4. Pokročilá iterativní metoda řešící problém batohu

Zadání úlohy:

- Zvolte si heuristiku, kterou budete řešit problém vážené splnitelnosti booleovské formule.
- Tuto heuristiku použijte pro řešení problému batohu.
- Hlavním cílem domácí práce je seznámit se s danou heuristikou, zejména se způsobem, jakým se nastavují její parametry.
- Problém batohu není příliš obtížný, většinou budete mít k dispozici globální maxima.
- Závěrečná úloha je co do nastavení a požadovaného výkonu heuristiky podstatně náročnější a může vyžadovat zcela jiné nastavení parametrů.

Popis zvoleného algoritmu:

Jako pokročilou heuristiku řešící problém batohu jsem si zvolil genetický algoritmus. Obecným schématem algoritmu je postupná tvorba generací různých řešení zadaného problému. Každý jedinec v generaci reprezentuje nějaké řešení. Nová generace je z předchozí vytvořena pomocí evolučních procesů, které se vyskytují v přírodě. Jsou jimi přirozený výběr, křížení, mutace a dědičnost.

Použité principy – vlastnosti.

- Selekce – z předchozí generace jsou přednostně vybráni jedinci, kteří mají vysokou hodnotu zdatnosti (fitness).
- Křížení – vzájemným křížením vybraných jedinců je vygenerován nový pro generaci následující.
- 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). Genetický algoritmus je často randomizovaný, kdy zdatnost určuje pravděpodobnost přenosu genetické informace a vlastností jedince na potomky.

Experiment:

Pro každý parametr jsem vygeneroval pomocí přiloženého generátoru vstupní data. Velikost testovaných instancí je 500 předmětů, celkem jsem otestoval 500 instancí pro každou kombinaci vstupních parametrů. Výsledný čas a relativní chyba je průměrem změřených hodnot pro testované instance.

Relativní chyba vůči optimálnímu řešení pro heuristické řešení je počítána podle doporučeného vzorce z eduxu. Hodnotu optimálního řešení jsem spočítal pomocí dynamického výpočtu s rozkladem podle ceny.

$$\varepsilon_{rel} = \frac{C(OPT) - C(APX)}{C(OPT)}$$

Použité přepínače kompilátoru GCC:

-Ofast -flto -Wno-sign-compare -std=c++11 -fopenmp -fstrict-aliasing -mfpmath=sse -mavx -mpc32 -funsafe-math-optimizations -ffast-math

Testovací soustava:

Intel(R) Core(TM) i5-3570K CPU @ 3.40GHz + 4 cores + Hyper Threading

Úlohy byly spouštěny v režimu jednoho vlákna a na PC neběžela žádná náročná úloha. K měření času jsem použil standardní knihovnu, se kterou mám dobré zkušenosti.

Cíle experimentu:

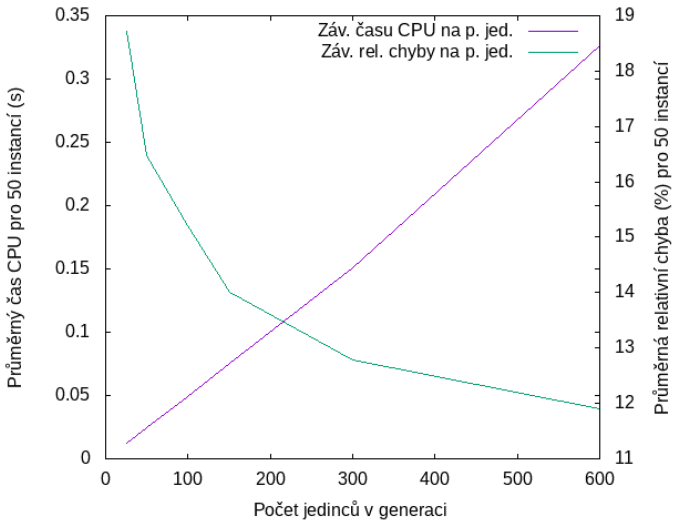
Vlastnosti algoritmu lze měnit pomocí různých parametrů. Mou implementaci lze ovlivnit následujícími parametry: celkový počet generací, počet jedinců v generaci, počet zachovaných nejlepších jedinců, počet zachovaných nejhorších jedinců, počet jedinců vzniklých křížením, počet jedinců vzniklých mutací.

Protože stavový prostor přípustných řešení není příliš nespojitý, můžeme v jeho průchodu uvažovat pouze přípustná řešení (tj. kombinace předmětů, které nepřesáhnou kapacitu).

Cílem experimentu je ozkoušet schopnost heuristiky řešit problém batohu 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: 5%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.

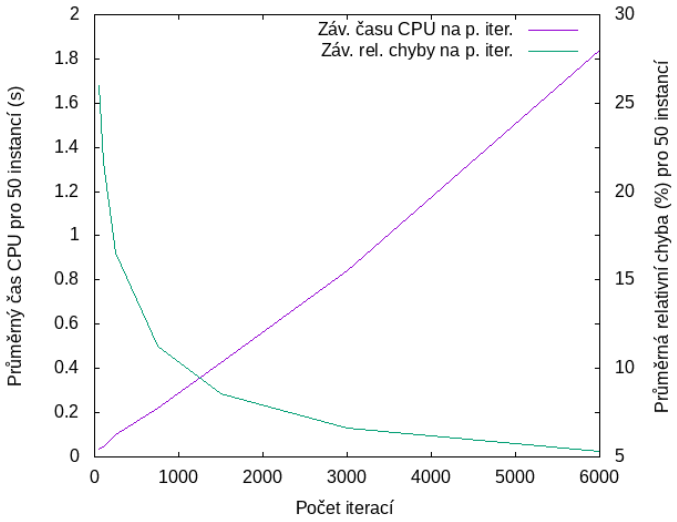


Graf [1] závislosti celk. času CPU a prům. rel. chyby v závislosti na počtu jedinců.

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.

Závislost na počtu generací:

Pro parametr celkového počtu iterací jsem zvolil rozsah <50, 6000> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet jedinců: 300, zachovaných nejlepších jedinců: 10%, zachovaných nejhorších jedinců: 7%, jedinců vzniklých křížením: 7%, faktor mutace: 5%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.

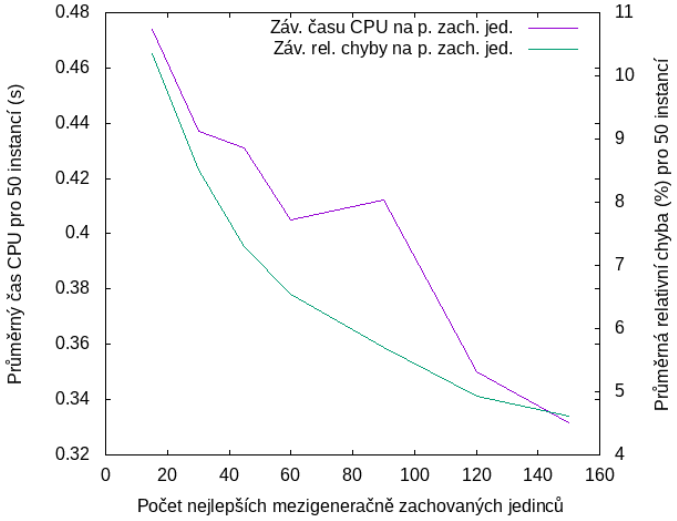


Graf [2] závislosti celk. času CPU a prům. rel. chyby v závislosti na počtu generací.

V grafu [2] jsou vyneseny průměrné časy a relativní chyby v závislosti na počtu generací. Graf odpovídá předpokladu, že v rostoucím počtem iterací klesá relativní chyba a zároveň se linerálně zvyšuje potřebný čas pro výpočet.

Závislost na procentu nejlepších jedinců:

Pro parametr ponechaného počtu nejlepších jedinců jsem zvolil rozsah <5%, 50%> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet jedinců: 300, počet iterací: 1500, zachovaných nejhorších jedinců: 7%, jedinců vzniklých křížením: 7%, faktor mutace: 5%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.

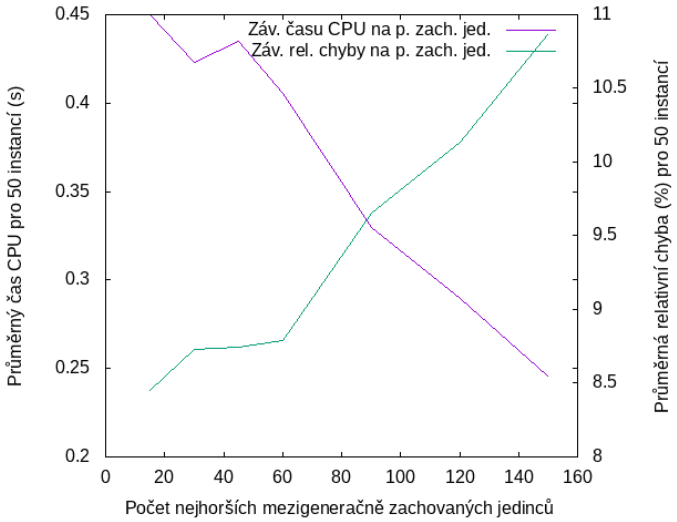


Graf [3] závislosti celk. času CPU a prům. rel. chyby v závislosti na počtu zachovaných jedinců.

V grafu [3] jsou vyneseny průměrné časy a relativní chyby v závislosti na počtu zachovaných jedinců. Průměrná relativní chyba se snižuje s roustoucím počtem zachovaných nejlepší jedinců. Potřebný čas CPU klesá, protože se zvyšujícím se poměrem zachovaných jedinců klesá průměrná složitost potřebná pro výpočet další generaci, protože zachování jedince je relativně levná operace.

Závislost na procentu nejhorších jedinců:

Pro parametr ponechaného počtu nejhorších jedinců jsem zvolil rozsah <5%, 50%> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet jedinců: 300, počet iterací: 1500, zachovaných nejlepších jedinců: 10%, jedinců vzniklých křížením: 7%, faktor mutace: 5%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.

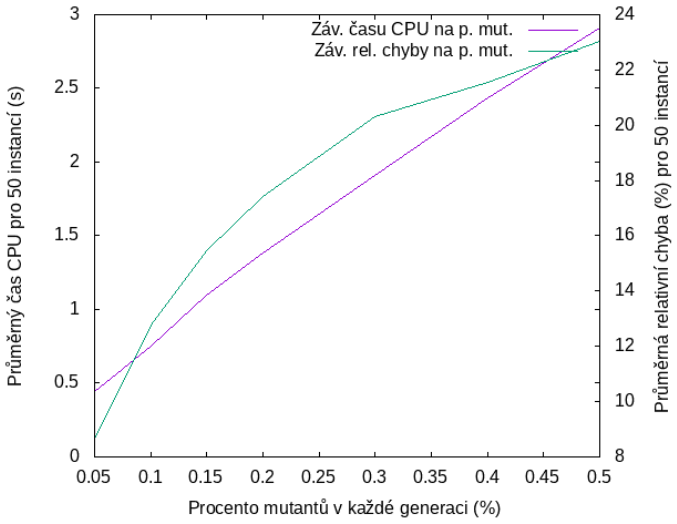


Graf [4] závislosti celk. času CPU a prům. rel. chyby v závislosti na počtu zachovaných jedinců.

V grafu [4] jsou vyneseny průměrné časy a relativní chyby v závislosti na počtu zachovaných jedinců. Průměrná relativní chyba se zvyšuje, což souvisí s tím, že zachování horších jedinců zvyšuje pravděpodobnost, že se jejich geny dostanou do další generace. Celkový čas CPU klesá, protože zachování jedince je levná operace, která v průměru urychlí čas výpočtu.

Závislost na procentu jedinců vzniklých mutací:

Pro parametr jedinců vzniklých mutací jsem zvolil rozsah <5%, 50%> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet jedinců: 300, počet iterací: 1500, zachovaných nejlepších jedinců: 10%, zachovaných nejhorších jedinců: 7%, jedinců vzniklých křížením: 7%, faktor mutace: 5%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.

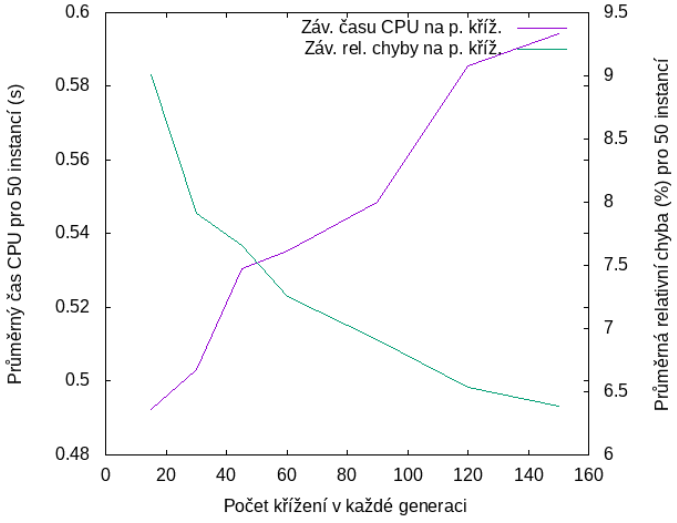


Graf [5] závislosti celk. času CPU a prům. rel. chyby v závislosti na procentu mutací.

V grafu [5] jsou vyneseny průměrné časy a relativní chyby v závislosti na procentu mutací. Čas CPU roste se zvyšujícím se počtem mutací v každé generaci. Je to proto, že mutace je relativně náročnou operací a rostoucí počet proto ovlivňuje celkový čas. Zároveň je vidět, že s rostoucím procentem mutací roste i relativní chyba. Procento mutací je příliš vysoké a negativně to ovlivňuje vlastnosti heuristiky. Proto je důležité držet procento mutací na poměrně nízké úrovni.

Závislost na procentu jedinců vzniklých křížením:

Pro parametr ponechaného počtu nejhorších jedinců jsem zvolil rozsah <5%, 50%> s nelineárním krokem. Zbývající parametry jsem zafixoval. Počet jedinců: 300, počet iterací: 1500, zachovaných nejlepších jedinců: 10%, zachovaných nejhorších jedinců: 7%, jedinců vzniklých křížením: 7%, zbylí jedinci byli mezigeneračně zkopírováni s pravděpodobností závislou na hodnotě fitness.



Graf [6] závislosti celk. času CPU a prům. rel. chyby v závislosti na procentu křížení.

V grafu [6] jsou vyneseny průměrné časy a relativní chyby v závislosti na procentu křížení. Čas CPU roste se zvyšujícím se počtem křížení v každé generaci. Je to proto, že křížení je relativně náročnou operací a rostoucí počet proto ovlivňuje celkový čas. Zároveň je vidět, že s rostoucím procentem křížení klesá relativní chyba. Vyšší procento jedinců vzniklých křížením snižuje rozptyl v populaci a tím umožňuje snadnější prostup lepších řešení (genů).

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.