Řešení problému vážené splnitelnosti booleovské formule simulovaným ochlazováním

NI-KOP DÚ 5

Tomáš Kalabis

Problém vážené splnitelnosti booleovské formule

Dána Booleova formule F o n proměnných, $X=(x_1, x_2, ... , x_n)$ v konjunktivní normální formě (CNF). Dále jsou dány celočíselné kladné váhy těchto n proměnných $W=(w_1, w_2, ... , w_n)$. Nalezněte přiřazení $Y=(y_1, y_2, ... , y_n)$ proměnných $x_1, x_2, ... , x_n$, takové, že $F(Y)=1$ a součet S vah proměnných nastavených do 1 je maximální.
Omezte se na vážený *3-SAT* problém, kde každá klauzule má právě 3 proměnné. Složitost problému je stejná, implementace a klasifikace instancí jsou jednodušší.

Simulované ochlazování

Simulované ochlazování je jedna z pokročilých iterativních metod použitelných pro řešení vážené splnitelnosti booleovské formule. Metoda nezaručuje správný výsledek, zato je velmi rychlá. Principiálně se podobá metodě *Hill climbing*, avšak při náhodném prohledávání sousedů ve stavovém prostoru může přijmout i horší stav. To je dáno pravděpodobností přijmutí horšího stavu, která je také ovlivněna mírou zhoršení nového stavu oproti starému stavu a aktuální *teplotou*. Výhodou této metody je také její parametrizovatelnost, kterou lze ovlivnit délku výpočtu / kvalitu výpočtu. Pseudokód mé implementace je následující:

```
stav = počáteční_stav
nejlepší_stav = None
while(not stav.frozen()) {
     while(not equilibrium()){
          stav = nový_stav()
          if (stav.lepší_než(nejlepší_stav)){
               nejlepší_stav = stav
          }
     }
     cool()
}
```

V průběhu výpočtu figuruje tzv. t*eplota*, která ovlivňuje počet navštívených stavů. Také s klesající teplotou klesá i pravděpodobnost přijetí horších stavů. Teplota je vymezena konstantou počáteční_teplota a metodou frozen(), která rozhoduje, zda je teplota dostatečně nízká, aby byl výpočet ukončen. Teplota je také ovlivňována metodou cool(), která určuje rychlost snižování

teploty. Teplota není snižována po každém vyzkoušeném stavu a snižuje se po dávkách vyzkoušených stavů. Počet těchto stavů určuje metoda equilibrium().
Metoda stav.lepší_než() určuje, který stav je hodnotnější. To je určeno přes tzv. cost() funkci, která určuje hodnotu stavu. Implementace této funkce má 2 hlavní přístupy. První přístup ohodnotí každý stav, který není splněný vždy hůře, než splněné stavy. Splněné stavy jsou dále hodnoceny podle optimalizačního kritéria. Druhý přístup je přístup tzv. *relaxovaný*, který penalizuje podle určitého kritéria nesplněné stavy.
U metody simulovaného ochlazování je také velmi důležitý spojitý průchod stavovým prostorem. S tím souvisí způsob určení nejbližšího souseda definující přechodové hrany mezi stavy v grafu stavového prostoru. Způsob určení nejbližšího souseda musí být navržen tak, aby byl stavový prostor spojitý a zároveň našel v rozumném čase stav, který je co nejpodobnější původnímu stavu co se týče její hodnoty. Dalším faktorem, který může hrát roli je určení počátečního stavu. To jsem si ověřil při experimentování nad předchozí úlohou, kde v problému batohu výrazně zvyšovalo efektivitu použití *greedy* řešení jako počáteční stav, oproti použití náhodného stavu.

Technické řešení

Řešení bylo rozděleno do 2 částí: řešič a analyzátor.

Řešič řeší 1 instanci problému a jako výstup je řešení, čas výpočtu, počet splněných klauzulí. Při spuštění s parametrem `-w` vrací také hodnotu `cost()` v jednotlivých iteracích běhu výpočtu a poměr splněných klauzulí v jednotlivých iteracích. Tyto informace jsou využity ve White Box fázi. Samotná implementace je napsaná v *Pythonu* a nachází se v souboru `simulated_annealing.py`. Program lze spustit pomocí příkazu `python3 ./simulated_annealing.py`. Nápověda jak spustit program se zobrazí pomocí `python3 ./simulated_annealing.py -h` .
Analyzátor je napsaný v *Python notebooku* pro možnost rychlého upravování. Analyzátor obsahuje White Box analytickou část a Black Box část.

Experimenty

Při řešení problému vážené splnitelnosti booleovské formule jsem se spíše snažil , aby formule byla splněná i přestože neoptimální, než aby měla vysokou váhu s pár nesplněnými klauzulemi. To však v únosném čase. Při větších instancích jsem již ustupoval kvalitou tak, aby výpočet netrval dlouho (cca 2s limit).
Experimentovalo se s parametry počáteční_stav, `počáteční_teplota`, `equilibrium()`, `frozen()`, `cool()` a testovali se různé implementace metody cost() a hledání nejbližšího souseda. Měřila se kvalita výsledků (počet splněných klauzulí a váha stavu) a rychlost výpočtu s těmito parametry.

Experimenty byly testovány na zařízení s následujícími parametry:

- Operační systém: Ubuntu 21
- RAM: 8GB DDR4
- Procesor: i5-6300HQ 2.3GHz

Průběh White Box fáze

White Box fáze testování probíhala na sadách *wuf-M1, wuf-R1* a *wuf-A1* na velikostech instancí, které mají uvedené optimální řešení. Testovalo se na 20 náhodně vybraných vzorcích z dané sady a velikosti.

Pozn.: Pro následující měření zachycené grafy bylo použito následující výchozí nastavení pokud není řečeno jinak: počáteční stav – náhodný, výchozí teplota – použití zpětnovazebního řízení, konstanta frozen – zastavení po převaze jedné hodoty v n posledních stavech, C – 0,7 a equilibrium – 200. Cost typu 2 (viz níže) a náhodné hledání nejbližšího souseda.

Hledání nejbližšího souseda

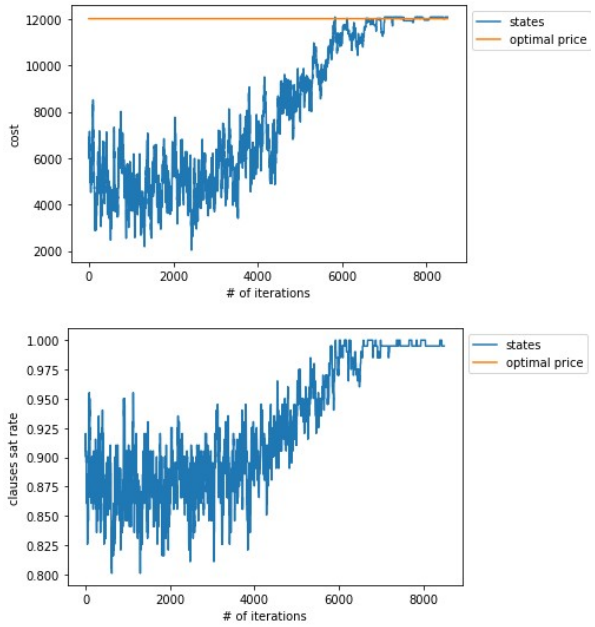
Hledání nejbližšího souseda bylo zpočátku implementováno jako prohození pravdivosti náhodné proměnné (*bitflip*). Dále jsem implementoval hledání prohozením pravdivosti náhodné proměnné v náhodné nesplněné klauzuli. Pokud byl stav již splněný, provedl se náhodný bitflip. Naproti očekávání byla tato metoda však ve *White Box* fázi méně účinná, než naprosto náhodné prohození. Myslel jsem, že stavový prostor je takhle „málo spojitý“. Proto jsem zkusil také tuto implementaci proložit *„operátorem katastrofy“*, který v 1% navrhne jako nový stav naprosto náhodný stav. To také nepomohlo.

Cost()

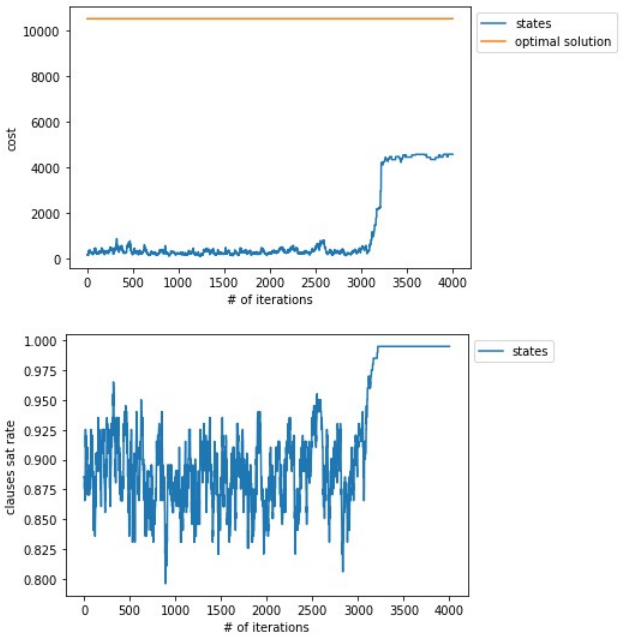
Tato funkce ohodnocuje stav dané konfigurace v rámci instance. Testovali se následující vzorce pro výpočet hodnoty `cost()`:

1. $[\textit{váha stavu}] \times [\textit{poměr splněných klauzulí}]$
2. $\dfrac{[\textit{váha stavu}]}{[\textit{počet nesplněných klauzulí}]^2 + 1}$
3. $[\textit{váha stavu}] \times [\textit{poměr splněných klauzulí}]^2$

Pro první a třetí uvedený vzorec byly upřednostňovány výsledky, které nebyly řešením, ale měli vyšší váhu než optimální řešení. Proto jsem tedy vyzkoušel 2. vzorec, který klade větší důraz na splnitelnost formule.
Ještě lepší výsledky ve *White Box* fázi však dávala kombinace třetího a druhého vzorce. První vzorec byl použit pro přechod mezi stavy a druhý vzorec byl použit pro ukládání dosavadního maxima a nejlepší nalezené hodnoty. Na grafech níže je vidět porovnání použití kombinace vzorců 3 + 2 a použití pouze vzorce 2. Horní grafy zobrazují cenu v průběhu iterací a dolní grafy zobrazují poměr splněných klauzulí. Levá dvojice zachycuje využití kombinace a pravá dvojice zachycuje použití vzorce 2 na všechno (i vybírání nejlepšího stavu).



Obrázek 1: kombinace 3 + 2



Obrázek 2: pouze vzorec 2.

Na malé testovací sadě o 20 prvcích měla kombinace relativní chybu **0,99** a použití pouze vzorce 2 mělo **0,95**. Kombinace měla však malou úspěšnost na sadách Q1 a R1.

Počáteční stav

Počáteční stav byl zvolen náhodně. Nalezení přibližného řešení (nebo neoptimálního řešení) by nemuselo být příliš k užitku v případe váženého SAT problému, jelikož stav může být v hlubokém lokálním maximu. Pro nalezení optimálního řešení by proto musela být nastavena vysoká počáteční teplota.

Počáteční teplota

Při nedostatečné teplotě je méně pravděpodobné nalezení globálního maxima a nalezení optimálního řešení. Teplota totiž ovlivňuje pravděpodobnost přijetí horšího stavu. Příliš vysoká hodnota je přebytečná a hodnoty velmi oscilují po příliš dlouhou dobu a nemusí tak dojít k fázi intenzifikace

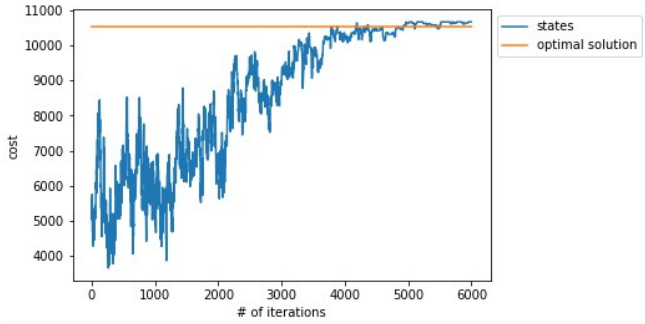
Pro pilotní testování byla nejdříve nastavena pevná hodnota (1000). Pro nalezení ideální počáteční teploty jsem potom použil zpětnovazební řízení. Jako hranici jsem nastavil hodnotu 0,5. Tedy pravděpodobnost přijetí horšího stavu musí být vyšší rovna než 0,5. Pro zrychlení nalezení této teploty jsem iniciální teplotu nastavil na hodnotu *`[součet_všech_vah_proměnných / počet_proměnných]`*. Tu jsem následně zvyšoval zdvojnásobováním. Velikost pravděpodobnostního vzorku jsem nastavil na *počet_proměnných.* Použití této metody výrazně urychlilo běh programu - zkrátilo fázi diverzifikace (potvrzené porovnání grafů a doby běhu).

Equilibrium()

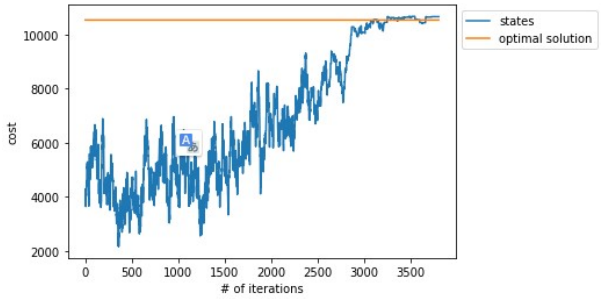
Při vyšším equilibriu je i vyšší doba výpočtů. Zároveň však při nízkém equilibriu je větší tendence zůstat v lokálním maximu. Je patrné, že s rostoucím equilibriem roste také počet iterací a rozsah oscilací stavů.

Nejprve bylo `equilibrium()` implemetováno pevně jako 200 pro všechny velikosti instancí. Tato hodnota byla zjištěna experimentálně ve White Box fázi a byla určena tak, aby kvalita výsledku byla uspokojivá a doba výpočtu trvala únosně dlouho (např. u velikosti instancí 50 do 1s). To ale po Black Box fázi nevracelo dostatečně kvalitní výsledky a byla proto stanoven výpočet equilibria $\lfloor \textit{počet proměnných} \rfloor \times 10$. Tento vzorec vracel mírně kvalitnější výsledky (viz Black

Box fáze). Na konstantu 10 se opět přišlo experimentálně ve White Box fázi, kde se brala opět v potaz doba výpočtu a kvalita výpočtu. Původně byla preferována konstanta 5, ale nepřinášela příliš přidané hodnoty. Na následujících grafech je průběh stavů a počet iterací instance `wuf50-123` ze sady M1.



Obrázek 3: equilibrium dle vzorce 500

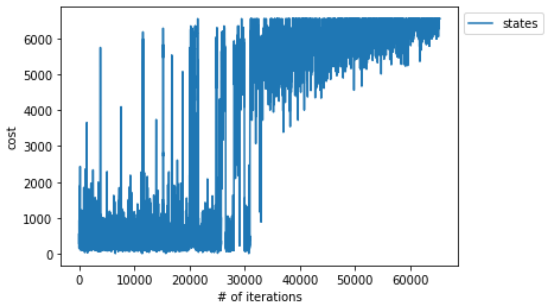


Obrázek 4: pevná konstanta 200

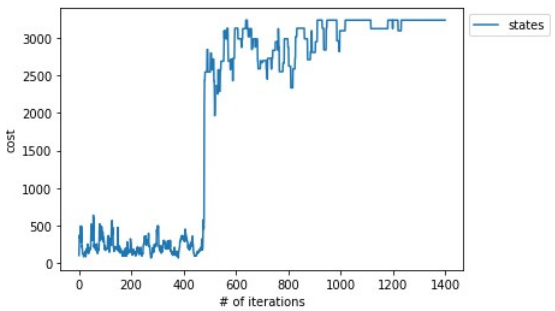
Cool()

Metoda `cool()` určuje míru snížení teploty. Ta je ovlivněna konstantou C, která typicky nabývá hodnot od 0,8 do 0,99. Je implementována tak, že násobí aktuální teplotu touto konstantou. Teplota tak klesá rychleji při vyšších teplotách a zároveň klesá výrazně rychleji při nižším C. To přímo ovlivňuje celkový počet iterací a tedy i dobu výpočtu (a z experimentálních zjištění ji nejvíce ovlivňuje). Na druhou stranu při vyšší hodnotě C je i vyšší pravděpodobnost nalezení optimálního řešení.

Funkce `cool()` je implementována jako $teplota \times c$, kde $c \in (0, 1)$. Při nastavení konstanty c na hodnot 0,99 je výpočet neúnosně dlouhý, ale kvalitní výsledky (100% opt.řeš. / 14s). Při hodnotě 0,5 je výpočet velmi rychlý, zato méně přesný (97% opt. Řeš. / 0,4 s). Jako optimální se ukázala hodnota 0,6 až 0,8. Konkrétně hodnota 0,6 má ~99% úspěšnost a trvá 0,3s na instancích velikosti 20 v sadě M1. Vyšší konstanta c totiž vykazuje při desetinásobném zvýšení času pouze jednotkové zlepšení kvality na vyšších velikostech instancí. Na grafech níže stojí za povšimnutí zejména počet iterací potřebných pro ukončení výpočtu.



Obrázek 5: c = 0,99

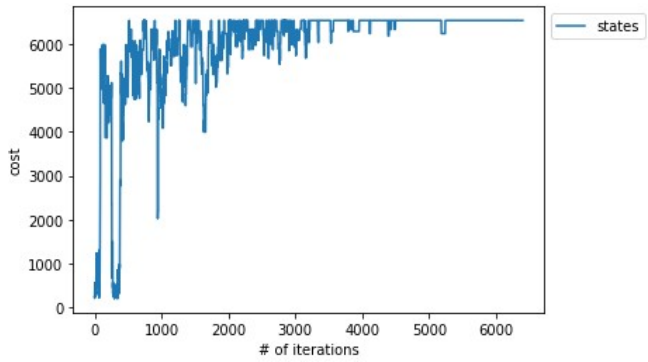


Obrázek 6: c = 0,6

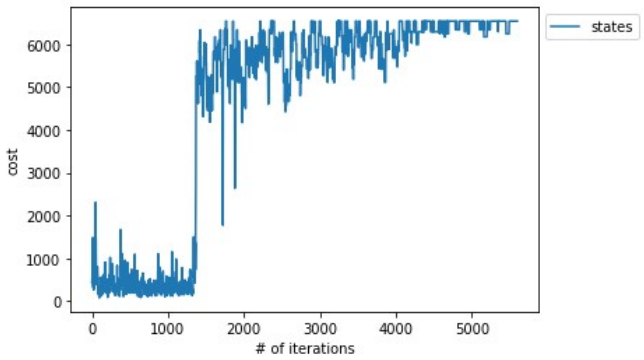
Frozen()

`Frozen()` určuje ukončení výpočtu. Je na místě odhadnout ideální konstantu. Pozdější ukončení zapříčiňuje neměnný stav, kdy je metoda „jistá svým výsledkem“ a příliš brzké ukončení skončí výpočet ve fázi, kdy je velká pravděpodobnost, že řešič najde ještě lepší řešení.

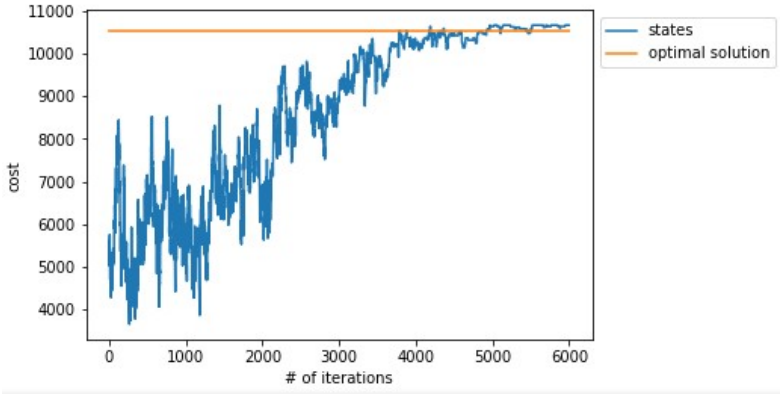
Implementace `frozen()` byla pro pilotní testování nejprve naprogramovaná jako dolní mez teploty. Výpočet se tedy zastavil, pokud se teplota snížila pod danou konstantu (100). Další verzí bylo ukončení výpočtu, pokud posledních 100 stavů bylo téže hodnoty. To velmi zrychlilo výpočet a ve *White Box* fázi vracely téměř vždy optimální výsledky. První 2 grafy zachycují vývoj ceny stavu instance o velikosti 20 proměnných ze sady *M1*. Graf vlevo zachycuje implementaci `frozen()`, kde se výpočet zastaví po 1000 stavů téže hodnoty. Graf vpravo pak po 100 stavech téže hodnoty. Tento způsob však neukončuje výpočet při vyšší teplotě (kdy se přijímá stav s nižší hodnotou) což je vidět na obou grafech. Byla proto naimplementována další verze, kdy se výpočet ukončí, jestliže v posledních 100 stavech převažuje jejich maximální hodnota. Hodnota 100 byla určena experimentálně. A dle grafů a výsledků z *White Box* fáze se také ukončuje rychleji. To zachycuje graf dole.



Obrázek 7: posledních 1000 iterací jsou stejné hodnoty



Obrázek 8: posledních 100 iterací jsou stejné hodnoty



Obrázek 9: posledních 100 převažuje maximální hodnota

Průběh Black Box fáze

Black Box fáze byla testována na všech instancích a na všech sadách. Měřila se opět *relativní chyba*, *čas výpočtu* a *splněnost klauzulí*. Na těchto měřeních se získávaly průměrné hodnoty (pro představu standardní hodnoty), a odchylky těchto hodnot (pro získání údaje různorodosti těchto hodnot). Byly měřeny také maximální a minimální hodnoty, ty však neuvádím, jelikož nejsou příliš zajímavé (překvapivé). Konečně tyto hodnoty pomohou určit, jak kvalitní toto nastavení je.

Nastavení 1

První zkoušené nastavení mělo následující parametry: *Počáteční stav – náhodný, počáteční teplota – počítána zpětnovazebním řízením, equilibrium – konstanta 200, funkce frozen – zastavení po převaze maximálního prvku ve 100 předchozích stavech, C – 0,7, cost – typu 2 (viz výše), nejbližší soused – náhodný bitflip.* Výsledky měření jsou zachyceny níže v tabulce.

	Prům. rel chyba	*Odchylka rel. chyby*	***Průměrný čas výpočtu (s)***	*Odchylka času výpočtu*	*Průměr splněných klauzulí*	*Odchylka splněné klauzule*
wuf20-88-A1	**0,11**	0,017	**0,2**	0,01	0,999	~ 0
wuf20-91-A1	**0,05**	0,008	**0,2**	0,01	0,978	~ 0
wuf100-430-A1	/	/	**2,4**	0,2	0,981	~ 0
wuf20-78-M1	**0,005**	0,008	**0,2**	0,01	1	0
wuf50-201-M1	**0,05**	0,002	**0,7**	0,04	0,999	~ 0
wuf75-301-M1	/	/	**1,4**	0,19	0,998	~ 0
wuf20-78-N1	**0,006**	0,001	**0,2**	0,01	0,999	~ 0
wuf50-201-N1	**0,04**	0,004	**0,7**	0,03	0,999	~ 0
wuf75-301-N1	/	/	**1,4**	0,2	0,999	~ 0
wuf20-78-Q1	**0,08**	0,011	**0,2**	0,01	0,974	~ 0
wuf50-201-Q1	**0,25**	0,025	**0,9**	0,08	0,988	~ 0
wuf75-310-Q1	/	/	**1,7**	0,17	0,988	~ 0
wuf20-78-R1	**0,08**	0,01	**0,2**	~ 0	0,999	~ 0
wuf50-201-R1	**0,23**	0,05	**0,9**	0,06	0,986	~ 0
wuf75-310-R1	/	/	**1,7**	0,2	0,988	~ 0

Z tabulky lze vyčíst, usoudit a okomentovat následující:

- Se zvyšujícími se počty proměnných narůstá průměrný čas výpočtu, průměr splněných klauzulí i průměrná relativní chyba. To samé platí i pro jejich odchylky.
- Sady Q1, R1 a A1 se liší od sad M1 a N1 v chybovosti a lze tak usoudit (a potvrdit komentáře k sadám ze zadání), že tyto sady jsou speciálně upravené, a existuje nejspíš mnoho řešení, u kterých je však těžké najít optimum.
- Rychlost výpočtu je uspokojivá zejména u výpočtu instancí velikosti 20 a 50. Vyšší počet instancí již překračuje 1s.
- Odchylka relativní chyby je u všech menší než 5%, což dává jasnou informaci, že míra chyby se příliš neliší napříč instancemi. U sad M1, N1 je to dokonce pod 1%.
- Splněnost napříč všemi sadami uspokojivá, avšak šla by zlepšit.

Z tohoto výsledku by šlo zlepšit kvalitu řešení na úkor času výpočtu (zejména u vyšších velikostech instancí). To jsem se snažil upravit v druhém měření.

Nastavení 2

Druhým zkoušeným nastavením bylo stejné jako předchozí, ale equilibrium odpovídalo hodnotě $počet\ proměnných \times 10$. Equilibrium se tedy lišilo, když $počet\ proměnných \neq 20$. Výsledky měření jsou v následující tabulce.

	Prům. rel chyba	*Odchylka rel. chyby*	***Průměrný čas výpočtu***	*Odchylka času výpočtu*	*Průměr splněné klauzule*	*Odchylka splněné klauzule*
wuf20-88-A1	**0,13**	0,02	**0,2**	0,01	0,999	~ 0
wuf20-91-A1	**0,06**	0,01	**0,2**	0,01	0,998	~ 0
wuf100-430-A1	/	/	**10**	7,5	0,994	~ 0
wuf20-78-M1	**0,005**	0,001	**0,2**	0,01	0,999	~ 0
wuf50-201-M1	**0,01**	0,001	**1,5**	0,2	0,999	~ 0
wuf75-301-M1	/	/	**4,4**	2	0,999	~ 0
wuf20-78-N1	**0,006**	0,001	**0,2**	0,01	1	0
wuf50-201-N1	**0,02**	0,001	**1,6**	0,6	0,999	~ 0
wuf75-301-N1	/	/	**4,4**	2	0,999	~ 0
wuf20-78-Q1	**0,09**	0,011	**0,2**	0,01	0,999	~ 0
wuf50-201-Q1	**0,20**	0,05	**2,0**	0,05	0,991	~ 0
wuf75-310-Q1	/	/	**5,3**	2,4	0,991	~ 0
wuf20-78-R1	**0,08**	0,01	**0,2**	0,01	0,999	~ 0
wuf50-201-R1	**0,23**	0,04	**2,0**	0,5	0,991	~ 0
wuf75-310-R1	/	/	**5,0**	1,9	0,991	~ 0

Tato tabulka ukazuje podobné výsledky jako předchozí nastavení, ale v zásadě výpočet u instancí, kde $počet\ proměnných > 20$ je výpočetně náročnější a kvalita se mírně zvýšila. Konkrétně u instancí velikosti 50 trvala ~1,5s (oproti 0,7s) a kvalita se zvýšila jak ve splnitelnosti klauzulí, tak v průměrné chybě.

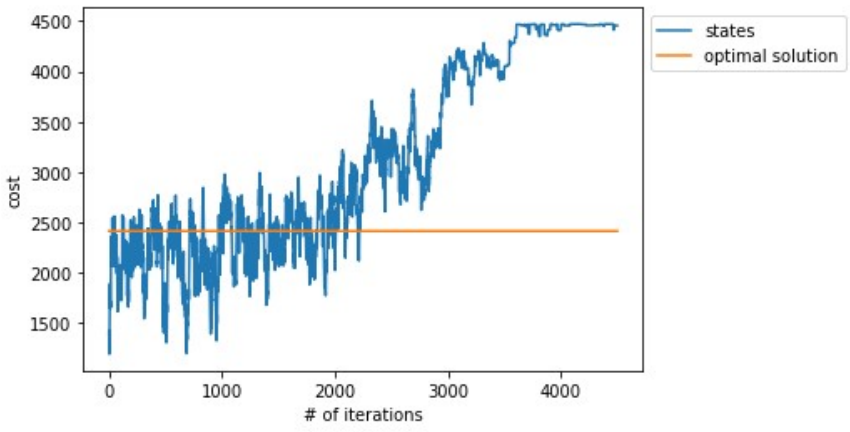
Nastavení 3

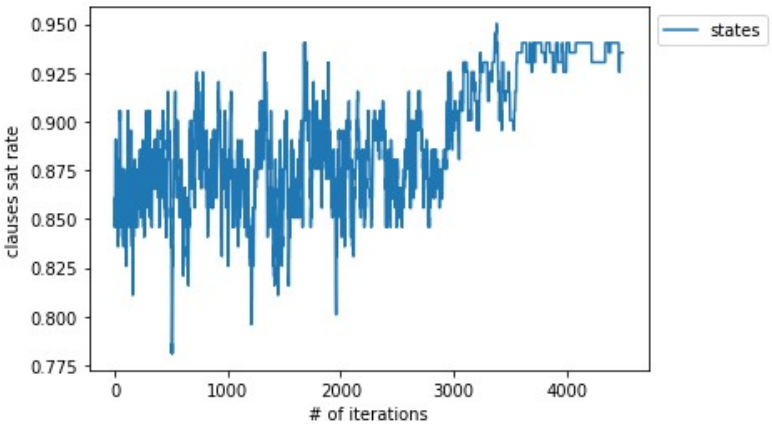
Další test měl následující nastavení: *Počáteční stav – náhodný, počáteční teplota – počítána zpětnovazebním řízením, equilibrium –* $\lceil počet\ proměnných \rceil \times 5$ *, funkce frozen – zastavení po převaze maximálního prvku ve 100 předchozích stavech, C – 0,7, cost – kombinace vzorce 2 a 3 (viz výše), nejbližší soused – náhodný bitflip. Také byl snížen počet vzorků pro získání pravděpodobnosti přijetí horších stavů na konstantu 20. U větších instancí byl vzorek zbytečně velký.*

	Prům. rel chyba	*Odchylka rel. chyby*	***Průměrný čas výpočtu***	*Odchylka času výpočtu*	*Průměr splněné klauzule*	*Odchylka splněné klauzule*
wuf20-88-A1	**0,03**	0,003	**0,2**	0,003	0,994	~ 0
wuf20-91-A1	**0,01**	0,001	**0,2**	0,003	0,996	~ 0

wuf100-430-A1	/	/	**6,1**	1,9	0,989	~ 0
wuf20-78-M1	**0,002**	0,0002	**0,2**	0,003	0,999	~ 0
wuf50-201-M1	**0,005**	0,0001	**1,4**	0,17	0,999	~ 0
wuf75-301-M1	/	/	**3,3**	0,5	0,998	~ 0
wuf20-78-N1	**0,001**	0,0001	**0,2**	0,002	0,999	~ 0
wuf50-201-N1	**0,005**	0,0001	**1,4**	0,1	0,998	~ 0
wuf75-301-N1	/	/	**3,3**	0,7	0,998	~ 0
wuf20-78-Q1	**0,20**	0,03	**0,2**	0,003	0,978	0,0002
wuf50-201-Q1	**0,74**	0,06	**1,6**	0,15	0,947	0,0001
wuf75-310-Q1	/	/	**3,8**	0,95	0,942	~ 0
wuf20-78-R1	**0,28**	0,04	**0,2**	0,003	0,978	0,0002
wuf50-201-R1	**0,73**	0,07	**1,6**	0,14	0,950	~ 0
wuf75-310-R1	/	/	**3,6**	0,7	0,943	~ 0

Výsledky tohoto měření byly velmi překvapivé. Na sadách A1, M1, N1 mělo toto nastavení výrazné zlepšení v průměrné relativní chybě. Chyba se snížila cca o polovinu u sad M1 a N1 a u sady A1 se snížila až pětinásobně. Časově se také sady zlepšily na těchto sadách. Například instance typu *wuf75-301-M1* se zlepšila z 4,4 na 3,3s. Na druhou stranu se zhoršila relativní průměrná chyba u sad Q1 a R1 a to až z 0,23 na 0,73. Snížila se rovněž splněnost klauzulí těchto sad a to až z 0,991 na 0,942. Toto zhoršení chybovosti bylo reálně znamenalo vrácení vyšší váhy a méně splněných stavů.
Po analýze ve *White Box* fázi tohoto jevu bylo zjištěno, že funkce vybírající nejlepší stav byla moc benevolentní vůči nesplněným klauzulím a vyplatilo se tedy vzít méně splněnou formuli, která měla mnohem větší hodnotu i přes značnou penalizaci. Na následujícím grafu je instance uf50-0470 ze sady Q1, jejíž výsledek vah byl 4457 se splněnými 191 klauzulemi z 201. Optimálním řešením bylo 2415, tedy skoro dvojnásobně méně. Z grafu lze vidět, že se metoda snaží konvergovat spíše k jakémusi kompromisu splnitelnosti a váhy stavů a nepreferuje tak počet splněných klauzulí. Řešením by mohlo být například normalizace nebo upravení `cost()` funkce.





Nastavení 4

Poslední experiment se stejným nastavením, akorát změnou cost funkce na $\frac{váha\,stavu}{2^{počet\ nesplněných\ klauzulí}}$ a equilibrium opět na $[počet\ proměnných]\times 10$.

To mělo dosavadně nejlepší výsledky co se týče splnitelnosti klauzulí. V relativní chybovosti bylo horší pro sady M1, N1 a A1. Pro sady Q1 a R1 o kousek nejlepší (wuf50-201-R1 s chybovostí 0,18). V čase výpočtu bylo však suverénně nejhorší v pro $počet\ proměnných>50$ kde pro 75 trval až 15s. Což není únosné. Detailní výsledky jsou v souboru `ex4.png` .

Optimální nastavení

Jako optimální nastavení jsem zvolil Nastavení 2, kde je obecně vysoká splněnost klauzulí, relativně nízká chybovost zejména pro sady M1, N1 a A1. Čas je také uspokojivý pro menší velikosti instancí. Pro vyšší už je v tomto ohledu toto nastavení horší.

Testování robustnosti

Jelikož je simulované ochlazování randomizovaný algoritmus, je třeba ověřit, zda vrací konzistentní výsledky. Pro ověření této skutečnosti jsem vybral náhodné instanci *wuf50-0123.mwcnf, wuf20-0123.mwcnf* a ty jsem spustil *20x.* Výsledkem byly data na následujícím obrázku. Z těch lze vyčíst podle jejich odchylek, že se liší nejvíce v čase výpočtu, avšak v kvalitě výpočtu (poměru splněných klauzulí a poměru vah ku optimálnímu řešení) se příliš neliší zejména u nižších velikostech.

```
Avg. sat rate: 1.0
Avg. weight rate: 1.0
Avg. exec time: 0.244
Var sat rate: 0.0
Var weight rate: 0.0
Var exec time: 0.005764
Max sat rate: 1.0
Max weight rate: 1.0
Max exec time: 0.44
Min sat rate: 1.0
Min weight rate: 1.0
Min exec time: 0.11
```

Obrázek 10: wuf20-0123.mwcnf

```
Avg. sat rate: 0.9972636815920397
Avg. weight rate: 0.9176091081593928
Avg. exec time: 2.288
Var sat rate: 1.3551644761268164e-05
Var weight rate: 0.0037655136842029467
Var exec time: 0.13678600000000002
Max sat rate: 1.0
Max weight rate: 0.9844402277039849
Max exec time: 3.43
Min sat rate: 0.9900497512437811
Min weight rate: 0.7896584440227704
Min exec time: 1.84
```

Obrázek 11: wuf50-0123.mwcnf

Závěr

Pro řešení váženého problému splnitelnosti booleovské formule jsem použil metodu simulovaného ochlazování. Provedl jsem experimenty zkoumající vliv parametrů na kvalitu a rychlost metody. Implementoval jsem několik verzí řešiče, popsal je a otestoval na testovacích sadách. Kvalitní výsledky se dařilo dosahovat zejména u instancí do velikosti 50 přičemž doba výpočtu byla únosná (do 1,5s). Velký problém však dělaly sady Q1 a R1, které těžko nalézaly řešení pod únosný čas (5s).