Problém batohu - řešení pokročilou iterativní metodou

Simulované ochlazování

Popis metody

Nejprve jsem řešil problém batohu metodou simulovaného ochlazování. Tato metoda funguje tak, že si nejprve vygeneruje nějaké řešení a to postupně mění. Každá změna přidá, nebo odebere věc z batohu a tím vytvoří novou konfiguraci. Pokud je nová konfigurace lepší, vždy je přijata, pokud je horší, je přijata s určitou pravděpodobností, která se ale časem snižuje.

Moje implementace

Mnou naprogramovaný algoritmus začíná s prázdným batohem. Ve vnější smyčce se pouze prování vnitřní smyčka a snižuje se teplota. Její snížení je dáno jedním z parametrů algoritmu. Tato smyčka se opakuje dokud je větší nežli minimální teplota. Ve vnitřní smyčce se vždy vygeneruje nový validní stav. Pokud je tento stav lepší než stav předchozí, je vždy přijat. Pokud je stav horší, může být přijat s určitou pravděpodobností, která je ale ovlivněna aktuální teplotou. Pokud je toto nové řešení lepší než doposud nejlepší nalezené řešení, je zapamatován. Po ukončení obou smyček je vráceno nejlepší nalezené řešení.

Měření kvality algoritmu

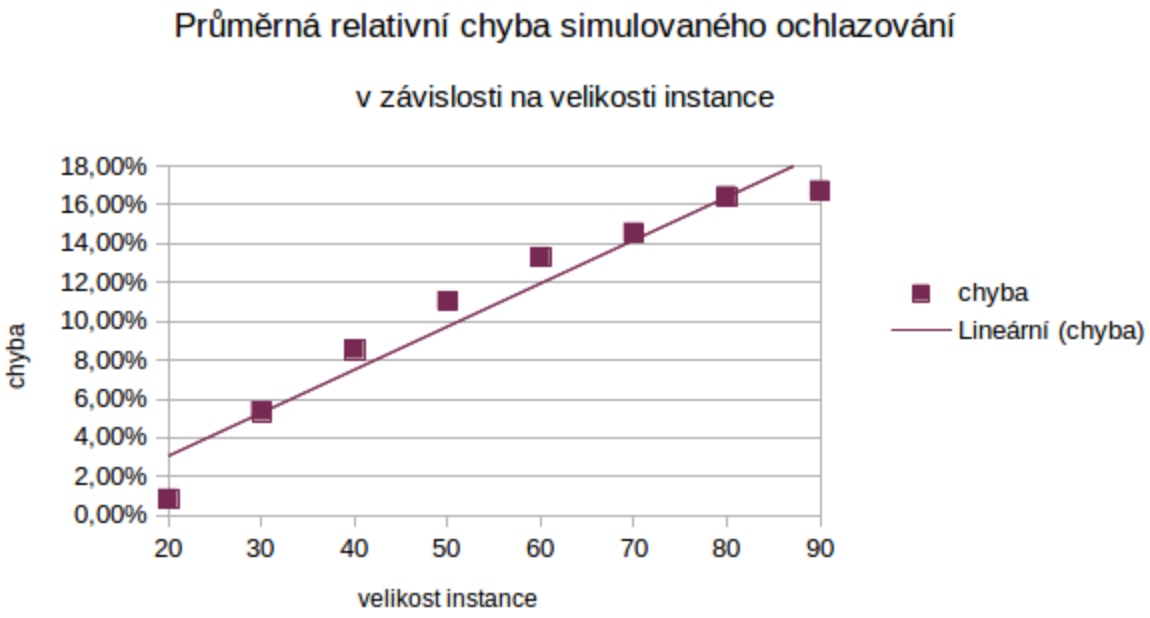
Tuto metodu jsem spouštěl s několika různými způsoby nastavení, ale její výsledky mě příliš nepřesvědčili. Hlavní část měření s podrobnější vyhodnocení výsledků jsem proto dělal u genetického algoritmu. Zde uvedu pouze hodnoty naměřené pro mnou nejlepší nalezenou konfiguraci parametrů. Jedná se o měření nad generovanými daty v rozmezí n od 20 do 90.

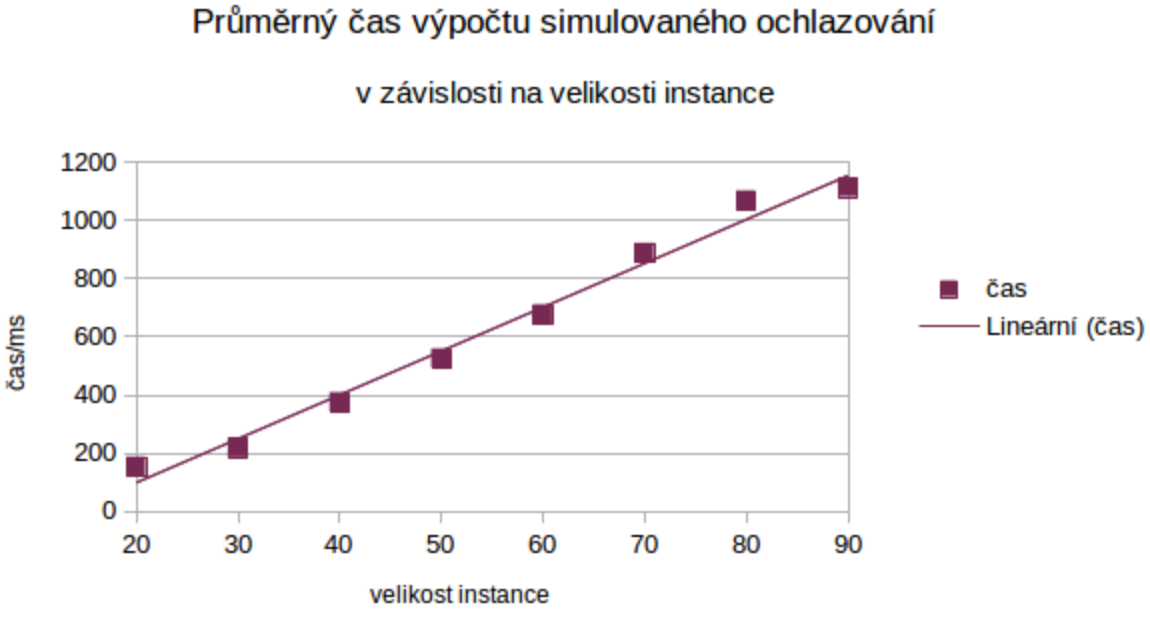
Nastavení parametrů

Parametry pro simulované ochlazování jsem zvolil následující:

- Ochlazovací koeficient: 0.95
- Počet opakování vnitřní smyčky: počet věcí v batohu * 100
- Počáteční teplota: 0,5
- Minimální teplota: 0,001

Tyto parametry jsem ladil pro instance velikosti 40 prvků a je to podle mě vidět i na relativní chybě.


Průměrná relativní chyba simulovaného ochlazování
v závislosti na velikosti instance
18,00%
16,00%
14,00%
12,00%
10,00%
8,00%
6,00%
4,00%
2,00%
0,00%
chyba
20
30
40
50
60
70
80
90
velikost instance
chyba
Lineární (chyba)


Průměrný čas výpočtu simulovaného ochlazování
v závislosti na velikosti instance
1200
1000
800
600
400
200
0
čas/ms
20
30
40
50
60
70
80
90
velikost instance
čas
Lineární (čas)

Genetický algoritmus

Popis metody

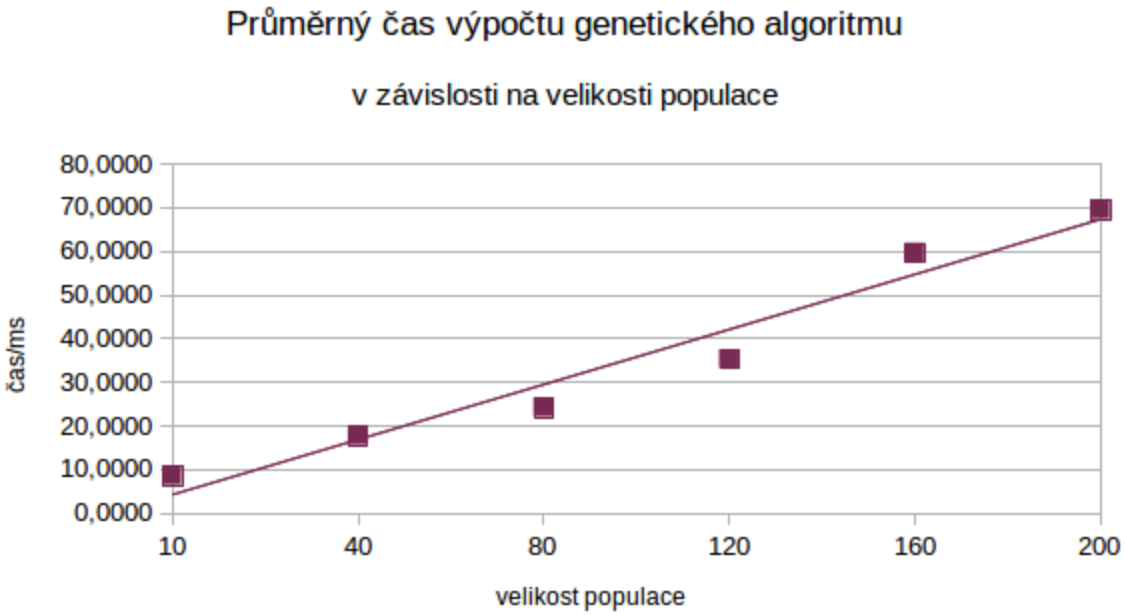
Tento algoritmus pracuje vždy s nějak velkou množinou stavů (populací). V každém kroku se z této populace vyberou ty nejnadějnější stavy a ty se namnoží do nové populace. V takto vzniklé populace probíhá nejprve křížení stavů mezi s sebou a potom s určitou pravděpodobností náhodné mutace. Na konci celého algoritmu se z poslední populace vybere to nejlepší řešení a vrátí se jako řešení celého problému

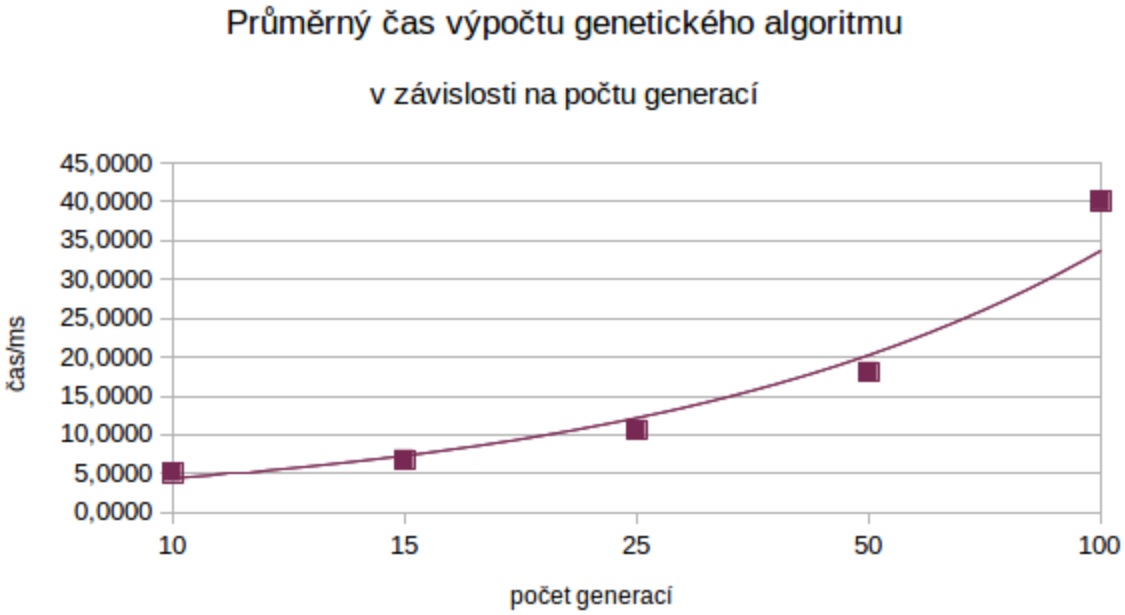
Moje implementace

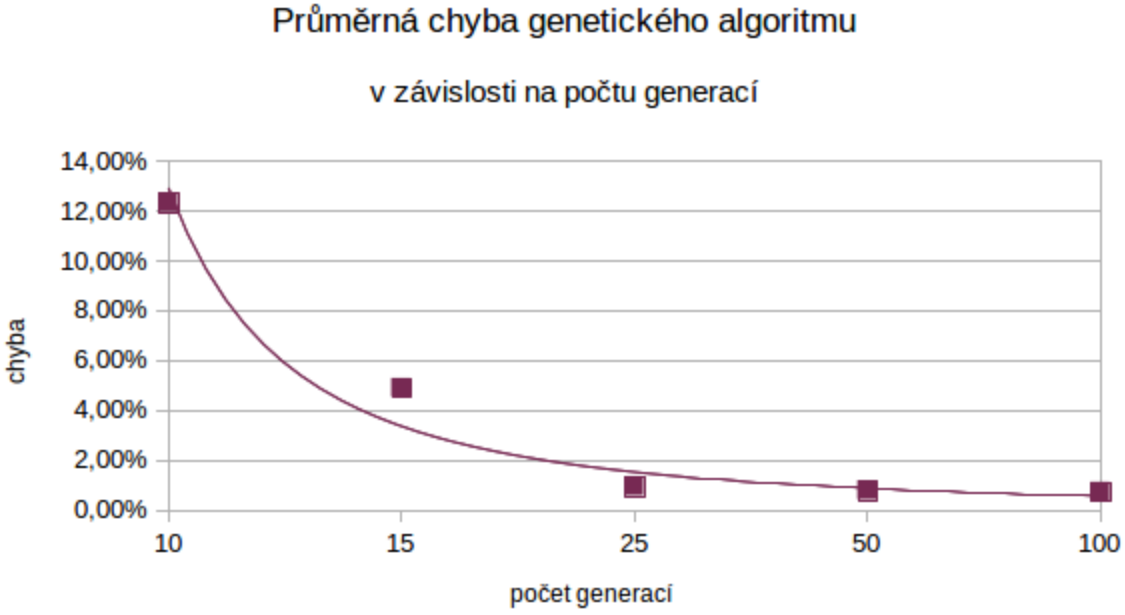
Program nejprve vygeneruje populaci velikosti zadané parametrem. V této populaci jsou náhodně vygenerované stavy, které však nepřekračují maximální hmotnost batohu. Poté se již vstupuje do smyčky, jejíž opakování je určeno parametrem počtu generací.
Nejprve se provádí takzvaná selekce. Pro selekci jsem se rozhodl použít turnajový výběr. Z aktuální generace vyberu vždy několik prvků (dáno parametrem) a u každého z nich vypočtu fitness funkci. Pokud hmotnost řešení nepřekračuje hmotnost batohu, je výsledkem fitness funkce cena řešení, jinak je tato cena mírně penalizována. Z takto vzniklé množiny poté vyberu ten nejlepší prvek a přidám ho do nové populace. To provádím tak dlouho, dokud se nová populace nenaplní.
Dalším krokem je křížení. Počet křížení je určen parametrem a jedná se o jednobodové křížení. Nejprve vyberu náhodně dva prvky z populace a u nich přehodím náhodný počet bitů řešení.
Posledním krokem je mutace. Pro každý prvek v populaci invertuji náhodný bit s pravděpodobností danou parametrem.
Po skončení celé evoluce vyberu řešení s nejlepší cenou které nepřekračuje kapacitu batohu a to vrátím jako správné řešení.

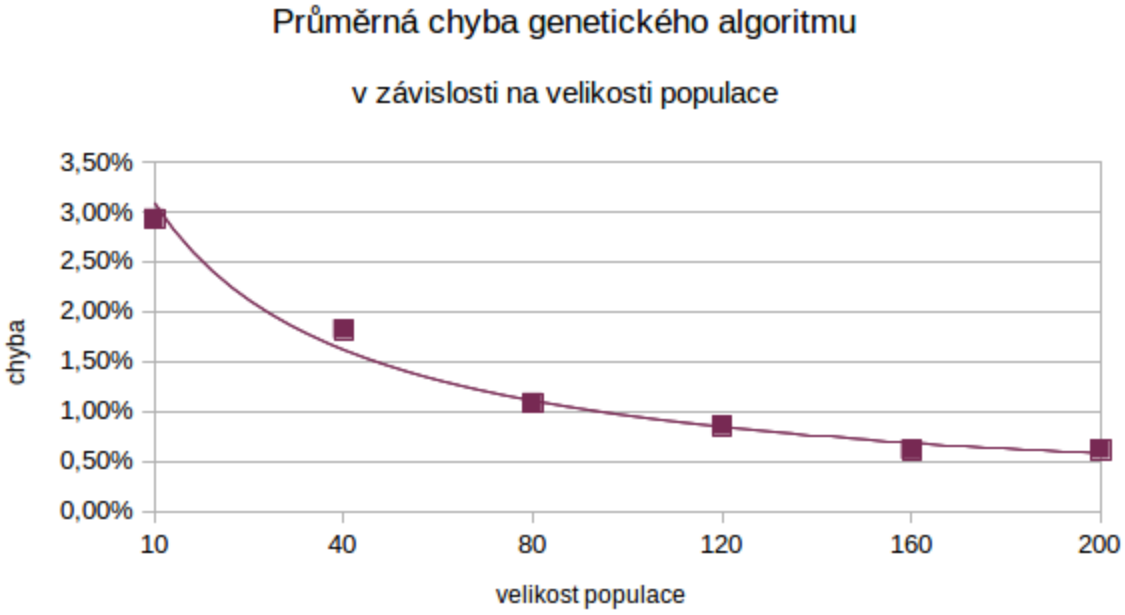
Měření kvality algoritmu

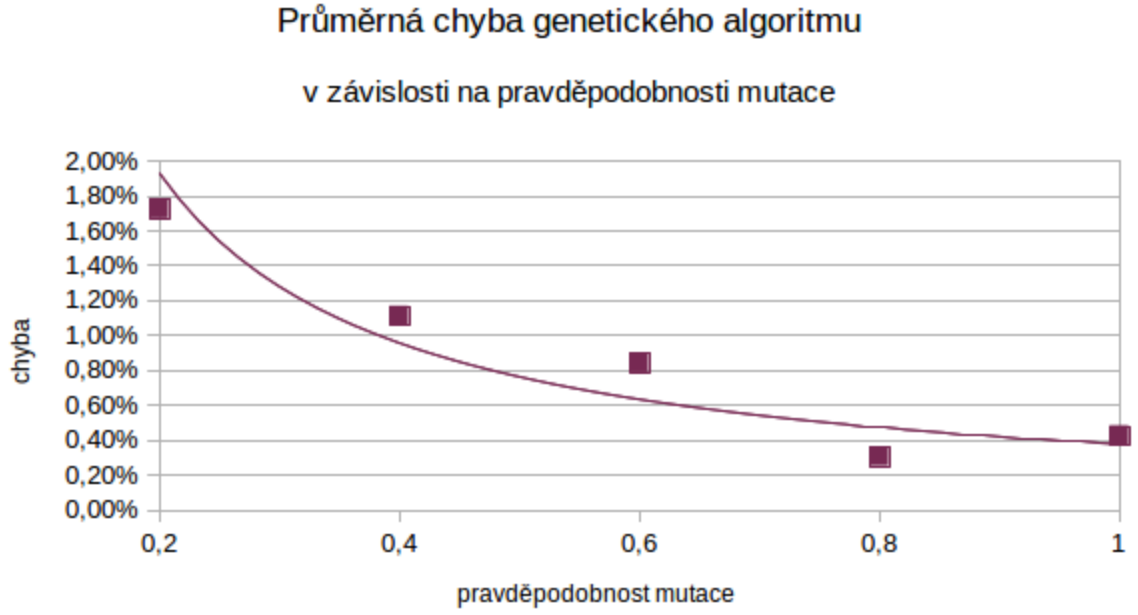
Dle mého měření dává tato metoda mnohem lepší výsledky než simulované ochlazování a i v lepším čase. Měření jsem prováděl na několika velikostech instancí batohu a s různými nastaveními parametrů. U této metody jsem vůbec netušil jak počítat počet prohledávaných stavů a proto jsem měřil čas výpočtu a relativní chybu.

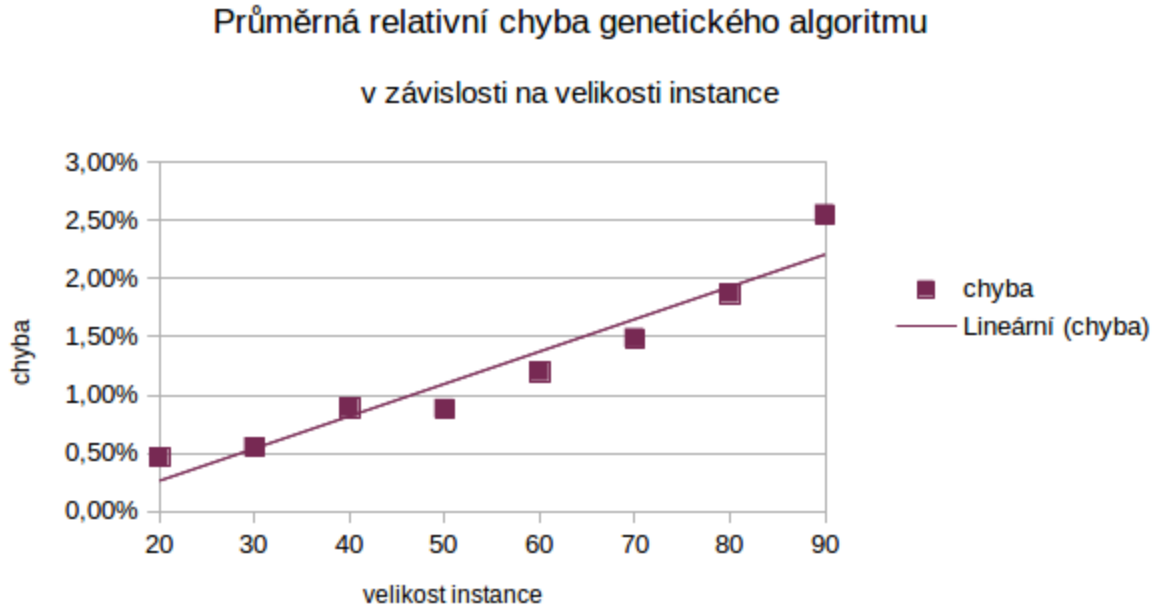

Průměrný čas výpočtu genetického algoritmu
v závislosti na velikosti populace
80,0000
70,0000
60,0000
50,0000
40,0000
30,0000
20,0000
10,0000
0,0000
čas/ms
10
40
80
120
160
200
velikost populace

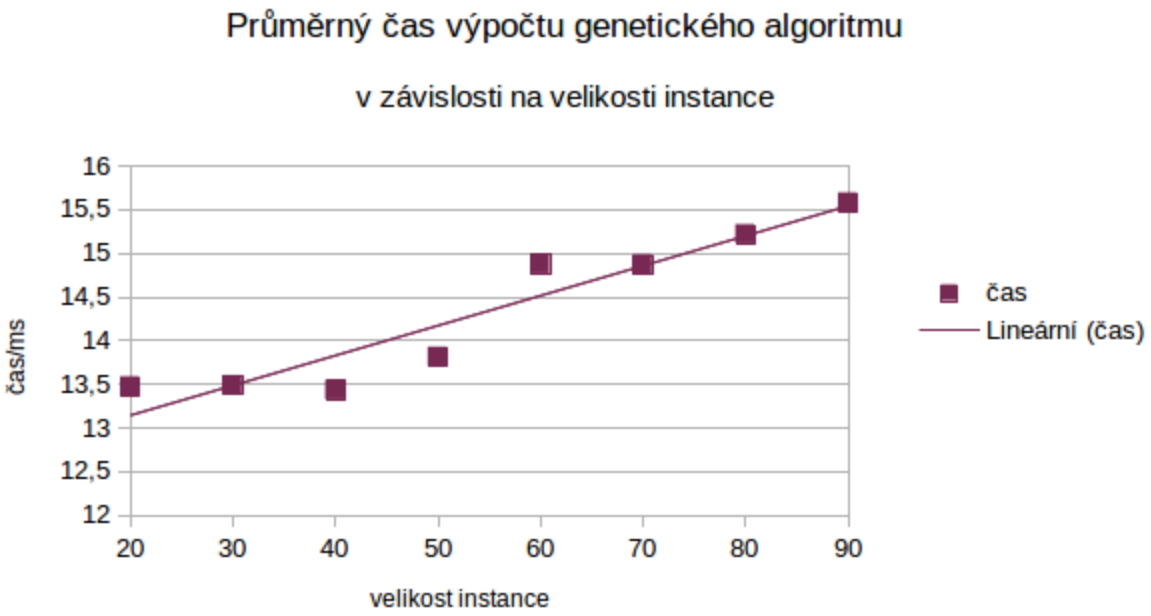

Průměrný čas výpočtu genetického algoritmu
v závislosti na počtu generací
45,0000
40,0000
35,0000
30,0000
25,0000
20,0000
15,0000
10,0000
5,0000
0,0000
čas/ms
10
15
25
50
100
počet generací


Průměrná chyba genetického algoritmu
v závislosti na počtu generací
14,00%
12,00%
10,00%
8,00%
6,00%
4,00%
2,00%
0,00%
chyba
10
15
25
50
100
počet generací


Průměrná chyba genetického algoritmu
v závislosti na velikosti populace
3,50%
3,00%
2,50%
2,00%
1,50%
1,00%
0,50%
0,00%
chyba
10
40
80
120
160
200
velikost populace


Průměrná chyba genetického algoritmu
v závislosti na pravděpodobnosti mutace
2,00%
1,80%
1,60%
1,40%
1,20%
1,00%
0,80%
0,60%
0,40%
0,20%
0,00%
chyba
0,2
0,4
0,6
0,8
1
pravděpodobnost mutace


Průměrná relativní chyba genetického algoritmu
v závislosti na velikosti instance
3,00%
2,50%
2,00%
1,50%
1,00%
0,50%
0,00%
chyba
20
30
40
50
60
70
80
90
velikost instance
chyba
Lineární (chyba)



Naměřené výsledky

Na času je vidět, že roste jak s rostoucí velikostí populace, tak i s rostoucím počtem generací. To je celkem očekávatelné. Chyba Klesá se zvětšujícím se počtem generací, od hodnoty přibližně 25 se ale její pokles velice zmírní. To stejné platí i o chybě při zvětšující se velikosti generace. Zde nastává zmírnění při velikosti cca 150. Naopak u pravděpodobnosti mutace chyba i roste. Nejlepší výsledky jsem naměřil u pravděpodobnosti 80%. Velikost instance hraje také důležitou roli. Podle mého měření je však mnohem méně významná než u simulovaného ochlazování. S rostoucí velikostí instancí roste chyba (pravděpodobně to je tím, že byly parametry odladěny na instance velikosti 40) a roste i čas (což je opět logické, jednak n zde hraje přímo roli a jednak se velikost instance objevuje i v parametru počtu opakování vnitřní smyčky).

Experiment

Zkoušel jsem při prvotním generování populace vložit i nějaké prvky, které vznikly pomocí jednoduché heuristické metody. Zkoumal jsem výsledek v kolika případech byl genetický algoritmus lepší než výsledek heuristiky a kolikrát horší. V naprosté většině případů se oba výsledky shodovaly (cca v polovině těchto případů byla chyba 0%). Cca ve 25 % byl výsledek genetiky lepší a v 10% byl horší. To znamená, že i genetika může zahodit lepší výsledek. Je možné, že se tento dobrý výsledek nedostal do výběru, nebo byl špatně zmutován nebo překřížen.
Možná by bylo lepší nečíst výsledek metody až po skončení celé smyčky, ale pamatovat si nejlepší vygenerovaný výsledek vůbec. Poté by rozhodně nemohlo dojít ke zhoršení.

Závěr

Metoda simulovaného ochlazování mě velice zklamala. Buďto jsem nedokázal dobře nastavit parametry (ale zkoušel jsem opravdu velice mnoho možností), nebo jsem měl algoritmus nějak špatně naprogramovaný, ale výsledky opravdu nebyly moc dobré a heuristika často dávala mnohem lepší řešení. Naopak genetický algoritmus mě překvapil. Jednak svojí malou chybou, relativní rychlostí a velkou stabilitou (i vůči špatně nastaveným parametrům). Pro poslední úlohu rozhodně zvolím genetický algoritmus.

Konfigurace počítače

Procesor Intel CORE i7, frekvence 1,8 GHz, Operační paměť 8Gb. Linux Ubuntu 64 bitů. Naprogramováno v jazyce Java.