NI-KOP 2. úkol

2022/23

JAN STEJSKAL

Obsah

1 Zadání

Zadáním úkolu je vyřešit problém maximální vážené splnitelnosti booleovské formule pomocí jedné ze dvou následujících heuristik na instancích v rozsahu 20–50 proměnných:

- Simulované ochlazování
- Genetický algoritmus

2 Řešení

2.1 Použitá heuristika

Pro vyřešení této úlohy jsem využil metodu simulovaného ochlazování.

2.2 Popis simulovaného ochlazování

Simulované ochlazování je jednou z metod prohledávání stavového prostoru. Od ostatních metod se liší tím, že s určitou, klesající pravděpodobností je schopna přijmout i horší stav. Tato pravděpodobnost je závislá na teplotě a koeficientu ochlazování.

2.3 Algoritmus

Pro implementaci jsem použil jazyk C++, kostra algoritmu simulovaného ochlazování byla převzata z přednášky.

2.3.1 Metoda try()

Pro prohledávání sousedních stavů jsem implementoval metodu try() následujícím způsobem. V 10% případů je nová konfigurace proměnných zcela náhodná. Ve zbylých 90% případů se konfigurace rozděluje na 2 případy, pokud je stávající konfigurace již řešením dané instance, pak se zvolí 1 náhodná proměnná, jejíž hodnota se prohodí, ale pokud konfigurace není řešením dané instance, pak se nalezne první klauzule, která není splněná a v ní se prohodí hodnota jedné náhodné proměnné.

Nová konfigurace je poté přijata v případě, že hodnota funkce cost() je pro ni vyšší než hodnota funkce cost() aktuální konfigurace, nebo pokud je nová konfigurace horší, pak může být přijata dle podmínky pro přijetí horšího stavu převzaté z přednášky. Jestliže ani při daném zhoršení nebyla konfigurace přijata, pak se zvýší čítač počtu konfigurací v řadě bez zlepšení, který pokud dosáhne nastaveného počtu, pak ukončí program.

2.3.2 Metoda cost()

Metoda cost(), která konfiguraci proměnných přiřazuje ohodnocení, tak provádí následujícím způsobem. Pokud konfigurace je řešením formule, pak vrací dosažený součet vah, v opačném případě vrací rozdíl počtu splněných klauzulí a celkového počtu klauzulí.

2.3.3 Metoda frozen()

Metoda frozen() vrací hodnotu true v případě, že aktuální teplota je rovna nebo menší nastavené koncové teplotě.

2.3.4 Metoda equilibrium()

Metoda equilibrium () vrací hodnotu true v případě, že počet iterací vnitřní smyčky algoritmu dosáhl nastaveného počtu.

2.3.5 Volba počátečního stavu

Počáteční stav je vždy zvolen zcela náhodně.

2.3.6 Volba počáteční teploty

Počáteční teplota je vždy pro danou instanci vypočtena automaticky. 1000x se spustí metoda try() a zaznamená se každé zhoršení hodnot funkce cost() pro aktuální a novou konfiguraci. Ze všech zhoršení se spočítá průměrné zhoršení. Počáteční teplota se poté spočítá jako:

*-1 * průměrné zhoršení* / *ln* (*pravděpodobnost přijetí horšího řešení*)

Za pravděpodobnost přijetí horšího řešení jsem zvolil 50%.

3 White box fáze

3.1 Test funkčnosti

Nejprve jsem otestoval celkovou funkčnost algoritmu na jedné instanci s 20 proměnnými a na jedné instanci s 50 proměnnými. Algoritmus neměl problém s instancí o 20 proměnných, ale instanci o 50 proměnných nebyl schopen vyřešit ani při velkém množství iterací s různými hodnotami parametrů heuristiky. Problém se nacházel v původní verzi metody try(), kdy v 90% případů, pokud konfigurace nebyla řešením formule, pak se ze všech klauzulí, které nebyly splněné, nashromáždily všechny jejich proměnné a z nich se poté vybrala jedna náhodná, která se prohodila. Tento způsob v případě mého algoritmu nebyl funkční a byl nahrazen způsobem, který je popsán v kapitole 2.3.1.

3.2 Ladění parametrů

V této fázi jsem ladil nastavení parametrů heuristiky. Použité parametry byly následující:

- Koeficient ochlazování
- Koncová teplota
- Mez ekvilibria
- Počet kroků pro zastavení bez zlepšení – fixní, hodnota 1000
- Počáteční teplota – nastavena automaticky

Pro účely ladění byl algoritmus puštěn vždy 50x na instanci *wuf50-218-M/wuf50-057.mwcnf* s příslušnými hodnotami parametrů. Pro porovnávání používám tyto metriky:

- **succ** - Počet úspěšných běhů, kdy algoritmus nalezne řešení, přičemž jsou všechny klauzule splněny.
- **iter** - Průměrný počet iterací úspěšných běhů
- **avg_re** - Průměrná relativní chyba úspěšných běhů (*optimum* - *váha* / *optimum*).

Pro názornost také přikládám grafy vývoje průběhu algoritmu a počtu splněných klauzulí aktuální konfigurace.

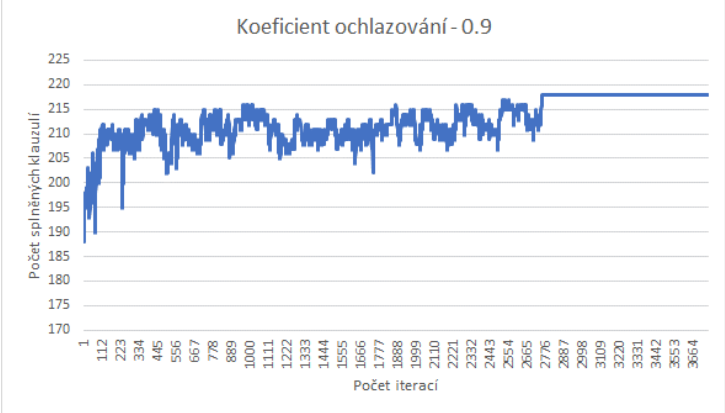
3.2.1 Koeficient ochlazování

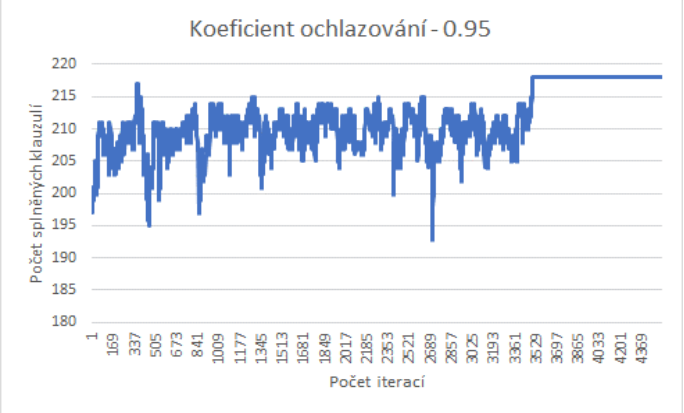
Koeficient ochlazování slouží ke snižování teploty v průběhu algoritmu. Testované hodnoty pro koeficient ochlazování jsou následující: 0.9, 0.95, 0.975, 0.995.

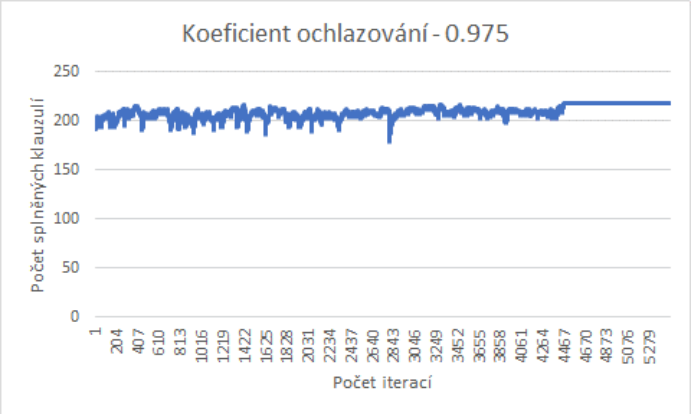
Hodnoty nastavených ostatních parametrů heuristiky:

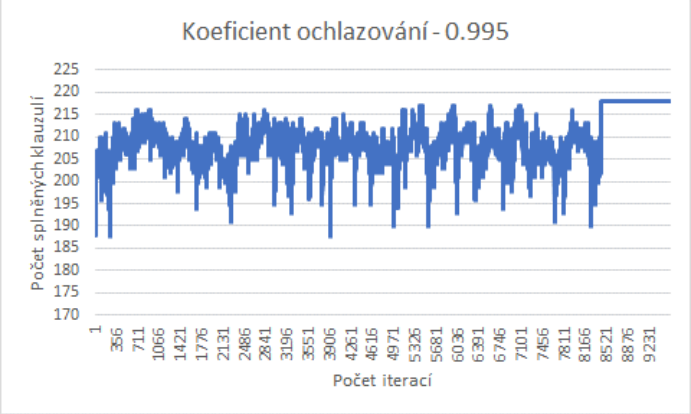
- Koncová teplota – 1
- Mez ekvilibria – 250
- Počet kroků pro zastavení bez zlepšení – 1000
- Počáteční teplota – nastavena automaticky

3.2.1.1 Průběhy algoritmu









3.2.1.2 Tabulka výsledků měření

Hodnota koeficientu	**succ**	**iter**	**avg_re**
0.9	78%	3025	0.076
0.95	88%	5641	0.0976
0.975	100%	8076	0.084
0.995	100%	11168	0.059

Z výsledků testování vidíme, že dvě hodnoty mají 100% úspěšnost a zbylé dvě mají úspěšnost menší než 100%. Z tohoto důvodu můžu koeficienty s hodnotou 0.9 a 0.95 rovnou zahodit. Rozhodl jsem se pro další testování zvolit hodnotu 0.995, protože má oproti hodnotě 0.975 o poznání lepší přesnost a nárůst průměrného počtu iterací není tak drastický.

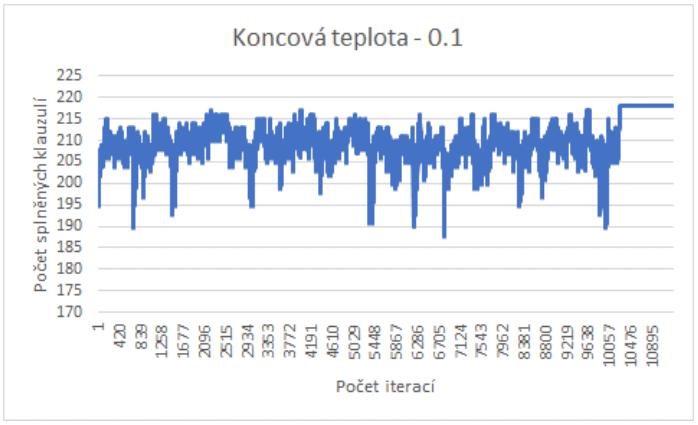
3.2.2 Koncová teplota

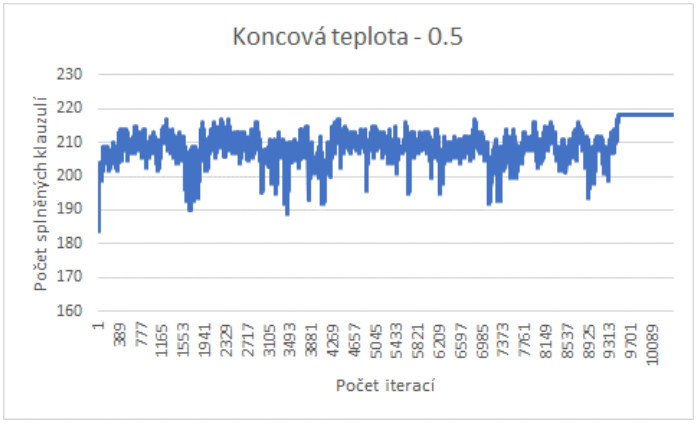
Koncová teplota slouží k ukončení iterování, pokud teplota klesne pod nastavenou hodnotu. Testované hodnoty pro koncovou teplotu jsou následující: 0.1, 0.5, 0.75, 1.

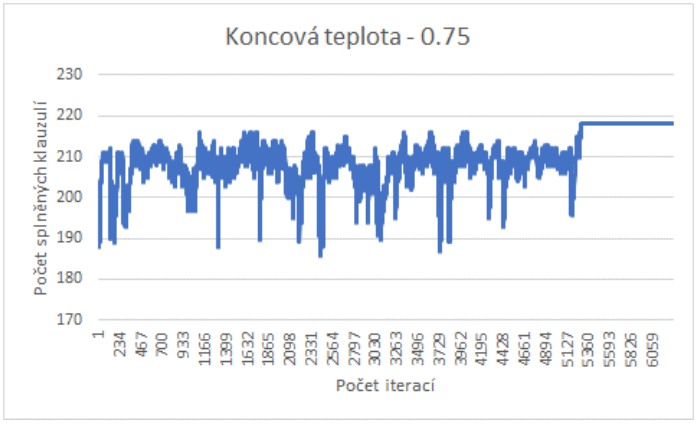
Hodnoty nastavených ostatních parametrů heuristiky jsou následující:

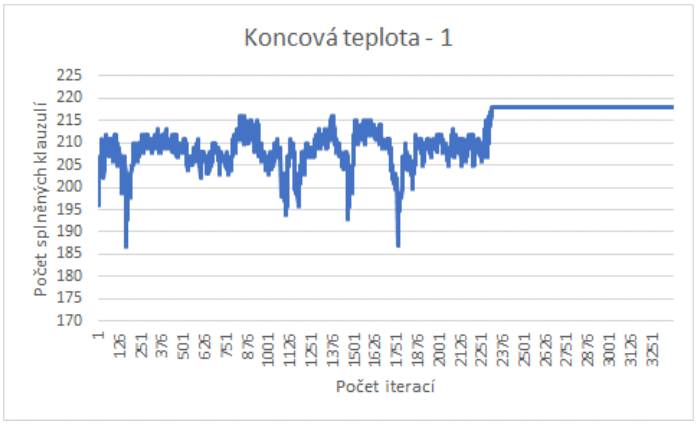
- Koeficient ochlazování – 0.995, jedná se o výsledek předešlého experimentu
- Mez ekvilibria – 250
- Počet kroků pro zastavení bez zlepšení – 1000
- Počáteční teplota – nastavena automaticky

3.2.2.1 Průběhy algoritmu:









3.2.2.2 Tabulka výsledků měření

Hodnota koncové teploty	**succ**	**iter**	**avg_re**
0.1	100%	14585	0.048
0.5	100%	13624	0.05
0.75	100%	12230	0.054
1	100%	11168	0.059

Na základě porovnání výsledků testování si můžeme všimnout, že se zvyšující hodnotou koncové teploty klesá počet iterací, ale zároveň se lehce zvyšuje relativní chyba algoritmu. V tomto případě jsem se rozhodl použít hodnotu 0.1, protože má nejvyšší přesnost a nárůst iterací je v porovnání s ostatními hodnotami relativně malý.

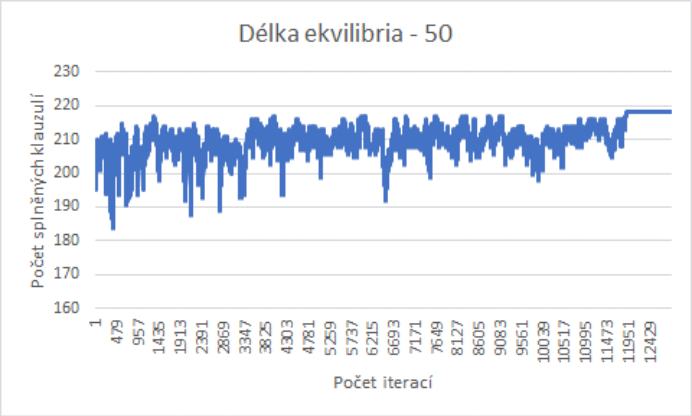
3.2.3 Délka ekvilibria

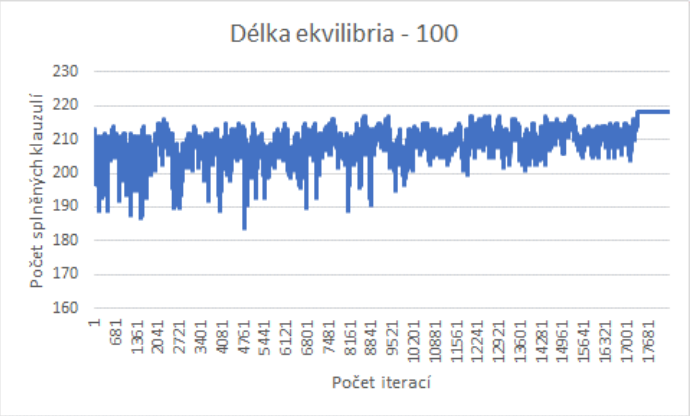
Parametr délky ekvilibria má za úkol ukončit iterování vnitřní smyčky algoritmu v případě, že počet iterací překročí nastavenou hodnotu. Testované hodnoty pro délku ekvilibria jsou následující: 50, 100, 250, 500.

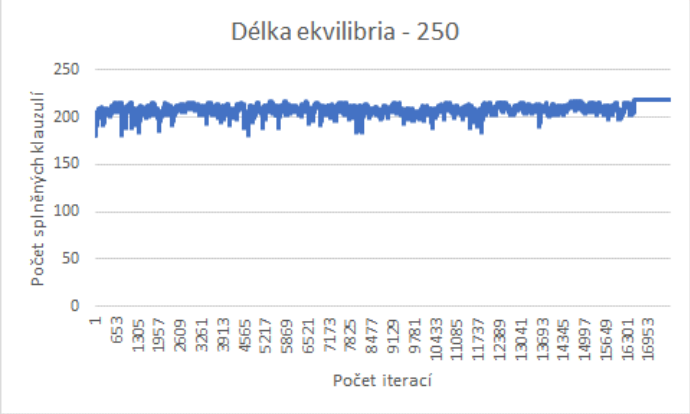
Hodnoty nastavených ostatních parametrů heuristiky jsou následující:

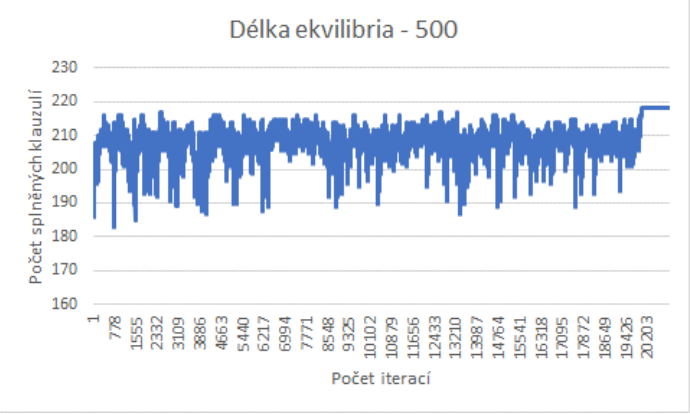
- Koeficient ochlazování – 0.995, jedná se o výsledek předešlého experimentu
- Koncová teplota – 0.1, jedná se o výsledek předešlého experimentu
- Počet kroků pro zastavení bez zlepšení – 1000
- Počáteční teplota – nastavena automaticky

3.2.3.1 Průběhy algoritmu:









3.2.3.2 Tabulka výsledků měření

Délka ekvilibria	**succ**	**iter**	**avg_re**
50	100%	7375	0.089
100	100%	10925	0.065
250	100%	11293	0.063
500	100%	15435	0.062

Z výsledků testování vidíme, že se zvyšující hodnotou délky ekvilibria se nám zvyšuje počet iterací, ale zároveň se snižuje průměrná relativní chyba. Rozdíl mezi hodnotou 100 a 250 v iteracích i průměrné relativní chybě je velmi nepatrný, nicméně zde hodnota 250 vychází v přesnosti o kousek lépe. Rozdíl u hodnot 250 a 500 v iteracích už je vzhledem k minimálnímu zlepšení relativní chyby větší, takže z tohoto důvodu jsem se rozhodl zvolit hodnotu 250.

3.2.4 Celkové výsledky

Po vykonání všech testů a porovnání jejich výsledků jsem se rozhodl použít následující hodnoty parametrů heuristiky pro blackbox fázi úlohy:

- Koeficient ochlazování – 0.995
- Koncová teplota – 0.1
- Mez ekvilibria – 250
- Počet kroků pro zastavení bez zlepšení – 1000
- Počáteční teplota – nastavena automaticky

Zvolené hodnoty dle mého názoru představují nejlepší kombinaci pro řešení většího množství instancí v rámci blackbox fáze. Preferovaným atributem u všech parametrů byla především co nejmenší relativní chybovost.

4 Blackbox fáze

V této fázi bylo provedeno závěrečné vyhodnocení heuristiky. Pro tento účel jsem použil následující sady instancí (každá jich obsahuje 100), přičemž pro každou instanci jsem zaznamenal 10 běhů:

- wuf20-71R-M
- wuf20-71R-N
- wuf20-71R-Q
- wuf20-71R-R
- wuf50-218R-M
- wuf50-218R-N
- wuf50-218R-Q
- wuf50-218R-R

Celkově jsem tedy provedl 8000 běhů. Zkoumané metriky jsou následující:

- **succ** - Počet úspěšných běhů, kdy algoritmus nalezne řešení, přičemž jsou všechny klauzule splněny
- **succ_opt** - Počet úspěšných běhů, kdy algoritmus nalezl řešení, přičemž jsou všechny klauzule splněny a zároveň bylo dosaženo optimální váhy
- **iter** - Průměrný počet iterací
- **avg_re** - Průměrná relativní chyba, (optimální váha – výsledek)/optimální váha
- **max_re** - Maximální relativní chyba

4.1 Zhodnocení výsledků

Sada	**succ**	**succ_opt**	**iter**	**avg_re**	**max_re**
wuf20-71R-M	100%	100%	165855	0	0
wuf20-71R-N	100%	100%	165099	0	0
wuf20-71R-Q	100%	93,4%	122491	0.0046	0.194
wuf20-71R-R	100%	92,7%	122183	0.0057	0.195
wuf50-218R-M	99,8%	53,2%	10810	0.04	0.57
wuf50-218R-N	100%	49,8%	10650	0.041	0.572
wuf50-218R-Q	99,7%	42,7%	11026	0.16	0.815
wuf50-218R-R	1%	45%	10309	0.19	0.818

Ze závěrečného měření vyplynuly poměrně zajímavé a překvapivé výsledky. Instance s 20 proměnnými byly vyřešeny s velmi malou relativní chybou a v případě verzí M, N se dostaly všechny běhy k výsledkům s optimální váhou, nicméně za cenu vyššího množství iterací. Iterace s 50 proměnnými už mají relativní chybu vyšší a počet běhů, které se dostaly k optimálním výsledkům se pohybuje od 42,7% do 53,2%, ale algoritmus na tyto řešení potřeboval malé množství iterací.

Algoritmus tedy u menších instancí s 20 proměnnými potřeboval více času, ale dosáhl optimálních výsledků, nebo k nim byl velice blízko. U větších instancí s 50 proměnnými sice algoritmus běžel rychle, ale výsledky z hlediska přesnosti už bohužel nebyly tolik uspokojivé.

5 Závěr

Implementoval jsem metodou simulovaného ochlazování algoritmus pro řešení problému maximální vážené splnitelnosti booleovské formule.

Následně jsem testoval různé hodnoty několika parametrů heuristiky a porovnával jejich výsledky na jedné instanci o 50 proměnných. Z těchto výsledků jsem vybral nejoptimálnější hodnoty a ty použil na testování algoritmu v Blackbox fázi.

Výsledky měření z Blackbox fáze, které proběhly na redukovaných sadách instancí o 20 a 50 proměnných ve verzích M, N, Q, R jsem zaznamenal do tabulky a vyhodnotil. Měření ukázalo menší přesnost algoritmu na větších instancích, naopak u menších instancí vyšší přesnost, ale také vyšší časovou zátěž. Pro potřeby úlohy považuji výsledky mého algoritmu za dostačující.