5. úloha – Řešení problému vážené splnitelnosti booleovské formule pokročilou iterativní metodou

Ondřej Červenka

29. ledna 2016

1 Specifikace úlohy

Je dána booleovská formule F s n proměnnými $X = \{x_1, x_2, \ldots, x_n\}$, v konjuktivní normální formě. Dále je každé proměnné x_i, $i \in \{1, \cdots, n\}$ přidělena váha w_i.

Hledáme takové ohodnocení proměnných $\{y_1, y_2, \ldots, y_n\}$, $y_i \in \{0, 1\}$ pro které bude formule F splněna a součet vah proměnných ohodnocených jedničkou bude maximální.

V této úloze budeme řešit variantu 3SAT, kdy jedna klauzule obsahuje právě 3 literály. Formule s n proměnnými a k klauzulemi má tedy tvar:

$$(a_1 \vee b_1 \vee c_1)_1 \wedge (a_2 \vee b_2 \vee c_2)_2 \wedge \ldots \wedge (a_k \vee b_k \vee c_k)_k$$

kde $a_1 \ldots a_k, b_1 \ldots b_k, c_1 \ldots c_k \in X$.

2 Simulované ochlazování

Jako pokročilou iterativní metodu jsem zvolil simulované ochlazování. Při tomto algoritmu nevolíme vždy momentálně nelepší tah (jako například u greedy heuristiky), ale náhodně zvolený tah[1]. Pokud tento tah vylepší dosud nalezené řešení, je proveden. Pokud ne, je proveden pouze s nějakou pravděpodobností p.

Tato pravděpodobnost závisí jednak na kvalitě tahu (jak moc zhorší dosud nalezené řešení) a jednak na paramteru T (teplota). Jako T je zvolena

[1] V našem případě tah představuje změnu ohodnocení jedné proměnné oproti současnému stavu.

nějaká počáteční hodnota, která se postupně snižuje (chladnutí), podle zvoleného parametru. Vyšší teplota pak zvyšuje pravděpodobnost provedení nezlepšujícího tahu.

Díky tomu, že algoritmus dovoluje s nějakou pravděpodobností provést i zhoršující kroky, je možné se dostat z lokálního maxima a nalézt lepší řešení.[1]

2.1 Kostra algoritmu

Kostra algoritmu je stejná jako v případě problému batohu, jediný rozdíl je v cenové funkci. U problému batohu cenová funkce vracela součet cen předmětů obsažených v dané konfiguraci a v případě přetížení batohu 0.

Při SAT problému cenová funkce vrací součet vah proměnných, které jsou v dané konfiguraci ohodnocney jedinčkou a 0 v případě, že pro toto ohodnocení není formule splněna. [1]:

```
//zacneme z nahodneho stavu
current = get_random_state()
T = T_i //pocatecni teplota
loop
        //chladnuti podle zvoleneho koeficientu
        T = T * cooling

        if T < frozen then //konec pri zamrznuti
                return current

        next = get_random_neighbour(current)

        EΔ = next.value - current.value

        if EΔ > 0 then
                current = next

        //pokud je EΔ < 0, tah se provede
        //s pravdepbodonosti e^(EΔ/T)
        else
                if p(e^(EΔ/T)) then
                        current = next
end loop
```

Jako parametry heuristiky je tedy třeba zvolit počáteční teplotu T_i, dále koeficient chladutí $cooling < 1$ a bod zamrznutí $frozen$. Nastavení těchto parametrů nám ovlivní jak dobu běhu, tak přesnost heuristiky.

Algoritmus je randomizovaný, výsledky jednotlivých běhů se tedy pro stejné instance a paramtery mohou lišit.

3 Měření

Vzhledem k tomu, že v tomto případě nemáme k dispozici předpočítané výsledky, je pro zkoumání chyby heuristiky nutné využít nějaký exaktní algoritmus. Ten nám pravděpodobně nedovolí spočítat příliš velké (ve smyslu počtu proměnných) instance, ale umožní nám lépe prozkoumat chování heuristiky na menších instancích.

Na malých instancích tedy budeme zkoumat především relativní chybu heuristiky. Na větších instancích nás pak bude zajímat vývoj kvality řešení při změně nastavení parametrů heuristiky a poměru klauzulí ku proměnným (o vlivu tohoto poměru více v sekci 3.2).

3.1 Podmínky měření

Algoritmus byl implementován v C a kompilován pomocí gcc 5.3.1. Při kompilaci nebyly použity žádné optimalizační přepínače. Program byl zkompilován a spouštěn na operačním systému GNU/Linux (Fedora) 64bit s verzí jádra 4.2.8. Procesor počítače je Intel Core i7-4510U s frekvencí 3.1 GHz.

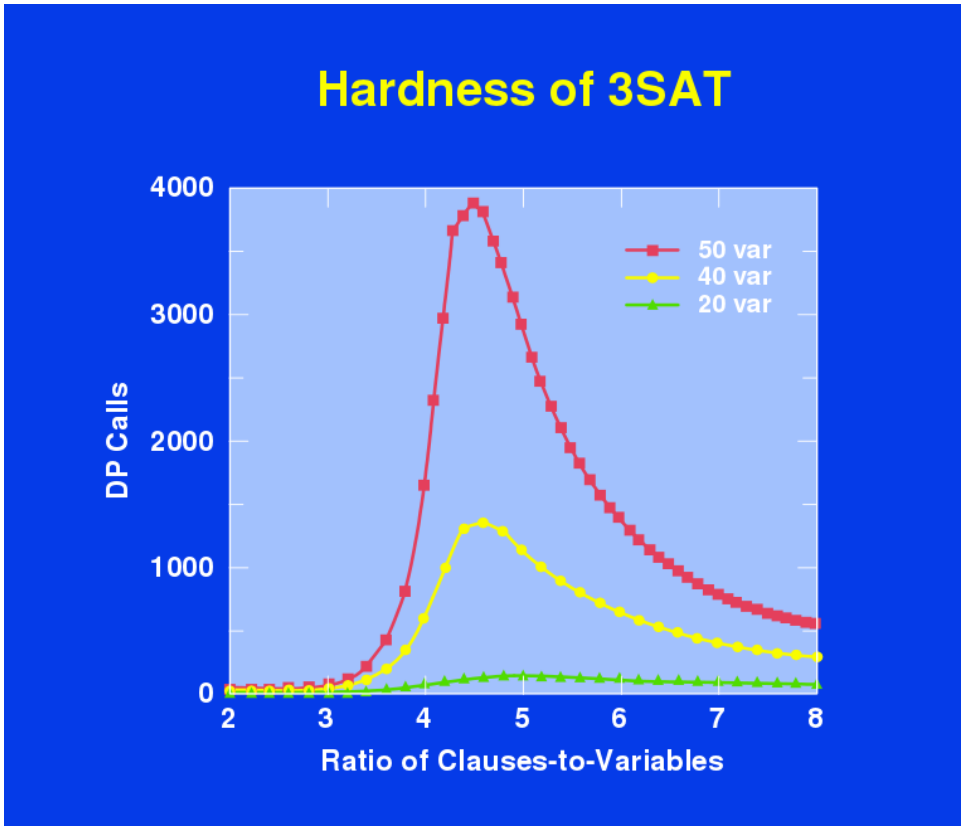
3.2 Volba instancí

Na obtížnost instance má kromě počtu proměnných n vliv také poměr počtu klauzulí k, tedy $r = k/n$ [2]. Tento vliv můžeme vidět na obrázku 1.

Existuje tedy nějaký kritický poměr $r' = k/n$, pro který pravděpodobnost vygenerování instance s alespoň nějakým řešením klesá k nule. Pro malé poměry r jsou podmínky splnění formule volnější a počet řešení větší. Jak se r přibližuje k r', počet možných řešení klesá. U instancí s poměrem $r > r'$ je již počet klauzulí příliš omezující, a tyto instance tedy většinou nemají žádné řešení.[3]

Hodnota kritckého poměru r' závisí na zvoleném počtu proměnných. Pro velká n se pohybuje kolem 4.26, pro malá n je o něco vyšší[3]. Vzhledem k tomu, že chceme aby vygenerované instance měly alespoň nějaké řešení, budeme se zaměřovat na ty s poměrem klauzulí ku proměnným nižším než 4.3.

Menší instance, které budeme zároveň počítat exaktním algoritmem, budou mít 18 proměnných. Větší instance budou mít 50 - 100 proměnných. Poměr $r = k/n$ budeme průběžně zvyšovat, abychom viděli, jak si heuristika poradí s obtížnějšími instancemi.



Obrázek 1: Vliv poměru klauzulí a proměnných na obtížnost instance[2]

3.3 Zkoumání relativní chyby heuristiky

3.3.1 Počáteční nastavení parametrů heuristiky a instancí

Na začátku testování jsem vyzkoušel několik náhodně generovaných instancí, abych odhadl vhodné počáteční nastavení parametrů. Ukázalo se, že podle očekávání má na kvalitu výsledku největší vliv rychlost ochlazování. Počáteční parametry heuristiky jsem tedy zvolil takto:

Počáteční teplota	80
Rychlost ochlazování	0.999
Bod zamrznutí	0.01

Tabulka 1: Počáteční nastavení heuristiky

Pro tyto parametry se ukázalo, že již v instancích s poměrem $r = 3.\overline{18}$ má heuristika problém nalézt alespoň nějaké řešení. Zvolil jsem tedy následující parametry instancí:

Počet proměnných n	18
Počet klauzulí k	36, 40, 44, 48
Poměr $r = k/n$	2, $2.\overline{22}$, $2.\overline{44}$, $2.\overline{66}$

Tabulka 2: Počáteční nastavení parametrů instancí

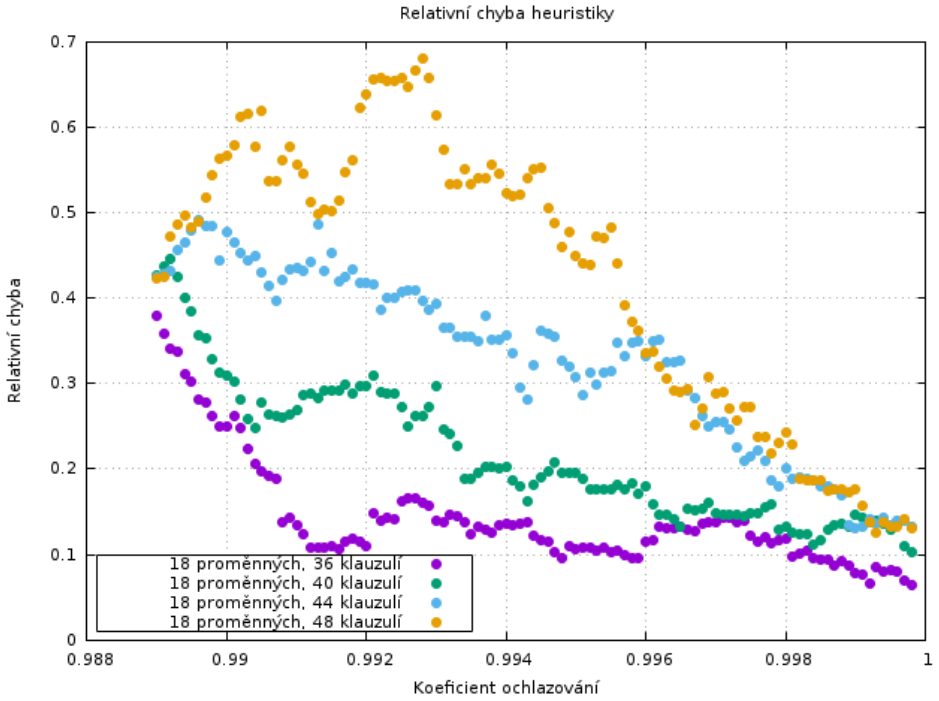
3.3.2 Měření relativní chyby

Nyní tedy budeme zkoumat, jak se bude vyvýjet relativní chyba heuristiky v závisloti na zvolených parametrech. Nejprve prozkoumáme vliv rychlosti ochlazování při počítání instancí s parametry popsanými v tabulce 2.

Průměrnou chybu budeme počítat z 50 náhodně vygenerovaných instancí s těmito parametry. Tím by se měl omezit vliv náhodné složky algoritmu.

Podle očekávání tedy relativní chyba klesá se zvyšujícím se koeficentem ochlazování, což můžeme pozorovat na grafu 2. Můžeme také vidět, že snížení poměru klauzulí k proměnným přineslo předpokládané zlepšení v chybě heuristiky. Chybu vznikající rostoucím poměrem se do jisté míry daří potlačit pomalejším ochlazováním, ovšem za cenu větší časové náročnosti.

Doba výpočtu naopak s klesající rychlostí ochlazování roste (graf 3), stejně jako u problému batohu. Poměr klauzulí a proměnných dobu běhu nijak neovlivnil. Výpočet cenové funkce je sice datově závislý i na počtu klauzulí, ovšem pouze lineárně, a tak je tato závislost utlumena.



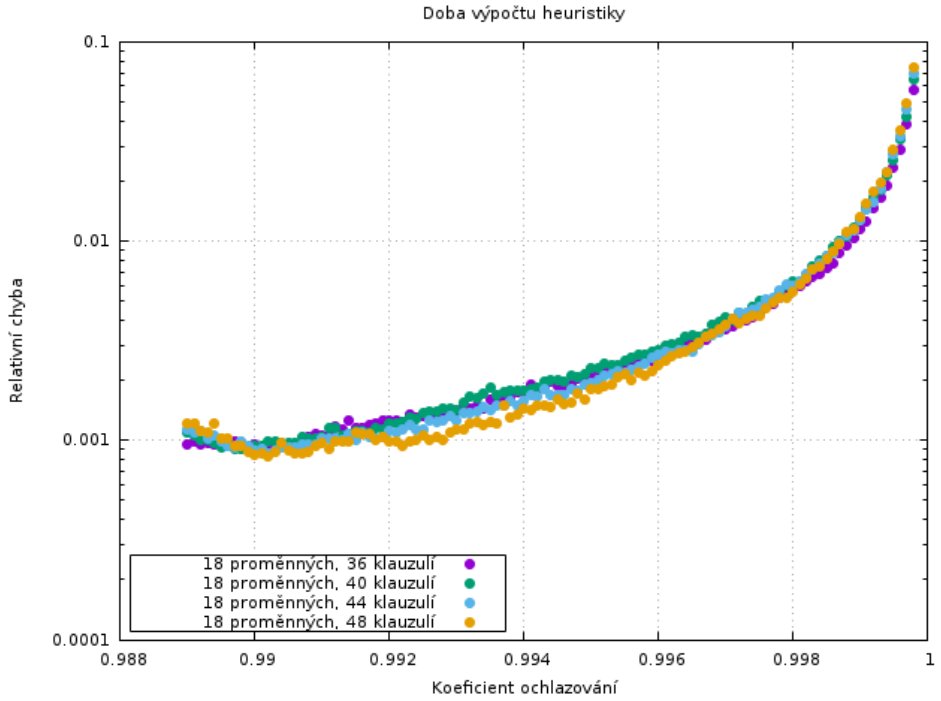
Obrázek 2: Vliv koeficentu ochlazování na relativní chybu u instancí s $n = 18$ a $k = 40$

3.4 Zkoumání vývoje kvality řešení na větších instancích

V sekci 3.3 jsme tedy prozkoumali, jak bude algoritmus reagovat na změnu rychlosti ochlazování a poměru klauzulí k proměnným při počítání malých instancí. Díky malému počtu proměnných jsme mohli najít také optimální řešení a spočítat relativní chybu.

V této sekci budeme zkoumat, jak se algoritmus chová při počítání větších instancí. Vzhledem k tomu, že neznáme optimální řešení nás bude zajímat vývoj kvality nalezeného řešení při zvyšování koeficientu ochlazování a poměru klauzulí k proměnným.

Větší instance jsou zajímavější také z toho důvodu, že vliv poměru klauzulí a proměnných výrazně stoupá (jak je vidět na obrázku 1). Budeme se tedy snažit volit takový poměr, abychom nalezli alespoň nějaké řešení.



Obrázek 3: Vliv koeficentu ochlazování na dobu výpočtu u instancí s $n = 18$ a $k = 40$

3.4.1 Počáteční nastavení parametrů heuristiky a instancí

Vzhledem k tomu, že v tomto případě nás bude zajímat vývoj ceny nalezeného řešení při změně parametrů, nebudeme generovat více instancí, ale v jeden okamžik budeme zkoumat pouze jednu instanci.

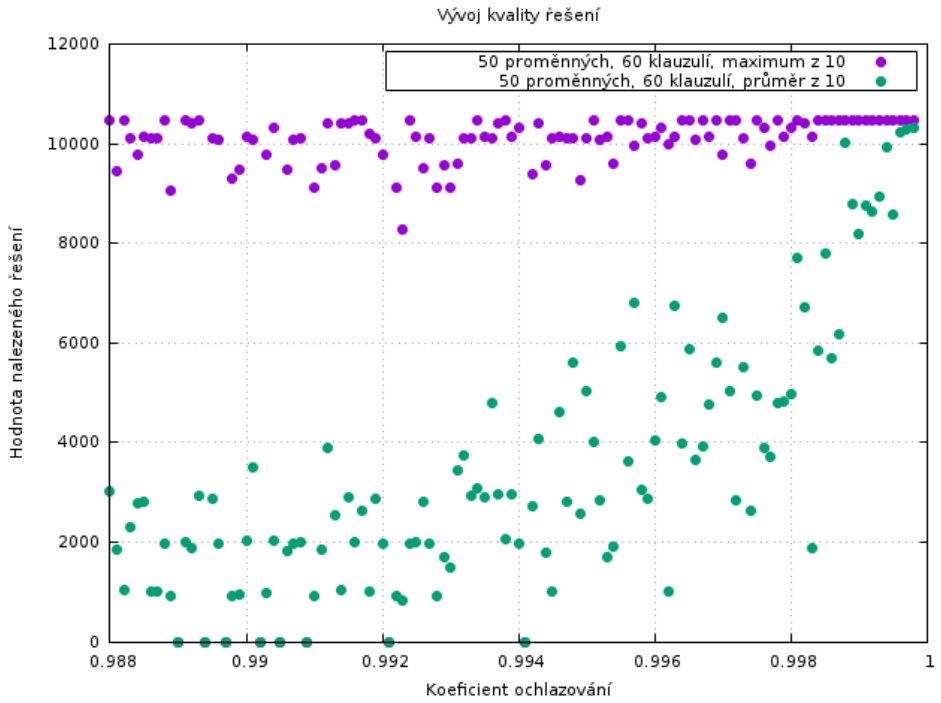
Abychom snížili vliv náhodných jevů, budeme výsledky každé instance a každého nastavení průměrovat z deseti běhů algoritmu.

Nejprve jsem se rozhodl otestovat instanci s $n = 50$ a $k = 60$ (tedy $r = 1.2$), se stejným počátečním nastavením heuristiky jako je v tabulce 1.

3.4.2 Měření vývoje kvality řešení

Ukázalo se, že i při tomto nízkém poměru (1.2) má v některých případech algoritmus potíže nalézt alespoň nějaké řešení, jak je vidět na grafu 4.

Také průměrná kvalita nalezených řešení značne kolísá kvůli většímu poměru nulových řešení v každém z deseti běhů algoritmu.



Obrázek 4: Vývoj kvality řešení při snižování koeficientu ochlazování u instancí s $n = 50$ a $k = 60$

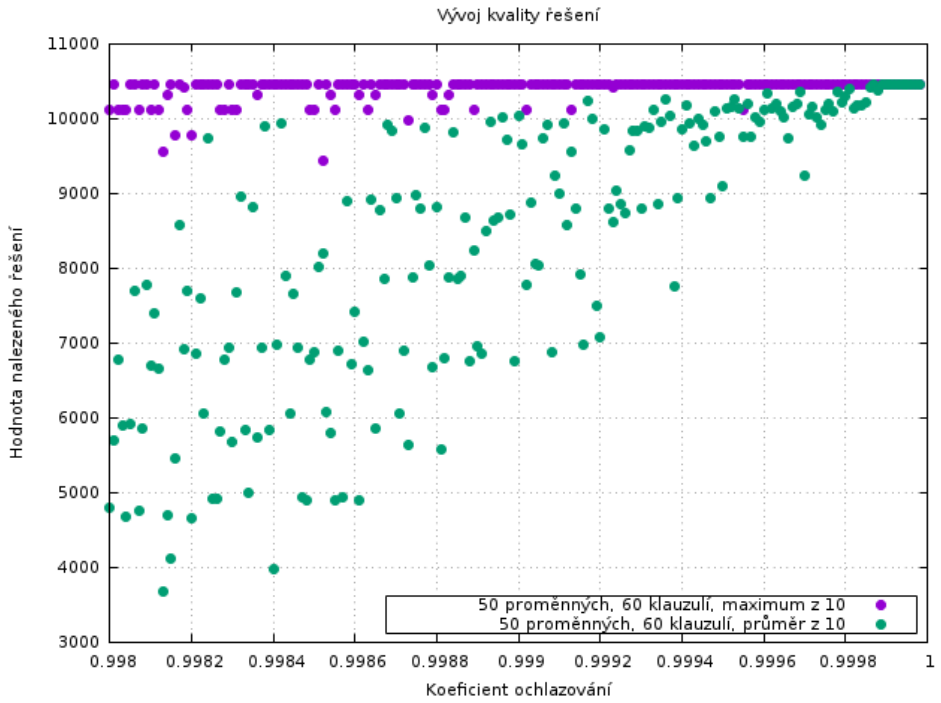
Dále jsem tedy zvýšil zkoumaný koeficient ochlazování (graf 5). Při tomto pokusu se již heuristice dařilo nalézt nějaké řešení časteji a průměrná kvalita řešení tedy stoupla. Hodnota nejlepšího nalezeného však byla stejná jako v předchozím pokusu.

Zde by tedy bylo možná výhodnější zvolit nižší koeficient ochlazování a provést více běhů algoritmu, neboť zvýšením koeficientu vzrostla pouze frekvence nacházení řešení, ale ne kvalita nejlepšího nalezeného řešení.

Každopádně se ukázalo, že vliv poměru klauzulí a proměnných zde působí mnohem silněji než u malých instancí, a je proto potřeba brát v úvahu nejen tento poměr, ale i velikost instance ke které ho vztahujeme.

V dalším pokusu jsem tedy zachoval poměr $k/n = 1.2$, ale snížil počet proměnných na 40. Na grafu 6 je vidět, že v tomto případě se již dařilo algoritmu najít řešení pravidelně, rozptyl v kvalitě řešení je též nižší. To je pravděpodobně opět dáno nižším počtem nulových řešení.

Celkově vývoj kvality nalezeného řešení při pokusech zobrazených na



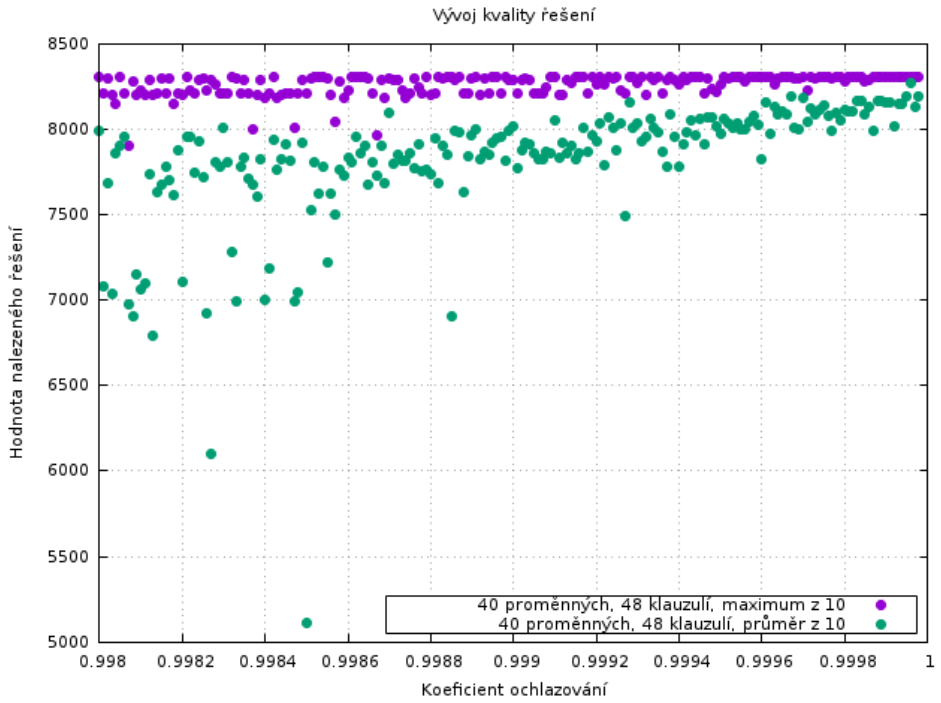
Obrázek 5: Vývoj kvality řešení při snižování koeficientu ochlazování u instancí s $n = 50$ a $k = 60$, (vyšší koeficient)

grafech 4, 5 a 6 odpovídal vývoji zobrazeném na grafu 2 v sekci 3.3.2. Opět se tedy ukázalo, že nepříznivý poměr klauzulí a proměnných můžeme částečně kompenzovat pomalejším ochlazováním, ovšem za cenu delší doby výpočtu.

3.4.3 Úspěšnost nalezení řešení

Nakonec jsem zkoumal úspěšnost nalezení alespoň nějakého řešení při rostoucím poměru klauzulí a proměnných r. V tomto pokusu jsme zaznamenával počet nenulových řešení z 30 běhů algoritmu na náhodně generovaných instancích s r rostoucím od 1 do 3.

Jak je vidět na grafu 7, úspěšnost rychle klesá k nule pro $r > 2$, a to i při poměrně vysokém koeficientu ochlazování (0.9999). Dále můžeme pozorovat, že mezi instancemi s 50 a 40 proměnnými není výrazný rozdíl v úspěšnosti, dá se ovšem předpokládat snížení průměrné kvality řešení, jak bylo naměřeno v sekci 3.4.2



Obrázek 6: Vývoj kvality řešení při snižování koeficientu ochlazování u instancí s $n = 40$ a $k = 48$, (vyšší koeficient)

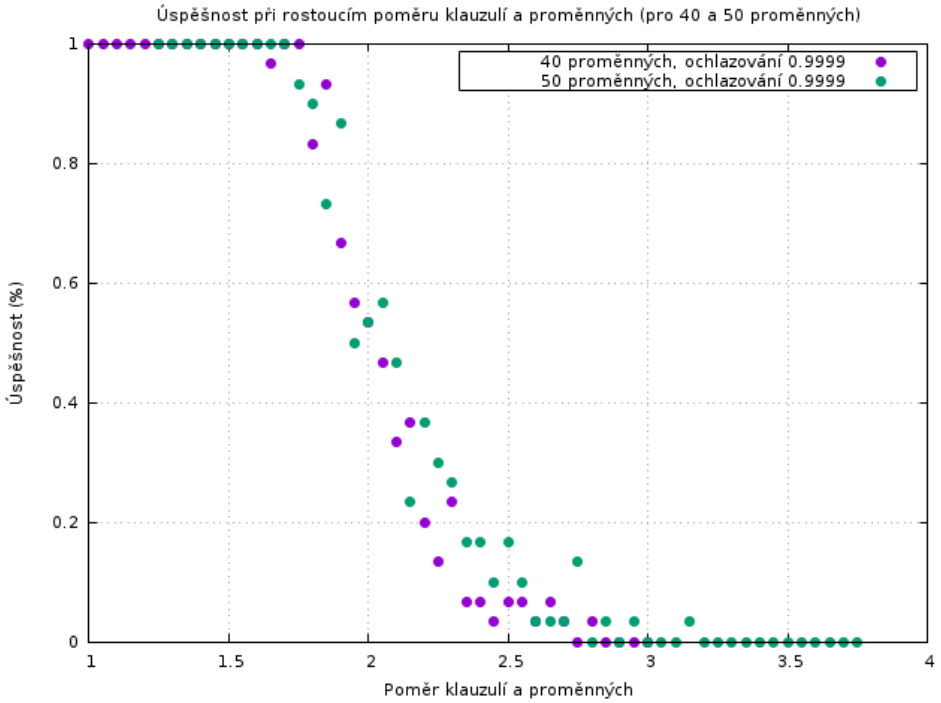
4 Závěr

V této úloze jsem implementoval algoritmus simulovaného ochlazování řešící 3SAT a otestoval jeho chování na náhodně generovaných instancích.

Ukázalo se, že při vhodném nastavení parametrů se poměrně úspěšně daří řešit instance s poměrem klauzulí a proměnných menším než 2. Rostoucí poměr nepříznivě ovlivňuje kvalitu řešení a při překročení hranice 2.5 se již algoritmu nedaří nalézt žádné řešení splňující zadanou formuli.

Nastavení parametrů heuristiky ovlivňuje míru prohledání stavového prostoru úlohy, a tedy podle očekávání i dobu běhu. Je tedy nutné najít kompromis mezi kvalitou řešení a časovou náročností.

Bohužel úspěšnost nacházení řešení klesá k nule mnohem dřív než je očekávaný mezní poměr 4.3[2]. Bylo by tedy dále vhodné prozkoumat možná vylepšení implementovaného algoritmu, jako například předzpracování vstupních dat pro volbu lepšího počátečního stavu.



Obrázek 7: Úspěšnost nalezení řešení při rostoucím poměru klauzulí a proměnných, pro $n = 40$ a $n = 50$

4.1 Vygenerované instance

K úloze přikládám instance použité k měření (složka `data`). Soubory s instancemi obsahují popis instance (počet klauzulí, proměnných, váhy a jednotlivé klauzule), a také nejlepší nalezené řešení spolu s parametry při kterých bylo nalezeno.

Malé instance (složka `data/small`) obsahují navíc i optimální řešení nalezené exaktním algoritmem a relativní chybu heuristiky.

Reference

[1] Russell, S. J.; Norvig, P.: *Artificial Inteligence - A Modern Approach.* Alan Apt, 1995, ISBN 0-13-103805-2.

[2] Selman, B.: *Stochastic Search and Phase Transitions: AI Meets Physics.* AT&T Bell Laboratories, [cit. 2016-01-21]. Dostupné z: `http://www.cs.`

cornell.edu/selman/compute/95ijcai/ijcai95aiphys.pdf

[3] Hoos, H. H.; Stützle, T.: *SATLIB: An Online Resource for Research on SAT*. University of British Columbia, [cit. 2016-01-23]. Dostupné z: http://www.cs.ubc.ca/~hoos/Publ/sat2000-satlib.pdf