

Projekt GA

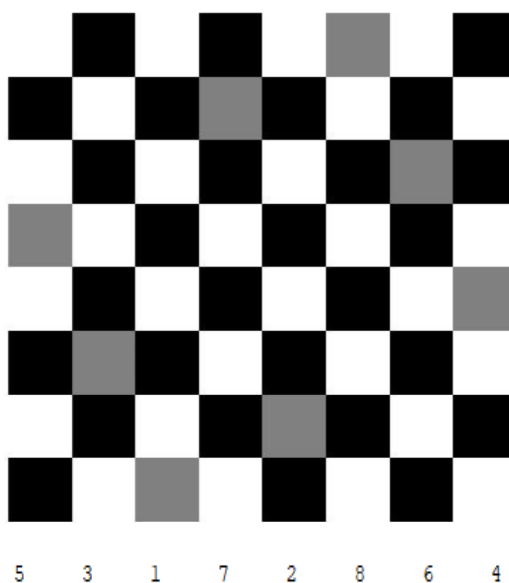
Riešitelia: Kutáš Branislav, Ján Škvara

Úloha: Našou úlohou bolo navrhnuť a implementovať genetický algoritmus pre riešenie úlohy 8 dám na šachovnici tak, aby jedna dáma neohrozovala inú dámu podľa šachových pravidiel.

Body úlohy:

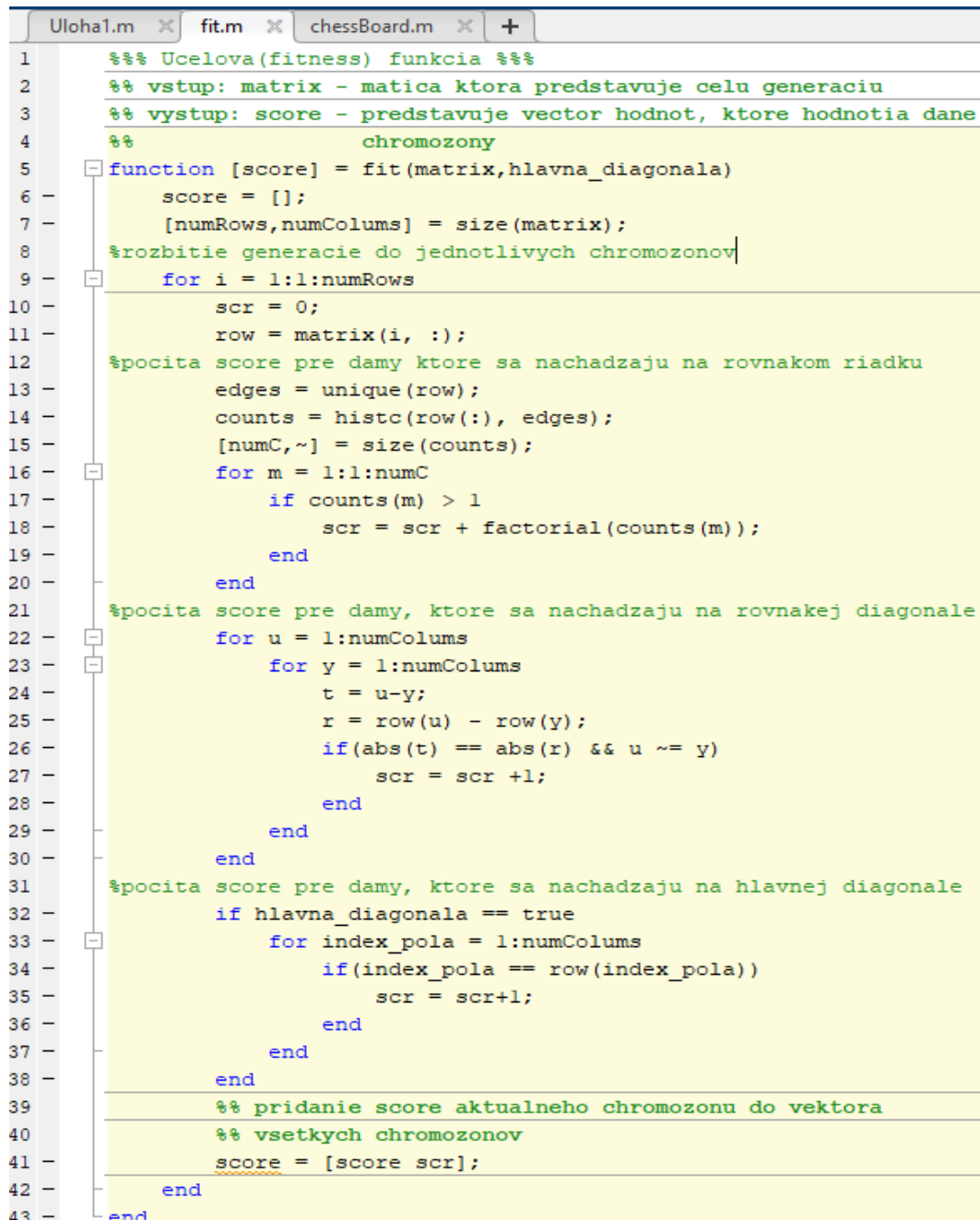
- Navrhните reprezentáciu jedinca populácie GA – zakódovanie úlohy do reťazca (chromozómu).
- Navrhните a implementujte účelovú funkciu (fitness funkciu), ktorá bude počítat kvalitu potenciálneho riešenia (reťazca).
- Pre kontrolu riešení vykonajte aj grafické zobrazenie polohy dám na šachovnici.

Zakódovanie reťazca: Náš reťazec (chromozóm) je zložený z ôsmych prvkov (génov). Index génu prestravuje pozíciu dámy v stĺpcoch. Hodnota daného prvku predstavuje riadok v ktorom sa dáma nachádza.



obrázok zobrazujúci rozostavenie dám (sivé políčka) na šachovnici. Pod obrázkom sa nachádza zakódovaný reťazec pre danú šachovnicu

Výpočet účelovej (fitness) funkcie: Naša fitness funkcia pozostáva z troch hlavných výpočtov pre daný chromozóm. Prvým je výpočet po riadkoch, určuje koľko krát sa dámy v určitých riadkoch stretli. Druhým je výpočet po diagonále a ten určuje koľko krát sa dámy stretli na rovnakej diagonále. Tretím je výpočet po hlavnej diagonále, ktorý je voliteľný, určuje koľko dām sa nachádza na hlavnej diagonále. Tieto čiastkové riešenia sa priebežne spočítavajú do celkového skóre. Dané skóre sa následne priradí do výstupného skóre, ktoré obsahuje skóre všetkých daných chromozónov.



```

1      %%% Ucelova(fitness) funkcia %%%
2      %% vstup: matrix - matica ktora predstavuje celu generaciju
3      %% vystup: score - predstavuje vector hodnot, ktore hodnotia dane
4      %% chromozony
5      function [score] = fit(matrix,hlavna_diagonala)
6          score = [];
7          [numRows,numColumns] = size(matrix);
8          %rozbitie generacie do jednotlivych chromozonov
9          for i = 1:1:numRows
10             scr = 0;
11             row = matrix(i, :);
12             %pocita score pre damy ktore sa nachadzaju na rovnakom riadku
13             edges = unique(row);
14             counts = histc(row(:), edges);
15             [numC,~] = size(counts);
16             for m = 1:1:numC
17                 if counts(m) > 1
18                     scr = scr + factorial(counts(m));
19                 end
20             end
21             %pocita score pre damy, ktore sa nachadzaju na rovnakej diagonale
22             for u = 1:numColumns
23                 for y = 1:numColumns
24                     t = u-y;
25                     r = row(u) - row(y);
26                     if(abs(t) == abs(r) && u ~= y)
27                         scr = scr +1;
28                     end
29                 end
30             end
31             %pocita score pre damy, ktore sa nachadzaju na hlavnej diagonale
32             if hlavna_diagonala == true
33                 for index_pola = 1:numColumns
34                     if(index_pola == row(index_pola))
35                         scr = scr+1;
36                     end
37                 end
38             end
39             %% pridanie score aktualneho chromozonu do vektora
40             %% vsetkych chromozonov
41             score = [score scr];
42         end
43     end
  
```

daný obrázok predstavuje kód našej účelovej funkcie s komentármi

Popis použitých GA: Náš GA delí populáciu na top jedincov a rodičov budúcej generácie. Rodičia sú vyberaní na základe turnajového výberu, následne ich krížime náhodne medzi sebou na 4 bodoch kríženia, nasleduje mierna mutácia 30%. Výslednú populáciu zaokrúhľujeme na celé čísla. Beží 10 pokusov, pre 2000 generácii. Následne vykreslí graf skóre pokusov a graf priemeru skóre pokusov.

Modifikovaná GA, má zvýšenú populáciu trojnásobne, počet top jedincov, a mieru mutácie pre vyššiu zmenu chromozómov.

```
for gen = 0:1:generacie
    fitness = fit(X, isDiag);
    selb = selbest(X, fitness, v); % vyber top jedincov podľa vektoru v
    TOP = selbest(X, fitness, [1]); % top jedinec jednej generacie
    Y = [Y fit(TOP, isDiag)]; % uloženie score top jedinca za generáciu pre grafy
    selt = seltourn(X, fitness, 14); % turnajový vyber rodičov budúcej populácie
    selt = crossov(selt, 4, 0); % náhodné kríženie rodičov na 4 miestach
    selt = mutx(selt, 0.3, S); % mierna globálna mutácia
    X = [selb; selt];
    X = round(X); % zaokrúhľovanie populácie po krížení a mutáciách
end
```

genetický algoritmus bez modifikácie

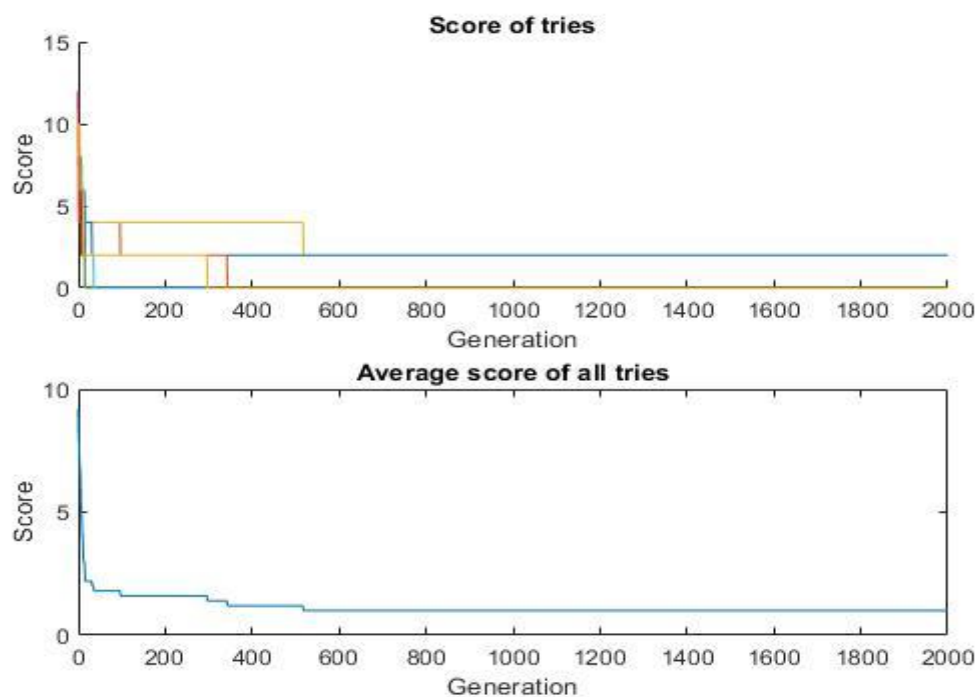
```
for gen = 0:1:generacie
    fitness = fit(X, isDiag);
    selb = selbest(X, fitness, v); % vyber vyššieho počtu top jedincov
    TOP = selbest(X, fitness, [1]);
    Y = [Y fit(TOP, isDiag)];
    selt = seltourn(X, fitness, 50); % zvýšený počet rodičov budúcej generácie
    selt = crossov(selt, 4, 0);
    selt = mutx(selt, 0.7, S); % zvýšená miera mutácie
    X = [selb; selt];
    X = round(X);
end
```

genetický algoritmus s modifikáciou

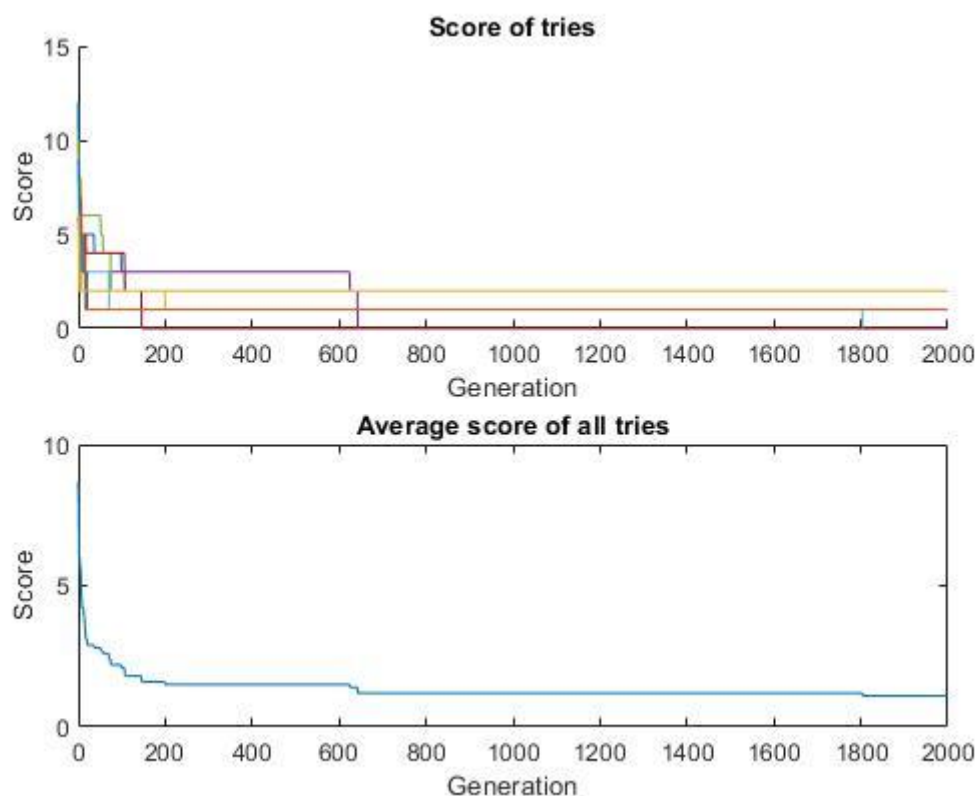
Výsledky:

	BEZ HLAVNEJ DIAGONÁLY		S HLAVNOU DIAGONÁLOU	
	BEZ MODIFIKÁCIE	S MODIFIKÁCIOU	BEZ MODIFIKÁCIE	S MODIFIKÁCIOU
ÚSPEŠNOSŤ (%)	50%	80%	30%	80%
PRIEMER	1	0,4	1,1	0,2

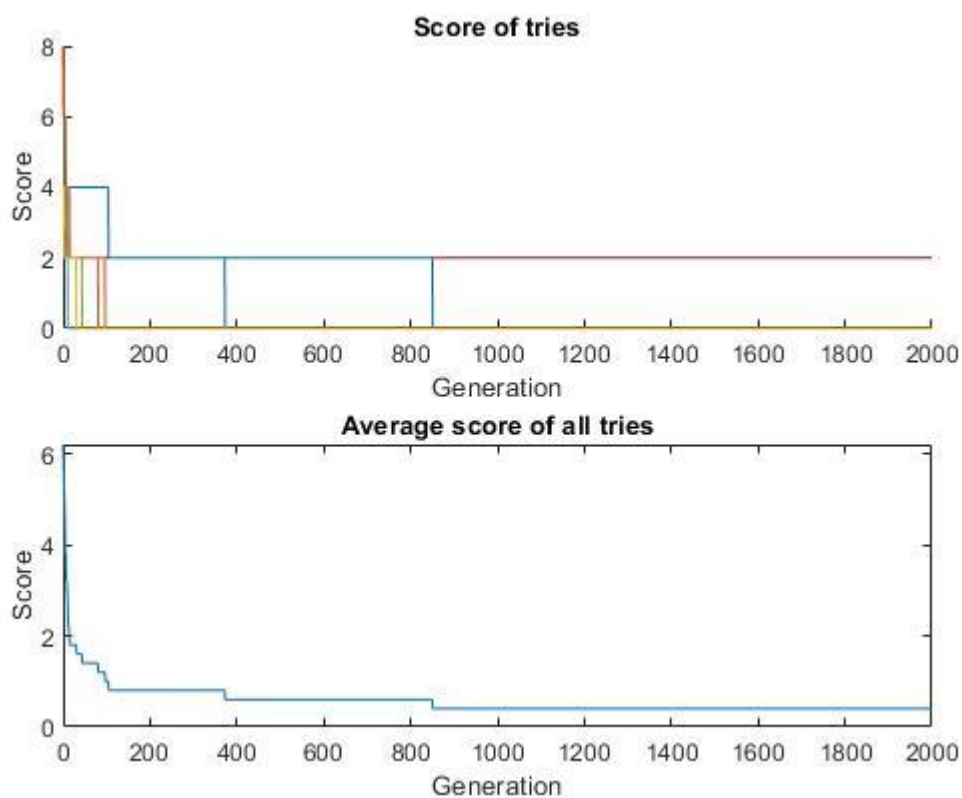
Tabuľka výsledkov



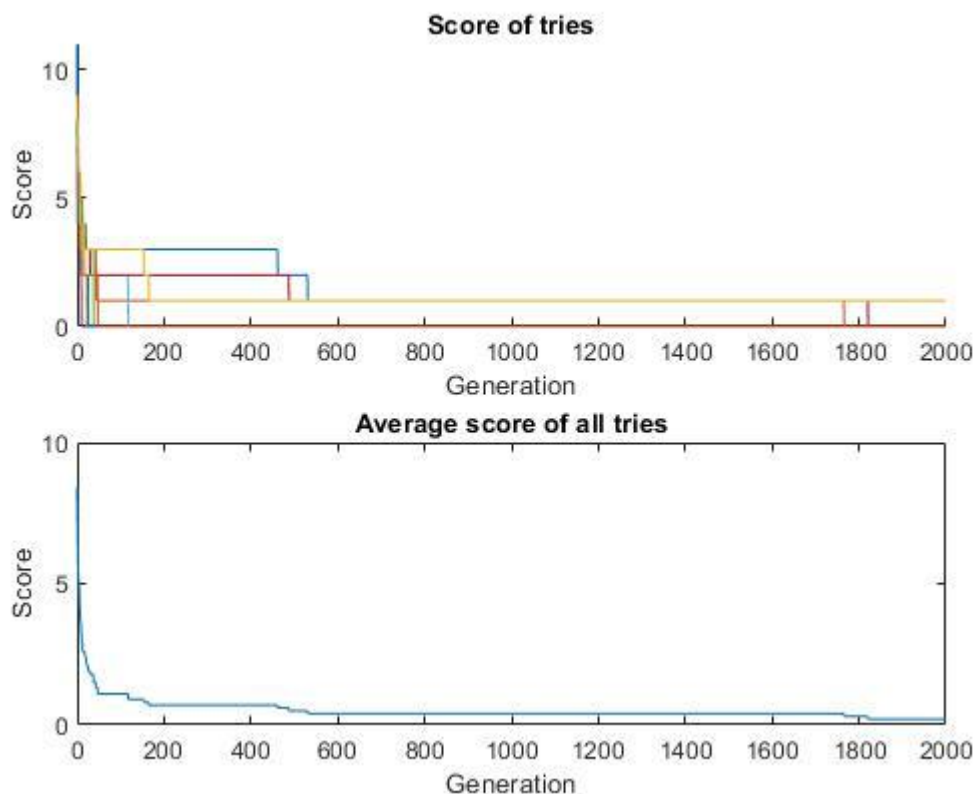
Priebeh fitness funkcií a priebeh priemeru týchto funkcií bez zarátavania hlavnej diagonály do fitness funkcie a bez danej modifikácie



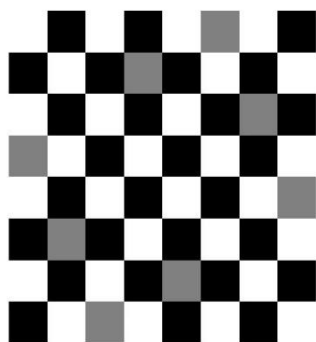
Priebeh fitness funkcií a priebeh priemeru týchto funkcií bez zarátavania hlavnej diagonály do fitness funkcie ale s danou modifikáciou



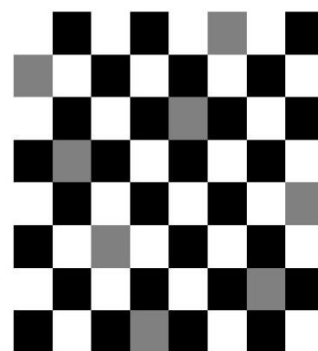
Priebeh fitness funkcií a priebeh priemeru týchto funkcií so zarátaním hlavnej diagonály do fitness funkcie a bez danej modifikácie



Priebeh fitness funkcií a priebeh priemeru týchto funkcií so zarátavaním hlavnej diagonály do fitness funkcie ale s danou modifikáciou



Na šachovnici je zobrazené jedno z možných správnych riešení rozmiestnenia dám (sivé políčka). Pri tomto danom scenári sa pokutovalo ak sa nejaká z dám nachádzala na hlavnej diagonále



Na šachovnici je zobrazené jedno z možných správnych riešení rozmiestnenia dám (sivé políčka). Pri tomto danom scenári sa nepokutovalo ak sa nejaká z dám nachádzala na hlavnej diagonále

Zhodnotenie: Z výsledkov nám vyplíva, že hľadanie správneho riešenia kde sa nezapočítavala hlavná diagonála do účelovej funkcie , bolo jednoduchšie ako hľadanie správneho riešenia so započítavaním hlavnej diagonály do účelovej funkcie (viď. Tabuľka výsledkov strana 4).

Pri tomto type problému sme sa snažili modifikovať náš GA na základe zistení z pokusov. Z ktorých sa nám najviac osvedčili GA ,ktoré mali väčšiu populáciu ako aj vyššiu mieru globálnej mutácie. Zároveň sme zvýšili počet výberu najlepších jedincov z predchádzajúcej populácie.

BONUSOVÁ ÚLOHA

Parametre GA a PGA:

Spoločné parametre:

Generácie: 2000

Veľkosť populácie: 30

Vektor selbest výberu: [2, 1, 1]

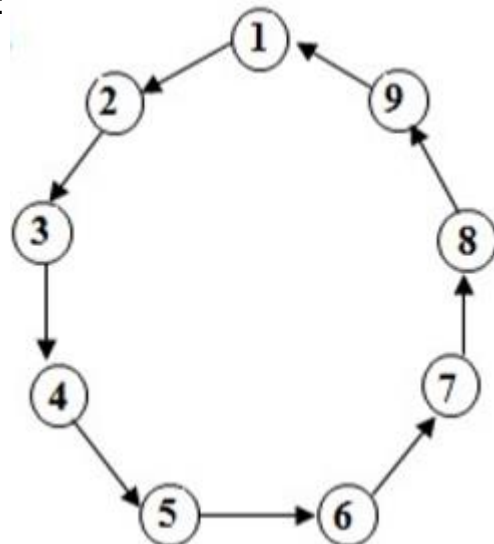
Veľkosť turnajového výberu: 26

Parametre PGA:

Frekvencia migrácie medzi subpopuláciami: raz za 50 generácií

Počet ostrovov: 5

Migráčná architekt:

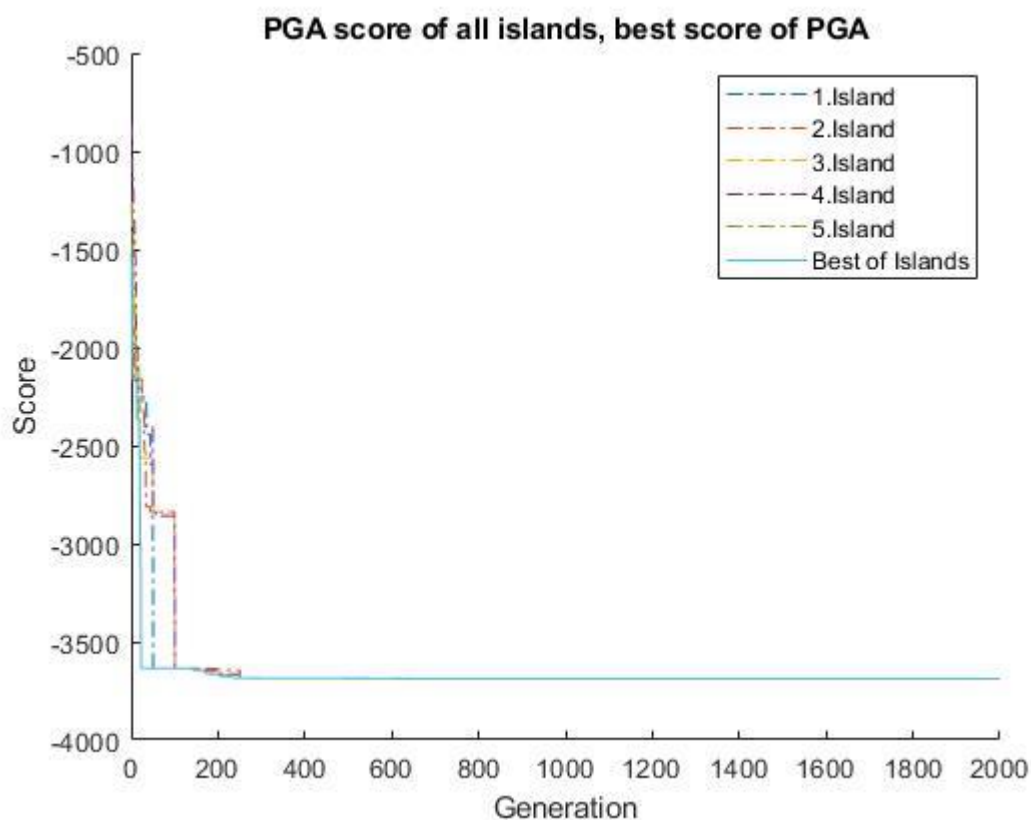


Výsledky:

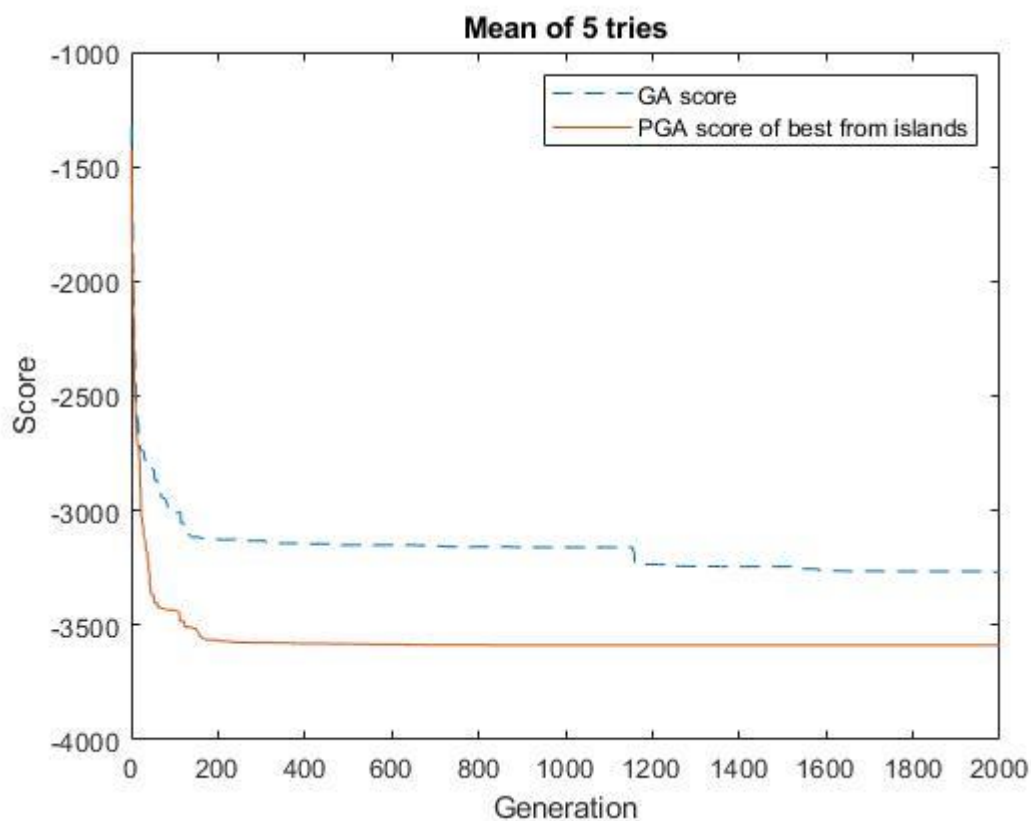
Zobrazené hodnoty predstavujú naše výstupy z GA(TOP) a PGA(output). Po prepočítaní našich výsledkov na skóre pomocou fit funkcie sme dostali lepšie výsledky od PGA než od GA.

```
TOP =  
  
    346.9636   499.0637   392.2502  -359.3667  -427.6422  
  
scoreGA =  
  
    -3.2067e+03  
  
output =  
  
   -456.4517   367.8969   499.5678   449.9688   463.3111  
  
scorePGA =  
  
    -3.4019e+03
```

Daný graf zobrazuje fitness ostrovov a fitness najlepšieho jedinca medzi ostrovmi. Z daného grafu je vidieť ako najlepší jedinec skáče medzi jednotlivými ostrovmi.



Daný graf zobrazuje priemery skóre GA a PGA z piatich spustení. Z grafu vyplíva že PGA konverguje k správne mu riešeniu výrazne rýchlejšie.



Zhodnotenie: PGA algoritmus dosiahol lepšie výsledky ako GA, pri rovnakých parametroch. Zároveň sme zistili že PGA algoritmus je náročnejší na čas výpočtu, je to spôsobené tým, že v každom ostrove prebieha viacero menších GA.