Umelá inteligencia, zadanie č.3

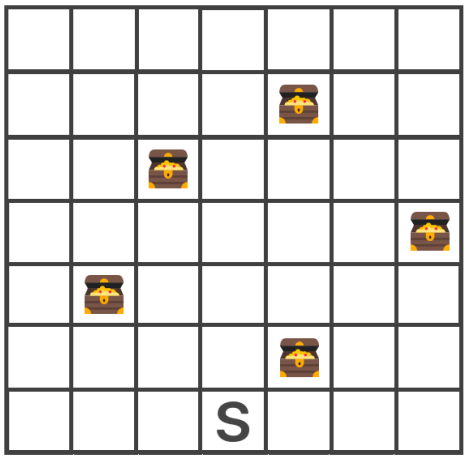
Strojové učenie sa – evolučný algoritmus a evolučné programovanie

Hľadanie pokladu (b)

Patrícia Huďanová

Definovanie riešeného problému

Majme hľadača pokladov, ktorý sa pohybuje vo svete definovanom dvojrozmernou mriežkou (viď. obrázok) a zbiera poklady, ktoré nájde po ceste. Začína na políčku označenom písmenom `S` a môže sa pohybovať štyrmi rôznymi smermi: `hore` **H**, `dole` **D**, `doprava` **P** a `doľava` **L**. K dispozícii má konečný počet krokov. Jeho úlohou je nazbierať čo najviac pokladov. Za nájdenie pokladu sa považuje len pozícia, pri ktorej je hľadač aj poklad na tom istom políčku. Susedné políčka sa neberú do úvahy.



Opis riešenia

Riešenie vo svojej podstate spočíva z nasledujúcich krokov:

1. načítanie vstupných parametrov (typ selekcie, výška a šírka mapy, počet pokladov a ich súradnice a súradnice začiatočného bodu) - objekt plocha
2. načítanie nastavení zo súboru nastavenia.txt do objektu nastavenia
3. vygenerovanie prvej generácie jedincov podľa nastavení
4. vytvorenie plochy podľa vstupu používateľa
5. ak prešiel maximálny počet generácií hľadanie sa stopne, vypíše sa najlepší jedinec a používateľ sa ďalej môže rozhodnúť či bude pokračovať v hľadaní lepšieho jedinca- maximálna povolená generácia sa zdvojnásobí a algoritmus pokračuje, alebo či ukončí hľadanie
6. vypočítanie sa hodnota fitness pre každého jedinca v generácii

1. paralelné počítanie fitness pre všetkých jedincov v generácii
2. určí sa fitness (pri výpočte som brala v úvahu aj počet vykonaných krokov) a cesta jedinca
3. ak sa našli všetky poklady algoritmus skončí

```
private static Jedinec vypocitajHodnotuFitness(Plocha plocha,
Nastavenia nastavenia, int x, int y) {
    top_fitness_jedinec= new Jedinec();

    //fitness sa počíta paralelne pre všetkých jedincov v generácii
    IntStream.range(0, (nastavenia.pocet_max)).parallel().forEach(i ->
{
        //určí sa fitness a cesta jedinca
        generaciaJedincov[i].urciFitnessACestuJedinca(plocha,
nastavenia, x, y, generaciaJedincov[i],i);
        //určí sa súčet fitness všetkých jedincov- používa sa ďalej pri
rulete
        sucet_fitness+= generaciaJedincov[i].getFitness();

        if
(generaciaJedincov[i].getFitness()>top_fitness_jedinec.getFitness() ){
            top_fitness_jedinec= generaciaJedincov[i];
            //ak sa našli všetky poklady algoritmus skončí
            if
(generaciaJedincov[i].getPoklady()==plocha.getPocet_pokladov()){
                return ;
            }
        }
    });
    return top_fitness_jedinec;
}
```

7. generácia sa zoradí podľa hodnoty fitness od najmenšej po najväčšiu
8. **ak sa našli všetky poklady tak sa vypíše najlepší jedinec a program skončí**
9. ak je povolený elitizmus, do novej generácie sa vyberú elitný jedinci a vráti sa index posledného
10. vytvorenie novej generácie (výber jedincov pomocou rulety alebo turnaju, a následne ich kríženie a mutácia)
11. výmena novej a aktuálnej generácie

    Nižšie môžeme vidieť implementáciu tohto riešenia (je mierne upravená oproti reálnemu kódu iba pre ukážku)

```
while(true){
    //3. podľa nastavení sa vytvorí prvá generácia jedincov
    vytvorPrvuGeneraciu(nastavenia);
    //4. vytvorenie plochy podľa vstupu používateľa
    Plocha plocha = new Plocha(X, Y, poklady, mapa);

    for(int gen=0;;++gen){

        //stopnutie hľadania ak prešiel maximálny počet generácií
        if (gen >= nastavenia.stop) {
```

```
            //vypísanie najlepšieho doteraz nájdeného jedinca
            //používateľ sa ďalej môže rozhodnúť či bude pokračovať v hľadaní
lepšieho              jedinca- maximálna povolená generácia sa zdvojnásobí a
algoritmus pokračuje,             alebo či ukončí hľadanie
        }

        //vypočíta sa hodnota fitness pre každého jedinca v generácii
        Jedinec result = vypocitajHodnotuFitness(plocha, nastavenia, X_start,
Y_start);
        //generácia sa zoradí podľa hodnoty fitness od najmenšej po najväčšiu
        Arrays.sort(generaciaJedincov);

        //vypísanie celej generácie
        for (int i=0; i<nastavenia.pocet_max;i++) {
            System.out.println("Jedinec c."+String.valueOf((i+1))
            +". : Hodnota fitness->"+ String.valueOf(generacia[i].getFitness())
            +", Cesta ->" + generaciaJedincov[i].getCesta());
        }

        //ak sa našli všetky poklady tak sa vypíše najlepší jedinec a program
skončí
        if (result.getPoklady() == plocha.pocet_pokladov) {
            System.out.println("Vsetky poklady najdene.\n GENERACIA c."+ gen;
            System.out.println("Najlepsi jedinec: \n"
            +"Hodnota fitness->"+ generaciaJedincov[nastavenia.pocet_max-
1].getFitness()+             "\n"+"Cesta->"+
generaciaJedincov[nastavenia.pocet_max-1].getCesta());
            System.exit(1);
         }

        // ak je povolený elitizmus, do novej generácie sa vyberú elitný jedinci
a vráti         sa index posledného
        int novy_index=vyberElitu(nastavenia);

        //vytvorí sa nová generácia (ruleta alebo turnaj)
        vytvorNovuGeneraciu(nastavenia,novy_index);

        //výmena novej a aktuálnej generácie
        var pomoc_generacia= generaciaJedincov;
        generaciaJedincov = novaGeneraciaJedincov;
        novaGeneraciaJedincov =pomoc_generacia;
    }
}
```

Virtuálny stroj

Virtuálny stroj som skonštruovala tak ako bol prezentovaný v zadaní, čiže pozná 4 inštrukcie (určené prvými 2 bitmi):

inkrementácia- 00xxxxxx

dekrementácia- 01xxxxxx

skok- 10xxxxxx

výpis- 11xxxxxx

Pre výpis som si stanovila nasledovné hodnoty: H(hore) 00

D(dole) 01

P(vpravo) 10

L(vľavo) 11

Virtuálny stroj sa zastaví v troch situáciách- ak sme vyšli mimo plochy (od fitness sa odpočíta 3), ak sa vykonalo 500 inštrukcií alebo ak sme našli všetky poklady.

Vytvorenie novej generácie

Elita

Ak je v nastaveniach povolený elitizmus tak sa do novej generácie prekopíruje určité percento najlepších jedincov z pôvodnej generácie- defaultne v nastaveniach 15%.

Ruleta

1. náhodne vygenerujem číslo v rozsahu 0-súčet fitness hodnôt
2. prechádzam všetkých jedincov v pôvodnej generácii a postupne od vygenerovaného čísla odrátavam ich hodnotu fitness
3. ak po odčítaní dostanem 0 alebo záporné číslo tak daného jedinca vyberiem do novej generácie a ruletu spustím znovu, pretože potrebujem dvoch jedincov

Turnaj

1. z aktuálnej generácie vyberám jedincov 4 randomných jedincov
2. zoradím vybraných jedincov
3. do novej generácie zoberiem druhého a tretieho jedinca z vybraných

Kríženie

Bod kríženia je vygenerovaný náhodne v rozmedzí minimálneho a maximálneho bodu, ktorý sa určuje v nastaveniach (defaultne je tam minimálny bod 22 aby sa zachovalo aspoň toľko buniek z pôvodného jedinca)

Mutácia

Mutovanie je možné 2 spôsobmi a to uplatnením xor na náhodný bit alebo na náhodnú bunku.

Voľba vhodnej paradigmy a programovacieho jazyka

Najideálnejšie sa mi zdalo program riešiť objektovým prístupom a zvolila som programovací jazyk java.

Spracovanie vstupu

Používateľ pri spustení programu dostane na výber či chce zadať vstup z konzoly alebo či ho chce načítať zo vstupného súbor.

Pri zadávaní vstupu z konzoly bude používateľ vyzvaný na zadanie typu selekcie, ktorú chce použiť na výber jedincov, výšku a šírku mapy, počet pokladov a ich súradnice a súradnice začiatočného bodu .

```
Zadajte typ vyberu jedinca:
0 -> SELEKCIA OHODNOTENIM (turnaj)
1 -> PROPORCIALNA SELEKCIA (ruleta)
1
Zadajte veklost mapy
x -> 3
y -> 3
Zadajte pocet pokladov -> 2
Suradnice policok s pokladmi v tvare 'x y'
1. poklad -> 1 2
2. poklad -> 0 1
Zadajte suradnice zaciatocneho policka v tvare 'x y' -> 1 1
```

Ak používateľ zvolí načítanie zo súboru, vstup sa načíta zo súboru vstup.txt, ktorý má tvar:

```
0
7:7
5
4:1
1:4
2:2
6:3
4:5
3:6
```

1. riadok: určuje typ selekcie (0- turnaj, 1- ruleta)
2. riadok: veľkosť plochy v tvare x:y
3. riadok: počet pokladov P
4. nasledujúcich P riadkov sú súradnice jednotlivých pokladov v tvare x:y
5. posledný riadok sú súradnice začiatočného políčka v tvare x:y

Súbor s nastaveniami vyzerá nasledovne:

```

16
250
150
true
15
22
40
```

1. riadok: prázdny
2. riadok: číslo- určuje koľko buniek sa inicializuje náhodne pre prvú generáciu
3. riadok: číslo- určuje počet jedincov v jednej generácii
4. riadok: číslo- určuje maximum generácii, ktoré sa vytvoria predtým než sa program stopne a počká na vstup od používateľa

5. riadok: true/false- určuje či sa použije elitizmus
6. riadok: číslo- určuje koľko percent jedincov sa vyberie ako elita z generácie
7. riadok: číslo- určuje minimálny bod kríženia
8. riadok: číslo- určuje maximálny bod kríženia

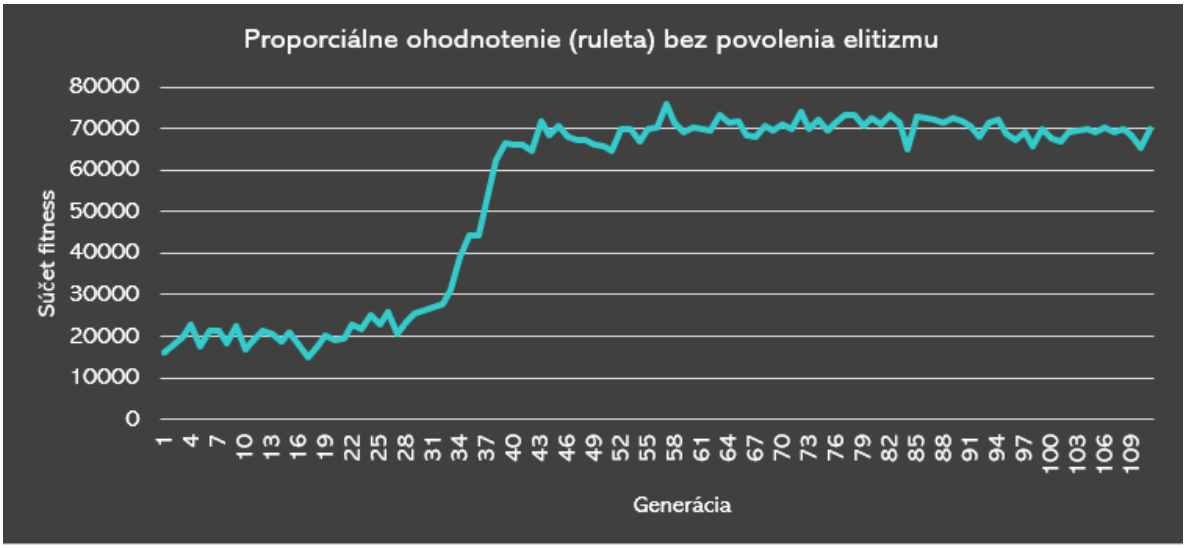
Testovanie

Priebežné testovanie programu som robila hlavne na malej vstupnej mape (3x3 alebo 4x3), ale hlavné testovanie som robila na vzorovom vstupe:

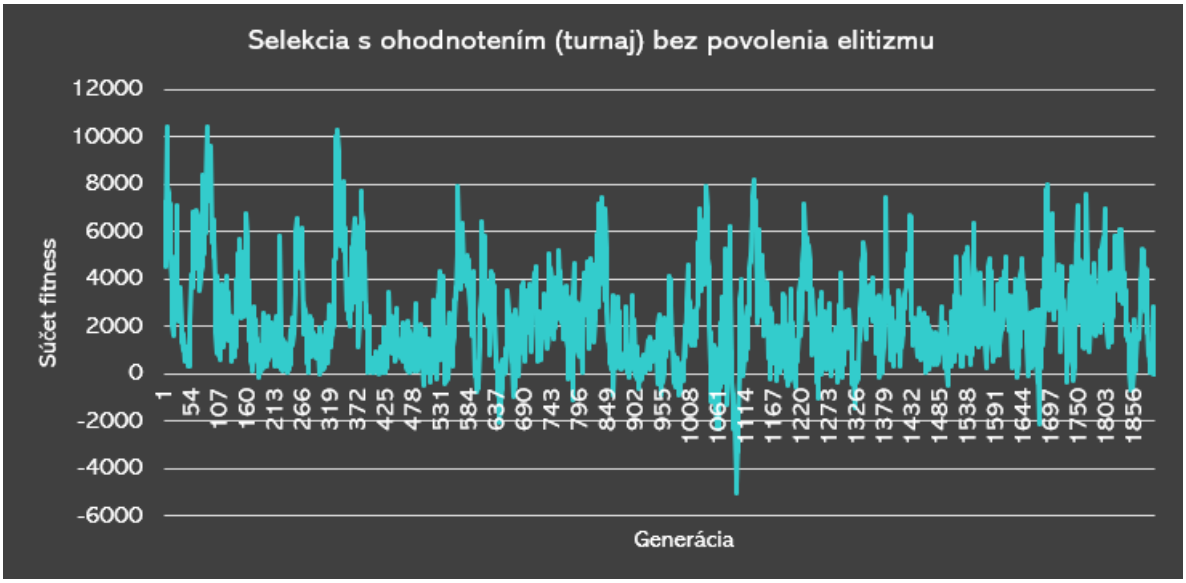
```
Zadajte veklost mapy
x -> 7
y -> 7
Zadajte pocet pokladov -> 5
Suradnice policok s pokladmi v tvare 'x y'
1. poklad -> 4 1
2. poklad -> 1 4
3. poklad -> 4 5
4. poklad -> 6 3
5. poklad -> 2 2
Zadajte suradnice zaciatocneho policka v tvare 'x y' -> 3 6


0  0  0  0  0  0  0
0  0  0  0  P  0  0
0  0  P  0  0  0  0
0  0  0  0  0  0  P
0  P  0  0  0  0  0
0  0  0  0  P  0  0
0  0  0  S  0  0  0
```

Pri testovaní som sledovala vývoj súčtu fitness funkcií pre jednotlivé generácie a podľa toho som sa snažila vylaďovať aj jednotlivé defaultné nastavenia.

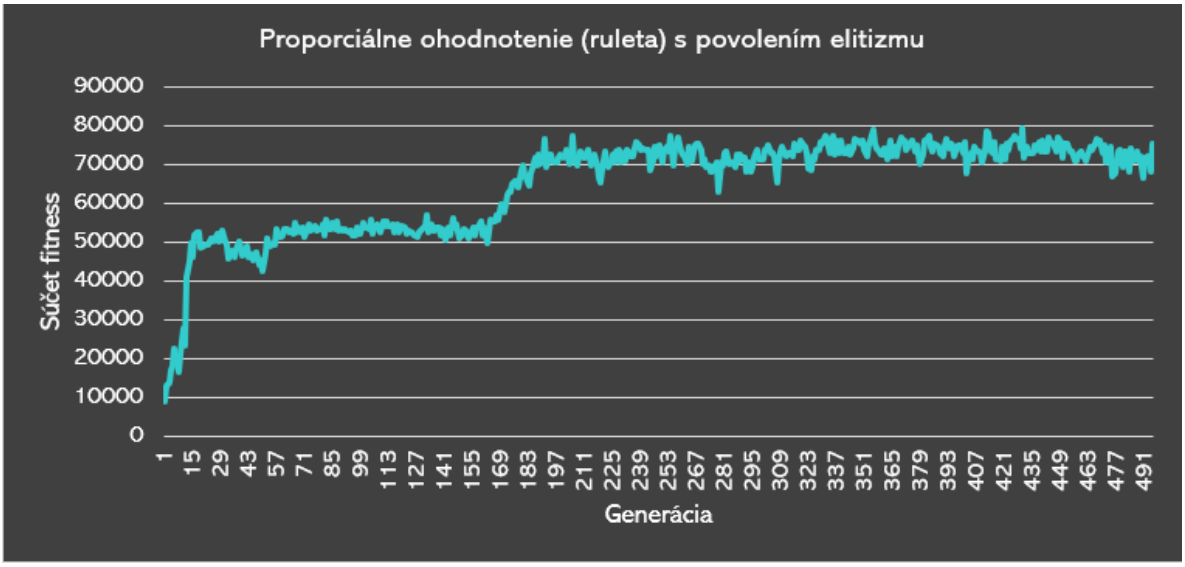
Tu je niekoľko výsledkov testov pre 4 rôzne situácie (vstup bol pri všetkých rovnaký):



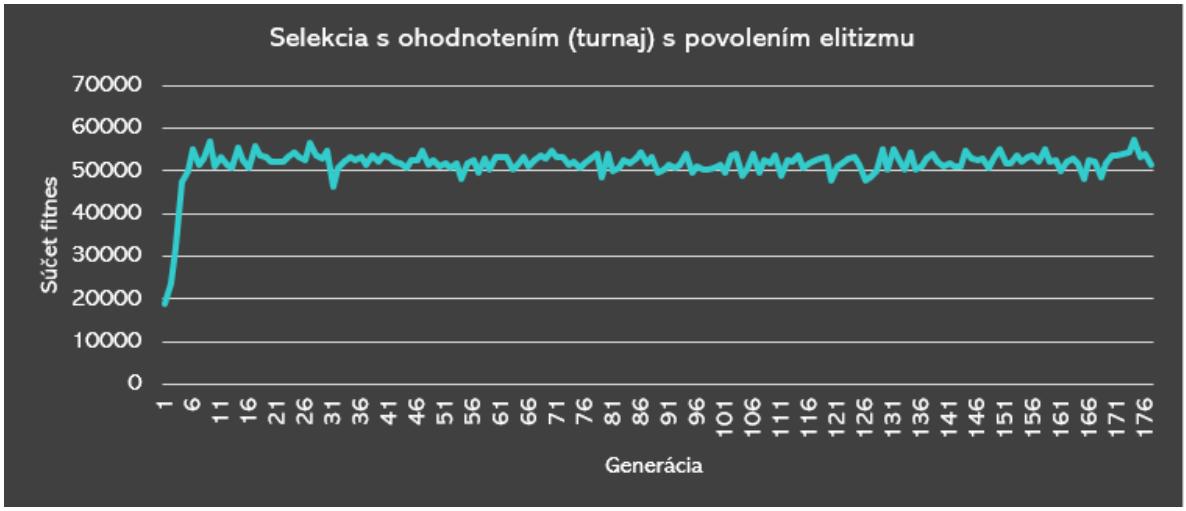

```
Vsetky poklady najdene.
GENERACIA c.114
Najlepsi jedinec:
Hodnota fitness->702
Cesta->LPPLHP#PLHLPPLHPP#LHLPDLHDDHLHL#DDL#HDPHLHLDDPHPPHPH#
```




```
Vsetky poklady najdene.
GENERACIA c.1906
Najlepsi jedinec:
Hodnota fitness->698
Cesta->HDHP#HHPHDP#LLLHHLHDPLLDP#HHPPD#DDLPHLPLHDLHLDPLDHLDHPDD#
```



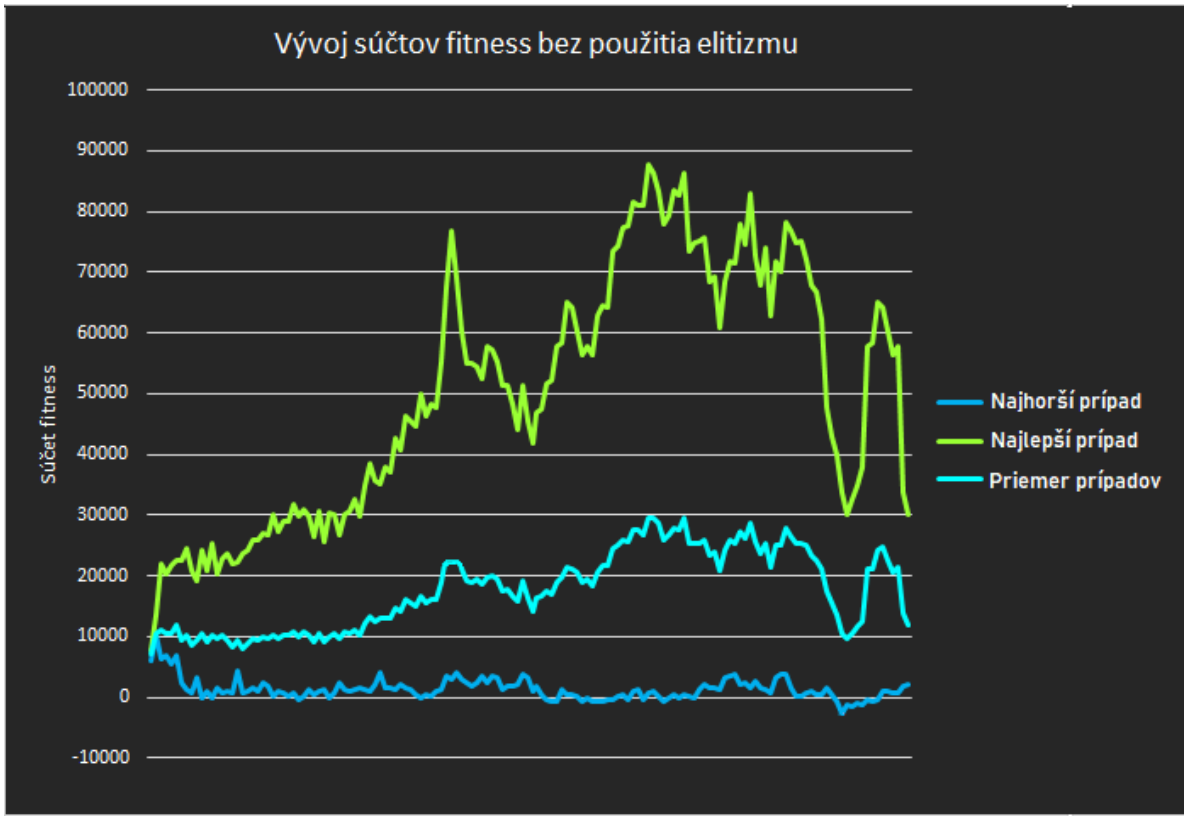

```
Vsetky poklady najdene.
GENERACIA c.496
Najlepsi jedinec:
Hodnota fitness->716
Cesta->PH#HLLL#HPH#PDDHDPPLLHPHPDDHLPP#LLHLHP#
```

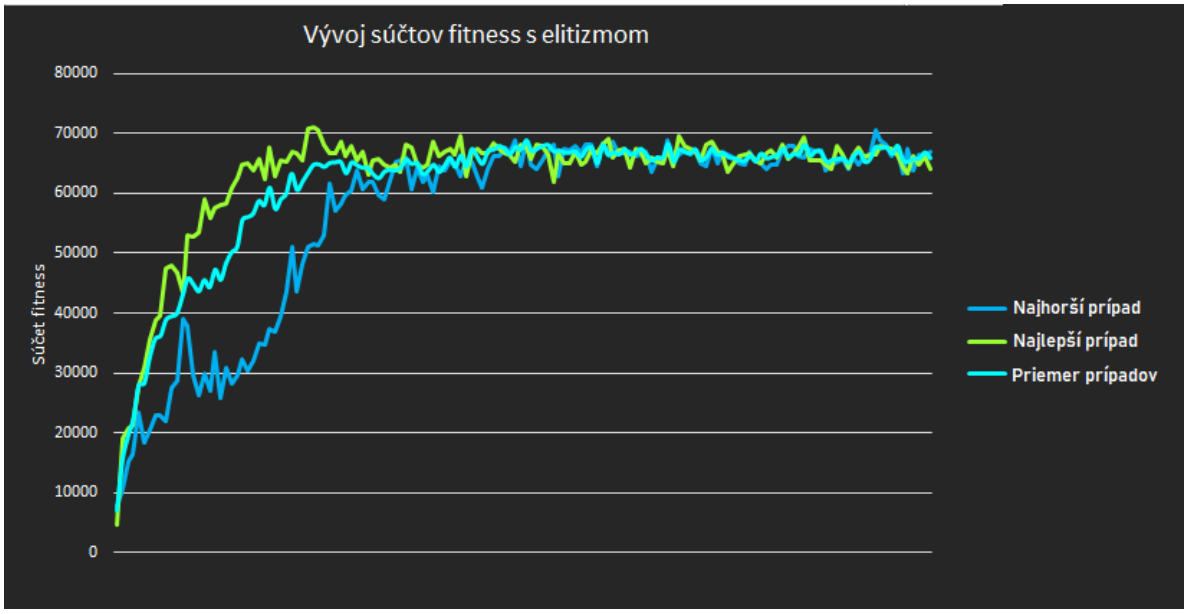



```
Successfully wrote to the file.
Vsetky poklady najdene.
GENERACIA c.180
Najlepsi jedinec:
Hodnota fitness->690
Cesta->PH#HLLPHDLPHDLPHDLPHDLPHPLPPP#LLDPLLHPLLLPLLPPLLDP#LPHPLPH#PLPHP#
```

Závery z testovania

Porovnávala som hlavne rozdiel vývoja pri použití elitizmu a bez neho. Rozdiel bol veľmi výrazný, keďže ak som elitizmus nepovolila, hodnoty súčtov fitness boli veľmi náhodne a niekedy prudko klesali (resp. prudko stúpli). Pri použití elitizmu sa jedinci postupne približovali elite- ale na druhú stranu to veľmi znižovalo variabilitu jedincov a nakoniec som mala veľa jedincov s rovnakou (ale relatívne vysokou) hodnotou fitness.





Zhodnotenie

Podľa môjho názoru moje riešenie spĺňa zadanie. Dokázala som týmto algoritmom vyriešiť aj zložitejšie zadania a väčšie vstupy (niekedy to trvalo dlhšie inokedy menej, v závislosti od prvej náhodne vygenerovanej generácie).

Zdroje

Návrat a kol.: Umelá Inteligencia, STU Bratislava, 2002, 2006, 2015. (Malé centrum)