Evoluční návrh přibližných násobiček pomocí CGP

František Horázný

Zadání (volně zjednodušeno):

Implementujte nebo aplikujte existující implementaci CGP na návrh n-bitových násobiček. Provádějte systematické experimenty a vyvoďte z nich závěr.

Cíle:

- Zprovoznit Tools4CGP (https://www.fit.vutbr.cz/~vasicek/cgp/tools/)
- Analyzovat kód Tools4CGP a pochopit.
- Pozměnit kód Tools4CGP. Přesněji fitness funkci a požadované chování výstupní funkce (n-bitová násobička)
- Analyzovat výsledky a zpracovat případné grafy a tabulky.

Zkrácený popis CGP:

CGP neboli kartézské genetické programování je velmi podobné genetickému programování. Rozdíl je v topologii. CGP má předem určenou mřížku N x M ve které se jednotlivé vrstvy mohou propojit a každá buňka v našem případě bude představovat jedno hradlo. Výsledkem však nemusí, ale může být všech N x M hradel, protože některé buňky svým výstupem nemusí mít vliv na výsledek.

Popis možných metrik chybovosti:

- Mean Absolute Error je průměrná absolutní tedy aritmetická chyba ve výpočtu.
- Worst Case Error je nejhorší možná chyba ve výpočtu.
- Mean Relative Error je průměrná relativní chyba. Tedy zohledňuje procentuální velikost chyby ku velikosti samotného správného výsledku.

Nástroje a využitelné řešení:

Ke generování násobiček byl využit set instrumentů Tools4CGP, který obsahuje kód pro evoluci, fittness funkci, nástroj pro generování hlavičkového souboru z pravdivostní tabulky tab2h i zobrazovač výsledných chromozomů. Tento soubor nástrojů bude využit jako hlavní zdroj již hotové implementace, ale i informací.

Implementace:

Implementován byl generátor pravdivostní tabulky *4bitnasobicka.h.* Při generování bylo zjištěno, že osmi bitová násobička je příliš složitá pro účely projektu a zároveň není složité případné rozšíření, proto dále bylo experimentováno a implementováno pouze s čtyř-bitovými násobičkami.

Dále byla doplněna funkce *int uzitosloupcu(chromozom p_chrom).* Tato funkce počítá podobným způsobem jako funkce *uzitobloku*. Výsledkem je počet využitých vrstev, tedy zpoždění násobičky.

Jako poslední bylo potřeba implementovat generátor různě velkých násobiček. Původní funkce *main* byla zaměněna za novou, která periodicky volá původní *main* s různou velikostí mřížky CGP. Výsledek vypisuje na standartní výstup a ten lze dále zpracovat.

Jeden z cílů změnit fitness funkci bohužel nebyl splněn, protože i po mnoha hodinách bádání a přemýšlení autor nedokázal vymyslet jak jednoduše reimplementovat funkci tak aby neutrpěla rychlost evoluce a psychické zdraví autora. Musel by se totiž reimplementovat celý nástroj tab2h, vymyslet novou strukturu vstupních dat, celý průběh by byl podstatně pomalejší, protože aktuální funkce je optimalizovaná na vyšší výkon a účinnou paralelizaci a velmi pravděpodobně by tato změna vedla k výslednému celému předělání původního nástroje na výpočet CGP. Toto nesplnění cíle vedlo k nemožnosti využití výše popsaných metrik chybovosti, avšak původní fitness funkce určitě lze také považovat za metriku chybovosti.

Experimenty:

Experimentováno bylo s velikostí CGP mřížky, s počtem generací, počtem mutací i s počtem různých běhů. Na počet hradel má z jasných důvodů největší vliv velikost CGP mřížky. Počet generací má velký vliv na dobu výpočtu a zpřesňuje (zlepšuje) poměr počtu hradel a fitness. To znamená že graf který je níže by měl body všechny více na jedné křivce (méně rozprostřené v prostoru) a touto křivkou by byla ROC křivka. Počet mutací má vliv na možnost změn, což znamená že rychleji konverguje k nějakému výsledku, ale následně je menší šance, že se nový jedinec přijme. Naopak má větší šanci se dostat z lokálního maxima. Počet různých běhů má pak čistě náhodný charakter. Tedy je šance začít v lepším místě a tím najít lepší řešení. Počet běhů je samozřejmě potřebný k vykreslení grafů.

S přihlédnutím na potřebu, aby experiment netrval věčně a zároveň dal smysluplné výsledky, byl konečný experiment spuštěn s těmito parametry:

- Počet generací 500000
- Maximální l-back
- 10 mutací
- 5 jedinců v populaci
- 5 běhů v jednom nastavení
- Nastavení mřížky 2x2 až 15x10 ve všech kombinacích

Výsledkem byl přibližně hodinový běh. Data jsou v souboru vysledky15x10.txt. Sloupce jsou:

- Fitness
- Počet hradel
- Počet vrstev hradel
- Počet sloupců generované mřížky
- Počet řádků generované mřížky

Výsledkem jsou přiložené dva grafy, kde je přidaný záznam (červeně) konvenční násobičky sestávající z 74 hradel v 12 vrstvách s maximální fitness.

Závěr:

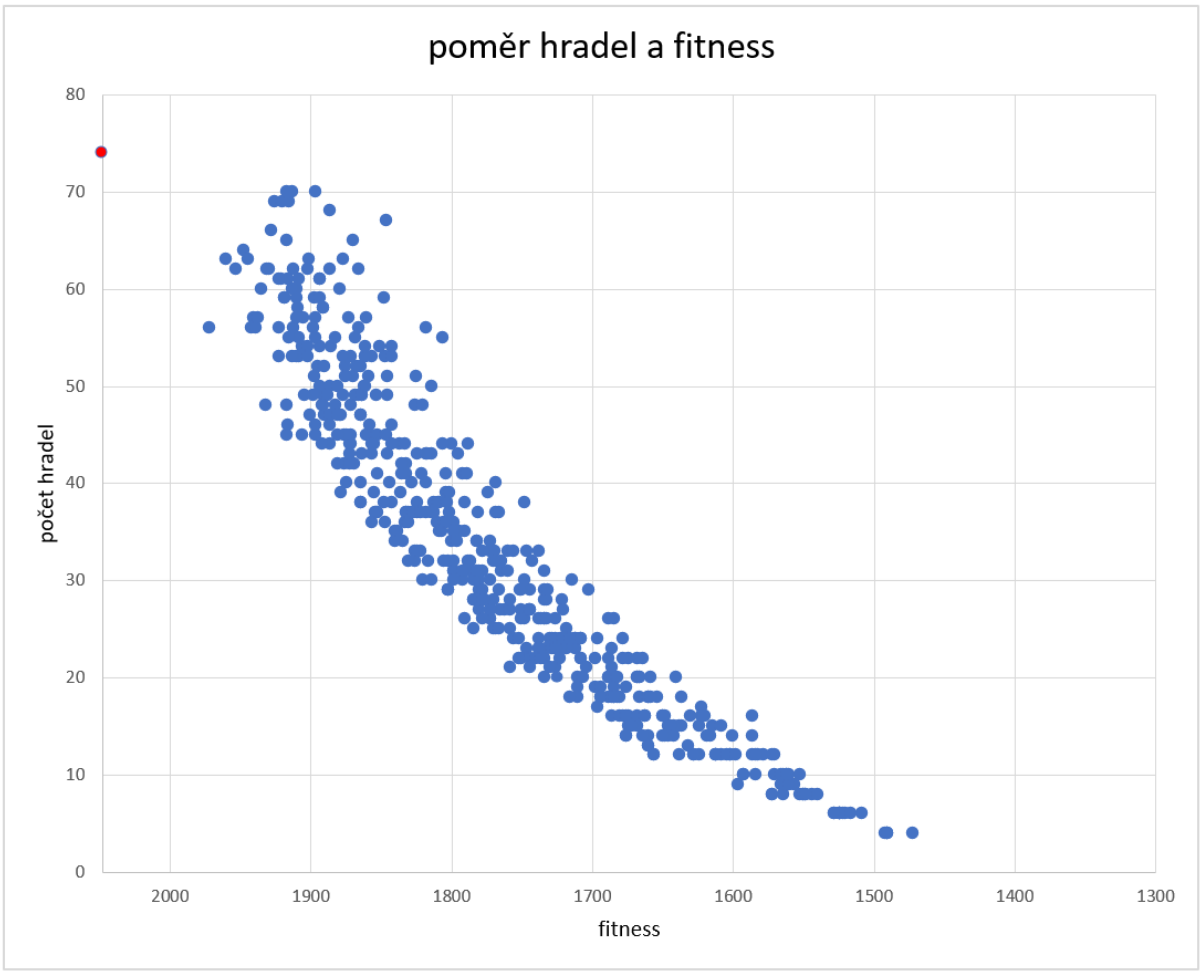
Bohužel se mi nepodařilo část zadání splnit, ale se splněnou částí jsem spokojen. Výsledné grafy velmi názorně ukazují možnost využití evolučních algoritmů v těchto odvětvích. Analytickým a konvenčním způsobem většinou nelze jednoduše najít rychlejší řešení za cenu určitých vědomých chyb. Evoluční algoritmus najde nejlepší možné řešení v zadaných mezích sám.

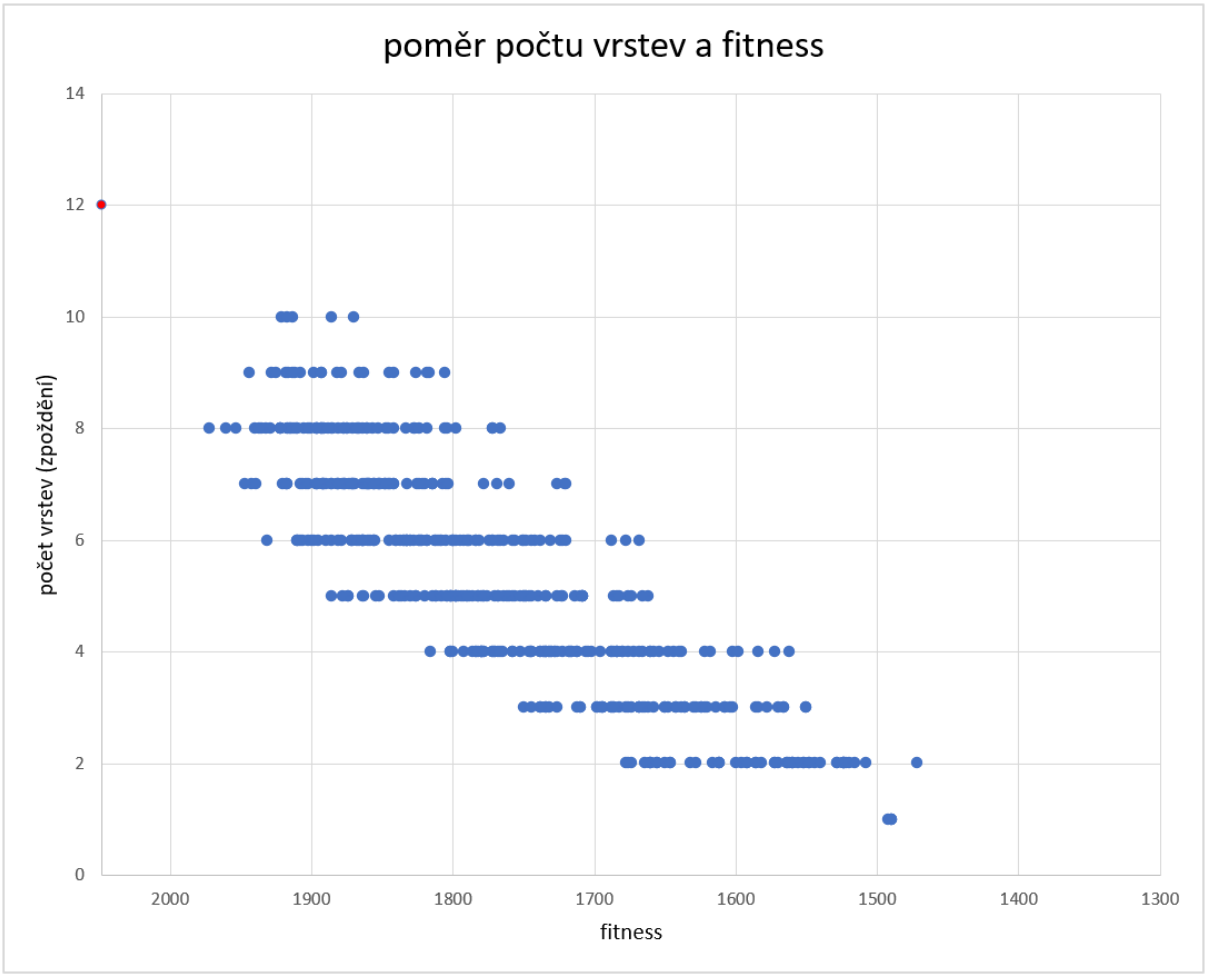
Spuštění:

```
python 4bnasobicka.py > 4bnasobicka.txt
        // vytvoří pravdivostní tabulku 4 bitové násobičky

tab2h 4bnasobicka.txt > 4bnasobicka.h
        // vytvoří hlavičkový soubor pro 4 bitovou násobičku

./cgp
```


poměr hradel a fitness
počet hradel
fitness
80
70
60
50
40
30
20
10
0
2000
1900
1800
1700
1600
1500
1400
1300


poměr počtu vrstev a fitness
počet vrstev (zpoždění)
fitness
14
12
10
8
6
4
2
0
2000
1900
1800
1700
1600
1500
1400
1300