2

Řešení problému max. vážené splnitelnosti booleovské formule(MWSAT) pokročilou iterativní metodou

Zkoušková úloha – NI-KOP

Winkler, Erich
ČVUT FIT

1 Table of Contents

2 Zadání

Problém řešte některou z pokročilých heuristik:

- simulované ochlazování
- genetický algoritmus

Heuristika musí zvádat instance rozdílných vlastností (zejména velikosti) bez interaktivních zásahů. Minimální rozsah velikosti je 20–50 pro nejméně kvalitní práce. Je třeba doložit práci heuristiky v *celém* stanoveném rozsahu (nikoliv pro jednu velikost někde v rozsahu).

Základní kód musí být vypracován samostatně. Jazyk je libovolný. Komponenty použité z jiných zdrojů (knihovny, moduly) musí být jasně citovány.

Po nasazení heuristiky ověřte její vlastnosti experimentálním vyhodnocením, které přesvědčivě doloží, jakou třídu (rozsah, velikosti…) instancí heuristika zpracovává. Doporučujeme používat metriky nezávislé na platformě. Zejména v případě použití nestandardních, např. originálních technik doložte jejich účinnost experimentálně (což vyloučí případné diskuse o jejich vhodnosti).

Zpráva musí dokládat Váš racionální přístup k řešení problému, tedy celý pracovní postup. Ve zprávě je nutno popsat obě fáze nasazení heuristiky, jak nastavení, **(white box fáze),** tak závěrečné vyhodnocení heuristiky **(black box fáze).** Práce bez jasně oddělených těchto dvou fází bude pokládána za **neúplnou**. Protože zpráva dokládá Váš postup, prosím v popisu white box fáze uvádějte i neúspěšné pokusy, slepé uličky atd. V

opačném případě vzniká podezření na volbu parametrů ad hoc. V popisu black box fáze uveďte, proč pokládáte výsledky za průkazné (alespoň úvahou).

Kompletní zadání naleznete zde: https://courses.fit.cvut.cz/NI-KOP/homeworks/files/task2.html

3 Heuristika – Simulované ochlazování

K řešení našeho problému jsem se rozhodl použít Simulované ochlazování. Nasazená heuristika bude řešit problém 3SAT. Tato heuristika se řadí mezi lokální metody prohledávání stavového prostoru.

4 Implementace a nasazení simulovaného ochlazování

Tato kapitola popisuje mnou vytvořenou implementaci simulovaného ochlazování a postupné nasazení této heuristiky. Pro přehlednost jsem se rozhodl rozdělit můj postup do iterací.

4.1 Popis implementace

Pro implementaci heuristiky byl vybrán jazyk C++ a to především z výkonnostních důvodů. Základem heuristiky je metoda nazvaná **simulation()**. Tato metoda implementuje cyklus naznačený v následujícím pseudokódu. Jak můžeme vidět, metoda obsahuje hlavní cyklus heuristiky a využívá několik metod.

4.1.1 Metoda Simulation()

```
void Annealing::simulation(SAT & _satProblem){
   currentState = state(_satProblem);
   bestState = currentState;
   while(frozen()){
      for(int i = 0; i < equilibrium; i++){
         state _newState(currentState);
         newState = _newState;
         newState.Neighbour();
         int diff = newState.calculateCost() - currentState.calculateCost();
         if(diff > 0 ){
            currentState = newState;
         }else if (accept()){
            currentState = newState;
         }
               IsItest();
      }
      cool();

   }
}
```

4.1.2 Metoda frozen()

Tato metoda určuje zda bylo dosaženo minimální teploty. Pokud ano, vrací true a cyklus končí.

4.1.3 Metoda CalculateCost()

Tato metoda je inspirována metodou Cost() z přednášky. Jejím účelem je „pokutovat“ stavy s velkým množstvím nesplněných klauzulí, a naopak oceňovat stavy blížící se optimálnímu řešení. Během tohoto výpočtu je potřeba zároveň brát v potaz váhu jednotlivých proměnných. Toho bylo dosaženo použitím následujícího vzorce.

$$Cost = Váha\ configurace + (Konstanta * počet\ splněných\ klauzulí)$$

4.1.4 Metoda Neighbour()

Tato metoda vytvoří novou konfiguraci proměnných pro otestování. Pro pilotní experimenty je implementována tak, aby vždy změnila hodnotu náhodně vybrané proměnné. Tento nový stav je přijat v případě, že jeho hodnota je vyšší než hodnota současného nebo s pravděpodobností danou metodou accept().

4.1.5 Metoda Accept()

Tato metoda určuje pravděpodobnost přijetí horšího stavu tak abychom unikli z lokálního maxima. Pravděpodobnost je určena následujícím vzorcem z přednášky.

$$Random(0,1) < Exp(\frac{NewState.cost - CurrentState.cost}{currT})$$

4.1.6 Metoda Cool()

Metoda přenásobuje aktuální teplotu chladícím koeficientem. Hodnota tohoto koeficientu je předmětem následujícího zkoumání.

4.2 Nasazení heuristiky

Samotné nasazení bude probíhat v několika iteracích. Nejdůležitějším předmětem zkoumání bude nastavení všech parametrů a jejich vlivu na chybovost a rychlost výpočtu. Stejně tak bude předmětem zkoumání úprava metody Neighbour – změna přechodu mezi stavy a nastavení počáteční konfigurace proměnných.

4.2.1 Specifikace platformy

Na testování a měření byl využit notebook s následujícími parametry.

Počítač:	Apple Macbook Air
CPU:	1.1GHz dual-core Intel Core i3
OS:	macOS
RAM:	8GB
Typ disku:	SSD
Programovací jazyk:	C++

4.3 White Box fáze

4.3.1 První iterace

Počáteční iterace sloužila především k prvotnímu seznámení s fungováním heuristiky a vytvoření nějakého prvotního odhadu vhodného nastavení parametrů.

Nastavení parametrů:

- Počáteční stav – Pro pilotní experiment jsou všechny proměnné ohodnoceny 0/false
- Počáteční teplota – Počáteční teplota je nastavena na hodnotu 10000
- Chladící koeficient – Pro pilotní experiment je nastaven na hodnotu 0.95
- Bod mrazu – Nastavena na pevně danou hodnotu 1.
- Equilibrium – Pro pilotní experiment nastaveno na hodnotu 50

Pilotní experiment byl proveden na datové sadě wuf20-91R-M. Čas potřebný k výpočtu byl změřen pomocí knihovny **std::chrono** v milisekundách.

4.3.1.1 Výsledek první iterace:

Vzhledem k tomu, že šlo o zcela první testování heuristiky, první čemu byla věnována pozornost je vývoj vah řešení během běhu programu. Tyto naměřená data jsou k dispozici ve složce Measurements. Ikdyž měření bylo provedeno pro všechny instance, pro přehlednost vykreslím graf pouze dvou běhů na dvou různých instancích.

V obou případech vidíme, že ke konci běhu se dostáváme k řešením s největší váhou. Viditelně i vidíme okamžiky kde bylo přijato horší řešení, tak abychom opustili lokální maximum. Graf ovšem vypadá trochu divoce a v dalších iteracích se pokusíme přijít na to proč. Vinu prozatím přikládám špatnému nastavení parametrů.

Úspěšnost nalezení optimálního řešení:

Další metrikou kterou je zajímavé sledovat jak často heuristika nalezla optimální řešení, kterou můžeme zjistit porovnáním s referenčními daty poskytnutými společně s testovacími sadami. Na této datové sadě s daným nastavením parametrů jsme dosáhli úspěšnosti přesahující 92%.

4.3.2 Druhá iterace

V první iteraci jsme si ověřili že daný algoritmus je schopen hledat optimální řešení pro výběr z instancí malého počtu proměnných. Nyní je potřeba zvolit co nejvhodnější parametry. Za tímto účelem musím brát v potaz tři věci. Kvalitu výsledku, čas výpočtu a relativní chybu. Relativní chyba udává, o kolik procent se průměrně nesprávný výsledek lišil od optimálního řešení. Prvním parametrem, který jsem se rozhodl otestovat je chladící koeficient.

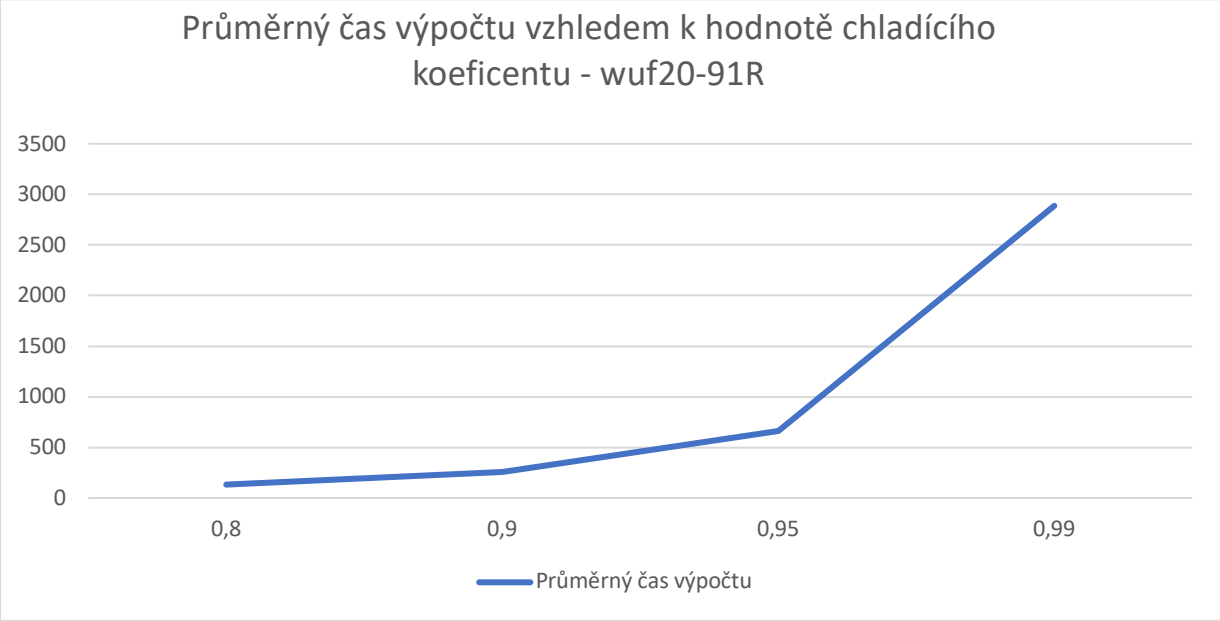
Pro tento experiment jsem se rozhodl využít zkrácené sady wuf20-91R.

Tabulka 1 – Vliv ochlazovacího koeficientu na výsledek

Chladící koeficient	Průměrný čas výpočtu	Procentuální úspěšnost nalezení optimálního řešení	Relativní chyba
0.8	132.68 ms	77 %	10.5 %
0.9	254.85 ms	87 %	10.14 %
0.95	660.06 ms	92 %	6.7 %
0.99	2887.96 ms	100 %	0%

Výsledek druhé iterace:

V tabulce 1 můžeme vidět, že úspěšnost nalezení optimálního řešení výrazně stoupala se zvětšujícím se chladícím koeficientem. Zároveň s tím výrazně klesala i relativní chyba u špatných výsledků. Úměrně tomu se i zvedá průměrný čas potřebný k vyřešení jedné instance, jak můžeme vidět na obrázku 3. Z toho je také patrné, že vývoj této křivky není lineární. U větších instancí by tedy koeficient blížící se k 0.99 mohl být časově velmi náročný. Nejlepší poměr čas/kvalita výsledku si zatím zachovává ochlazovací koeficient 0.95. Rád bych v rámci této iterace provedl obdobný test na instancích o více proměnných. K tomu jsem si vybral datovou sadu wuf50-218R.



1. Průměrný čas výpočtu potřebný k vyřešení jedné instance

Výsledek pro wuf50-218R:

Podle mých očekávání, úspěšnost algoritmu výrazně klesla jak ukazuje tabulka níže. Nicméně trend zůstává stejný, čím vyšší ochlazovací koeficient, tím lepších výsledků algoritmus dosahuje. Nadále se budu zabývat pouze koeficienty 0.9+, protože z dostupných měření, nižší hodnoty mají velmi neuspokojivé výsledky.

Chladící koeficient	Průměrný čas výpočtu	Procentuální úspěšnost nalezení optimálního řešení	Relativní chyba
0.8	146.36 ms	39 %	9.8 %
0.9	338.8 ms	51 %	7.02 %
0.95	622.06 ms	68 %	5.7 %
0.99	3419.68 ms	91 %	2.7 %

Tabulka 2: Vliv ochlazovacího koeficientu na úspěšnost algoritmu

4.3.3 Třetí iterace – Počáteční teplota a ekvilibrium

V minulých iteracích jsme si ověřili funkčnost algoritmu a zásadní vliv ochlazovacího koeficientu na kvalitu výsledku. Nyní bych se rád zaměřil na počáteční teplotu a ekvilibrium. Tedy na dva parametry, které hrají zásadní roli v počtu vnitřních kroků algoritmu. Od zvýšení počáteční teploty a equilibria si slibuji výrazné zlepšení výsledků pro větší instance.

První, na co jsem se rozhodl zaměřit je počáteční teplota. Zvolil jsem 3 různé varianty pro které jsem se rozhodl provést měření. První varianta je konstanta kterou jsem použil pro pilotní experiment v první iteraci. Dále jsem zvolil výpočet dle vzorce $(počet\_Proměnných * Počet\_Klauzulí)$. Tuto variantu v ve výsledkových tabulka nazývám *Varianta A*. Jako poslední možnost jsem zvolil $(Počet_{Proměnných} * Počet_{Klauzulí}) * Konstanta$. Pro následující měření jsem konstantu nastavil na hodnotu 100. V této iteraci budu používat sadu wuf50-218R.

Ochlazovací koeficient	Počáteční teplota	Úspěšnost
0,9	10000	51%
	Varianta A	52%
	Varianta B	55%
0.95	10000	68%
	Varianta A	68%
	Varianta B	74%
0.99	10000	91%
	Varianta A	93%
	Varianta B	92%

Tabulka 3: Vliv počáteční teploty na úspěšnost algoritmu

K mému velkému překvapení, varianta A se jeví pro některé případy lepší než Varianta B. Obě se ale jeví lepší než počáteční teplota z pilotního experimentu. Myšlenka se tedy jeví jako dobrá, nicméně bude záležet na správné zvolené konstantě K. Proto jsem se rozhodl snížit konstantu K na hodnotu 10 a pokusit se najít vhodnou hodnotu equilibria. I v tomto případě se konstantní varianta jeví jako nevhodná. Proto jsem se rozhodl tuto hodnotu navázat na počet proměnných. Jedna varianta nazvaná EQA, nastavuje ekvilibrium čistě na počet proměnných. Druhá varianta násobí počet proměnný konstantou. Pro toto měření jsem se rozhodl nastavit tuto konstantu na hodnotu 10. V tabulce níže je tato varianta nazvána EQB.

Ochlazovací koeficient	Equilibrium	Úspěšnost
0,9	50	55%
	EQA	55%
	EQB	90%
0.95	50	74%
	EQA	74%
	EQB	94%
0.99	50	93%
	EQA	93%
	EQB	98%

Tabulka 4: Equlibrium - wuf50-218R

V tabulce 4, můžeme vidět, že varianta EQB se jeví jako velmi úspěšná. Pro ochlazovací koeficient se dokonce dostáváme na úspěšnost 98%. Další informace kterou jsem z měření získal se týká času výpočtu. Se zvyšováním equlibria se výrazně zvyšuje čas výpočtu. Pro ochlazovací koeficient 0.99 a variantu EQB trvalo nalezení výsledku pro jednu instanci průměrně 25 vteřin. Proto v budoucnu bude možná nutné jeden z parametrů snížit.

4.3.4 Čtvrtá iterace – Změna generování následujícího stavu

Dalším způsobem, který by mohl značně ovlivnit fungování heuristiky je způsob generování následujícího stavu. Až doposud byla náhodně změněna jedna proměnná v konfiguraci proměnných. Otázkou je, zda by nebylo lepší změnit více proměnných najednou. V této iteraci jsem se rozhodl otestovat dvě další varianty. Změnu náhodného počtu proměnných a změnu proměnné která se vyskytuje v nesplněné klauzuli. Osobně si od této varianty slibuji největší úspěch. I pro tuto iteraci jsem vybral sadu wuf50-218R.

Chladící koeficient	0.9	0.95	0.99
Změna jedné proměnné	90 %	94 %	98 %
Změna náhodného počtu	63 %	70 %	74 %
Změna proměnné z nesplněné klauzule	90 %	95 %	98 %

Tabulka 5: Vliv změny způsobu generování nového stavu na úspěšnost heuristiky

Jak můžeme vidět v tabulce 5, varianta se změnou náhodného počtu proměnných se jeví jako zcela nepoužitelná. Špatnou zprávou je, že zlepšení s moji novou metodou není zdaleka takové, v jaké jsem doufal. Malé pozitivum ovšem vidím v tom, že přestože tato varianta dosahuje stejné procentuální úspěšnosti, snížila se relativní chyba oproti optimálnímu výsledku. Pro další měření tedy použiji variantu č. 3, ale k dosažení lepších výsledků budu muset evidentně použít jiný mechanismus.

4.3.5 Pátá iterace – Adaptační mechanismus – Reset Teploty

V této iteraci jsem se rozhodl implementovat adaptační mechanismus za účelem zvýšení úspěšnosti algoritmu. Jako první jsem si vybral resetování teploty.

Postup:

- Po zamrznutí dojde k resetování teploty a nový stav je náhodně vygenerován.
- Algoritmus restartujeme dvakrát
- Budeme sledovat, zda se skutečně zvýšila úspěšnost a jak moc se zvedla časová náročnost

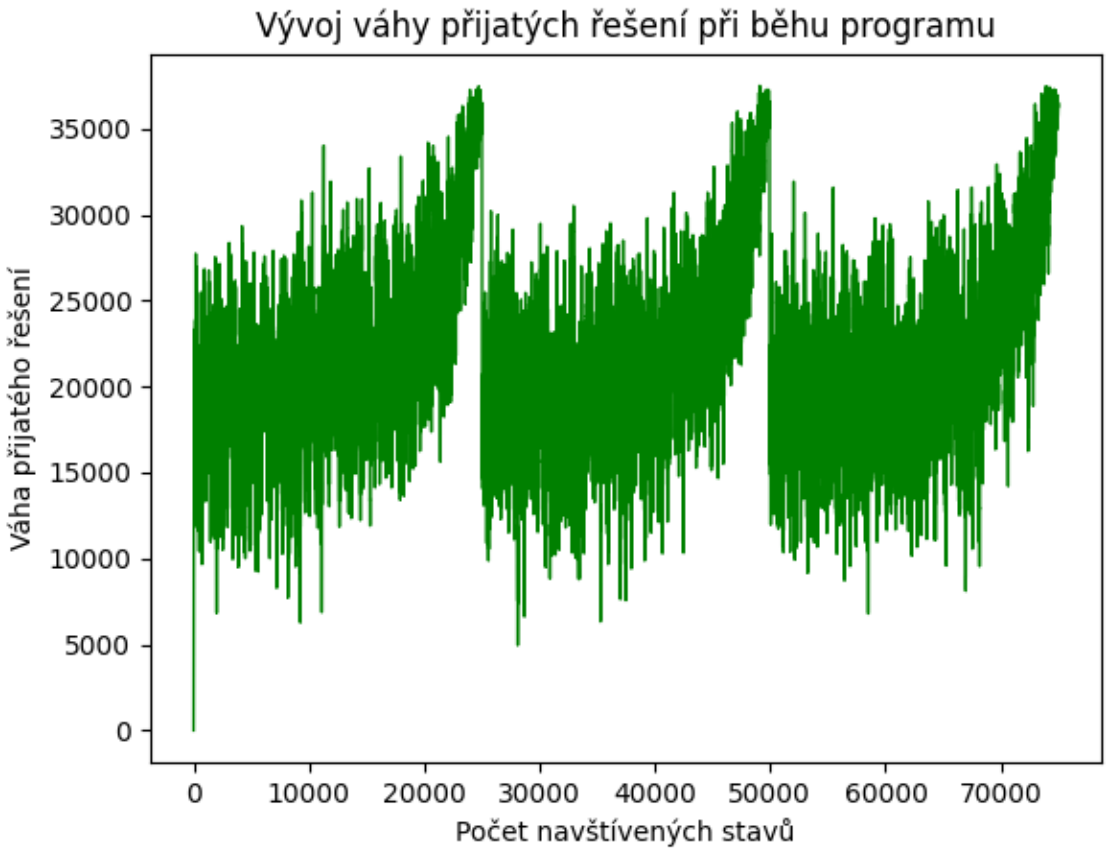
Výsledek měření páté iterace:

V tabulce 6 můžete vidět souhrn výsledku měření. Osobně mě zklamalo, že ani s tímto adaptačním mechanismem jsem na této datové sadě nedosáhl úspěšnosti 100 %. Nicméně můžeme vidět, že jsme schopni dosáhnout 98 % úspěšnosti i se sníženým ochlazovacím koeficientem. Výsledkem je, že algoritmus dosáhne stejných výsledků za nižší čas. A to o průměrně o téměř 7 vteřin na instanci. Tento rozdíl je naprosto zásadní. A je hlavním přínosem resetu teploty na výsledky měření.

Ochlazovací koeficient	Varianta	Úspěšnost	Čas	Relativní chyba
0.9	Bez Resetu	90%	3668,23	6,19%
	Reset(2x)	98%	9389,3	6,19%
0.92	Bez Resetu	90%	3925,29	4,73%
	Reset(2x)	98%	11453,08	6,19%
0.95	Bez Resetu	95%	5105,72	4,19%
	Reset(2x)	98%	18775,2	6,19%

Tabulka 6: Porovnání vlivu resetu teploty na výsledky – wuf50-218R

Zároveň jsem se rozhodl stejné měření provést na testovací sadě *wuf20-91R.* Výsledky tohoto měření byly velice pozitivní. Pro všechny 3 hodnoty ochlazovacího koeficientu jsem s Resetem dosáhl **úspěšnosti 100 %.** Jako nejvýhodnější ochlazovací koeficient v této konfiguraci se jeví jako 0.92. A to především kvůli jeho poměru (*čas výpočtu/ kvalita výsledku).* Z obrázku 3, můžeme vidět průběh programu pro instanci wuf-50-0110. Tento graf ukazuje váhu aktuálního stavu v závislosti k počtu navštívených stavů. Můžeme tak vidět, že váha navštívených stavů postupně stoupá. Zároveň vidíme, že algoritmus v začátku přijímá i výrazně horší stavy a postupně se ustáluje, což je očekávané chování. Z grafu se jeví chování jako očekávané. Na grafu vidíme dva resety a přijímání většího počtu „horších“ stavů na začátku běhu, tak abychom unikli z lokálního maxima.



2. Průběh programu pro wuf50-110

4.3.6 Šestá iterace – Změna funkce cost()

Tato iterace byla vytvořena kvůli problému, který vznikl při prvním pokusu o měření v Black box fázi. Po změření celé datové sady wuf20-91M (1000 instanci) se jevilo vše v největším pořádku a úspěšnost přesahovala hodnotu 99 %. Nicméně po stejném experimentu na datové sadě wuf20-91N úspěšnost spadla pod 40 %.
Z materiálu na cvičení víme, že i tato sada by měla navádět algoritmus ke správnému řešení. Není tedy důvod aby můj algoritmus přestal na této sadě tak zásadně fungovat.

Nakonec jsem dospěl k závěru, že chyba byla ve špatně implementované funkci cost(). Jak uvádím výše, tato funkce pokutovala stavy, které obsahovaly nevyřešené klauzule. Čím více nevyřešených formulí, tím menší hodnota byla. Prvním problémem se ukázalo, že hodnota cost vstupovala do záporných hodnot. Druhý problém byl, že počet nesplněných klauzulí byl násoben konstantou 1000 a nereflektoval váhy stavů kterých mohl daný 3SAT problém nabývat.

Rozhodl jsem se tedy implementovat tuto funkci následovně:

$$StateCost = Weight + (Satisfied\ Clauses * (\ MaxWeight + 1))$$

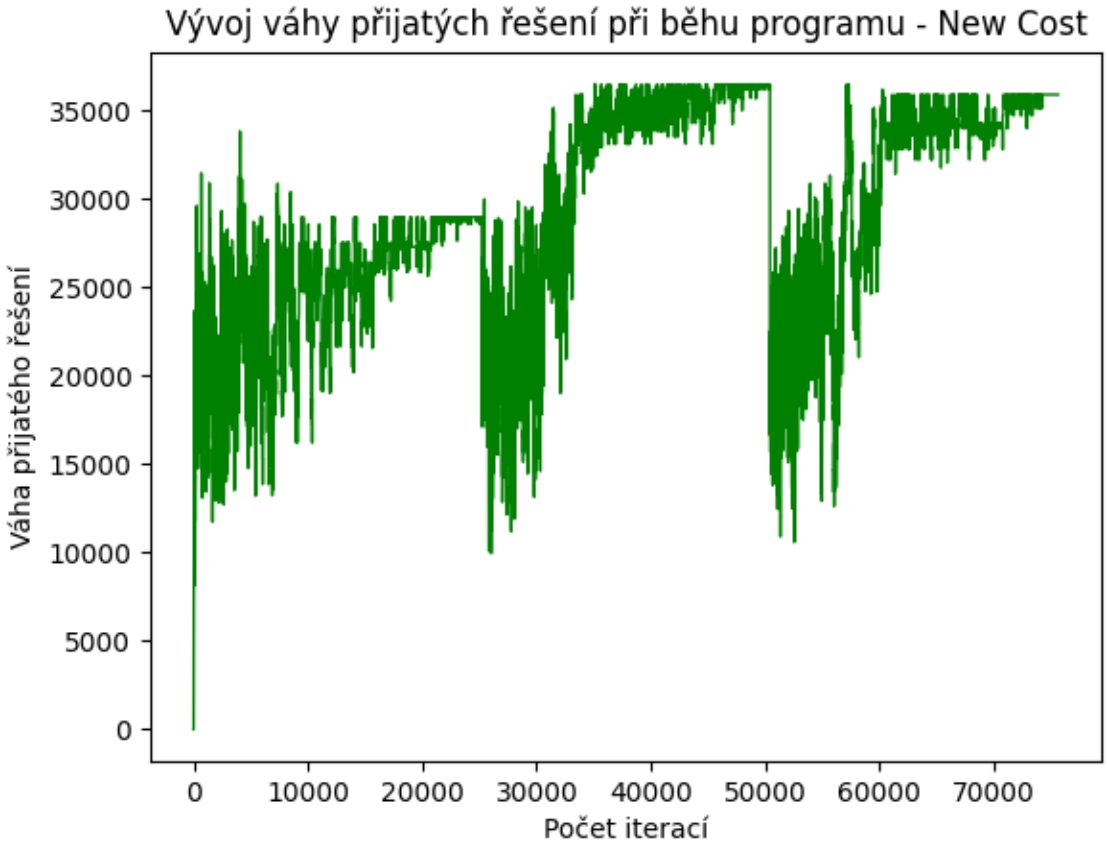
Tímto stylem se nikdy nedostane hodnota StateCost do záporných hodnot a zároveň funkce odměňuje stavy s větším počtem splněných klauzulí úměrně váze proměnných. Tuto funkci jsem se rozhodl experimentálně ověřit na sadách wuf20-91R – M a N.

Výsledek šesté iterace:

V tabulce 6 můžeme vidět, že změna funkce cost zásadně zlepšila úspěšnost algoritmu na datové sadě N a to na hodnotu okolo 98%. To považuji za výborný výsledek. Dodatečně jsem spustil již upravený algoritmus na datovou sadu typu wuf20-91R - Q. K mému překvapení dosáhla úspěšnosti 100%. Vzhledem k tomu, že tato sada má vytvářet zavádějící úlohy, beru to jako velký úspěch. Z obrázku 4 můžeme vidět, že pro instanci wuf50-010 dojde pokaždé k ustálení v lokálním maximu. V prvním případě se ovšem nejedná o globální maximum a algoritmus uvázne v lokálním maximu z kterého se mu již nepodařilo dostat. Nicméně, to je něco co poměrně efektivně řeší reset teploty z minulé iterace.

	Old cost()	New cost()
Sada M	100%	100%
Sada N	42%	98%

Tabulka 5: Úspěšnost algoritmu po změně cost()



3. Průběh wuf-50-010

4.4 Black Box fáze

4.4.1 Finální nastavení

- Počáteční stav – Pro pilotní experiment jsou všechny proměnné ohodnoceny 0/false
- Počáteční teplota – Počáteční teplota je vypočtena vzorcem $\text{Počet proměnných} * \text{Počet Klauzulí} * 10$
- Chladící koeficient – Pro pilotní experiment je nastaven na hodnotu 0.95
- Bod mrazu – Nastavena na pevně danou hodnotu 1.
- Equilibrium – Nastaveno na $Počet\ proměnných * 10$

Výsledky finálního měření:

Tabulka 6 zobrazuje výsledky měření blackboxové fáze na plných sadách. Bohužel, u žádné sady se nepodařilo dosáhnout úspěšnosti 100 %. Nicméně, datové sady pro 20 proměnných se této hodnotě velmi blížili, a to včetně zavádějící datové sady Q. K mému velkému překvapení, na této datové sadě fungoval algoritmus zcela nejlépe. Ze všech 1000 instancí se dopustil pouze jediné chyby.

Zklamáním je pokles úspěšnosti na datových sadách pro 50 proměnných. Navzdory tomu, že se ve White box fázi jevilo, že úspěšnost pro tyto datové sady by mohla být velmi podobná, důkladné měření toto bohužel nepotvrdilo. Pravdou ale je, že pokles nebyl tak extrémní a relativní chyba oproti optimálnímu řešení se pohybovala v řádech jednotek procent.

K dalšímu poklesu úspěšnosti došlo pro datové sady pro 75 proměnných. Dobrou zprávou je, že i pro datové sady M a N algoritmus pořád dosahuje úspěšnosti blížící se 80%. Výrazně horší je datové sada Q je úspěšnost spadla až k 60%.

sada	úspěšnost	čas výpočtu
wuf 20-91-M	99,50%	5300
wuf 20-91-N	95,50%	4345
wuf 20-91-Q	99,90%	4238,15
wuf-50-218-M	93%	21206,89
wuf-50-218-N	89%	31370,56
wuf75-325-M	78%	47359,63
wuf75-325-N	77%	50025,16
wuf75-325-N	62%	52034,29

Tabulka 6: Výsledky měření

4.4.2 Závěr

V rámci této úlohy jsem jsem implmentoval algoritmus simulovaného ochlazování pro problém vážené splnitelnosti formule. V rámci nasazení algoritmu bylo vytvořeno několik verzí. Důkladným zkoumáním během White box fáze jsem dospěl k hodnotám parametrů, které se z měření na zkrácených datových sadách jevili jako nejlepší ve vztahu k úspěšnosti algoritmu. Výsledkem tohoto zkoumání je nastavení, které je popsáno na začátku Black Box fáze. Pro další zlepšení úspěšnosti algoritmu jsem implementoval adaptivní mechanismus – **Reset Teploty.** Který výrazně napomohl ke zlepšení výsledků.

V rámci Black box fáze byly naměřeno chování algoritmu pro plné datové sady. Pro datové sady z *wuf-20-91* algoritmus fungoval velmi dobře. A to včetně zavádějící datové sady Q, kde z 1000 instancí se algoritmu nepodařilo nalézt optimální řešení pouze pro jednu.

Pro datové sady *wuf-50-218*, úspěšnost heuristiky lehce klesla na hodnoty pohybující se okolo 90% jak naznačuje tabulka 6. S tímto výsledkem osobně nejsem spokojený, vzhledem k tomu, že po měřeních ve White Box fázi jsem si sliboval výrazně lepší výsledky po nasazení adaptivního mechanismu a úpravy funkce cost. Nicméně je třeba brát v úvahu, že se jedná o problém patřící k nejtěžším ve třídě NPO a úspěšnost okolo 90% s relativně malou odchylkou od optimálních hodnot u špatných řešení se dá brát jako úspěch.

Na závěr jsem změřil plné datové sady *wuf75-325.* I v tomto případě došlo k poklesu úspěšnosti. V datových sadách M a N na hodnoty okolo 78%. K výraznějšímu poklesu došlo u zavádějící datové sady Q a to až na hodnotu 62%. Dobrou zprávou je, že ve všech případech se relativní chyba pohybovala do 3%. Tedy heuristika nacházela řešení velice blízko optimálnímu řešení.

Osobně považuji cíle této práce za splněné. Pokročila heuristika simulované ochlazovaní byla nasazena na daný problém. Na základě důkladného měření byly postupně upraveny funkce vytvářející algoritmus a byly nalezeny „rozumné“ parametry. Ideální konfigurace na základě White box fáze byla použita pro rozsáhle měření pro Black box fázi jejíž výsledky můžeme vidět v tabulce 6 a čistá naměřená data ve složce Measurements-final.