# 5. Řešení problému vážené splnitelnosti booleovské formule pokročilou iterativní metodou

## Zadání úlohy:

* Je dána booleovská formule F proměnnných X=(x1, x2, … , xn) v KNF. Dále jsou dány celočíselné kladné váhy W=(w1, w2, … , wn). Najděte ohodnocení Y=(y1, y2, … , yn) proměnných x1, x2, … , xn tak, aby F(Y)=1 a součet vah proměnných, které jsou ohodnoceny jedničkou, byl maximální.
* Je přípustné se omezit na problém 3 SAT.

## Popis zvoleného algoritmu:

Jako pokročilou heuristiku řešící problém 3SAT jsem si zvolil genetický algoritmus, se kterým jsem pracoval na 4. úloze. Algoritmus je iterativní a funguje na principu postupné tvorby generací, které se skládají z jedinců. Každý jedinec v generaci reprezentuje nějaké řešení. Nová generace je z předchozí vytvořena pomocí principů odvozených z evolučních procesů, které se vyskytují v přírodě. Jsou jimi přirozený výběr, křížení, mutace a dědičnost.

* Selekce – z předchozí generace jsou přednostně vybráni jedinci, kteří mají vysokou hodnotu zdatnosti (fitness).
* Křížení – vzájemným křížením vybraných jedinců je vygenerován nový.
* Dědičnost – vlastnosti jedinců jsou mezigeneračně přenášeny, tzn. generace se od sebe liší pouze do určité míry.
* Mutace – jedinec je mezigeneračně zkopírován, ale část jeho informace je náhodně pozměněna.

Po každém vytvoření nového jedince je nutné spočítat hodnotící kritérium (zdatnost, fitness). Genetický algoritmus je randomizovaný, kdy zdatnost určuje pravděpodobnost přenosu genetické informace a vlastností jedince na potomky.

## Rozbor problému:

Cílem semestrální práce je vytvořit program, řešící problém 3 SAT pomocí pokročilé iterativní heuristiky. V důsledku velkého rozsahu stavového prostoru neexistuje bruteforce řešení v „rozumném“ čase na současném HW.

Omezující podmínkou je, aby byl výraz po ohodnocení konfiguračními proměnnými splněn.

Předpokládáme, že řešení jsou ve stavovém prostoru rozprostřena nahodile a že nedochází k shlukování. Zároveň platí, že počet řešení, která splňují omezující podmínky, je poměrně nízký.

Z velikosti problému a nízkého počtu konfigurací, které splňují omezení lze usoudit, že křížení dvou výhodných jedinců nemusí vůbec vést na řešení. Proto budeme v algoritmu ponechávat i konfigurace, které nesplňují omezující podmínky a budeme se soustředit zejména na faktor mutace a selekce.

## Experiment:

Protože je složitost problém odvozena od poměru klauzulí a počtu proměnných, bude nutné stanovit jejich „rozumný“ počet a s ním dále počítat. Většina dostupných instancí problému neobsahuje váhy pro jednotlivé proměnné. V průběhu experimentu budeme proto váhy generovat pomocí uniformního rozdělení v intervalu <1, 100>.

V průběhu experimentu se budeme snažit vyhodnotit vlastnosti algoritmu nezávisle na vstupních instancích. Zejména schopnost reagovat na změny parametrů a možnou konvergenci k řešení.

Protože jsou instance náhodně generovány neexistuje snadný způsob, jak srovnat relativní kvality řešení. V našem experimentu budeme hodnotit kvalitu (fitness) pomocí sumy vah jednotlivých proměnných.

Pro každou kombinaci parametrů algoritmu budeme testovat vstupní balík 500 instancí, kde každá má 250 klauzulí a 1056 proměnných. Instance jsou staženy z doporučovaného zdroje DIMACS. Pro zjednodušení problému jsem po počátečním otestování omezil počet klauzulí omezil na 100 a proměnné na ty, co jsou v nich obsaženy.

## Implementace:

Implementaci algoritmu jsem provedl v jazyce C++ pomocí objektového programování. Kódování jedince je jednoduhé binární pole o délce rovnající se počtu proměnných. Logická hodnota udává přímo nastavení příslušné proměnné. Na vstupu jsou proměnné navíc namapovány na čísla <1, n> pro snadnější implementaci.

Proces vytvoření nové generace:

* Je spočítána fitness pro všechny jedince
* Podle této hodnoty je pole jedinců seřazeno s tím, že přednost mají jedinci splňující formuli
* Na určeném početu jedinců proběhne mutace
* Následuje selekce a křížení.

Zásadním parametrem genetického algoritmu je selekční tlak. Celý proces selekce jsem se rozhodl implementovat pomocí exponenciálního rozdělení. Jedinci jsou seřazeni podle svých hodnot fitness. Následně je náhodně vygenerován index z rozsahu <0, n-1>, kde n značí počet jedinců. Tato náhodná hodnota je generována s exponenciálním rozdělením a proto je pravděpodobnost, že je vybrán lepší jedinec, vyšší. Parametr **λ** ovlivňuje hodnotu exponenciálního rozdělení a umožňuje tím změnu selekčního tlaku.

## Cíle experimentu:

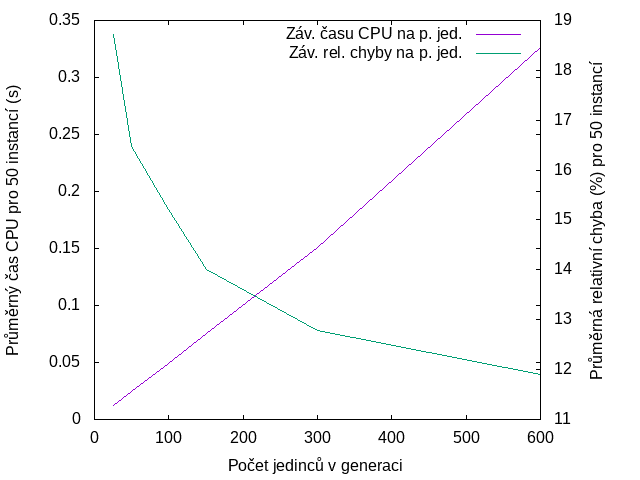
Vlastnosti algoritmu lze měnit pomocí různých parametrů. Mou implementaci lze ovlivnit následujícími parametry: celkový počet generací, počet jedinců v generaci, počet zachovaných nejlepších jedinců, počet zachovaných nejhorších jedinců, počet jedinců vzniklých křížením a pravděpodobnost mutace.

Protože je stavový prostor přípustných řešení příliš nespojitý, budeme průchodu uvažovat i nepřípustná řešení.

Cílem experimentu je ozkoušet schopnost heuristiky řešit problém 3 SAT v závislosti na zvolených parametrech a instanci dat. Nastavování parametrů je důležitou součástí postupu a je určující pro výslednou kvalitu řešení.

## Závislost na počtu instancí:

Pro parametr celkového počtu jedinců jsem zvolil rozsah <25, 600> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet iterací: 500, zachovaných nejlepších jedinců: 10%, zachovaných nejhorších jedinců: 7%, jedinců vzniklých křížením: 7%, faktor mutace: 0.1%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.



Graf [1] závislosti celk. času CPU a prům. rel. chyby v závislosti na počtu jedinců.

Z grafu [1] je čitelné, že s rostoucím počtem jedinců v generaci stoupá i potřebný výpočetní čas, podle odhadů lineárně, což podporují zobrazené hodnoty. Relativní chyba s rostoucím počtem jedinců klesá z počátku velice rychlým tempem.

## Závěr:

Během experimentu jsme zkoumali pokročilou iterativní heuristiku, genetický algoritmus. Ověřili jsme závislost vlastností algoritmu na ovládacích parametrech. Řešení se zdá být poměrně závislé na vhodné kombinaci těchto parametrů. Proto je důležité prozkoumat vlastnosti instancí, zejména jejich velikosti, a stanovat požadovanou relativní chybu, případně spotřebovaný čas.

Je nutné si uvědomit, že genetický algoritmus je pouze metodou průchodu stavového prostoru zkoumaného problému, proto je citlivá na uváznutí v lokálních extrémech.

Stejně tak si musíme uvědomit, že algoritmus je randomizovaný a proto pro různá spuštění algoritmu můžeme dostat různé výsledky. Má implementace má stanovaný počet generací, které vždy proběhnou, jedná se tak o metodu Monte Carlo, kdy je výsledek náhodná proměnná, ale potřebný čas je nezávislá proměnná. Případně by šlo algoritmus pozměnit a jako zastavovací podmínku učit relativní chybu, v tom případě by šlo o metodu Las Vegas a spotřebované výpočetní prostředky by tvořily náhodnou proměnnou.