarmOptimization` ostvaren je algoritam roja čestica opisan u poglavlju 6. Ostvareno je tradicionalno ažuriranje brzine s potpuno povezanom topologijom uz mogućnost odabira promjenjivog faktora inercije. Parametar *wType* poprima vrijednost 0 za konstantan faktor inercije odnosno 1 za promjenjiv faktor inercije. Parametri algoritma *C1*, *C2* i *w* odgovaraju opisanim parametrima C_1, C_2 i ω, a parametrom *vMax* definira se maksimalna dozvoljena vrijednost komponenata brzine. Ako je odabran promjenjiv faktor inercije potrebno je definirati i parametar *wMin* koji određuje minimalnu vrijednost faktora inercije, te ECF parametar *term.maxgen* koji određuje broj generacija tijekom kojeg se faktor inercije mijenja. Svaka jedinka ima i tri dodatna `FloatingPoint` genotipa za pohranu brzine jedinke, najboljeg položaja jedinke te najveće dobrote jedinke.

Razredom `EvolutionStrategy` ostvarena je evolucijska strategija opisana u poglavlju 7. Ostvarena je inačica $(\mu/1+\lambda)$-ES. Parametar algoritma *lambda* odgovara opisanom parametru λ, a ECF parametar *population.size* odgovara opisanom parametru μ. Svaka jedinka ima i dodatni `FloatingPoint` genotip za pohranu strategije jedinke.

Razredom `CMAES` ostvarena je adaptacija evolucijske strategije kovarijacijskom matricom opisana u poglavlju 7.1. Opisane inačice ostvarene su razredima `IPOPCMAES`, `IPOPaCMAES` i `BIPOPCMAES` koji nasljeđuju razred `CMAES`. Algoritmi ne primaju niti jedan parametar, a ECF parametar *population.size* odgovara opisanom parametru λ. Algoritmi ovise o biblioteci *Commons Math*.

8.2. Ispitne funkcije

Za potrebe ispitivanja algoritama implementirano je nekoliko ispitnih funkcija [16].

Ackleyeva funkcija

Funkcija je ostvarena razredom `Ackley`. Funkcija je određena izrazom 8.1.
Uobičajen prostor pretraživanja je $-32.768 \leq x_i \leq 32.768$, a globalni optimum $f_1(\mathbf{x}) = 0$ postiže se za $\mathbf{x} = (0.0, 0.0, \ldots, 0.0)$.

$$f_1(\mathbf{x}) = -20 \cdot e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e \tag{8.1}$$

Griewangkova funkcija

Funkcija je ostvarena razredom `Griewangk`. Funkcija je određena izrazom 8.2. Uobičajen prostor pretraživanja je $-600 \leq x_i \leq 600$, a globalni optimum $f_2(\mathbf{x}) = 0$ postiže se za $\mathbf{x} = (0.0, 0.0, \ldots, 0.0)$.

$$f_2(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{8.2}$$

Rastriginova funkcija

Funkcija je ostvarena razredom `Rastrigin`. Funkcija je određena izrazom 8.3. Uobičajen prostor pretraživanja je $-5.12 \leq x_i \leq 5.12$, a globalni optimum $f_3(\mathbf{x}) = 0$ postiže se za $\mathbf{x} = (0.0, 0.0, \ldots, 0.0)$.

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i)) \tag{8.3}$$

Rosenbrockova funkcija

Funkcija je ostvarena razredom `Rosenbrock`. Funkcija je određena izrazom 8.4. Uobičajen prostor pretraživanja je $-2.048 \leq x_i \leq 2.048$, a globalni optimum $f_4(\mathbf{x}) = 0$ postiže se za $\mathbf{x} = (1.0, 1.0, \ldots, 1.0)$.

$$f_4(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right) \tag{8.4}$$

Schwefelova funkcija

Funkcija je ostvarena razredom `Schwefel`. Funkcija je određena izrazom 8.5. Uobičajen prostor pretraživanja je $-500 \leq x_i \leq 500$, a globalni optimum $f_4(\mathbf{x}) = -418.9829n$ postiže se za $\mathbf{x} = (420.9687, 420.9687, \ldots, 420.9687)$.

$$f_5(\mathbf{x}) = \sum_{i=1}^{n} (-x_i \sqrt{\sin |x_i|}) \tag{8.5}$$

8.3. Primjeri i rezultati ispitivanja

Ispitano je ponašanje svih algoritama na problemu pronalaska globalnog minimuma ispitnih funkcijama iz prethodnog poglavlja kod dimenzija $n = 5$ i $n = 20$. Ispitni parametri algoritma definirani su tablicom 8.1 prema uputama iz literature. Parametri `FloatingPoint` genotipa $lbound$ i $ubound$ postavljeni su prema uobičajenim granicama prostora pretraživanja. Svi algoritmi testirani su s najviše $term.maxgen = 2000$ generacija i populacijom veličine $population.size = 20$, osim CMA-ES i njegovih inačica koji su testirani s populacijom veličine $population.size = 4 + \lfloor 3 \ln D \rfloor$ odnosno 8 za $n = 5$ i 12 za $n = 20$.

Tablica 8.1: Ispitni parametri algoritama

Algoritam	Parametar	Vrijednost
ABC	*limit*	100
CS	*pa*	0.25
	alpha	$0.01(ubound - lbound)$
	beta	1.5
CLONALG	*n*	20
	beta	0.2
	c	0.2
	tauB	30
DE	*F*	0.5
	CR	0.5
	DEStrategy	$DERandToBest1$
PSO	*C1*	2.0
	C2	2.0
	wType	1
	w	0.9
	wMin	0.4
	vMax	$0.2(ubound - lbound)$
ES	*lambda*	40

U tablicama 8.2 i 8.3 prikazan je najbolji rezultat od 10 pokrenutih testova za svaki algoritam. Ukupno najbolje rezultate postižu algoritmi IPOP-CMA-ES, IPOPa-CMA-ES i BIPOP-CMA-ES koji jedini uspijevaju pronaći globalno optimalno rješenje za Rosenbrockovu funkciju za dimenziju 20 sa zadovoljavajućom preciznošću. Vrlo dobra rješenja daje i algoritam ABC koji jedini pronalazi globalni optimum Schwefelove funkcije za dimenziju 20. Učinkovitost većine algoritama, a pogotovo algoritma ES pogoršava se s porastom dimenzije. Kod optimiranja Ackleyeve funkcije za obje dimenzije većina je algoritama uspješna.

Tablica 8.2: Rezultati ispitivanja za $n = 5$.

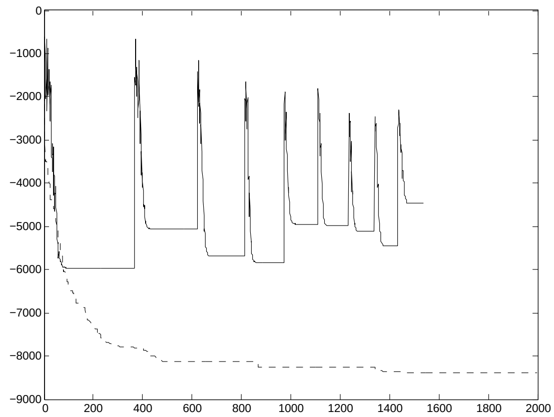
Algoritam	f_1	f_2	f_3	f_4	f_5
ABC	$3.99 \cdot 10^{-15}$	$3.7 \cdot 10^{-13}$	0	0.036	-2094.91
CS	$4.44 \cdot 10^{-16}$	0.012	0	0.07	-2094.91
CLONALG	$2.8 \cdot 10^{-6}$	0.015	$9.02 \cdot 10^{-12}$	0.052	-1976.48
DE	$4.44 \cdot 10^{-16}$	0.015	0	0	-2094.91
PSO	$3.99 \cdot 10^{-15}$	0	0	0.06	-1976.48
ES	$3.99 \cdot 10^{-15}$	0.015	0.99	$7.78 \cdot 10^{-4}$	-1976.48
CMA-ES	$3.99 \cdot 10^{-15}$	0.012	0.99	$9.12 \cdot 10^{-30}$	-1621.16
IPOP-CMA-ES	$1.14 \cdot 10^{-13}$	$5.55 \cdot 10^{-16}$	0	$2.48 \cdot 10^{-15}$	-2094.91
IPOP-aCMA-ES	$2.66 \cdot 10^{-12}$	$1.55 \cdot 10^{-14}$	$2.3 \cdot 10^{-14}$	$2.07 \cdot 10^{-15}$	-2094.91
BIPOP-CMA-ES	$8.57 \cdot 10^{-14}$	$1.77 \cdot 10^{-15}$	0	$9.20 \cdot 10^{-16}$	-2094.91

Tablica 8.3: Rezultati ispitivanja za $n = 20$.

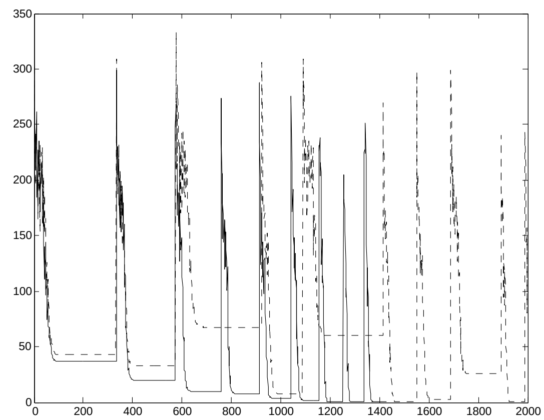
Algoritam	f_1	f_2	f_3	f_4	f_5
ABC	$3.95 \cdot 10^{-14}$	$1.64 \cdot 10^{-12}$	0	0.16	-8379.66
CS	$4.88 \cdot 10^{-5}$	0.72	95.33	7.86	-5295.41
CLONALG	0.002	$1.08 \cdot 10^{-4}$	$2.35 \cdot 10^{-5}$	3.38	-7669.03
DE	$7.55 \cdot 10^{-15}$	1.18	4.13	19.7	-7688.76
PSO	$1.72 \cdot 10^{-10}$	0.02	8.95	14.3	-7429.12
ES	1.98	0.81	132.12	15.2	-5538.98
CMA-ES	$3.99 \cdot 10^{-15}$	0	35.82	0.18	-6009.32
IPOP-CMA-ES	$2.18 \cdot 10^{-12}$	$8.33 \cdot 10^{-15}$	$4.80 \cdot 10^{-14}$	$5.19 \cdot 10^{-5}$	-6149.02
IPOP-aCMA-ES	$1.73 \cdot 10^{-11}$	$9.66 \cdot 10^{-15}$	$5.32 \cdot 10^{-14}$	$5.95 \cdot 10^{-13}$	-6149.02
BIPOP-CMA-ES	$1.43 \cdot 10^{-10}$	$5.44 \cdot 10^{-15}$	$2.23 \cdot 10^{-13}$	$7.50 \cdot 10^{-4}$	-6543.83

Slika 8.2 prikazuje promjenu vrijednosti funkcije najboljeg rješenja u populaciji kroz generacije algoritama ABC i IPOP-aCMA-ES za Schwefelovu funkciju dimenzije 20. Vidljivo je kako ABC s lakoćom pronalazi globalni optimum dok IPOP-aCMA-ES tijekom svakog ponovnog pokretanja konvergira prema nekom od lokalnih optimuma.

Slika 8.3 prikazuje promjenu vrijednosti funkcije najboljeg rješenja u populaciji kroz generacije algoritama BIPOP-CMA-ES i IPOP-aCMA-ES za Rastriginovu funkciju dimenzije 20. Vidljivo je kako kod algoritma IPOP-aCMA-ES porastom populacije pretraživanje postaje više globalno i algoritam konvergira prema globalnom umjesto prema lokalnim optimumima. Slično ponašanje pokazuje i algoritam BIPOP-CMA-ES s time da kod trećeg, petog i osmog ponovnog pokretanja dolazi do konvergencije prema lokalnim optimumima jer se radi o pokretanjima s malom veličinom populacije.



Slika 8.2:
Usporedba algoritama ABC (isprekidana linija) i IPOP-aCMA-ES (puna linija) na f_5 za $n = 20$.



Slika 8.3:
Usporedba algoritama BIPOP-CMA-ES (isprekidana linija) i IPOP-aCMA-ES (puna linija) na f_3 za $n = 20$.

9. Zaključak

Metaheuristike zbog svoje relativno niske računske složenosti predstavljaju snažan alat za optimizaciju problema velike složenosti koji se ne mogu riješiti analitički ili primjenom iscrpne pretrage. U radu je opisano nekoliko metaheuristika iz područja evolucijskog računanja: algoritam kolonije pčela, algoritam kukavičje pretrage, algoritam klonske selekcije, diferencijska evolucija, algoritam roja čestica te evolucijske strategije. Navedeni algoritmi ostvareni su u okviru radnog okruženja za evolucijsko računanje, izložena je njihova implementacija te su ispitani na nekoliko višemodalnih ispitnih funkcija. Inačice evolucijskih strategija IPOP-CMA-ES, IPOP-aCMA-ES i BIPOP-CMA-ES, iako nešto računski zahtjevnije od ostalih algoritama, opravdale su status široko priznatih algoritama za optimizaciju problema s kontinuiranom domenom, osobito na Rosenbrockovoj funkciji gdje ostali algoritmi nisu uspjeli konvergirati prema globalnom optimumu niti kada su našli rješenja u njegovoj blizini. Posebno je zanimljiv utjecaj ponovnog pokretanja i veličine populacije na konvergenciju tih algoritama. Od ostalih algoritama istaknuo se algoritam ABC koji je jedini za veću dimenziju pronašao globalni optimum Schwefelove funkcije koja jedina ima neravnomjerno raspoređene lokalne optimume uz rub prostora pretraživanja. Ipak, navedene rezultate treba uzeti s rezervom jer parametri nisu prilagođeni specifičnom optimizacijskom problemu no daju okvirnu sliku osobina algoritama. U budućnosti bi ostvarene algoritme trebalo optimirati te ispitati na većem skupu ispitnih optimizacijskih problema kako bi se statističkom obradom rezultata mogla dobiti detaljnija slika njihovih dobrih i loših osobina.

LITERATURA

[1] A. Auger i N. Hansen. A Restart CMA Evolution Strategy With Increasing Population Size. U *IEEE Congress on Evolutionary Computation, CEC 2005, Proceedings*, 2005. URL `http://www.lri.fr/~hansen/cec2005ipopcmaes.pdf`.

[2] H.-G. Beyer i H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 2002. URL `http://www.springerlink.com/content/2311qapbrwgrcyey/fulltext.pdf`.

[3] V. Cutello, G. Narzisi, G. Nicosia, i M. Pavone. Clonal Selection Algorithms: A Comparative Case Study Using Effective Mutation Potentials. U *Artificial Immune Systems: 4^{th} International Conference, ICARIS 2005, Proceedings*, 2005. URL `http://www.cs.unict.it/~nicosia/papers/conferences/Nicosia-ICARIS05-optIAvsCLONALG.pdf`.

[4] L.N. de Castro i F.J. Von Zuben. The Clonal Selection Algorithm with Engineering Applications. U *Proceedings of the Genetic and Evolutionary Computation Conference*, 2000. URL `http://www.dca.fee.unicamp.br/~vonzuben/research/lnunes_dout/artigos/gecco00.pdf`.

[5] M.A. Montes de Oca, T. Stützle, M. Birattari, i M. Dorigo. Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 2009. URL `http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2007-006r002.pdf`.

[6] R.C. Eberhart i J. Kennedy. A New Optimizer Using Particle Swarm Theory. U *Proceedings of the 6^{th} International Symposium on Micro Machine and Human Science*, 1995. URL `http://140.133.35.1/faculty/pwu/heuristic/PSO_2.pdf`.

[7] N. Hansen. The CMA evolution strategy: a comparing review. U *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*. 2006. URL http://www.lri.fr/~hansen/hansenedacomparing.pdf.

[8] N. Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. U *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, 2009. URL http://hal.inria.fr/docs/00/38/20/93/PDF/hansen2009bbi.pdf.

[9] N. Hansen i A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. U *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, 1996. URL http://www.lri.fr/~hansen/CMAES.pdf.

[10] N. Hansen i A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 2001. URL http://www.lri.fr/~hansen/cmaartic.pdf.

[11] N. Hansen i R. Ros. Benchmarking a Weighted Negative Covariance Matrix Update on the BBOB-2010 Noiseless Testbed. U *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, 2010. URL http://hal.archives-ouvertes.fr/docs/00/54/57/36/PDF/ws1p32-hansen.pdf.

[12] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, 2005. URL http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf.

[13] D. Karaboga i B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 2007. URL http://sci2s.ugr.es/eamhco/pdfs/ABC-algorithm-numerical-function-2007.pdf.

[14] J. Kennedy i R.C. Eberhart. Particle Swarm Optimization. U *Proceedings of the IEEE International Conference on Neural Networks*, 1995. URL http://140.133.35.1/faculty/pwu/heuristic/PSO_1.pdf.

[15] J. Kennedy, R.C. Eberhart, i Y. Shi. *Swarm Intelligence*. 2001.

[16] M. Molga i C. Smutnicki. Test functions for optimization needs, 2005. URL `http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf`.

[17] R. Storn i K. Price. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, 1995. URL `http://www.icsi.berkeley.edu/~storn/TR-95-012.pdf`.

[18] E. Talbi. *Metaheuristics: From Design to Implementation*. 2009.

[19] X.-S. Yang i S. Deb. Cuckoo search via Lévy flights. U *Proceedings of World Congress on Nature & Biologically Inspired Computing*, 2009. URL `http://arxiv.org/pdf/1003.1594.pdf`.

[20] X.-S Yang i S. Deb. Engineering Optimisation by Cuckoo Search. *nternational Journal of Mathematical Modelling and Numerical Optimisation*, 2010. URL `http://arxiv.org/pdf/1005.2908v3.pdf`.

Radno okruženje za ispitivanje metaheuristika

Sažetak

Metaheuristike se primjenjuju na širok skup optimizacijskih problema velike složenosti koji nisu rješivi uporabom tradicionalnih pristupa. U središtu interesa ovog rada nalaze se prirodom inspirirane metaheuristike iz područja evolucijskog računanja. Dana je kratka teorijska pozadina te pregled inačica algoritma umjetne kolonije pčela, algoritma kukavičje pretrage, algoritma klonske selekcije, diferencijske evolucije, algoritma roja čestica te evolucijskih strategija. Posebna pažnja je posvećena inačici evolucijske strategije CMA-ES koja danas predstavlja jednu od široko priznatih optimizacijskih tehnika za kontinuirane probleme. Navedene metaheuristike ostvarene su u okviru radnog okruženja za evolucijsko računanje u Javi te je uspoređeno njihovo ponašanje na nekoliko ispitnih funkcija iz literature.

Ključne riječi: prirodom inspirirane metaheuristike, kontinuirani optimizacijski problemi, radno okruženje za evolucijsko računanje

Metaheuristics Evaluation Framework

Abstract

Metaheuristic optimization algorithms have become a popular choice for solving intractable optimization problems which are difficult to solve using traditional methods. In this thesis focus is set on solving real-valued problems using nature-inspired metaheuristics in the field of evolutionary computation. A brief theoretical background and an overview of variants of some of them is given, namely artificial bee colony, cuckoo search, clonal selection algorithm, differential evolution, particle swarm optimization and evolution strategies including CMA-ES which represents the state-of-the-art in evolutionary optimization of real-valued problems. These metaheuristics are implemented using evolutionary computation framework in Java and their performance is compared on a few benchmark functions.

Keywords: nature-inspired metaheuristics, continuous optimization problems, Evolutionary Computation Framework