SLOVENSKÁ TECHNICKÁ UNIVERZITA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGII

Strojové učenie sa
Evolučný algoritmus a evolučné programovanie

Predmet	Umelá inteligencia
Vypracoval	Tomáš Babjak, 91986
Akademický rok	2018/2019

7.4.2019 Bratislava

Obsah

1. Zadanie úlohy

Zenová záhradka je plocha vysypaná hrubším pieskom (drobnými kamienkami). Obsahuje však aj nepohyblivé väčšie objekty, ako napríklad kamene, sochy, konštrukcie, samorasty. Mních má upraviť piesok v záhradke pomocou hrablí tak, že vzniknú pásy ako na nasledujúcom obrázku.

Pásy môžu ísť len vodorovne alebo zvislo, nikdy nie šikmo. Začína vždy na okraji záhradky a ťahá rovný pás až po druhý okraj alebo po prekážku. Na okraji – mimo záhradky môže chodiť ako chce. Ak však príde k prekážke – kameňu alebo už pohrabanému piesku – musí sa otočiť, ak má kam. Ak má voľné smery vľavo aj vpravo, je jeho vec, kam sa otočí. Ak má voľný len jeden smer, otočí sa tam. Ak sa nemá kam otočiť, je koniec hry. Úspešná hra je taká, v ktorej mních dokáže za daných pravidiel pohrabať celú záhradu, prípade maximálny možný počet políčok. Výstupom je pokrytie danej záhrady prechodmi mnícha.

Úlohu je možné rozšíriť tak, že mních navyše zbiera popadané lístie. Listy musí zbierať v poradí: najprv žlté, potom pomarančové a nakoniec červené. Príklad vidno na obrázku nižšie. Listy, ktoré zatiaľ nemôže zbierať, predstavujú pevnú prekážku. Je potrebné primerane upraviť fitness funkciu. Za takto rozšírenú úlohu je možné získať navyše jeden bonusový bod.

Uvedenú úlohu riešte pomocou evolučného algoritmu. Maximálny počet génov nesmie presiahnuť polovicu obvodu záhrady plus počet kameňov, v našom príklade podľa prvého obrázku 12+10+6=28. Fitnes je určená počtom pohrabaných políčok. Výstupom je matica, znázorňujúca cesty mnícha. Je potrebné, aby program zvládal aspoň záhradku podľa prvého obrázku, ale vstupom môže byť v princípe ľubovoľná mapa..

2. Reprezentácia záhradky

Trieda Garden predstavuje záhradku, konkrétne dvojrozmerné pole gardenMap naplnené hodnotami 0 alebo -1 pre pôdu resp. kameň. Ďalej obsahuje premenné x a y , ktoré slúžia na vytvorenie záhradky daných rozmerov v týchto osiach. Funkciou je createGarden naplní pole gardenMap hodnotami 0 a -1. Funkcia insertStone slúži na naplnenie mapy kameňmi a je volaná vo funkcii createGarden.

3. Reprezentácia jedinca

Jedinca predstavuje trieda Monk, a obsahuje pole genes ako reprezentáciu chomozómu čiže poľa génov daného mnícha. Pomocou funkcie initializeGenes() sa prvotne inicialzujú náhodné gény jedinca, ktoré sú z daného intervalu od 0 po veľkosť obvodu záhradky-1. Každý jedinec ďalej obsahuje hodnoty ako sila(strength), fitness alebo blocked a counter, ktoré sú potrebné pre vyhodnotenie sily alebo správnosti jedinca. Ďalšie funkcie mnícha sú v nasledujúcej časti dokumentácie.

4. Algoritmus hrabania záhradky

Inštancia mnícha vytvorí inštanciu záhradky vo funkcii rakeGarden() a volaním g.createGarden() naplní jej polia. Na základe mníchovho génu sa vypočíta vstupná súradnica a určí sa z ktorej strany záhradky vošiel a ktorým smerom ju má hrabať. Gény mnícha sú teda použité iba ako počiatočné súradnice vstupu do záhrady. Mních sa teda pohybuje daným smerom vypočítaným z jeho génu a pred každým posunom na ďalšie políčko sa zisťuje či sa na danom políčku, kde by sa mal posunúť nenachádza prekážka a to vo forme kameňa alebo už pohrabanej záhradky. To sa vykoná volaním funkcie blockAhead(), ktorá vracia do podmienky if() true ak sa nachádza blok na ďalšom políčku a false ak nie. Ak vráti true, mních sa potrebuje otočiť a na výber má dva smery. Ak ide zhora alebo zdola, môže ísť vpravo alebo vľavo, tento prípad je zabezpečený funkciou changeDirection(). Do tejto funkcie ako parametre vojdú súradnice miesta, kde práve stojí mních a zistí sa či sa môže posunúť do daných strán čiže vpravo alebo vľavo. Najprv sa mních pozrie doprava (na políčko s väčšou x-ovou súradnicou) a potom doľava (na políčko s menšou x-ovou súradnicou) a kde sa nachádza nepohrabané políčko tam bude nasledovať. Ak sa nemôže pohnúť ani jedným smerom nastaví premennej blocked hodnotu true.

Po vrátení blocked ako true sa skončí hľadanie cesty a vyhodnotí sa fitness daného mnícha. Podobne to funguje aj keď mních ide sprava alebo zľava (funkcia

changeDirectionUpDown()) a na výber má cestu dole (na políčko s väčšou y-ovou súradnicou) a hore (na políčko s menšou y-ovou súradnicou). Následne je jeho smer určený funkciami up(), down(), right() a left() ktoré svojimi názvami indikujú ktorým smerom sa bude mních uberať. Ich funkcionalita je rovnaká ako pri vstupe (vyššie opísanom).

Na záver sa vypočíta fitness mnícha, čiže počet políčok, ktoré mních pohrabal, a vypíše sa cesta záhradkou (tá sa kvôli prehľadnosti po novom vypíše iba pri poslednom teda úspešnom prechode záhradkou). Vo funkcii main() si program od používateľa vyžiada informáciu o tom, či chce pokračovať vo vykonávaní ako prijímač alebo vysielač správy. Pre odoslanie inej správy je potrebné program vypnúť a nanovo zapnúť na oboch uzloch.

5. Algoritmus pre vylepšovanie jedincov

Na vylepšovanie jedincov slúži trieda Evolution, v ktorej sa na začiatku inicializuje prvá generácia jedincov pomocou funkcie initialize(). Pre každého mnícha z populácie vygeneruje náhodné gény a následne mních pohrabe záhradku – proces opisovaný v predošlej kapitole. Na základe výslednej hodnoty fitness sú mnísi zoradený a je im vypočítaná ich sila pomocou funkcie findStrength(). Pri výpočte sily som sa inšpiroval knihou P. Návrata: Umelá inteligencia, s.350. Metóda nájde silu pre všetkých jedincov populácie usporiadaných podľa ich fitness hodnoty.

Následne je vytvorená nová generácia mníchov na základe tejto prvotnej. Tento proces, ktorý je kľúčový pre program sa odohráva vo funkcii newPopulation(). Robí to podobne ako funkcia initialize(), ale namiesto vygenerovania náhodných génov pre jedincov, volá na základe hodnoty v EVOLUTIONTYPE funkcie evolve(1,2,3...), ktoré vylepšujú gény jedincov napríklad kríženiami alebo mutáciami.

Funkcie evolve() vznikali postupne, ako som sa snažil vylepšiť a analyzovať jednotlivé prístupy k mutáciám a kríženiam. Na výber jedinca som použil dve spôsoby a to ruletu a turnaj.

Na výber pomocou turnaja (funkcia tournament()) som vybral dve náhodné čísla z rozsahu 0..numberOfMonks(počet mníchov v populácii) – 1 a vybral som toho z nich, ktorý ml vyššiu hodnotu sily (tým pádom aj hodnotu fitness, keďže sila sa vyvíja úmerne s fitness).

Pri výbere ruletou som najprv použil výber na základe sily, ktorú som vypočítal na základe knihy P. Návrata: Umelá inteligencia, spočítal som sily všetkých jedincov a vynásobil náhodným číslom z intervalu (0,1>. Následne od tejto hodnoty odčítaval sily jedincov populácie

od najslabšieho, čiže s najnižšou hodnotou strength. Jedinec, pri ktorom sa hodnota „kvázi súčtu“ zníži po odčítaní pod nulu sa stane vyhľadaným rodičom pre kríženie resp. mutáciu.

Tento spôsob mi pri testovaní neprinášal však požadované hodnoty a v porovnaní s turnajom bola omnoho slabšia, a preto som sa rozhodol vytvoriť novú funkciu, ktorá by nehodnotila jedincov podľa sily, ale podľa samotnej fitness hodnoty jedincov. Túto funkciu som nazval myRoulette() a vykonáva sa v podstate rovnakým spôsobom ako predošlá, avšak s rozdielom že sa spočítavajú a následne odčítavajú hodnoty fitness jedincov. Táto funkcia, ako bude spomenuté v kapitole Testovanie, prinášala markantne lepšie výsledky pri kríženiach a mutáciách.

5.1 Kríženie chromozómov

Kríženie chromozómov mníchov sa vykonáva vo funkciách crossoverTournament() a crossoverRoulette() a jednoduché rozoznať podľa názvu aký spôsob výberu rodičov sa kde využíva. Prvá zo spomínaných funkcií zobrazuje spôsob kríženia, pri ktorom sa pomocou turnaja vyberú dvaja rodičia a vypočíta sa limit pre kríženie – náhodné číslo z intervalu (0,1> sa vynásobí počtom génov jedinca. Na základe tohto limitu sa vyberú gény pre nového jedinca od prvého rodiča(od začiatku chromozómu po daný limit) a od druhého rodiča (od limitu po koniec chromozómu). Inšpirácie ku kríženiu: Návrat P.: Umelá inteligencia s.348 a samozrejme cvičenia ku predmetu UI. Druhá zo spomínaných funkcii funguje na rovnakom princípe avšak rodičia sú vyberaný pomocou funkcii roulette() a myRoulette() – ktorá z nich sa využije je určená boolean hodnotou myRoulette (true – pouťitá funkcia myRoulette).

5.2 Mutácia chromozómov

Mutáciu chromozómov som riešil pomocou komplementu k danému číslu, ktoré reprezentuje gén v chromozóme a to tak že od maximálnej veľkosti čísla v géne odčítal hodnotu génu rodiča spravil som absolútnu hodnotu tohto výsledného čísla. To sa deje vo funkciách mutationComplementTournament() a mutationComplementRoulette().

Pri tvorení nových generácii vo funkciách evolve() som použil okrem predošlých spôsobov aj elitizmus, ktorý je znázornený funkciou elitism() (len kvôli lepšej prehľadnosti, dalo sa to spraviť jednoduchšie), kde vyjadrujem, že jedinci, ktorý majú najvyššiu hodnotu

fitness majú právo zotrvať aj v nasledujúcej generácii (Návrat P.: Umelá inteligencia s.349). Tento spôsob som využil pri desatine všetkých jedincov a pri skoro každej z evolve() funkcii.

Taktiež som využil formu mutácie, kde som vymenil celý chromozóm jedinca, ktorý mal najnižšiu hodnotu fitness a nahradil ju náhodnými génami aby som takto zamedzil uviaznutiu pri vysokých fitness hodnotách a priniesol tzv. „novú krv“ teda potenciálne lepšie gény na kríženie.

5.3 Typy evolúcii

Evolve1: elitizmus, kríženie a výber ruletou(fitness), kríženie a výber turnajom, mutácia a turnaj, mutácia a ruleta(fitness), náhodná generácia

Evolve2: elitizmus, kríženie ruleta(sila), mutácia ruleta(sila), náhodná generácia

Evolve2.1: elitizmus, kríženie ruleta(fitness), mutácia ruleta(fitness), náhodná generácia

Evolve3: elitizmus, kríženie turnaj, mutácia turnaj, náhodná generácia

Evolve4: elitizmus, kríženie turnaj, kríženie ruleta(sila), náhodná generácia

Evolve4.1: elitizmus, kríženie turnaj, kríženie ruleta(fitness), náhodná generácia

Evolve5: elitizmus, mutácia ruleta(sila), mutácia turnaj, náhodná generácia

Evolve5.1: elitizmus, mutácia ruleta(fitness), mutácia turnaj, náhodná generácia

Evolve6: kombinácia 4.1 pre fitness najlepšieho prvku hodnotu menšiu ako maximálna možná fitness -1 a 2.1 ak sa rovná maximálnej možnej fitness bez jednej, chcel som použiť to najlepšie z kríženia ak je ešte ďaleko od maxima a pridať mutáciu keď sa blíži k maximu aby sa nezaseklo na lokálnom maxime.

Evolve7 – random: náhodné gény vygenerované pre každú populáciu.

Na záver sa vykoná hrabanie záhradky novou generáciou mníchov, vyhodnotí sa ich fitness a sila a zistí sa, či sa v novej populácii nenašiel mních, ktorý pohrabal celú záhradku. Ak sa našiel, ukončí sa vykonávanie procedúry a prípadne sa vyhodnotia populácie a mních, v opačnom prípade sa opäť zavolá procedúra newPopulation(), ale s novou populáciou ako parametrom. Taktiež ak počet generácií presiahol stanovenú hodnotu v konštante

MAXGENERATIONS skončí sa vykonávanie a vráti mnícha, ktorý mal najlepšiu fitness hodnotu doteraz.

6. Postupné vylepšovanie programu

Program som postupne vylepšoval napríklad tak, že som hľadal ako optimalizovať riešenie a prišiel som k záveru, že mnoho génov mnícha sa nikdy ani nedostane do záhradky (políčko na vstupnom mieste je už pohrabané) a preto som tieto gény z mníchovho chromozómu vymazal (v triede Monk, ak sa dostal gén do záhrady inkrementovala sa jeho hodnota validGene) t.j. gény s hodnotou validGene == 0 som zmenil na -1. Následne pri tvorbe novej generácie som nahradil gény s -1 novými náhodnými génmi. Táto oprava génu významne zvýšila efektivitu programu.

Pri testovaní programu som sa snažil maximalizovať funkciu na modifikovanie génov a to tak, že som hľadal ktorý spôsob poskytne vo finále najlepšie výsledky. Na základe pozorovania a znalostí som vypracoval funkciu evolve6(), ktorá by mala byť najúspešnejšou. Je kombináciou evolve4.1(), ktorá prináša trvalo najlepšie výsledky pri testovaní vďaka kríženiu a kombinácii výberu jedincov ako turnajom tak aj ruletou(pomocou fitness hodnoty) a evolve2.1(), ktorá tiež prináša stabilné výsledky a znižuje, pri dosiahnutí lokálneho maxima, šancu na zaseknutie vďaka mutácii.

Pri testovaní som prišiel k záveru, že výber rodičov pomocou rulety mi pre testovanú mapu dáva lepšie výsledky (porovnaj hodnoty funkcii evolve2 a evolve3) a preto som ruletu vo funkcii evolve6() použil pri väčšej časti kríženia a pri mutácii.

Výsledky testovania výberu jedincov pomocou rulety ďalej ukázali, že výber pomocou hodnoty fitness dosiahli približne rovnaké ale o niečo lepšie výsledky ako výber na základe sily preto som ju použil aj vo funkcii evolve6().

7. Testovanie

Testovanie prebiehalo na dvoch mapách (pozri súbor Mapy.txt) rôznych rozmerov a usporiadaní kameňov. Testoval som správnosť vykonávaného algoritmu hrabania záhradky a efektivitu krížení resp. mutácii a jednotlivých funkcii na vývin chromozómov. Testovanie prebiehalo na vzorke 100 jedincov v populácii. Počet opakovaní – teda počet vytvorení prvej náhodnej populácie 100 mníchov s náhodnými génmi a jej vývin a hľadanie riešenia (trieda Evolution) – sa opakuje podľa konštanty zadefinovanej v triede Main ako NUMBEROFREPS. Pri veľkých testoch so toto číslo nastavil na 1000 a pri malých na 100 opakovaní. Vysoký počet

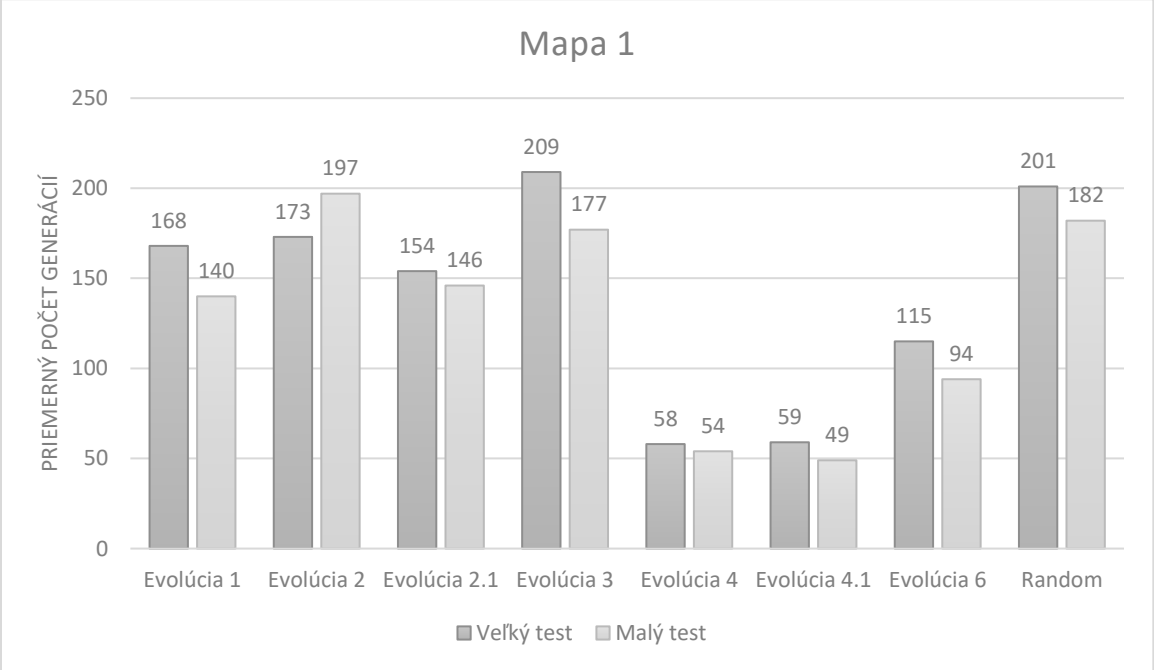
opakovaní som zvoli aby som s vyššou presnosťou určil priemerný počet generácii potrebný na nájdenie riešenie pri rôznych druhoch evolúcii. Počet generácii je ukladaný do textového súboru a následne sa hodnoty z tohto súboru sčítajú a vydelia počtom opakovaní.

7.1 Testovanie správnosti

Správnosť algoritmu na hrabanie záhradky som vykonával manuálne a to výpisom pohrabanej záhradky mníchom, v skoršom štádiu postupnými krokmi hrabania a porovnávaním s génmi daného mnícha – či naozaj vošiel odkiaľ podľa génu mal vojsť do záhradky. Preto nie sú vo výpise pohrabanej záhradky cesty v poradí postupne 1,2,3.., ale sú tam čísla génov(+1) avšak stále v rastúcom poradí 1,3,4,7.. . Správnosť si môžete overiť v priložených textových súboroch, kde sú výpisy máp po testovaní a nad každou z nich gén mnícha, ktorý túto mapu vygeneroval.

7.2 Testovanie efektivity

Efektivitu riešenia som testoval na základe veľkých aj malých testov a priemerného počtu generácií mníchov pri hľadaní riešenia. Výsledky testovania nájdete v priložených textových súboroch a súhrnné výsledky v grafoch:



Hodnoty pre Evolúciu 5 a 5.1 som neuvádzal kvôli vysokej odchýlke od ostatných meraní.

8. Zhodnotenie

Z grafu Mapa 1 vyplýva, že ako najefektívnejšia sa pre danú mapu javia metódy evolúcie 4 a 4.1, čiže tie, pri ktorých sa používa najviac kríženie a metóda mutácie iba formou výmeny

celého chromozómu slabých jedincov. Celkom dobre sa ukázal aj spôsob 6, ktorý prekonal náhodné gény dvojnásobne. Čakal som od neho však lepšie výsledky, keďže mal výhody mutácií a kríženia, ale očividne spôsob mutácie (komplement), ktorý som našiel v knihe Umelá inteligencia a použil v takmer všetkých funkciách okrem 4 a 4.1 sa pri tomto type problému neosvedčil.

Na druhej strane, použité kríženie pri Evolúcii 4, 4.1 znížilo počet generácii oproti náhodne vygenerovaným číslam takmer štvornásobne, čo dokazuje efektivitu kríženia a celkovo aj evolučného programovania ako takého. Predpokladám, že pri mapách väčších rozmerov alebo pri zložitejších mapách s väčším počtom kameňov by sa tento rozdiel ešte viac prehlboval.

9. Používateľská príručka

Ak chcete zadefinovať novú záhradu, ktorú má mních hrabať, je potrebné nastaviť údaje v triede Main a to konštanty X a Y pre rozmery mapy a NUMBEROFSTONES ako počet kameňov ktoré budete vkladať. Ďalej je potrebné tieto kamene poukladať na svoje pozície v triede Garden, konkrétne vo funkcii insertStone(), kde je prvý rozmer poľa gardenMap chápaný ako súradnica Y a druhý rozmer súradnica X.

Ak si prajete zmeniť veľkosť populácie je potrebné vo triede Main zmeniť konštantu NUMBEROFMONKS a ak chcete zmeniť počet opakovaní, pre ktoré sa vykonáva evolúcia populácie musíte zmeniť konštantu NUMBEROFREPS.

Používateľ dostane po spustení a zmehnutí programu správu do konzoly, v ktorej sa bude nachádzať NUMBEROFREPS génov, máp a správ o počte generácii potrebných na nájdenie pohrabanej mapy. Možný výstup pre danú konštantu rovnajúcu sa jednej je:

```
7 41 10 27 22 31 42 25 -1 0 -1 -1 8 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 30 -1 -1
Number of generations: 5
10  10  10  10  10  10  4  1  13  13  3  5
7  7  7  7  7  -1  4  1  13  13  3  5
2  -1  7  7  7  7  4  1  13  13  3  5
2  7  7  7  -1  7  4  1  13  13  3  5
2  7  -1  7  7  7  4  1  13  13  3  5
2  7  6  6  6  6  4  1  13  13  3  5
2  7  6  26  26  6  4  1  -1  -1  3  5
```

2 7 6 26 26 6 4 1 8 8 3 5

2 7 6 26 26 6 4 1 8 8 3 5

2 7 6 26 26 6 4 1 8 8 3 5

File deleted successfully

Priemerny pocet generácií mnichov je 5

Process finished with exit code 0

Pre viac testov navštívte priečinok tests, kde sa nachádza veľké množstvo výstupov pre rôzne typy evolúcii a počiatočné gény. Kvôli prehľadnosti výstupu pri väčších množstvách opakovaní sa v ňom neuvádzajú postupne všetky gény a cesty záhradou a ani priemerný a najlepší chromozóm generácie. Každopádne aj výpis maximálnej a priemernej fitness hodnoty sa dá zapnúť odkomentovaním kódu na riadku 161 v triede Evolution alebo si ho môžete pozrieť aj s grafmi v prílohách 1 a 2, kde boli použité evolučné funkcie 6 resp. 4.

Na to, aby hľadanie vhodného chromozómu netrvalo príliš dlho a ak sa program zasekne niekde v lokálnom maxime a nedokáže sa pohnúť slúži na riešenie týchto situácii konštanta MAXGENERATIONS, ktorá určuje koľko najviac generácií mníchov sa môže vytvoriť.

Zoznam použitej literatúry

Pavol Návrat a kol. – Umelá inteligencia (2002)

Príloha 1

Average fitness is 77 Max fitness is 106

Average fitness is 78 Max fitness is 108

Average fitness is 76 Max fitness is 112

Average fitness is 78 Max fitness is 112

Average fitness is 79 Max fitness is 112

Average fitness is 82 Max fitness is 112

Average fitness is 81 Max fitness is 112

Average fitness is 81 Max fitness is 112

Average fitness is 86 Max fitness is 112

Average fitness is 86 Max fitness is 112

Average fitness is 81 Max fitness is 112

Average fitness is 85 Max fitness is 112

Average fitness is 84 Max fitness is 113

Average fitness is 75 Max fitness is 112

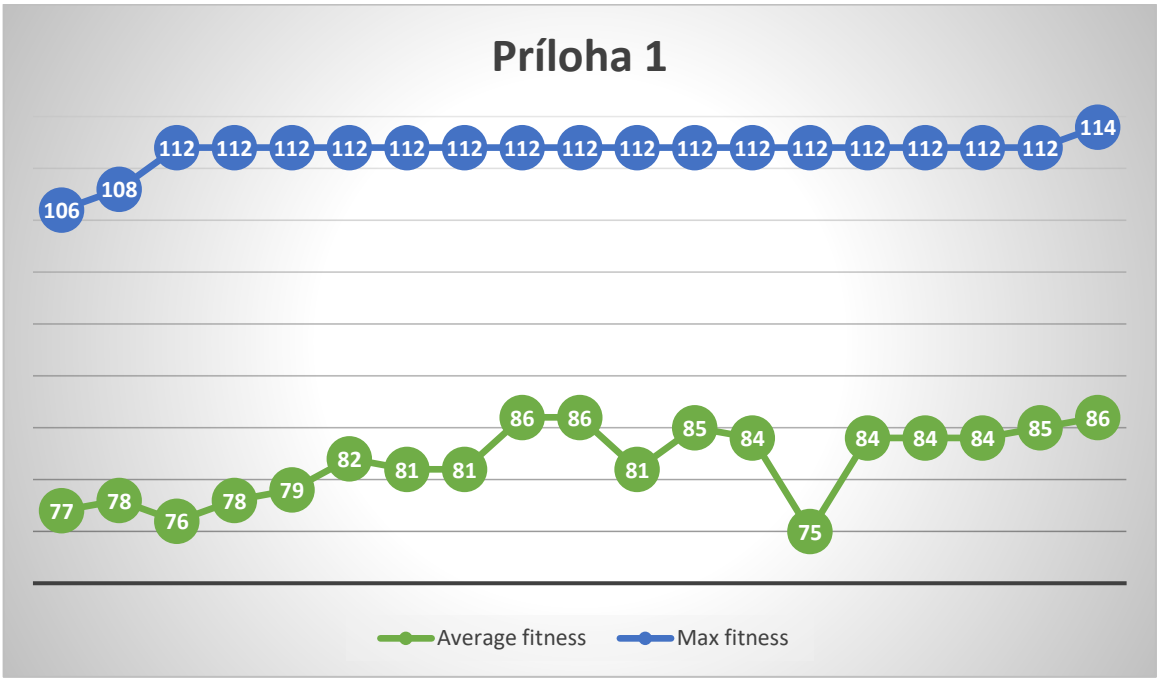
Average fitness is 84 Max fitness is 112

Average fitness is 84 Max fitness is 112

Average fitness is 84 Max fitness is 112

Average fitness is 85 Max fitness is 112

Average fitness is 86 Max fitness is 114



Number of generations: 18

Príloha 2

Average fitness is 73 Max fitness is 102

Average fitness is 73 Max fitness is 105

Average fitness is 76 Max fitness is 108

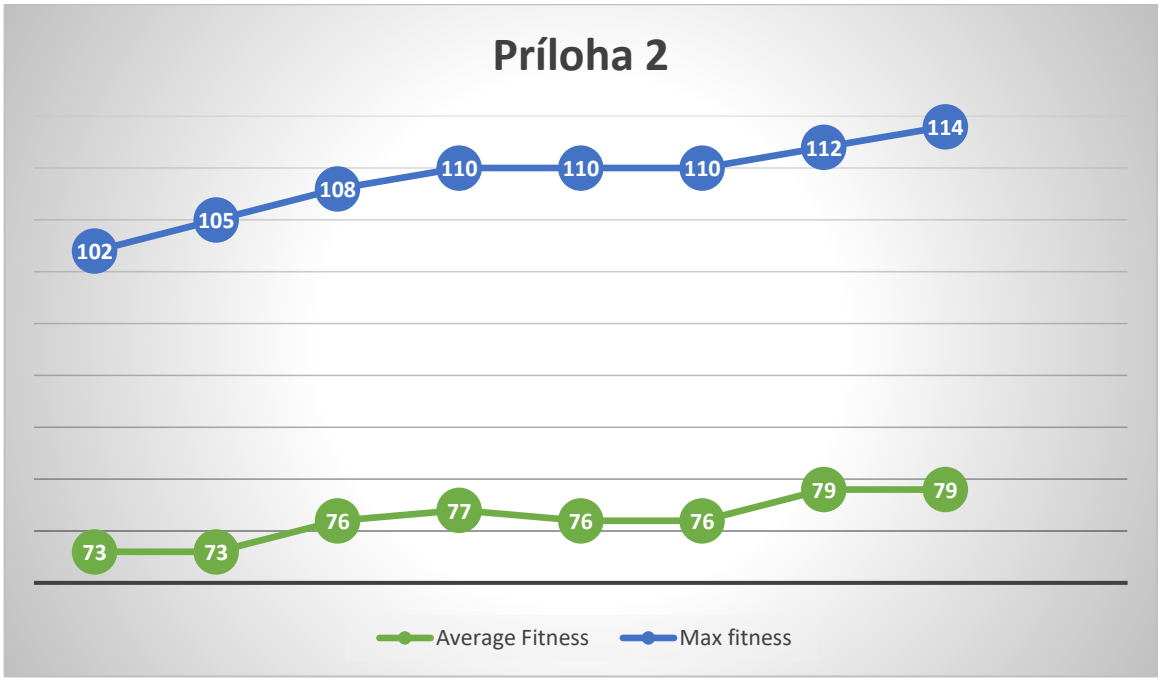
Average fitness is 77 Max fitness is 110

Average fitness is 76 Max fitness is 110

Average fitness is 76 Max fitness is 110

Average fitness is 79 Max fitness is 112

Average fitness is 79 Max fitness is 114



Number of generations: 7