# Aproximace obvodů – programová dokumentace

Petr Dvořáček – xdvora0n@stud.fit.vutbr.cz

## 1 Přeložení knihovny a spuštění projektu

Knihovna byla implementována v jazyce Cython. Jedná se o jazyk Python obohacený o statické typování. Díky tomu jsou pak výsledné programy rychlejší, než kdyby byly napsány v čistém Pythonu.

Před instalací je nutné se ujistit, že pracujete s 64 bitovým strojem s Linuxem. Dále je nutné mít stažený Cython ve verzi 0.20.1 a interpret Pythonu ve verzi 3.4.0. Pak stačí spustit příkaz `make` ve složce se zdrojovými kódy.

Vešekerá instalace byla provedena na operačním systému Arch Linux ve verzi 3.14.1-1. Celkový přehled příkazů:

```
$ pacman python
$ pacman cython
$ make
```

Pro spuštění vyhledávání aproximačních obvodů pro 3+3 sčítačku spusťte příkaz:

```
$ python3 run.py
```

Pro spuštění vyhledávání aproximačních obvodů pro 3x3 násobičku spusťte příkaz:

```
$ python3 run2.py
```

Pro spuštění vyhledávání aproximačních obvodů pro 4+4 sčítačku spusťte příkaz:

```
$ python3 run3.py
```

## 2 Použití knihovny – tutoriál

Pro použití knihovny spusťte interpret pythonu, ve složce obsahující soubor `cgp.cpython-34m.so`. V interpretu, či zdrojovém kódu, pak importujte knihovnu.

```
from cgp import Cgp
```

Vytvoření nového objektu, vyžaduje tři parametry. Z toho dva jsou údaje o velikosti grafu CGP – počet řádků a sloupců. Ten třetí pak značí míru propojení grafu.

```
cgp = Cgp(row, col, lback)
```

Kvůli rychlosti běhu programu, jsou operace uzlů grafu vždy stejné a uživatel je nemůže měnit. Množina operací je pak následující $\Gamma = \{id, and, or, xor, not, nand, nor, nxor\}$.

Dále je nutné naplnit objekt vstup/výstupními daty. Nejsnadnější variantou je, že se vytvoří soubor, kde na každém řádku jsou data zapsána ve formátu: `vstup : výstup`. Například pro poloviční jednobitovou sčítačku by soubor obsahoval:

```
00 : 00
01 : 10
10 : 10
11 : 01
```

Počet vstupů a výstupů do grafu se odvodí právě z formátu souboru. Pro naplnění obejktu daty, pak stačí zadat příkaz:
```
cgp.file("scitacka.txt")
```

Pro spuštění algoritmu můžete použít dvě metody výpočtu fitness funkce. První z nich se snaží minimalizovat Hammingovu vzdálenost (SHD) mezi referenčním a výstupem z GP. Druhý z nich se snaží minimalizovat Hammingovu vzdálenost s využitím priorit. Je to chybně označeno jako SAD.
```
cgp.runSHD(generation, population, mutation, runs, acc)
cgp.runSAD(generation, population, mutation, runs, acc)
```
První parametr značí počet generací, druhý velikost populace, třetí četnost mutací, čtvrtý počet běhů a poslední maximální počet použitých bloků.

Po doběhnutí CGP lze manipulovat s nalezenými výsledky. Pro nejlepší nalezený chormozóm zadejte:
```
cgp.chrom
```

Chcete-li tento chromozom ve formátu vhodného pro CGP-viewer, či pouze do lépe čitelné formy, pak spusťte příkaz:
```
cgp.resultChrom()
cgp.showChrom()
```

Počet použitých uzlů:
```
cgp.usedNodes
```

Nejlepší nalezená fitness:
```
cgp.fitness
```

Pro aproximační obvody lze použít chybovou tabulku, kde prvním parametrem je chromozom. Druhý resp. třetí parametr je počet vstupních bitů prvního resp. druhého operandu.
```
cgp.createErrorTable(chrom, col_bin, row_bin)
```

Byl použit upravený program z mé BP.
Petr Dvořáček: Kartézské genetické programování v jazyce Python, bakalářská práce, Brno, FIT VUT v Brně, 2013