Řešení problému vážené splnitelnosti booleovské formule pokročilou iterativní metodou

Specifikace úlohy

Je dána booleovská formule F proměnnných $X=(x_1, x_2, \dots, x_n)$ v konjunktivní normální formě (tj. součin součtů). Dále jsou dány celočíselné kladné váhy $W=(w_1,w_2, \dots, w_n)$. Najděte ohodnocení $Y=(y_1, y_2, \dots, y_n)$ proměnných $x_1, x_2, \dots, x_n$ tak, aby $F(Y)=1$ (formule byla splněna) a součet vah proměnných, které jsou ohodnoceny jedničkou, byl maximální.

Varianty řešení

Pro řešení problému jsem si vybral jednu z pokročilých lokálních heuristik (simulované ochlazování, genetické algoritmy, tabu prohledávání). Moje volba je genetický algoritmus. Dále jsem také naprogramoval řešení hrubou silou, abych u menších instancí mohl srovnávat spolehlivost genetického algoritmu.

Popis řešení

Při psaní genetického algoritmu jsem vycházel ze základní kostry podle přednášek a doplnil jsem ji o specifikaci jednotlivých kroků.

1. Vygeneruj náhodnou populaci.
2. počet generací krát opakuj:
    I. Selekce - výběr nejnadějnějších prvků
        a. elitismus - vyber n nejlepších prvků z předešlé populace
        b. turnajový výběr - doplň populaci pomocí turnaje n prvků
    II. Křížení
        a. dvoubodové křížení
        b. jednobodové křížení
        c. improvement - vylepšení několika prvků
    III. Mutace - náhodná mutace s určitou pravděpodobností
    IV. Výběr nejlepšího prvku - je nějaký prvek z aktuální populace lepší než dosavadní nejlepší prvek?
3. Vrať nejlepší prvek

Generování náhodné populace probíhá primitivně. S pravděpodobností 50% se každé proměnné přiřadí true nebo false. Poté se již jde do hlavní smyčky, která probíhá pro každou generaci. Nejprve probíhá selekce.

<u>Selekce</u>

Vytvoří se prázdná populace a nejprve se do ní vloží globálně nejlepší řešení. Poté se uplatňuje takzvaný elitismus. Do nové generace se vloží n nejlepších prvků z generace

předchozí. Poté se do populace vloží jeden náhodně vygenerovaný prvek (aby populace tak rychle nezdegenerovala) a nakonec se populace doplní pomocí turnajového výběru. Velikost turnaje je určena parametrem.

Fitness funkce

Fitness funkce určuje který prvek je lepší. Základním měřítkem je váha (čím větší, tím lepší). K této váze se však ještě připočítává bonus za splnění klauzule (za každou splněnou klauzuli je přičtena hodnota daná parametrem) a bonus za splnění celé formule. Kdyby se tyto bonusy nepřičítaly, výsledkem by byla sice největší váha, ale formule ty téměř jistě nebyla splněna.

Křížení

Při křížení ze dva náhodné prvky překříží do dvou nových. Křížení je několik druhů. Ve svém algoritmu jsem použil tři z nich. Prvním je jednobodové křížení. Zde se přehodí náhodný počet proměnných od začátku. Druhým druhem je dvoubodové křížení, zde se prohazují proměnné uprostřed. Počet těchto křížení je dán parametrem a o tom, které křížení se provede rozhoduje náhoda (s větší pravděpodobností pro jednobodové křížení). Posledním druhem křížení je Improvement.

Improvement

Tato metoda je něco mezi křížením a mutací. Vybere se několik náhodných prvků a u každého se postupně pokusí invertovat každá proměnná. Pokud je výsledek lepší, proměnná se ponechá invertovaná, jinak se vrátí na původní hodnotu.

Mutace

Mutace je pouhé invertování libovolné proměnné pro každý prvek s pravděpodobností danou parametrem.

Naměřené výsledky

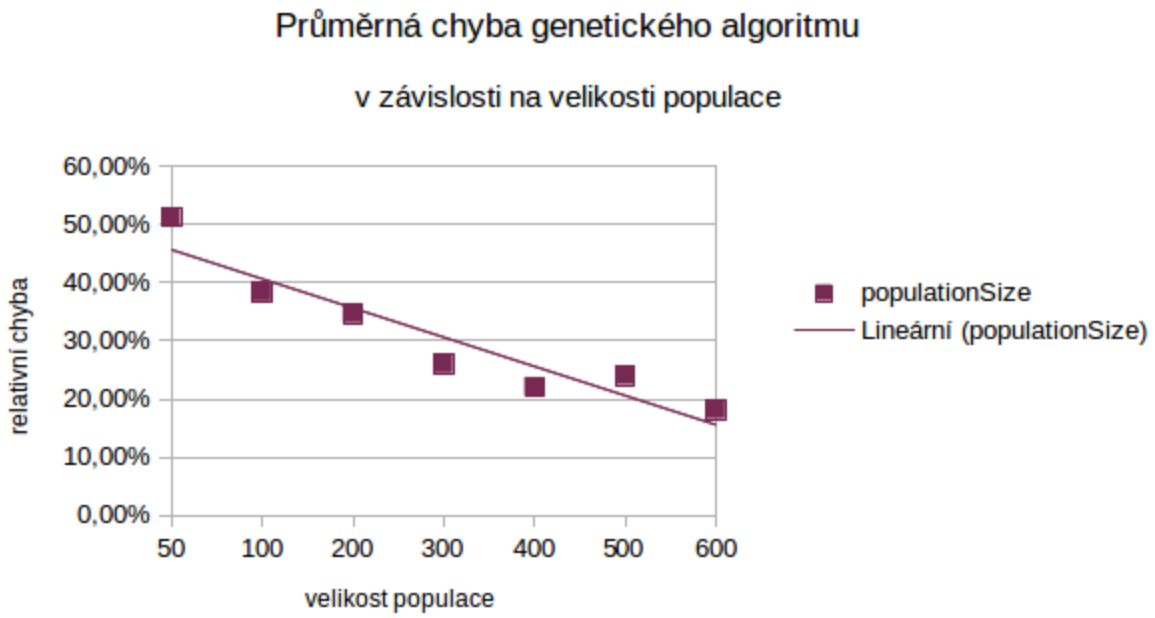
Po několika pokusech s parametry jsem určil sadu parametrů, pro která algoritmus nevracel špatné výsledky a na této sadě jsem začal testovat jejich význam a hlavně optimální nastavení. Vždy jsem tedy měnil pouze jeden parametr a sledoval, jak se vyvíjí chyba ve srovnání s výsledkem vypočteným hrubou silou.

Základní parametry pro všechna měření jsou:

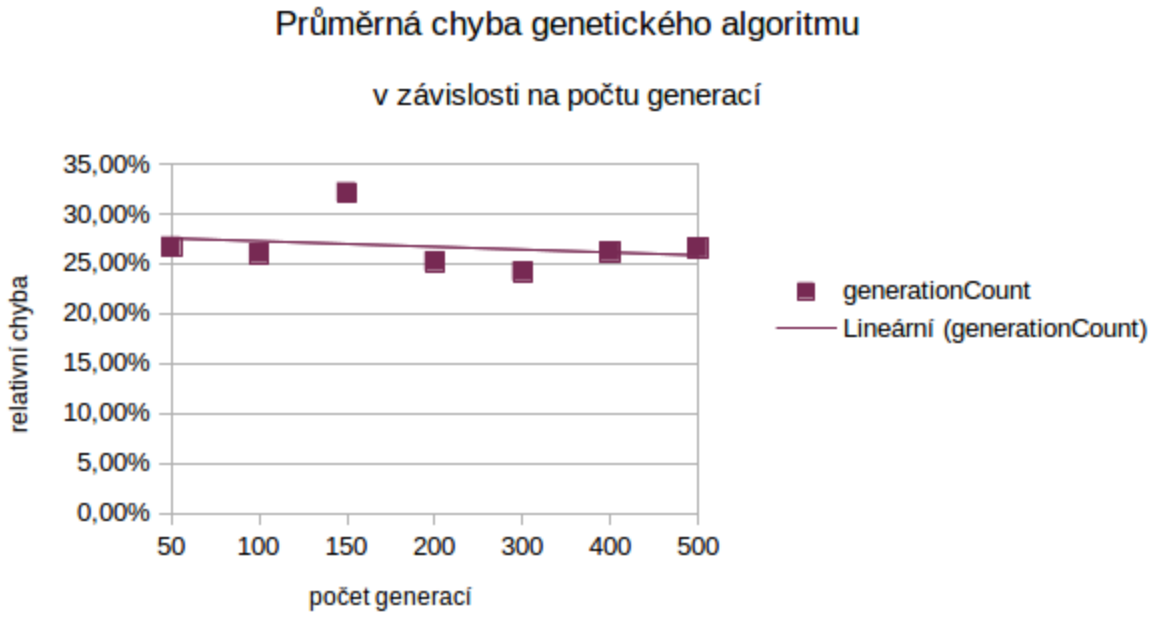
```
int populationSize = 300;
int generationsCount = 100;
int crossbreedingCount = populationSize / 10;
float mutationProbability = 0.6f;
int selectionSize = 3;
int bonusClause = sumOfWeight / 10;
int bonusFormula = sumOfWeight;
int elitism = populationSize / 60;
int improvementCount = populationSize / 50;
```

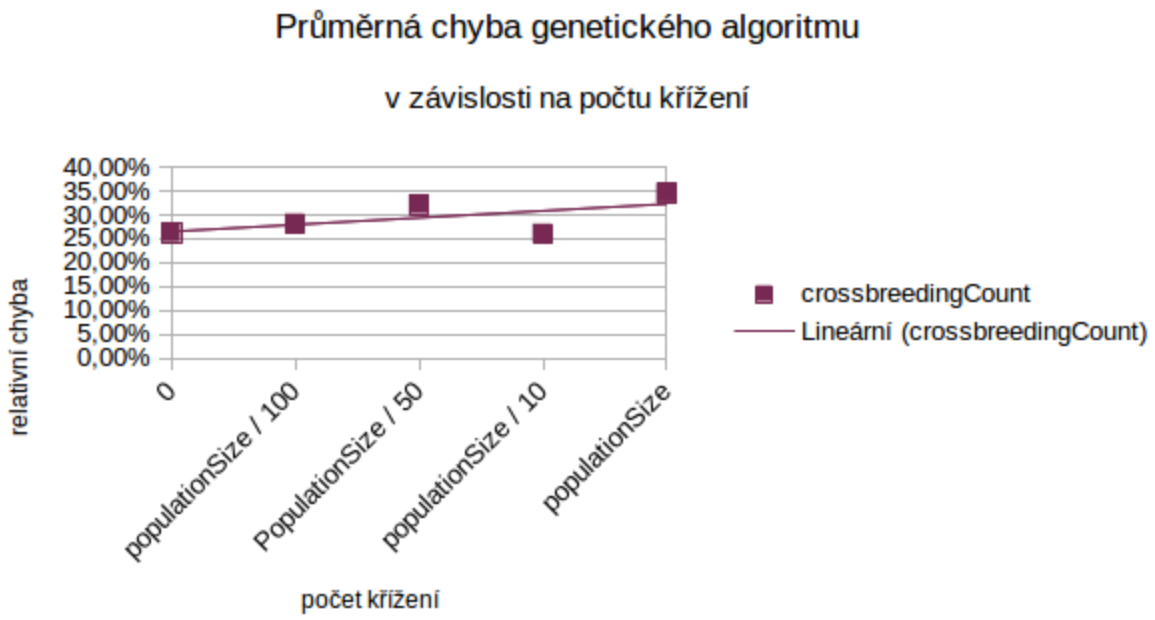
Některé parametry jsem vztáhl k sumární váze a některé jsou závislé na jiných parametrech. Čas výpočtu jsem neměřil, je však zcela jasné, že parametry jako velikost populace a hlavně počet generací hrají s dobou výpočtu hlavní roli.

Instance pro základní měření jsem použil s 20ti proměnnými a 91 klauzulemi a se 3mi proměnnými v klauzuli (3SAT). Váhy byly dogenerovány náhodně od 0 do 100. Pro měření jsem používal 50 instancí, jejichž výsledky jsem průměroval.

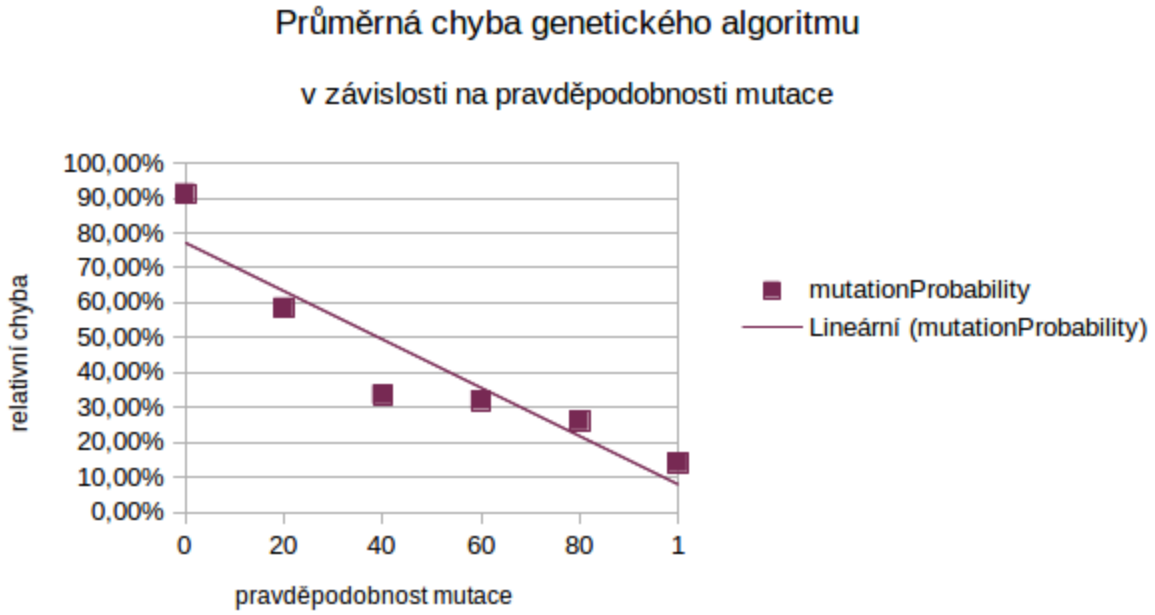


Jak je vidět, s rostoucí velikostí populace chyba klesá, ale také roste čas výpočtu. Naopak počet generací nehraje tolik významnou roli.

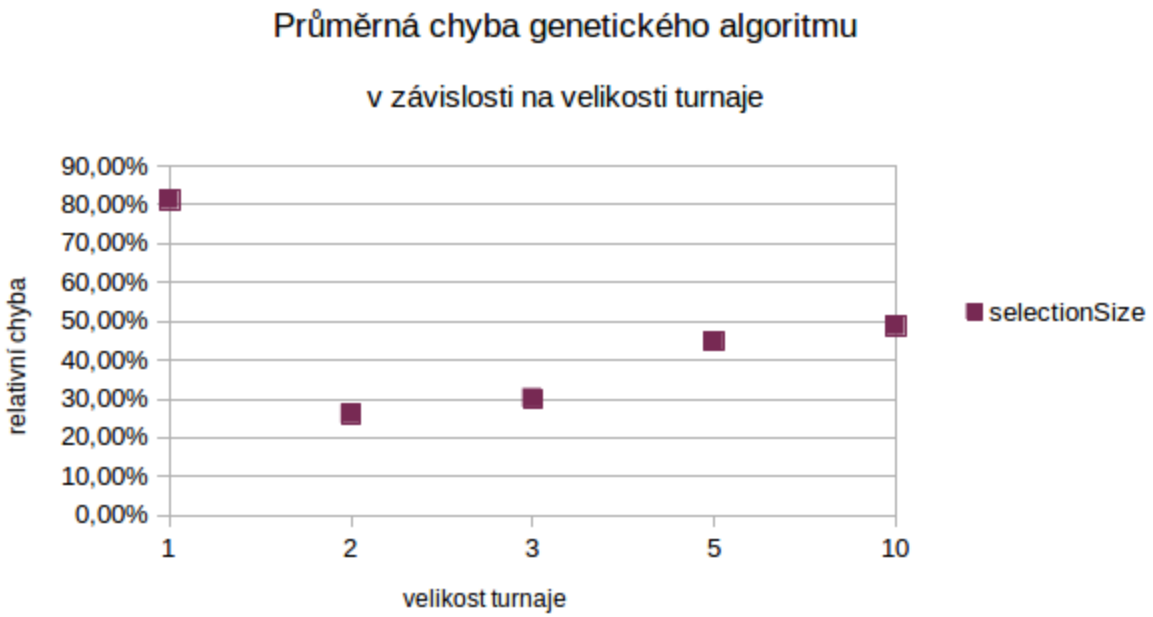




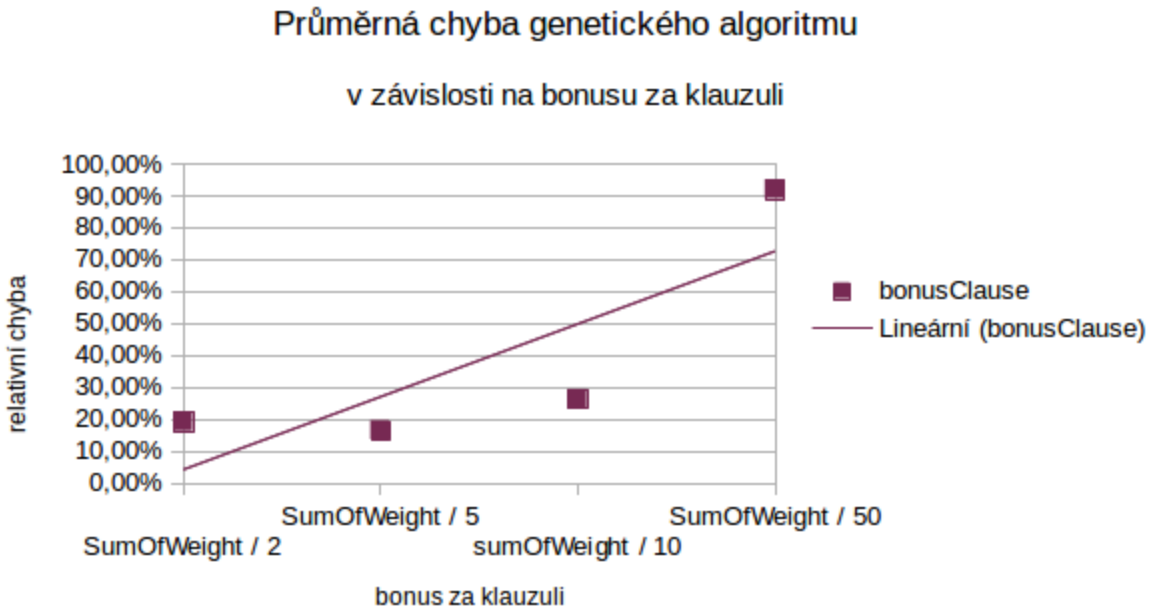
Počet křížení také není příliš výrazným parametrem. Zajímavější je pohled na druh křížení. Při jednobodovém křížení jsem opakovaně dosahoval lepších výsledků než při dvoubodovém křížení. Proto jsem ve výsledném algoritmu jejich poměr upravil ve prospěch jednobodové varianty. Při tomto křížení se chyba pohybovala kolem 26% a při dvoubodovém kolem 34%. Jejich kombinace v poměru 50:50 dávala výsledky asi 29%.



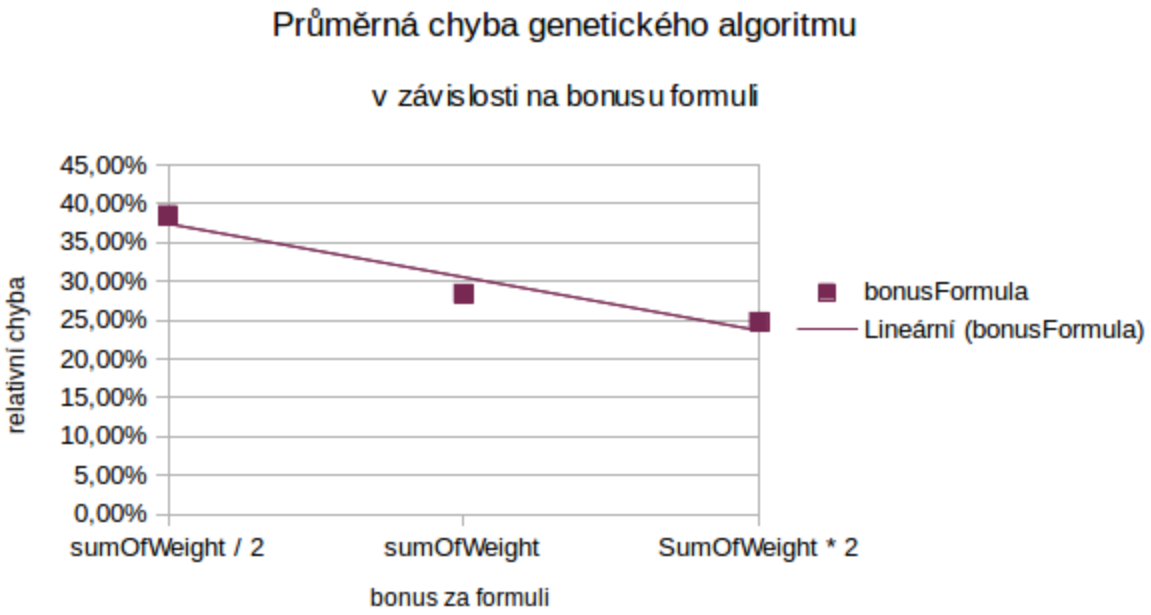
Graf pravděpodobnosti mutace je pro mě velice zarážející, ale s rostoucí pravděpodobností chyba klesá a to jsem tato měření prováděl opakovaně. Dávám to za vinu typu algoritmu, ve kterém je nejobtížnější nalézt splnitelnou formuli a mutace je jeden způsob, jak změnit jednotlivé proměnné. Zkoušel jsem tento parametr měnit i v kombinaci s jiným nastavením ostatních parametrů, ale moje měření se vždy potvrdilo.



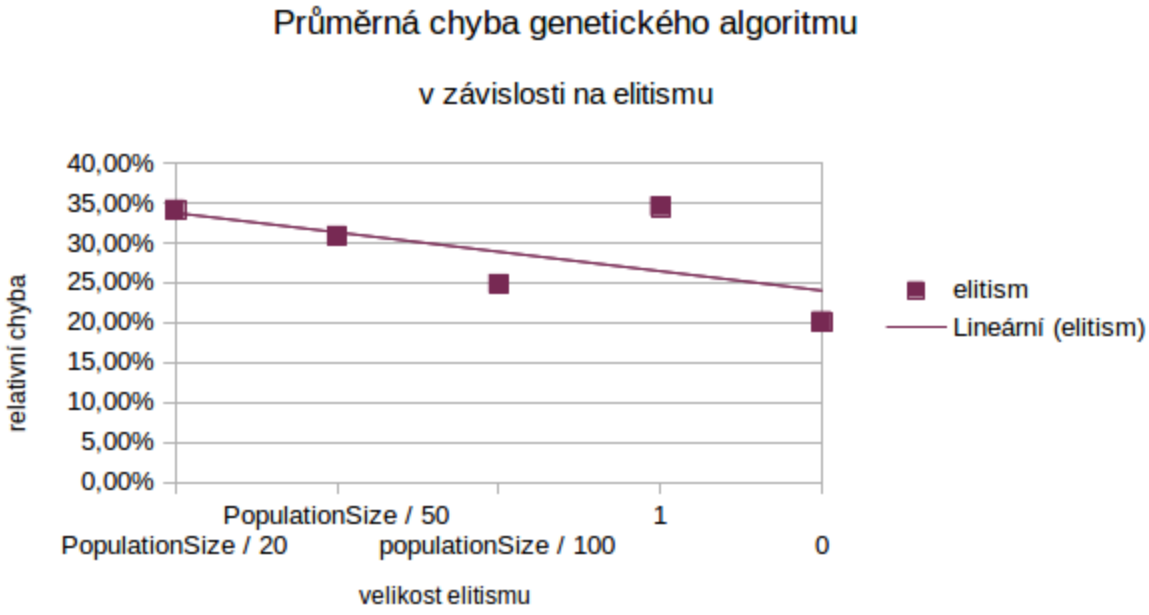
Pro velikost turnaje se osvědčilo brát pouze málo čísel. Ideálně 2-3 prvky.



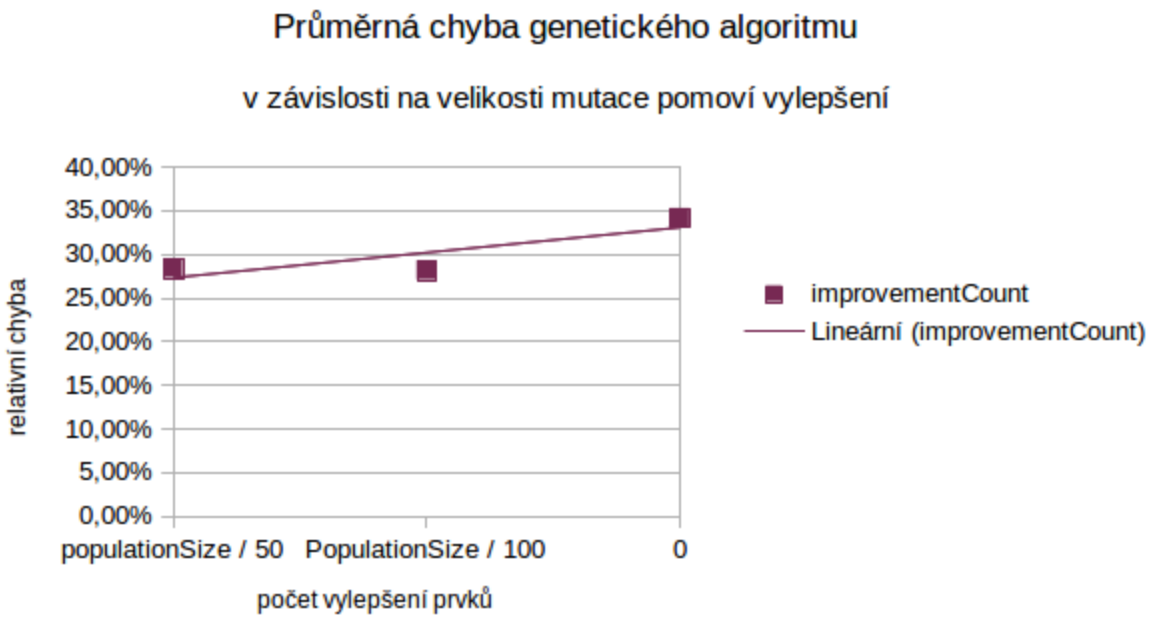
Jak se potvrdilo, tento parametr je velice důležitý, a pokud je špatně nastaven, roste chyba až ke 100%. Jako dobré se ukázalo použít pětinu sumy všech vah.



Bonus za splnění celé formule je také důležitý a čím je větší, tím lépe. To je způsobeno tím, že hlavním problémem je nalézt splnitelnou formuli a pokud se to již povede, nesmíme ji ztratit špatnou selekcí, ale musí zůstat v populaci a být postupně vylepšována.



Elitismus se ukázal jako ne příliš přínosný. Čím méně nejlepších prvků se zkopíruje, tím lépe. Je to divné, ale tento parametr pravděpodobně způsobuje rychlou degeneraci celé populace.



Posledním parametrem je Improvement (princip je popsán výše). Tato technika opravdu výsledku pomáhá a je v rozumné míře přínosná.

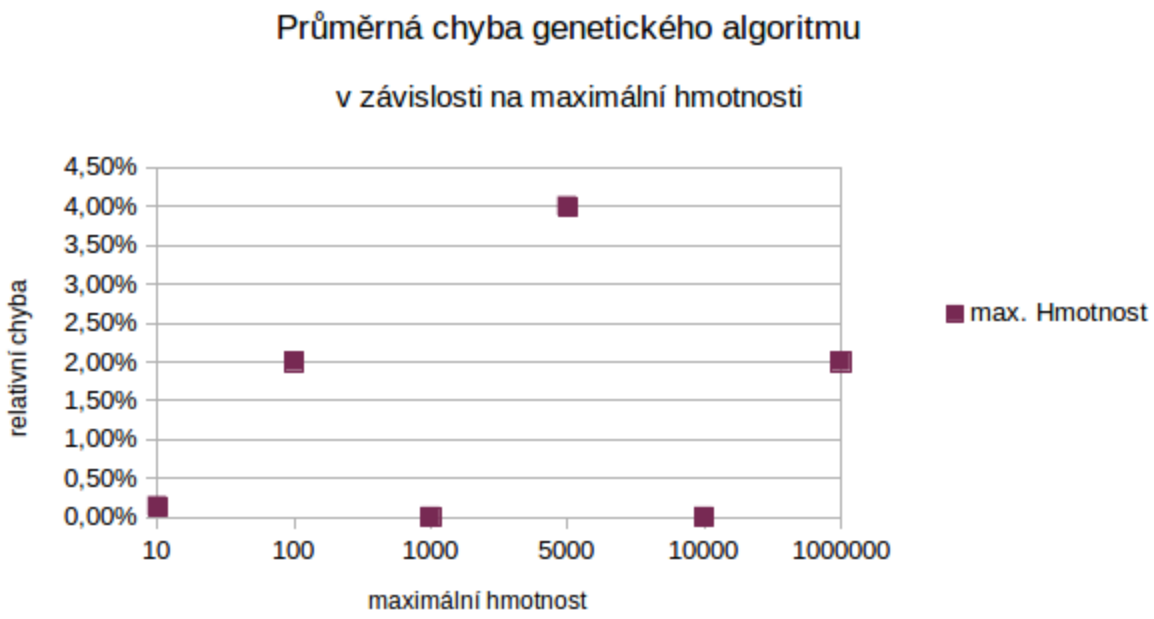
Po provedeném měření jsem ve všech kategoriích nastavil parametry na nejlepší možné hodnoty a výsledek předčil očekávání. Zatímco průměrná chyba s předcházející konfigurací se pohybovala kolem 25%-35%, s následující konfigurací jsem opakovaně dosáhl průměrné chyby pouhá 2%. Ze všech 50ti výsledků se nepodařilo formuli splnit pouze 1 a jednou se lišila výsledná váha.

Nejlepší konfigurace:

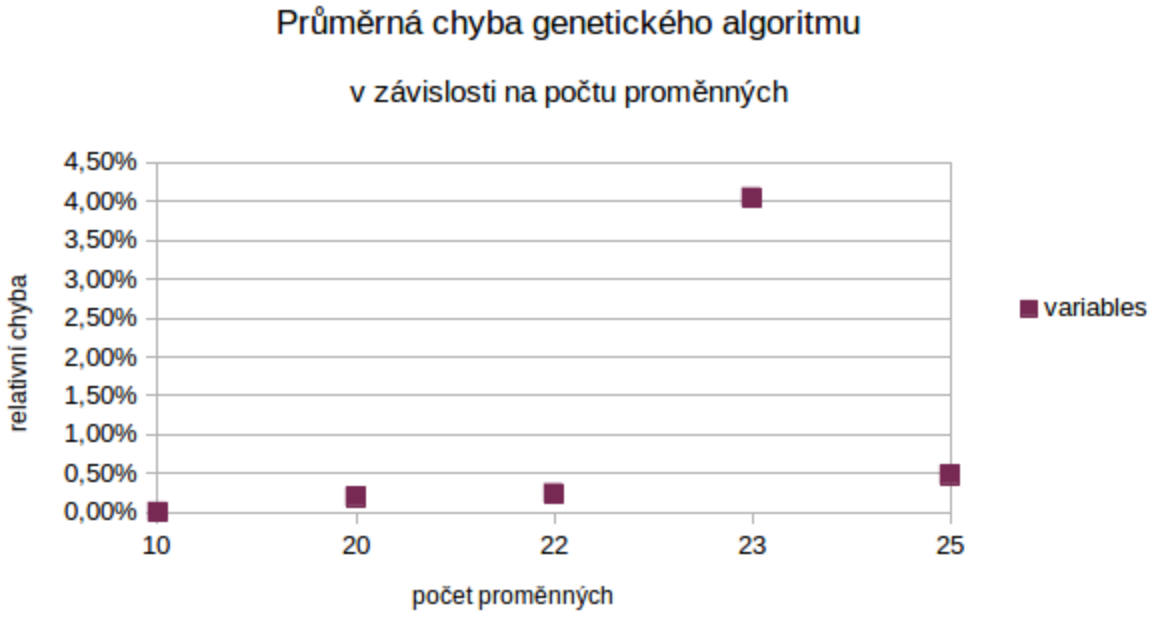
```
int populationSize = 400;
int generationsCount = 100;
int crossbreedingCount = populationSize / 10;
float mutationProbability = 0.8f;
int selectionSize = 2;
int bonusClause = sumOfWeight / 5;
int bonusFormula = sumOfWeight * 2;
int elitism = 0;
int improvementCount = populationSize / 50;
```

Algoritmus s touto konfigurací jsem spustil na 500 instancích stejných parametrů jako byly popsány v úvodu a výsledná chyba byla pouze 1,7%.

Dále jsem používal tyto vylepšené parametry a pokusil jsem se změřit úspěšnost programu v závislosti na různých datech. Nejprve jsem měřil vliv maximální (a průměrné) hmotnosti jednotlivých proměnných. Zde by se mohla projevit špatná definice fitness funkce nebo bonusových bodů za splnění klauzule.

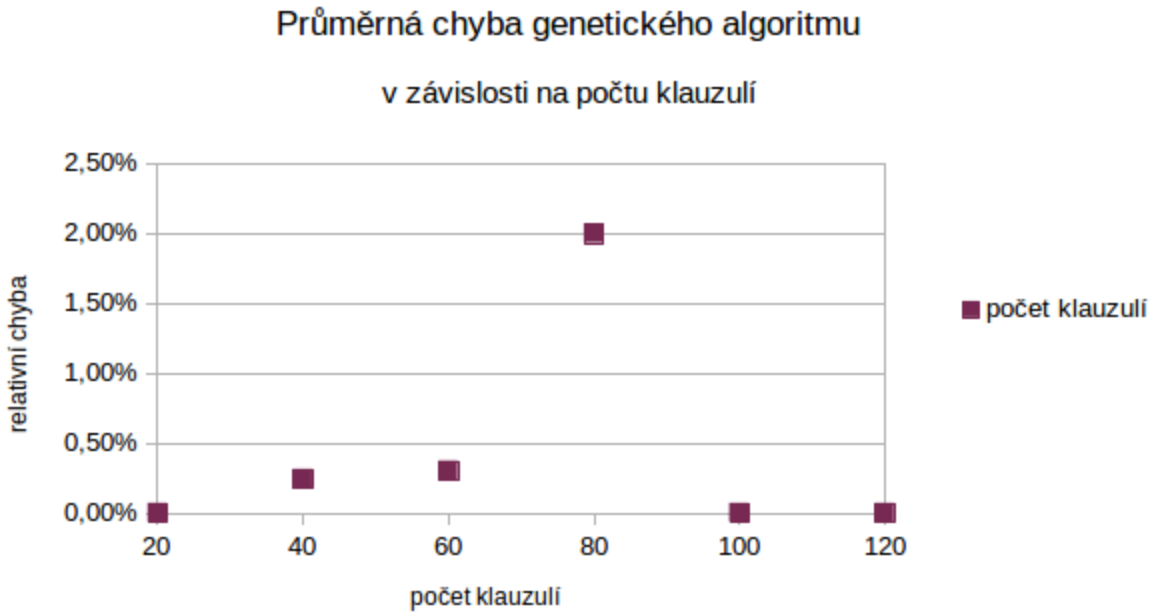


To se naštěstí nepotvrdilo a díky vazbě na sumární hmotnost se výsledky téměř nelišily ani při řádové změně cen. (Chyba 2% znamená jednu nevypočtenou hodnotu z 50ti)



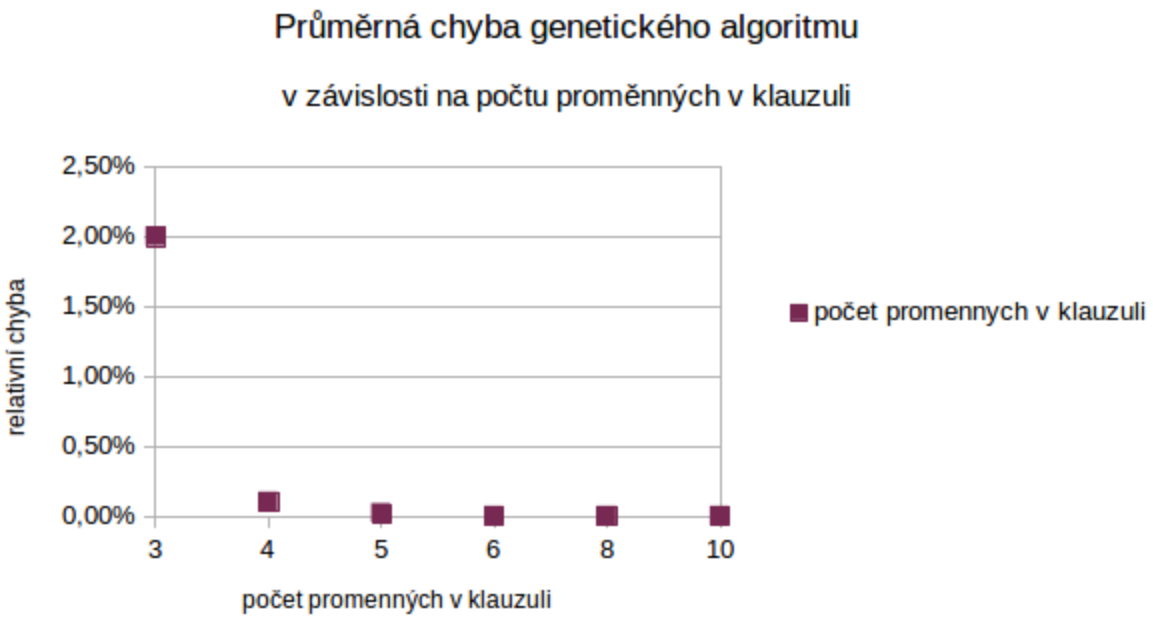
Dále jsem se pokusil změřit vliv chyby na počet proměnných. Ověřit výsledek v rozumném čase jsem byl schopen pouze do cca 25 proměnných. Jak je vidět na předcházejícím grafu, chyba mírně roste, ale nejedná se o nějaký problém, protože rozdíl je v řádu desetin procent. Pokusil jsem se ještě zvýšit počet generací, to však výsledek nijak pozitivně neovlivnilo (chyba cca 2,5%) a naopak čas výrazně vzrostl. Také jsem zvýšil velikost populace (ve vztahu k počtu proměnných) ale výsledek byl ještě horší (chyba cca 4%). Ponechal jsem tedy původní hodnoty.

Další hodnotou je počet klauzulí, zde jsem opět byl schopen ověřit pouze cca 120 klauzulí. Následující graf ukazuje naměřené výsledky.



Jak je vidět, chyba nemá žádnou rostoucí ani klesající tendenci a je zdá se být náhodná.

Poslední změnou je změna počtu proměnných v jedné klauzuli. Můj algoritmus není omezen pouze na 3SAT (i když jsem zatím pracoval pouze s formulemi, které mají 3 proměnné v klauzuli), ale zvládá obecný problém SAT. Na následujícím grafu je vidět opět vývoj chyby v závislosti na počtu proměnných. Chyba klesá a poté dosahuje 0%.



Posledním krokem měření jsem se pustil do velkých instancí, které není možné ověřit pomocí hrubé síly. Vygeneroval jsem si několik sad po 50ti instancích a všechny splnitelné. Prohnal jsem je genetickým algoritmem a měřil jsem. kolik instancí vrátí hmotnost 0 - tedy že se nepovedlo splnit formuli.
Počítal jsem s instancemi o 50ti proměnných a 100 klauzulích (3 proměnné na klauzuli) a algoritmus opakovaně vypočetl všechny instance. Jestli však byla hmotnost maximální říci nedokážu.

Ještě je třeba podotknout, že chyba je v naprosté většině případů způsobena nesplněním celé formule (výsledek u dané instance je 100%). Pokud se však algoritmus nalezl splnitelnou formuli, téměř vždy byla váha zcela správně (chyba 0%). Pokud tedy algoritmus vrátí výslednou váhu 0 (nebyla splněna formule), vyplatí se ho spustit znova a pokud již vrátí nějakou nenulovou váhu, bude výsledek s vysokou pravděpodobností dobře.

Závěr

Z minulé úlohy (problém batohu) jsem se poučil a pozměnil trochu průběh algoritmu, jednak jsem přidal nějaké další metody (elitismus, improvement, dvoubodové křížení...), ale hlavně nevybírám nejlepší řešení pouze z poslední populace, ale aktualizuji ho průběžně z každé populace. Proto se mi nemůže stát, že bych nalezl nějaké řešení, které bych později pokazil a výsledek by tudíž nebyl tak dobrý.
Celkový výsledek s nastavenými parametry podle měření hodnotím jako velice dobrý a dobře použitelný. Chyba je opravdu velice malá a naprostá většina instancí je vyřešena úplně přesně.

Dalo by se tento algoritmus ještě hodně vylepšovat a měřit spoustu dalších vlivů a nastavení, ale ze mnou testovaných hodnot se ukazuje tato verze jako vhodná pro širokou škálu instancí a v rozumném čase. S rostoucí velikostí populace velmi roste čas a výsledek není o mnoho lepší. Dle mého názoru je tedy aktuální nastavení nejlepším kompromisem přesnosti a doby výpočtu.

Konfigurace počítače

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