Nelder-Meadov algoritam s heuristikama

Mateo Dujić, Matija Šantek
mentor: doc. dr. sc. Goranka Nogo

Prirodoslovno-matematički fakultet

24. siječanj, 2022

Sadržaj

Sadržaj

Općenito

- Nelder-Meadov algoritam dizajniran je za rješavanje optimizacijskog problema minimizacije nelinearne funkcije $f : \mathbb{R}^n \to \mathbb{R}$

Općenito

- Nelder-Meadov algoritam dizajniran je za rješavanje optimizacijskog problema minimizacije nelinearne funkcije $f : \mathbb{R}^n \to \mathbb{R}$
- metoda koristi samo funkcijske vrijednosti u nekim točkama iz $\mathbb{R}^n$

Općenito

- Nelder-Meadov algoritam dizajniran je za rješavanje optimizacijskog problema minimizacije nelinearne funkcije $f : \mathbb{R}^n \to \mathbb{R}$
- metoda koristi samo funkcijske vrijednosti u nekim točkama iz $\mathbb{R}^n$
- ne pokušava računati približnu vrijednost gradijenta na ikojoj od tih točaka

Sadržaj

Nelder-Meadov algoritam

- algoritam je baziran na simpleksima

Nelder-Meadov algoritam

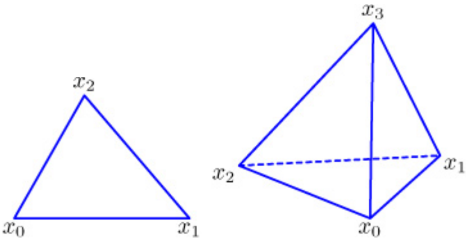
- algoritam je baziran na simpleksima
- simplex $S \in \mathbb{R}^n$ je konveksna ljuska $n+1$ vrhova $x_0, \ldots, x_n \in \mathbb{R}^n$

Nelder-Meadov algoritam

- algoritam je baziran na simpleksima
- simplex $S \in \mathbb{R}^n$ je konveksna ljuska $n+1$ vrhova $x_0, \ldots, x_n \in \mathbb{R}^n$
- npr. simpleks u $\mathbb{R}^2$ je trokut, a u $\mathbb{R}^3$ je tetraedar

Nelder-Meadov algoritam

- algoritam je baziran na simpleksima
- simplex $S \in \mathbb{R}^n$ je konveksna ljuska $n+1$ vrhova $x_0, \ldots, x_n \in \mathbb{R}^n$
- npr. simpleks u $\mathbb{R}^2$ je trokut, a u $\mathbb{R}^3$ je tetraedar



Slika: simpleksi u $\mathbb{R}^2$ i $\mathbb{R}^3$

Opis algoritma

- Pretraživanje započinje s $n+1$ točkom koje se smatraju vrhovima radnog simpleksa i pripadajućim funkcijskim vrijednostima u tim točkama $f_j = f(x_j)$, $j = 0, \dots, n$

Opis algoritma

- Pretraživanje započinje s $n+1$ točkom koje se smatraju vrhovima radnog simpleksa i pripadajućim funkcijskim vrijednostima u tim točkama $f_j = f(x_j)$, $j = 0, \ldots, n$
- Vrhovi početnog simpleksa ne smiju pripadati istoj hiperravnini

Opis algoritma

- Pretraživanje započinje s $n+1$ točkom koje se smatraju vrhovima radnog simpleksa i pripadajućim funkcijskim vrijednostima u tim točkama $f_j = f(x_j)$, $j = 0, \ldots, n$
- Vrhovi početnog simpleksa ne smiju pripadati istoj hiperravnini
- Metoda tada vrši niz transformacija radnog simpleksa S, koje su usmjerene *smanjivanju* funkcijskih vrijednosti vrhova

Opis algoritma

- Pretraživanje započinje s $n+1$ točkom koje se smatraju vrhovima radnog simpleksa i pripadajućim funkcijskim vrijednostima u tim točkama $f_j = f(x_j)$, $j = 0, \ldots, n$
- Vrhovi početnog simpleksa ne smiju pripadati istoj hiperravnini
- Metoda tada vrši niz transformacija radnog simpleksa S, koje su usmjerene *smanjivanju* funkcijskih vrijednosti vrhova
- U svakom koraku, transformacije su određene računanjem jednog ili više novih testnih vrhova koje se uspoređuju s već izračunatim vrhovima na simpleksu

Opis algoritma

- Pretraživanje započinje s $n+1$ točkom koje se smatraju vrhovima radnog simpleksa i pripadajućim funkcijskim vrijednostima u tim točkama $f_j = f(x_j)$, $j = 0, \dots, n$
- Vrhovi početnog simpleksa ne smiju pripadati istoj hiperravnini
- Metoda tada vrši niz transformacija radnog simpleksa S, koje su usmjerene *smanjivanju* funkcijskih vrijednosti vrhova
- U svakom koraku, transformacije su određene računanjem jednog ili više novih testnih vrhova koje se uspoređuju s već izračunatim vrhovima na simpleksu
- Proces se završava kada radni simplex S postane dovoljno malen u nekom smislu ili kada funkcijske vrijednosti f_j budu dovoljno blizu u nekom smislu (dano je da su f s kojima radimo neprekidne)

Psudokod

Nelder-Meadov algoritam

1: **procedure** SimplexLocalSearch(početna točka, λ, parametri za uvjete zaustavljanja)

Psudokod

Nelder-Meadov algoritam

procedure SIMPLEXLOCALSEARCH(početna točka, λ, parametri za uvjete zaustavljanja)
    konstruiraj simpleks iz dane početne točke i λ
    while nisu ispunjeni uvjeti zaustavljanja **do**
        izračunaj info za zaustavljanje
        transformiraj simpleks
    end while
    return najbolji vrh simpleksa
end procedure

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:
  1. Poredamo vrhove. Označimo indekse h, s, l kao najgori (highest), drugi najgori (second highest) i najbolji (lowest) vrh, redom. Oznake su onda:

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:
  1. Poredamo vrhove. Označimo indekse h, s, l kao najgori (highest), drugi najgori (second highest) i najbolji (lowest) vrh, redom. Oznake su onda:

$$f_h = \max_j f_j, \qquad f_s = \max_{j \neq h} f_j, \qquad f_l = \min_j f_j.$$

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:
  1. Poredamo vrhove. Označimo indekse h, s, l kao najgori (highest), drugi najgori (second highest) i najbolji (lowest) vrh, redom. Oznake su onda:

$$f_h = \max_j f_j, \qquad f_s = \max_{j \neq h} f_j, \qquad f_l = \min_j f_j.$$

  2. Izračunamo središte c najbolje strane - to je ona nasuprot najgorem vrhu x_h:

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:
  1. Poredamo vrhove. Označimo indekse h, s, l kao najgori (highest), drugi najgori (second highest) i najbolji (lowest) vrh, redom. Oznake su onda:

$$f_h = \max_j f_j, \qquad f_s = \max_{j \neq h} f_j, \qquad f_l = \min_j f_j.$$

  2. Izračunamo središte c najbolje strane - to je ona nasuprot najgorem vrhu x_h:

$$c := \frac{1}{n} \sum_{j \neq h} x_j.$$

Transformacije simpleksa

- Jedna iteracija Nelder-Meadova algoritma sastoji se od sljedeća tri koraka:
  1. Poredamo vrhove. Označimo indekse h, s, l kao najgori (highest), drugi najgori (second highest) i najbolji (lowest) vrh, redom. Oznake su onda:
  $$f_h = \max_j f_j, \qquad f_s = \max_{j \neq h} f_j, \qquad f_l = \min_j f_j.$$
  2. Izračunamo središte c najbolje strane - to je ona nasuprot najgorem vrhu x_h:
  $$c := \frac{1}{n} \sum_{j \neq h} x_j.$$
  3. Primijenimo odgovarajuću transformaciju: izračunamo novi radni simpleks iz prethodnog.

Glavna ideja transformacija

- Prvo, pokušajmo zamijeniti samo najgori vrh x_h boljim vrhom koristeći refleksiju, ekspanziju ili kontrakciju u odnosu na najbolju stranu.

Glavna ideja transformacija

- Prvo, pokušajmo zamijeniti samo najgori vrh x_h boljim vrhom koristeći refleksiju, ekspanziju ili kontrakciju u odnosu na najbolju stranu.
  - Uočimo da sada sve testne točke leže na pravcu koji prolazi kroz točke x_h i c te **najviše dvije** se računaju u jednoj iteraciji.

Glavna ideja transformacija

- Prvo, pokušajmo zamijeniti samo najgori vrh x_h boljim vrhom koristeći refleksiju, ekspanziju ili kontrakciju u odnosu na najbolju stranu.
  - Uočimo da sada sve testne točke leže na pravcu koji prolazi kroz točke x_h i c te **najviše dvije** se računaju u jednoj iteraciji.
- Ako ovo uspije, prihvaćena točka postaje novi vrh radnog simpleksa i transformacija je gotova.

Glavna ideja transformacija

- Prvo, pokušajmo zamijeniti samo najgori vrh x_h boljim vrhom koristeći refleksiju, ekspanziju ili kontrakciju u odnosu na najbolju stranu.
    - Uočimo da sada sve testne točke leže na pravcu koji prolazi kroz točke x_h i c te **najviše dvije** se računaju u jednoj iteraciji.
- Ako ovo uspije, prihvaćena točka postaje novi vrh radnog simpleksa i transformacija je gotova.
- Ako ne uspije, skupljamo simpleks prema najboljem vrhu x_l.

Glavna ideja transformacija

- Prvo, pokušajmo zamijeniti samo najgori vrh x_h boljim vrhom koristeći refleksiju, ekspanziju ili kontrakciju u odnosu na najbolju stranu.
  - Uočimo da sada sve testne točke leže na pravcu koji prolazi kroz točke x_h i c te **najviše dvije** se računaju u jednoj iteraciji.
- Ako ovo uspije, prihvaćena točka postaje novi vrh radnog simpleksa i transformacija je gotova.
- Ako ne uspije, skupljamo simpleks prema najboljem vrhu x_l.
  - Samo u ovom slučaju, **računamo** n **vrhova odjednom**.

Formalizacija ideje transformacija

- Transformacije su kontrolirane s 4 parametra:
  1. α za refleksiju,
  2. β za kontrakciju,
  3. γ za ekspanziju i
  4. δ za skupljanje.

Formalizacija ideje transformacija

- Transformacije su kontrolirane s 4 parametra:
  1. α za refleksiju,
  2. β za kontrakciju,
  3. γ za ekspanziju i
  4. δ za skupljanje.
- Trebaju zadovoljavati sljedeće uvjete:

$$\alpha > 0, \quad 0 < \beta < 1, \quad \gamma > 1, \gamma > \alpha, \quad 0 < \delta < 1.$$

Formalizacija ideje transformacija

- Transformacije su kontrolirane s 4 parametra:
  1. α za refleksiju,
  2. β za kontrakciju,
  3. γ za ekspanziju i
  4. δ za skupljanje.
- Trebaju zadovoljavati sljedeće uvjete:

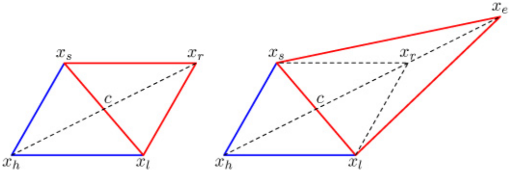
$$\alpha > 0, \quad 0 < \beta < 1, \quad \gamma > 1, \gamma > \alpha, \quad 0 < \delta < 1.$$

- Obično se koristi:

$$\alpha = 1, \quad \beta = \frac{1}{2}, \quad \gamma = 2, \quad \delta = \frac{1}{2}.$$

Formalizacija ideje transformacija

- **Refleksija**: izračunamo točku refleksije $x_r := c + \alpha(c - x_h)$ i $f_r := f(x_r)$. Ako je $f_l \leq f_r \leq f_s$, prihvatimo x_r i završi iteraciju.

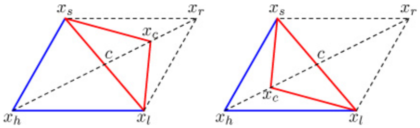


Slika: Refleksija i ekspanzija

- **Ekspanzija**: ako $f_r < f_l$, izračunaj točku ekspanzije $x_e := c + \gamma(x_r - c)$ i $f_e := f(x_e)$. Ako $f_e < f_r$, prihvati x_e i završi iteraciju. Inače (ako je $f_e \geq f_r$), prihvati x_r i završi iteraciju.

Formalizacija ideje transformacija

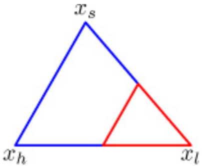
- **Kontrakcija**: Ako $f_r \geq f_s$, izračunaj točku kontrakcije x_c koristeći bolju od točaka x_h i x_r.
  - **Vanjska**: Ako $f_s \leq f_r < f_h$ izračunaj $x_c := c + \beta(x_r - c)$ i $f_c := f(x_c)$. Ako $f_c \leq f_r$, prihvati x_c i završi iteraciju. Inače, primijeni skupljanje.
  - **Unutarnja**: Ako $f_r \geq f_h$, izračunaj $x_c := c + \beta(x_h + c)$ i $f_c := f(x_c)$. Ako $f_c < f_h$, prihvati x_c i završi iteraciju. Inače, primijeni skupljanje.



Slika: Vanjska i unutarnja kontrakcija

Formalizacija ideje transformacija

- **Skupljanje**: Izračunaj n novih vrhova $x_j := x_l + \delta(x_j - x_l)$ i $f_j := f(x_j)$, za $j = 0, \ldots, n$,, pri čemu $j \neq l$.



Slika: Skupljanje

Sadržaj

Meta-heuristike

- Kako Nelder-Meadov algoritam funkcionira po principu 'spuštanja niz padinu', rješenja će uglavnom biti u lokalnim optimumima, a ne u globalnim.

Meta-heuristike

- Kako Nelder-Meadov algoritam funkcionira po principu 'spuštanja niz padinu', rješenja će uglavnom biti u lokalnim optimumima, a ne u globalnim.
- Da bismo izbjegli 'zaglavljivanje', predstavljamo nekoliko principa bježanja iz lokalnih optimuma.

Meta-heuristike

- Kako Nelder-Meadov algoritam funkcionira po principu 'spuštanja niz padinu', rješenja će uglavnom biti u lokalnim optimumima, a ne u globalnim.
- Da bismo izbjegli 'zaglavljivanje', predstavljamo nekoliko principa bježanja iz lokalnih optimuma.
- To su redom:
  1. Iterirani slučajni početak
  2. Usmjereni bijeg
  3. Ne-Tabu pretraživanje
  4. Simulirano kaljenje

Meta-heuristike

- Kako Nelder-Meadov algoritam funkcionira po principu 'spuštanja niz padinu', rješenja će uglavnom biti u lokalnim optimumima, a ne u globalnim.
- Da bismo izbjegli 'zaglavljivanje', predstavljamo nekoliko principa bježanja iz lokalnih optimuma.
- To su redom:
  1. Iterirani slučajni početak
  2. Usmjereni bijeg
  3. Ne-Tabu pretraživanje
  4. Simulirano kaljenje
- Svaka ima neke svoje parametre, ali sve metode imaju parametar "Najveći broj iteracija u Nelder-Meadovu algoritmu" kojim zaustavljamo traženje nakon dovoljnog broja iteracija.

Meta-heuristike

- Kako Nelder-Meadov algoritam funkcionira po principu 'spuštanja niz padinu', rješenja će uglavnom biti u lokalnim optimumima, a ne u globalnim.
- Da bismo izbjegli 'zaglavljivanje', predstavljamo nekoliko principa bježanja iz lokalnih optimuma.
- To su redom:
  1. Iterirani slučajni početak
  2. Usmjereni bijeg
  3. Ne-Tabu pretraživanje
  4. Simulirano kaljenje
- Svaka ima neke svoje parametre, ali sve metode imaju parametar "Najveći broj iteracija u Nelder-Meadovu algoritmu" kojim zaustavljamo traženje nakon dovoljnog broja iteracija.
- Koristimo ga jer nam je iskustvo u traženju funkcija pokazalo da u većini slučajeva algoritam, sam po sebi, ne pronalazi bolja rješenja te ga nema smisla čekati.

Iterirani slučajni početak

- Ova metoda sastoji se od ponovnog pokretanja algoritma od slučajnog rješenja svaki put kada simpleks konvergira po kriteriju ϵ ili kada je dostignut maksimalan broj evaluacija M.

Iterirani slučajni početak

- Ova metoda sastoji se od ponovnog pokretanja algoritma od slučajnog rješenja svaki put kada simpleks konvergira po kriteriju ϵ ili kada je dostignut maksimalan broj evaluacija M.
- U svakoj iteraciji, na slučajan način se izabere točka i simpleks se ponovno izgradi iz te točke.

Iterirani slučajni početak

- Ova metoda sastoji se od ponovnog pokretanja algoritma od slučajnog rješenja svaki put kada simpleks konvergira po kriteriju ϵ ili kada je dostignut maksimalan broj evaluacija M.
- U svakoj iteraciji, na slučajan način se izabere točka i simpleks se ponovno izgradi iz te točke.
- Kad god se najbolje rješenje dodatno popravi, provodi se novo lokalno pretraživanje s kriterijem zaustavljanja ϵ', za profinjenje lokalnog optimuma.

Pseudokod

Iterirani slučajni početak

procedure $\textsc{IteratedSimplex}(\epsilon,\ \epsilon',\ \lambda,\ M)$
    $k = 0$
    while $k < M$ **do**
        $x =$ slučajno odabrana točka u danim ogradama
        $x = SimplexLocalSearch(x, \lambda, \epsilon, M - k)$
        $k+ =$ ukupan broj evaluacija funkcije
        if x^* nije inicijaliziran **ili** x bolji od x^* **then**
            $x^* = SimplexLocalSearch(x, \epsilon', \lambda, M - k)$
            $k+ =$ ukupan broj evaluacija funkcije
        end if
    end while
    return x^*
end procedure

Usmjereni bijeg

- Kada simpleks konvergira po kriteriju ϵ , počni širiti simpleks kroz njegov najbolji vrh (stalno ažurirajući poredak vrhova).

Usmjereni bijeg

- Kada simpleks konvergira po kriteriju ϵ , počni širiti simpleks kroz njegov najbolji vrh (stalno ažurirajući poredak vrhova).
- Ekspanzija će isprva smanjiti kvalitetu točaka, ali nakon određenog broja ponavljanja, dostignut će lokalni 'pessimum' i ekspanzije koje se budu dalje vršile će voditi napretku.

Usmjereni bijeg

- Kada simpleks konvergira po kriteriju ϵ , počni širiti simpleks kroz njegov najbolji vrh (stalno ažurirajući poredak vrhova).
- Ekspanzija će isprva smanjiti kvalitetu točaka, ali nakon određenog broja ponavljanja, dostignut će lokalni 'pessimum' i ekspanzije koje se budu dalje vršile će voditi napretku.
- Dozvolit ćemo da se ekspanzija vrši dok god se ne popravi najgora točka simpleksa. U toj točki očekujemo da se nalazimo na drugoj strani brda.

Usmjereni bijeg

- Kada simpleks konvergira po kriteriju ϵ , počni širiti simpleks kroz njegov najbolji vrh (stalno ažurirajući poredak vrhova).
- Ekspanzija će isprva smanjiti kvalitetu točaka, ali nakon određenog broja ponavljanja, dostignut će lokalni 'pessimum' i ekspanzije koje se budu dalje vršile će voditi napretku.
- Dozvolit ćemo da se ekspanzija vrši dok god se ne popravi najgora točka simpleksa. U toj točki očekujemo da se nalazimo na drugoj strani brda.
- Dakle, ako algoritam ponovno pokrenemo iz te točke, očekujemo da ćemo doseći drugi lokalni optimum.

Pseudokod

Usmjereni bijeg

procedure $\textsc{DirectionalEscape}(\epsilon, \epsilon', \lambda, M)$
    $k = 0$
    $x =$ slučajno odabrana točka u danim ogradama
    while $k < M$ **do**
        $x = SimplexLocalSearch(x, \lambda, \epsilon, M - k)$
        $k+ =$ ukupan broj evaluacija funkcije
        if x^* nije inicijaliziran **or** x bolji od x^* **then**
            $x^* = SimplexLocalSearch(x, \epsilon', \lambda, M - k)$
        end if
        while $xWorst$ nije inicijaliziran **or** x bolji od $xWorst$ **do**
            $xWorst = x$
            $x = \gamma \cdot s^* + (1 - \gamma)\hat{s}$
            $s^* = x$
        end while
        $k+ =$ ukupan broj evalucija funkcije
    end while
    return x^*
end procedure

Ne-Tabu pretraživanje

- Istraživanjem meta-heuristika nad Nelder-Meadovim algoritmom ustanovilo se da će algoritam pronaći dobra rješenja ako pretražujemo područja prethodnih lokalnih optimuma, umjesto da ih izbjegavamo, kao što je to slučaj kod Tabu pretraživanja.

Ne-Tabu pretraživanje

- Istraživanjem meta-heuristika nad Nelder-Meadovim algoritmom ustanovilo se da će algoritam pronaći dobra rješenja ako pretražujemo područja prethodnih lokalnih optimuma, umjesto da ih izbjegavamo, kao što je to slučaj kod Tabu pretraživanja.
- Objašnjenje ovog zaključka svodi se na to da lokalni optimumi često znaju biti blizu drugih lokalnih optimuma.

Ne-Tabu pretraživanje

- Istraživanjem meta-heuristika nad Nelder-Meadovim algoritmom ustanovilo se da će algoritam pronaći dobra rješenja ako pretražujemo područja prethodnih lokalnih optimuma, umjesto da ih izbjegavamo, kao što je to slučaj kod Tabu pretraživanja.
- Objašnjenje ovog zaključka svodi se na to da lokalni optimumi često znaju biti blizu drugih lokalnih optimuma.
- Dakle, moglo bi imati smisla pretražiti područje oko prethodnih optimuma, umjesto da ih izbjegavamo.

Ne-Tabu pretraživanje

- Odatle naziv Ne-tabu pretraživanje. U ovome algoritmu, područje oko lokalnog optimuma se pretražuje pogađanjem slučajnih rješenja u njegovoj blizini i ponovnim pokretanjem pretraživanja iz njih.

Ne-Tabu pretraživanje

- Odatle naziv Ne-tabu pretraživanje. U ovome algoritmu, područje oko lokalnog optimuma se pretražuje pogađanjem slučajnih rješenja u njegovoj blizini i ponovnim pokretanjem pretraživanja iz njih.
- Parametri koje metoda koristi su:
  1. σ - zadaje udaljenost od trenutnog baznog rješenja, koristi se za dobivanje novih rješenja

Ne-Tabu pretraživanje

- Odatle naziv Ne-tabu pretraživanje. U ovome algoritmu, područje oko lokalnog optimuma se pretražuje pogađanjem slučajnih rješenja u njegovoj blizini i ponovnim pokretanjem pretraživanja iz njih.
- Parametri koje metoda koristi su:
  1. σ - zadaje udaljenost od trenutnog baznog rješenja, koristi se za dobivanje novih rješenja
  2. R - zadaje broj pokušaja pogađanja oko svakog baznog rješenja (nakon R koraka bazno rješenje se mijenja novim najboljim pronađenim rješenjem)

Ne-Tabu pretraživanje

- Odatle naziv Ne-tabu pretraživanje. U ovome algoritmu, područje oko lokalnog optimuma se pretražuje pogađanjem slučajnih rješenja u njegovoj blizini i ponovnim pokretanjem pretraživanja iz njih.
- Parametri koje metoda koristi su:
  1. σ - zadaje udaljenost od trenutnog baznog rješenja, koristi se za dobivanje novih rješenja
  2. R - zadaje broj pokušaja pogađanja oko svakog baznog rješenja (nakon R koraka bazno rješenje se mijenja novim najboljim pronađenim rješenjem)
- Napomenimo odmah da je teško pogoditi dobre parametre.

Pseudokod

Ne-Tabu pretraživanje

procedure NonTabuSearch(ϵ, ϵ', λ, M, σ, R)
 $k = 0$, $x =$ slučajno odabrana točka u danim ogradama
 $x = \mathit{SimplexLocalSearch}(x, \lambda, \epsilon, M - k)$
 $k+ =$ ukupan broj evaluacija funkcije
 $x^* = x$, $y = x$
 while $k < M$ **do**
  for $i = 1$ **to** R **do**
   $x_j = y_j \pm \sigma \cdot (u_j - l_j)$
   $x = \mathit{SimplexLocalSearch}(x, \lambda, \epsilon, M - k)$
   $k+ =$ ukupan broj evaluacija funkcije
   if x bolji od x^* **then**
    $x = \mathit{SimplexLocalSearch}(x, \lambda, \epsilon', M - k)$
    $k+ =$ ukupan broj evaluacija funkcije
   end if
   if x' nije inicijaliziran **or** x bolji od x' **then**
    $x' = x$
   end if
  end for
  $y = x'$
 end while
 return x^*
end procedure

Simulirano kaljenje

- U ovome slučaju heuristika se svodi na sljedeće: prilikom početne veće temperature, s većom vjerojatnosti prihvaćamo lošija rješenja u odnosu na slučaj kada je temperatura niža i manja vjerojatnost da prihvatimo lošija rješenja.

Simulirano kaljenje

- U ovome slučaju heuristika se svodi na sljedeće: prilikom početne veće temperature, s većom vjerojatnosti prihvaćamo lošija rješenja u odnosu na slučaj kada je temperatura niža i manja vjerojatnost da prihvatimo lošija rješenja.
- Svakako, ako u koraku pronađemo rješenje koje je bolje nego korak prije, na njemu vršimo Nelder-Meadov algoritam.

Simulirano kaljenje

- U ovome slučaju heuristika se svodi na sljedeće: prilikom početne veće temperature, s većom vjerojatnosti prihvaćamo lošija rješenja u odnosu na slučaj kada je temperatura niža i manja vjerojatnost da prihvatimo lošija rješenja.
- Svakako, ako u koraku pronađemo rješenje koje je bolje nego korak prije, na njemu vršimo Nelder-Meadov algoritam.
- Rješenja pronalazimo u okolini radijusa z kojeg smanjujemo ako pronađemo lošije rješenje, a povećavamo ako pronađemo bolje rješenje.

Simulirano kaljenje

- U ovome slučaju heuristika se svodi na sljedeće: prilikom početne veće temperature, s većom vjerojatnosti prihvaćamo lošija rješenja u odnosu na slučaj kada je temperatura niža i manja vjerojatnost da prihvatimo lošija rješenja.
- Svakako, ako u koraku pronađemo rješenje koje je bolje nego korak prije, na njemu vršimo Nelder-Meadov algoritam.
- Rješenja pronalazimo u okolini radijusa z kojeg smanjujemo ako pronađemo lošije rješenje, a povećavamo ako pronađemo bolje rješenje.

Pseudokod

Simulirano kaljenje

procedure $\textsc{SimulatedAnnealing}(\epsilon,\ \lambda,\ T,\ \beta,\ \mu)$
    $x =$ slučajno rješenje i postavi radijus z
    while $T > 0$ **do**
        $k = 0$
        while $k \leq \mu$ **do**
            generiraj susjedna rješenja u listu L i zapamti u x' najbolje
            $\Delta E = f(x') - f(L[-1])$
            if $\delta E < 0$ **then**:
                $x = \mathit{SimplexLocalSearch}(x', \epsilon, \lambda)$
                zapamti taj x i $z = 2 \cdot z$
            else
                if $\mathit{rand}() < \mathit{exp}(-\Delta E / T)$ **then**
                    zapamti taj x i $z = 0.5z$
                end if
            end if
        end while
        $T = T - \beta$
    end while
    $x^* =$ najbolje nađeno rješenje
    $x^* = \mathit{SimplexLocalSearch}(x^*, \epsilon, \lambda)$
    return x^*
end procedure

Sadržaj

Funkcije, rezultati i analiza

- U ovom odjeljku vizualiziramo u tri dimenzije sve funkcije koje smo testirali običnim Nelder-Meadovim algoritmom i svim funkcijama koje smo naveli.

Funkcije, rezultati i analiza

- U ovom odjeljku vizualiziramo u tri dimenzije sve funkcije koje smo testirali običnim Nelder-Meadovim algoritmom i svim funkcijama koje smo naveli.
- Svaka funkcija je testirana 50 puta svakom metodom te su rezultati navedeni u odgovarajućim tablicama.

Funkcije, rezultati i analiza

- U ovom odjeljku vizualiziramo u tri dimenzije sve funkcije koje smo testirali običnim Nelder-Meadovim algoritmom i svim funkcijama koje smo naveli.
- Svaka funkcija je testirana 50 puta svakom metodom te su rezultati navedeni u odgovarajućim tablicama.
- Također, da bismo vizualizirali kvalitetu pojedine metode, ugrubo ćemo ih bodovati.

Funkcije, rezultati i analiza

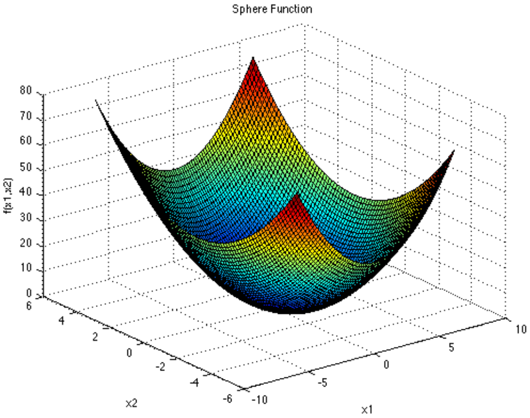
- U ovom odjeljku vizualiziramo u tri dimenzije sve funkcije koje smo testirali običnim Nelder-Meadovim algoritmom i svim funkcijama koje smo naveli.
- Svaka funkcija je testirana 50 puta svakom metodom te su rezultati navedeni u odgovarajućim tablicama.
- Također, da bismo vizualizirali kvalitetu pojedine metode, ugrubo ćemo ih bodovati.
- Zasebno promatramo prosječno nađena rješenja i najbolje nađena rješenja te redom dodijeljujemo bodova za svaku funkciju svakoj metodi, ovisno koja je ona u poretku (4 bodova najbolje rješenje, 3 drugo najbolje, pa 2 i 1).

Funkcije, rezultati i analiza

- U ovom odjeljku vizualiziramo u tri dimenzije sve funkcije koje smo testirali običnim Nelder-Meadovim algoritmom i svim funkcijama koje smo naveli.
- Svaka funkcija je testirana 50 puta svakom metodom te su rezultati navedeni u odgovarajućim tablicama.
- Također, da bismo vizualizirali kvalitetu pojedine metode, ugrubo ćemo ih bodovati.
- Zasebno promatramo prosječno nađena rješenja i najbolje nađena rješenja te redom dodijeljujemo bodova za svaku funkciju svakoj metodi, ovisno koja je ona u poretku (4 bodova najbolje rješenje, 3 drugo najbolje, pa 2 i 1).
- Dvama najgorim rješenjima ne dodjeljujemo bodove.

Sferna funkcija

$$f(x) = \sum_{i=1}^{N} x_i^2, x_i \in [-30, 30], i = 1, \ldots, N$$



Rezultati za sfernu funkciju

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	9.3772e-48	0.0	1	100000	/	/	/
Iterirani slučajni početak	2.5e-323	0.0	1	100000	/	/	/
Usmjereni bijeg	9.3402e-42	0.0	0.1	100000	/	/	/
Ne-tabu pretraživanje	7.6601e-11	4.2881e-20	10	1000	1	2	/
Simulirano kaljenje	5.0515e-75	5.4294e-192	10	100000	/	/	100

Slika: Sferna funkcija testirana na svim metodama

Analiza rezultata za sfernu funkciju

- Sve osim ne-tabu i simuliranog kaljenja našle su u nekom trenutku najbolje rješenje, ali prosječno najbolje su bile iterirani slučajni početak i simulirano kaljenje.

Analiza rezultata za sfernu funkciju

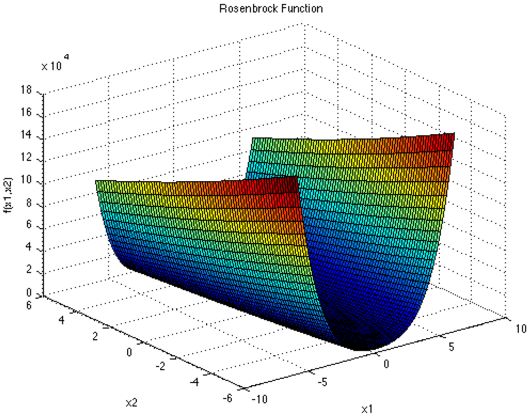
- Sve osim ne-tabu i simuliranog kaljenja našle su u nekom trenutku najbolje rješenje, ali prosječno najbolje su bile iterirani slučajni početak i simulirano kaljenje.
- Za ovu funkciju je zato najbolje rezultate pokazao iterirani slučajni početak.

Analiza rezultata za sfernu funkciju

- Sve osim ne-tabu i simuliranog kaljenja našle su u nekom trenutku najbolje rješenje, ali prosječno najbolje su bile iterirani slučajni početak i simulirano kaljenje.
- Za ovu funkciju je zato najbolje rezultate pokazao iterirani slučajni početak.
- Iskustvo testiranja je pokazalo da je najbolje za ovu funkciju pustiti da se Nelder Meadov algoritam izvrši do kraja jer ona nema lokalnih minimuma (osim globalnog u nuli koji je rješenje).

Rosenbrock

$$f(x) = \sum_{i=1}^{d-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2, x_i \in [-30, 30], i = 1, \ldots, N$$



Rezultati za Rosenbrock

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	2972616.8259	0.1100	0.1	100000	/	/	/
Iterirani slučajni početak	781414.9711	1.737e-17	1	100000	/	/	/
Usmjereni bijeg	0.0	0.0	10	1000	/	/	/
Ne-tabu pretraživanje	508.6913	6.7596e-20	10	1000	1	2	/
Simulirano kaljenje	4.0122e-17	0.0	10	1000	/	/	100

Slika: Rosenbrock testiran na svim metodama

Analiza rezultata za Rosenbrock

- Otprilike pola funkcija je pronašlo najbolje rješenje, međutim pogotovo uspješnim pokazao se usmjereni bijeg koji je u svakom mogućem pokretanju pronašao najbolje rješenje.

Analiza rezultata za Rosenbrock

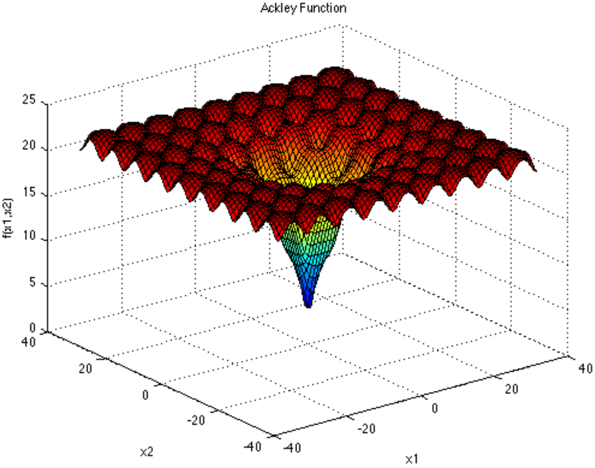
- Otprilike pola funkcija je pronašlo najbolje rješenje, međutim pogotovo uspješnim pokazao se usmjereni bijeg koji je u svakom mogućem pokretanju pronašao najbolje rješenje.
- S druge strane, većina ostalih funkcija imali su jako veliku prosječnu vrijednost.

Analiza rezultata za Rosenbrock

- Otprilike pola funkcija je pronašlo najbolje rješenje, međutim pogotovo uspješnim pokazao se usmjereni bijeg koji je u svakom mogućem pokretanju pronašao najbolje rješenje.
- S druge strane, većina ostalih funkcija imali su jako veliku prosječnu vrijednost.
- Najgorim se pokazao obični Nelder-Meadov algoritam i po ovoj funkciji vidimo korist heuristika u odnosu na obični algoritam.

Ackley

$$f(x) = -20\ \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{N} x_i^2}\right) - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right) + 20 + e, \quad x_i \in [-30, 30], \quad i = 1, \ldots, N$$



Rezultati za Ackley

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	1.4103	3.5527e-15	10	100000	/	/	/
Iterirani slučajni početak	0.2718	3.5527e-15	10	100000	/	/	/
Usmjereni bijeg	2.7711e-15	0.0	10	1000	/	/	/
Ne-tabu pretraživanje	2.4825e-11	3.5527e-15	10	1000	1	2	/
Simulirano kaljenje	2.0322e-14	3.5527e-15	10	1000	/	/	100

Slika: Ackley testiran na svim metodama

Analiza rezultata za Ackley

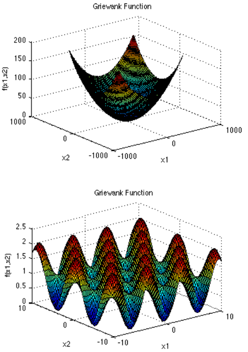
- Sve metode, osim usmjerenog bijega pronašle su isto najbolje rješenje, $3.5527 \cdot 10^{-15}$.

Analiza rezultata za Ackley

- Sve metode, osim usmjerenog bijega pronašle su isto najbolje rješenje, $3.5527 \cdot 10^{-15}$.
- Funkcija Ackley pokazala se kao tvrd orah, koju smo jedino uspjeli riješiti usmjerenim bijegom. Najgorim se opet pokazao obični Nelder-Meadov algoritam.

Griewank

$$f(x) = \sum_{i=1}^{N} \frac{(x_i - 100)^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1, \quad x_i \in [-600, 600], \quad i = 1, \ldots, N$$



Rezultati za Griewank

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	2.1931	0.6378	1	100000	/	/	/
Iterirani slučajni početak	1.4187	2.8755e-14	1	100000	/	/	/
Usmjereni bijeg	0.0683	0.0	0.1	5000	/	/	/
Ne-tabu pretraživanje	1.7222e-05	0.0	0.1	1000	1	2	/
Simulirano kaljenje	0.0273	0.0	0.1	5000	/	/	10

Slika: Griewank testiran na svim metodama

Analiza rezultata za Griewank

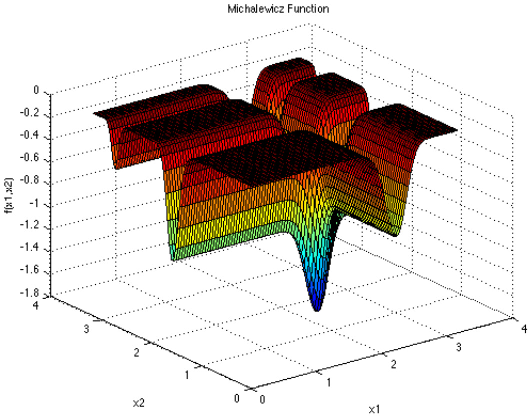
- Sve metode, osim običnog N.M. i iteriranog slučajnog početka uspjele su u nekom trenutku pronaći najbolje rješenje.

Analiza rezultata za Griewank

- Sve metode, osim običnog N.M. i iteriranog slučajnog početka uspjele su u nekom trenutku pronaći najbolje rješenje.
- Za ovu funkciju, prosječno najboljom pokazalo se ne-tabu pretraživanje, najgorom opet Nelder-Meadov algoritam.

Michalewicz

$$f(x) = -\sum_{i=1}^{N} \sin(x_i) \cdot \sin^{2m}\left(\frac{i \cdot x_i^2}{\pi}\right), \quad x_i \in [0, \pi], \quad i = 1, \ldots, N$$



Rezultati za Michalewicz

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	-3.5838	-5.3661	10	100000	/	/	/
Iterirani slučajni početak	-4.5594	-6.3828	10	100000	/	/	/
Usmjereni bijeg	-5.8197	-7.4364	10	1000	/	/	/
Ne-tabu pretraživanje	-6.1145	-7.4298	10	1000	0.1	10	/
Simulirano kaljenje	-8.5811	-9.5785	10	1000	/	/	100

Slika: Michalewicz testiran na svim metodama

Analiza rezultata za Michalewicz

- Ova funkcija pokazala se također tvrdim orahom i nijedna metoda nije uspjela pronaći njezin minimum.

Analiza rezultata za Michalewicz

- Ova funkcija pokazala se također tvrdim orahom i nijedna metoda nije uspjela pronaći njezin minimum.
- Najbliže najboljem rješenju bilo je simulirano kaljenje, a ona se pokazala i prosječno najboljom.

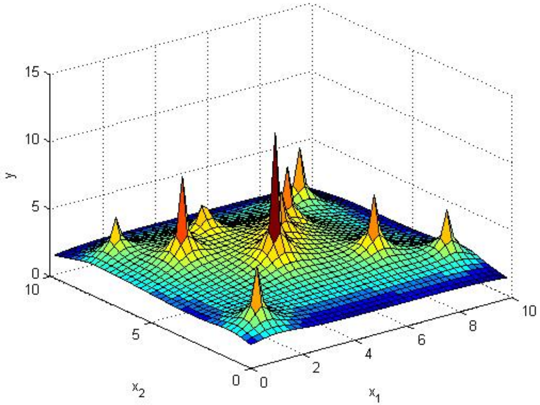
Analiza rezultata za Michalewicz

- Ova funkcija pokazala se također tvrdim orahom i nijedna metoda nije uspjela pronaći njezin minimum.
- Najbliže najboljem rješenju bilo je simulirano kaljenje, a ona se pokazala i prosječno najboljom.
- Obični Nelder-Meadov algoritam opet je bio najgori.

Shekel

$$f(x) = -\sum_{i=1}^{m}\left(\frac{1}{\sum_{j=1}^{N}(x_j - C_{ji})^2 + \beta_i}\right), \quad x_i \in [0, 10], \quad i = 1, \ldots, N$$

2D Shekel function



Rezultati za Shekel

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	-6.4879	-10.5364	10	100000	/	/	/
Iterirani slučajni početak	-9.9228	-10.5364	10	100000	/	/	/
Usmjereni bijeg	-10.1498	-10.5364	10	1000	/	/	/
Ne-tabu pretraživanje	-10.5362	-10.5364	10	1000	1	2	/
Simulirano kaljenje	-7.9302	-10.5364	10	5000	/	/	10

Slika: Shekel testiran na svim metodama

Analiza rezultata za Shekel

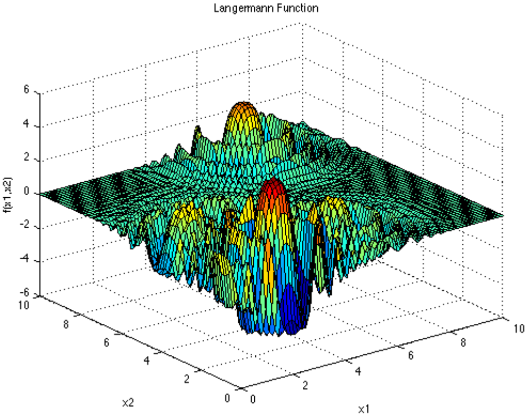
- Ova funkcija bila je u manje dimenzija nego funkcije prije pa su sve metode uspjele pronaći najbolje rješenje.

Analiza rezultata za Shekel

- Ova funkcija bila je u manje dimenzija nego funkcije prije pa su sve metode uspjele pronaći najbolje rješenje.
- Prosječno najboljim pokazalo se ne-tabu pretraživanje, a najgorim ponovno obični N.M. algoritam.

Langermann

$$f(x) = -\sum_{i=1}^{m} c_i \cos\left(\pi(x_1 - a_{i1})^2 + (x_2 - a_{i2})^2\right) e^{-\frac{(x_1-a_{i1})^2+(x_2-a_{i2})^2}{\pi}}, \quad x_i \in [0,10], \quad i = 1, \ldots, N$$



Rezultati za Langermann

Metoda	Prosječna nađena vrijednost	Najbolja nađena vrijednost	Lambda	Najviše it. u N-M	Sigma	R	Temp
Obični Nelder-Meadov algoritam	-1.6914	-4.0614	10	100000	/	/	/
Iterirani slučajni početak	-1.9287	-4.1276	10	100000	/	/	/
Usmjereni bijeg	-2.6621	-4.1223	10	1000	/	/	/
Ne-tabu pretraživanje	-3.8754	-4.1556	10	1000	0.1	2	/
Simulirano kaljenje	-3.1604	4.1557	10	5000	/	/	10

Slika: Langermann testiran na svim metodama

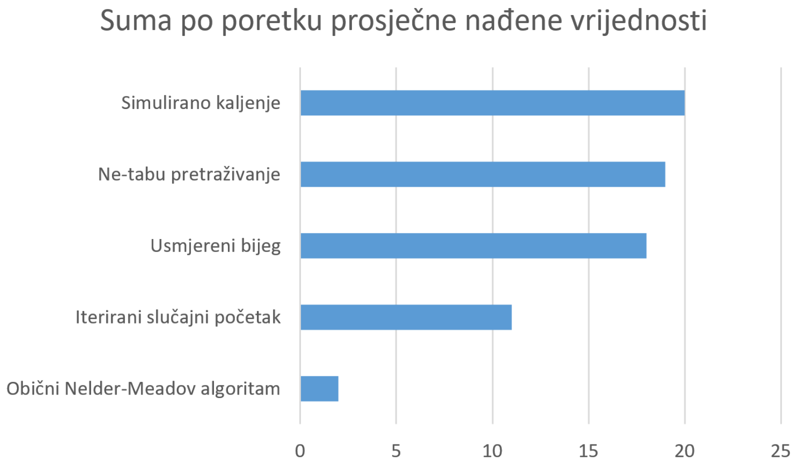
Analiza rezultata za Langermann

- Ova funkcija, unatoč što je samo u 2 dimenzije, nije bila lako rješiva u metodama. Nijedna metoda nije pronašla najbolje rješenje.

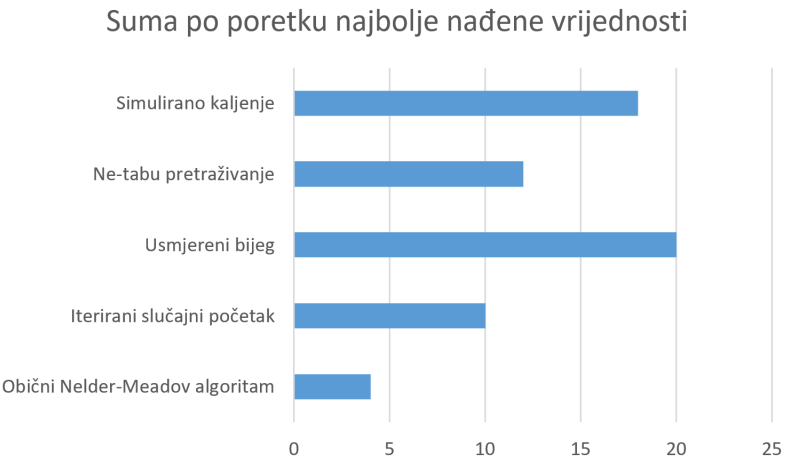
Analiza rezultata za Langermann

- Ova funkcija, unatoč što je samo u 2 dimenzije, nije bila lako rješiva u metodama. Nijedna metoda nije pronašla najbolje rješenje.
- Najbolje rješenje (u odnosu na druge metode) pronašlo je simulirano kaljenje, a prosječno najbolje ne-tabu pretraživanje.

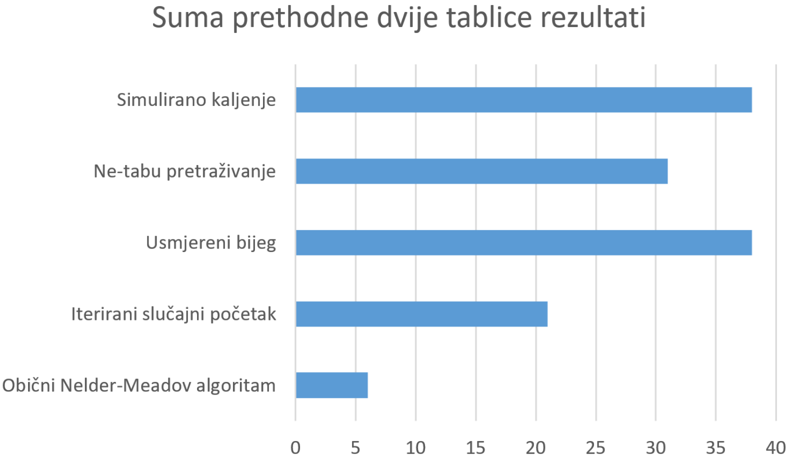
Rezultati po poretku



Rezultati po poretku



Rezultati po poretku



Literatura

- Saša Singer and John Nelder, *Nelder-Mead algorithm*, http://www.scholarpedia.org/article/Nelder-Mead_algorithm (2009)
- João Pedro Pedroso, *Simple meta-heuristics using the simplex algorithm for non-linear programming*, (2007)
- Ahmed Fouad Ali,*Hybrid Simulated Annealing and Nelder-Mead algorithm for solving large-scaleglobal optimization problems*, (2014)