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

Anna Kapitánová

January 25, 2023

1 Zadání

Je dán vektor X o n proměnných $(x_1 \dots x_n)$, $x_i \in \{0, 1\}$, dále Booleova formule těchto proměnných v konjunktivní normální formě o m klauzulích (součtových termech), dále pak pro každou proměnnou x váha $w(x)$. Úkolem je sestrojit ohodnocení Y proměnných X takové, že $F(Y) = 1$ a počet proměnných ohodnocených 1, je maximální. Odkaz na teorii.

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). Odkaz na zadání.

2 Algoritmus a implementace

2.1 Simulované žíhání

Jedná se o heuristickou optimalizační metodu, která je inspirovaná žíháním kovových slitin za úkolem získání jejich optimálních vlastností. Obecně u se u optimalizačních problémů snažíme maximalizovat resp. minimalizovat kvalitu řešení, kterou měříme pomocí nějaké kriteriální funkce F.

V simulovaném žíhání se díváme na momentální řešení a hledáme další přípustná řešení mezi sousedními řešeními. Pokud existuje v jeho okolí lepší řešení standardně k němu přejdeme, pokud ovšem ne, potenciálně i tak přejdeme ke horšímu řešení a to s pravděpodobností závislou na míře zhoršení a aktuální teplotě T. Míra zhoršení je definována jako: F(y) - F(x). Čím větší zhoršení sousední řešení představuje, tím menší je pravděpodobnost, že k němu přejdeme. Parametr teploty je zadán na počátku jako počáteční teplota a v průběhu hledání řešení je teplota postupně ochlazována v závislosti na ochlazovacím koeficientu. Platí, že čím je teplota vyšší tím je vyšší pravděpodobnost akceptování horšího řešení.

$P(F(x), F(y), T) = e^{\frac{F(y)-F(x)}{T}}$

2.2 Stavový prostor

Jako stav jsou považovány jednotlivé konfigurace, tedy ohodnocení proměnných v patřičné formuli.

Kriteriální funkce, kterou jsem měřila kvalitu stavu, resp. jsem pomocí ní porovnávala dva stavy, byla vybrána následovně:

- Jeden stav je splnitelný a druhý není – v tomto případě je lepším stavem ten, který je splnitelný.
- Oba stavy jsou nesplnitelné:
  - liší se v počtu splněných klauzulí – ten stav s více splněnými klauzulemi je lepší,
  - počet splněných klauzulí je u obou stavů stejný – lepší je ten stav jehož vážený součet proměnných je vyšší.
- Oba stavy jsou splnitelné: lepší je ten stav, jehož vážený součet proměnných je vyšší.

2.3 Implementace

Kód je psaný v jazyce Python 3, vygenerované grafy jsem vytvářela s pomocí knihovny Matplotlib.

Třída *Sat* reprezentuje jednotlivé instance. Obsahuje obsažené klauzule jako instance třídy *Clause*, udržuje informace o počtu klauzulí, počtu proměnných, či váhy jednotlivých proměnných. Obsahuje např. metodu pro zjištění, zda je formule splnitelná pro konkrétní ohodnocení (řešení).

Třída *Clause* reprezentuje jednu klauzuli. Udržuje proměnné, které klauzule obsahuje a metodu zjišťující, zda je klauzule splnitelná pro konkrétní ohodnocení.

Třída *SimAnnealSolver* zaštiťuje kostru algoritmu simulovaného žíhání a obsahuje metody specifické pro konkrétní problémy, jako je porovnání jednotlivých řešení, hledání nejbližších sousedů aktuálního stavu, funkci `optimize` (samotný algoritmus), apod. Udržuje si konkrétní problém tedy instanci třídy *Sat*, nejlepší doposud nalezené řešení jako instanci třídy *Solution* a hodnoty jednotlivých parametrů, které figurují v algoritmu simulovaného žíhání, jako je koeficient chlazení, počáteční teplota apod.

Třída *Solution* reprezentuje jednotlivá řešení, udržuje informace jako ohodnocení proměnných, vážený součet vah proměnných, hodnotu fitness (udržuje hodnotu kriteriální funkce pro dané řešení), informaci o tom, zda je řešení splnitelné, a počet splněných klauzulí při daném ohodnocení.

Soubor *main.py* obsahuje funkce pro spouštění algoritmu, načtení a zápis dat apod.

2.4 Měření výsledků

Veškerá měření jsem prováděla na PC s procesorem Intel Core i5-8250U CPU @ 1.60GHz × 8. Čas výpočtu jsem měřila pomocí knihovny *timeit*.

Zaznamenávala jsem dvě relativní chyby algoritmu, jednak vzhledem k sumě vah a jednak vzhledem k počtu splněných klauzulí oproti referenčnímu optimálnímu řešení. Během průběhu výpočtu jsem měřila relativní chyby pro nejlepší řešení i pro momentálně nalezené řešení v jednotlivých iteracích. Relativní chyby byly vypočteny pomocí těchto vzorců:

- $rel\_err\_weights = \frac{(opt\_weigth - sol\_weights\_sum)}{opt\_weights\_sum}$,
- $rel\_err\_clauses = \frac{(clauses\_number - sol\_is\_sat\_clauses)}{clauses\_number}$.

Při analýze naměřených výsledků jsem se zaměřovala na metriky jako vývoj počtu splněných klauzulí, váženého součtu vah, či relativní chyby v závislosti na čísle iterace. Dále jsem se pak zaměřovala např. na dobu běhu v závislosti na instanci a dalších parametrech, které blíže rozeberu v patřičných sekcích.

3 White box fáze

Ladění v rámci white box fáze jsem rozdělila do několika iterací, kde se každá iterace zaměřuje na určitou oblast algoritmu, jaký na ní mají jednotlivé parametry vliv a jaké hodnoty parametrů byly na základě měření vybrány.

3.1 První iterace

V první fázi jsem počítala s následujícím nastavením:

- Počáteční stav jsem volila tak, že jsem ohodnotila jednotlivé proměnné náhodně.
- Přechodovou funkci jsem nastavila tak, že vybere náhodně takový sousední stav, který se od momentálního stavu liší v ohodnocení maximálně jedné proměnné.
- Počáteční teplotu jsem nastavila jako průměrný součet vah všech zadaných proměnných z formule. Tento způsob byl zvolen, neboť je citlivý na odlišné rozsahy vah mezi jednotlivými instancemi.
- Koeficient chlazení jsem nastavila na hodnotu 0,95.
- Bod mrazu jsem definovala tak, že ke zmražení dojde v tom případě, když momentální teplota klesne pod mez, která je definovaná jako startovní teplota vynásobená koeficientem koncové teploty.
- Koeficient koncové teploty jsem nastavila na 0,25.
- Ekvilibrium jsem nastavila na pevný počet kroků 100.
- Hodnotu `take rate` jsem nastavila na 0,02.

Cílem první iterace bylo ověření, že algoritmus funguje správně. Pro testování jsem tedy volila instance ze sady `wuf20-71`. Vybrala jsem instance ze všech sad M, N, Q, R. Obrázek 1 zobrazuje průběh algoritmu u instance 01 z datové sady M. Tabulka 3.1 zobrazuje výsledky pro jednotlivé instance. Je vidět, že algoritmus nalezl u všech instancí řešení, na instancích ze sady M nalezl i vždy nejlepší řešení.

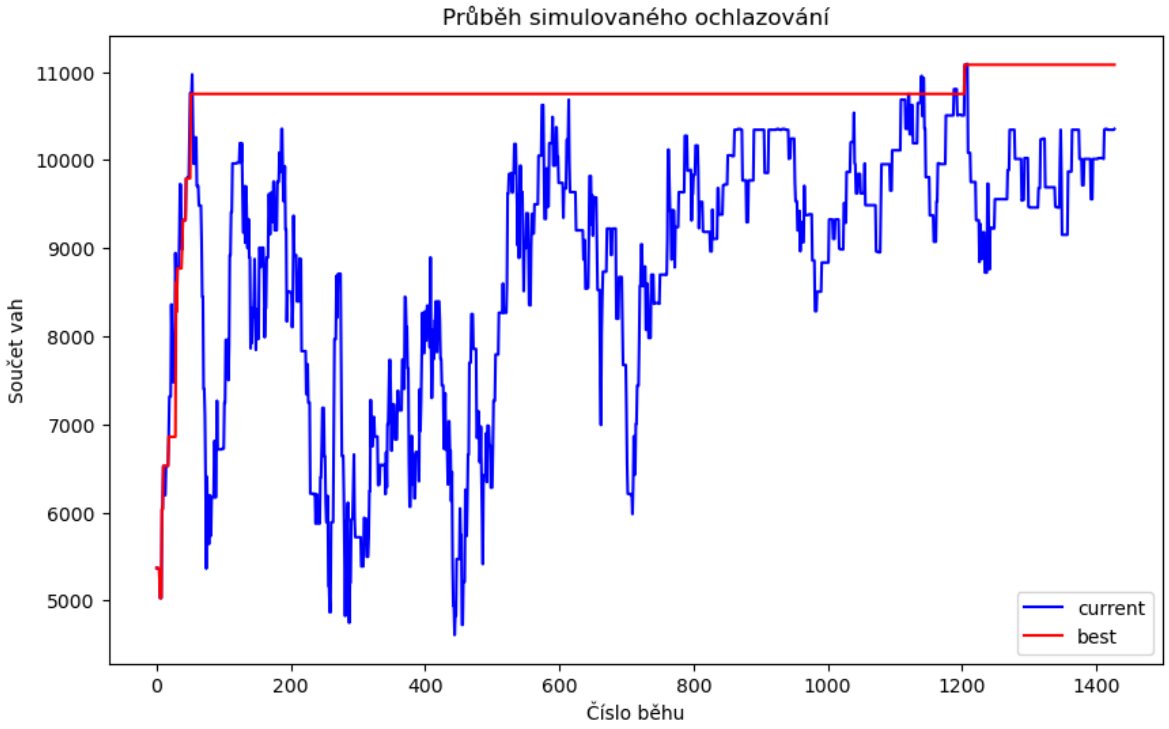

Figure 1: Ukázka průběhu algoritmu na instanci 01 z datové sady M

sada	instance	splnitelná	nalezená hodnota	optimální hodnota
M	01	true	6829	6829
M	09	true	11081	11081
M	010	true	15 483	15 483
N	01	true	141 661	141 661
N	02	true	111 770	118 347
N	010	true	130 165	130 165
Q	03	true	2999	4077
Q	04	true	3663	3988
Q	010	true	8320	9133
R	02	true	92 299	100 666
R	03	true	70 095	95 341
R	010	true	84 014	84 014

3.2 Druhá iterace

Předmětem druhé iterace bylo ladění parametru koeficientu ochlazování. Koeficient ochlazovaní má vliv na to, jak rychle se bude teplota ochlazovat mezi jednotlivými běhy vnějšího cyklu.

Vyzkoušela jsem různé hodnoty koeficientu: 0.975, 0.9775, 0.98, 0.9825, 0.985, 0.9875, 0.99, 0.9925, 0.995 a 0.9975. Ostatní parametry jsem nechala shodné jako v první iteraci. Na tuto a všechny další iterace jsem používala testovací instance z datových sad `wuf20-91` a `wuf50-218`. Bylo provedeno vždy 20 běhů pro danou hodnotu a výsledky pak byly průměrovány.

Během výzkumu byla potvrzena premisa, že čím vyšší koeficient zvolíme, tím delší čas nám algoritmus zabere. Závislost délky běhu algoritmu na koeficientu ochlazování zobrazuje graf 2. Ukázalo se také, že zvyšováním koeficientu ochlazování klesá relativní chyba splněných klauzulí a naopak roste relativní chyba sum vah. Tabulka 3.2 tento trend zobrazuje. Na základě naměřených dat bylo rozhodnuto nadále pracovat s koeficientem ochlazování 0,99.

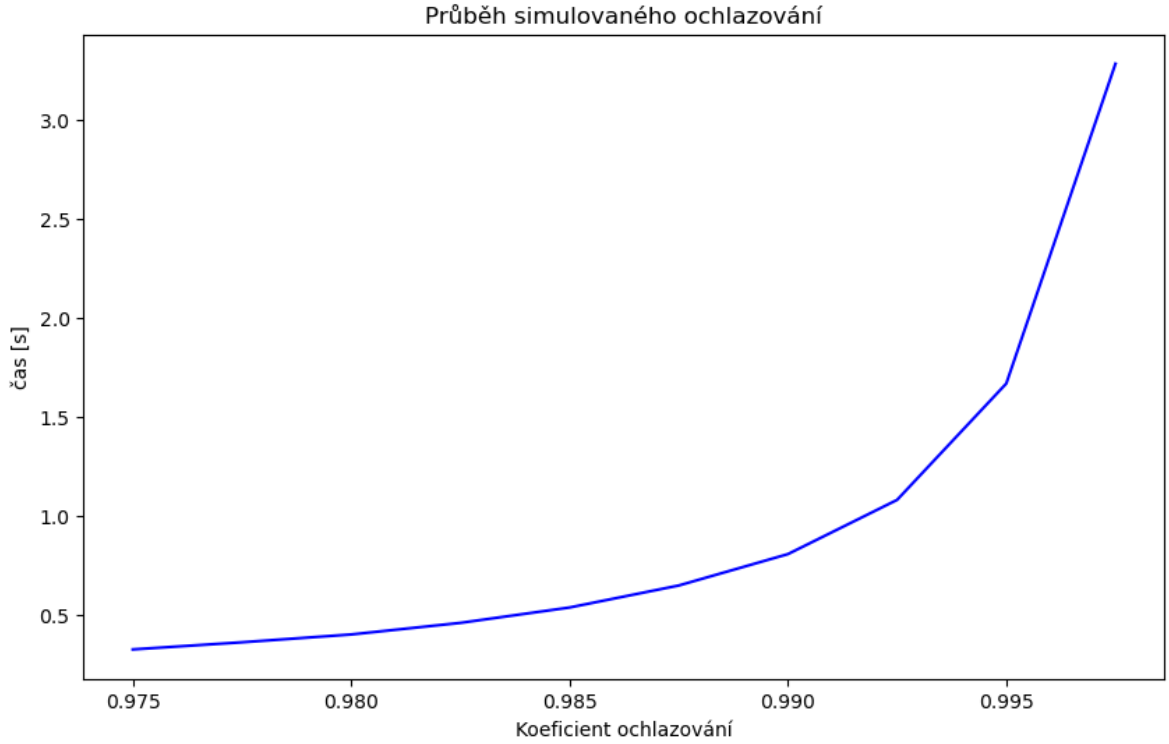

Figure 2: Závislost délky běhu algoritmu na koeficientu ochlazovaní

Koeficient ochlazovaní	Průměrná relativní chyba sum vah	Průměrná relativní chyba počtu splněných klauzulí
0,975	0,12	0,121
0,9775	0,22	0,125
0,98	0,25	0,101
0,9825	0,23	0,76
0,985	0,27	0,065
0,9875	0,26	0,034
0,99	0,26	0,014
0,9925	0,28	0,013
0,995	0,27	0,003
0,9975	0,30	0,005

3.3 Třetí iterace

Předmětem třetí iterace bylo zjistit ideální hodnotu koncové teploty, tedy kdy dojde ke zmražení. Zmražení jsem implementovala dvěma různými způsoby. Konec nastane, buď pokud momentální teplota klesne pod mez, která je definovaná jako startovní teplota krát koeficient koncové teploty, nebo pokud algoritmus už delší dobu nejeví známky zlepšení, což je definováno pomocí parametru `take rate`.

Z měření vyplynulo, že čím vyšší koeficient koncové teploty nastavíme tím kratší čas nám celkový výpočet zabere, jak zobrazuje graf 3. Zároveň ale vzhledem k tomu, že má algoritmus menší čas na možné zlepšení rostou relativní chyby, jak je vidět na grafech 3. Bylo tedy nutné najít rozumný kompromis mezi přijatelnou dobou běhu algoritmu a jeho výkonem. Z těchto byla na základě naměřených výsledků vybrána hodnota koeficientu 0,1.

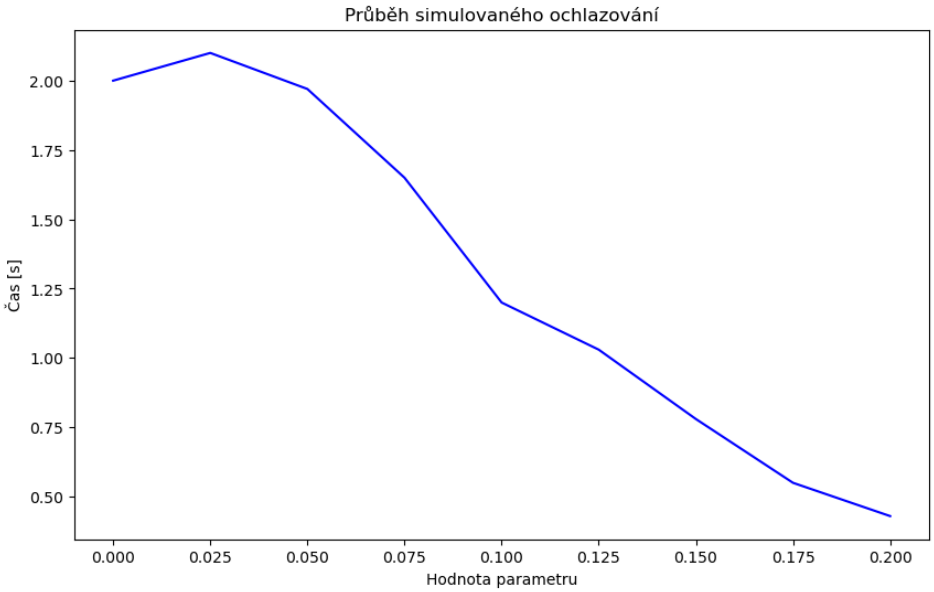

Figure 3: Závislost délky běhu algoritmu na koeficientu koncové teploty

Druhým krokem třetí iterace bylo ladění parametru `take rate`. Obvykle se v literatuře dočteme doporučovanou hodnotu tohoto parametru ve velmi nízkých číslech v řádu desetin procent, či maximálně jednotek procent. Byly zkoumány hodnoty parametru: 0,005, 0,01, 0,015, 0,02 a 0,025. Grafy 5 zobrazují srovnání vlivu parametru take rate na součet vah v závislosti na čísle iterace algoritmu na jedné instanci. Hodnota parametru pro další pokračování byla zvolena 0,05.

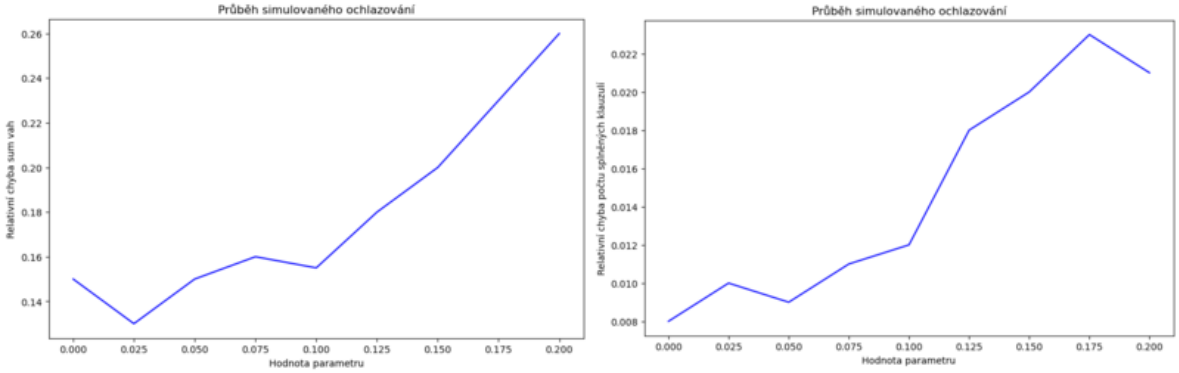

Figure 4: Závislost relativních chyb algoritmu na koeficientu koncové teploty

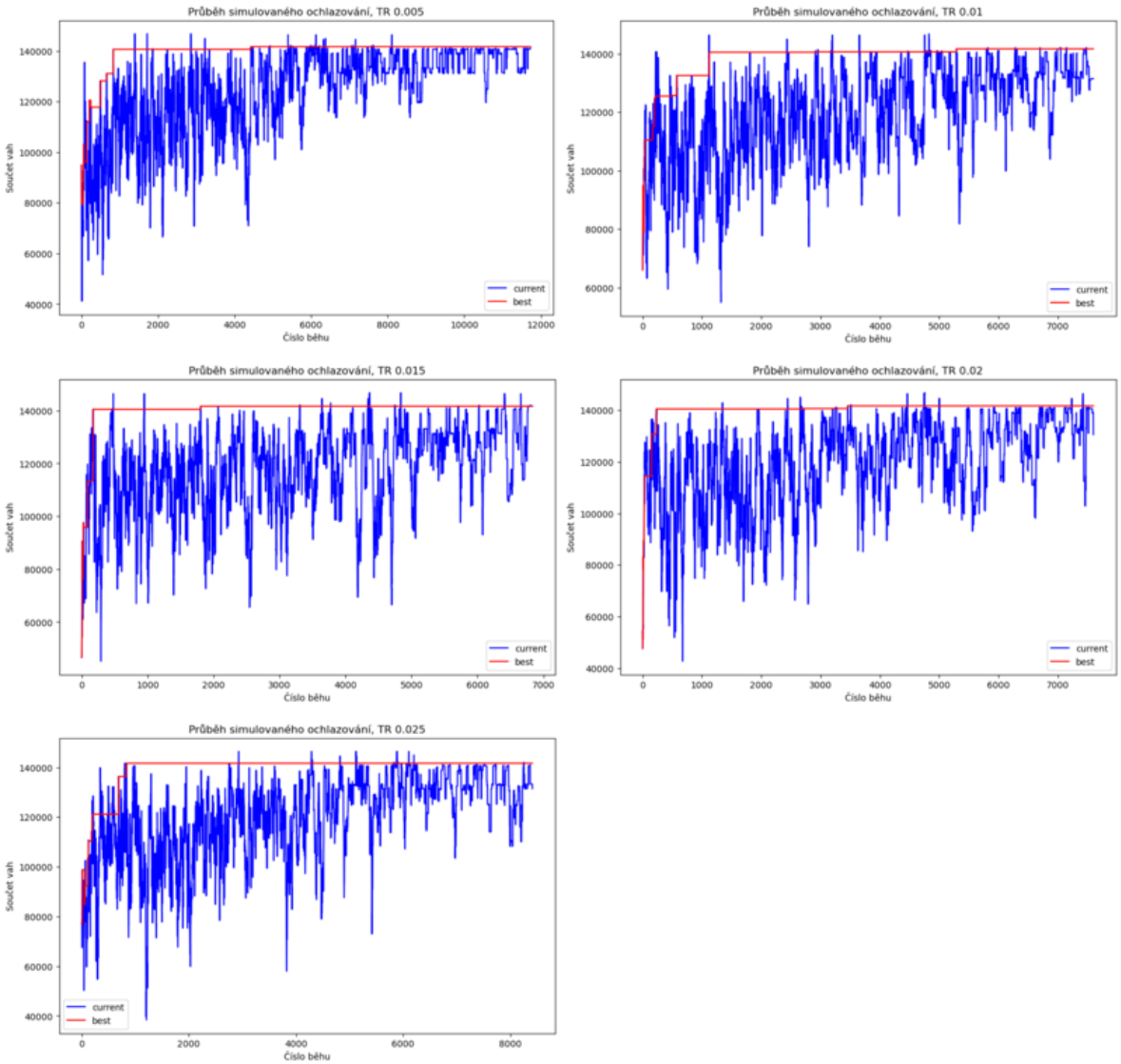

Figure 5: Srovnání vlivu parametru `take rate` na součet vah v závislosti na čísle iterace algoritmu

3.4 Čtvrtá iterace

V rámci čtvrté iterace byl laděn parametr mezi ekvilibria, který ovlivňuje počet vnitřních cyklů. Pokud počet vnitřních cyklů překročí hranici ekvilibria dojde k ukončení vnitřního cyklu. Byly zkoumány tyto hodnoty: 50, 100, 150 a 200. Jak je vidět na grafu 6, čas běhu algoritmu lineárně stoupal se zvyšující se hodnotou parametru. Grafy 8 zobrazují srovnání vlivu parametru na součet vah v závislosti na čísle iterace algoritmu na jedné instanci. Se zvyšující hodnotou parametru došlo ke snížení obou měřených relativních chyb, jak je vid2t na grafech 7. Na základě poznatků z měření byla stanovena hodnota 200 hodnotou parametru do dalších měření.

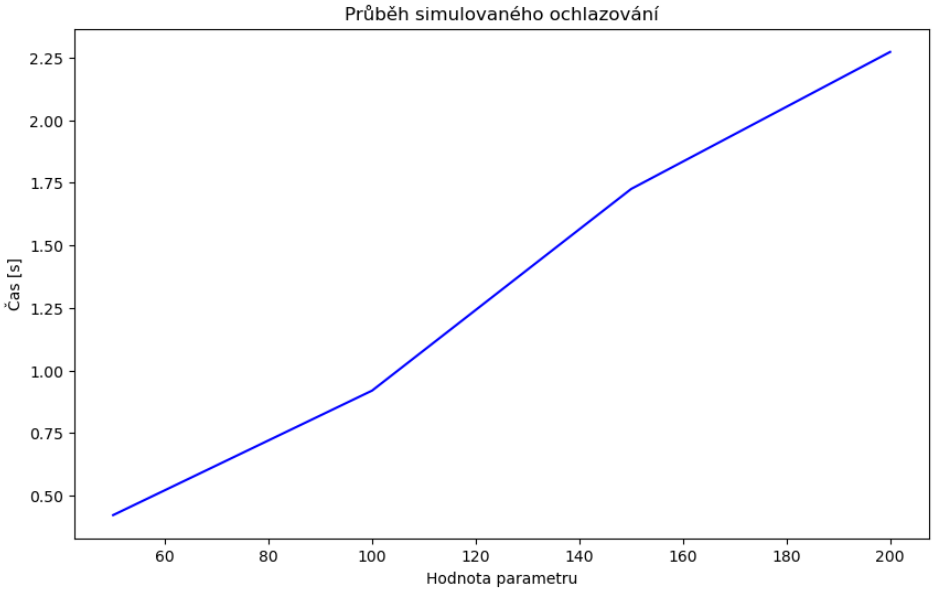

Figure 6: Závislost délky běhu algoritmu na mezi equilibria

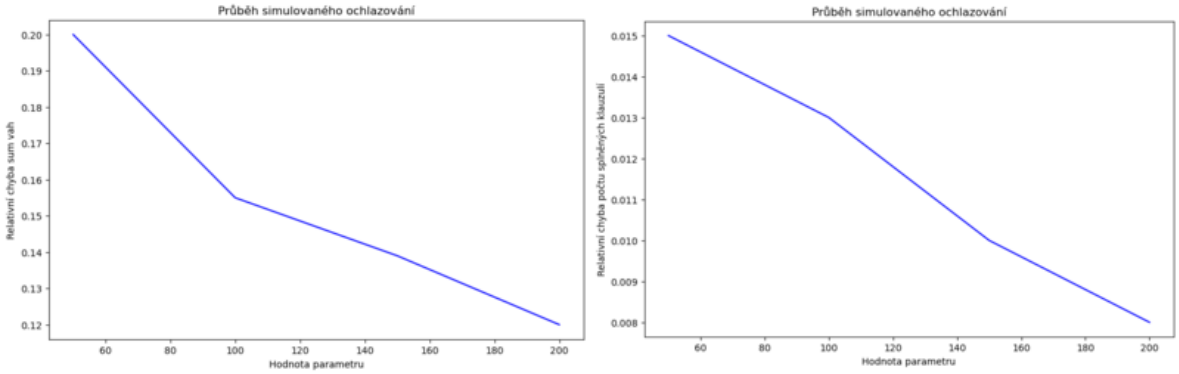

Figure 7: Závislost relativních chyb algoritmu na mezi equilibria

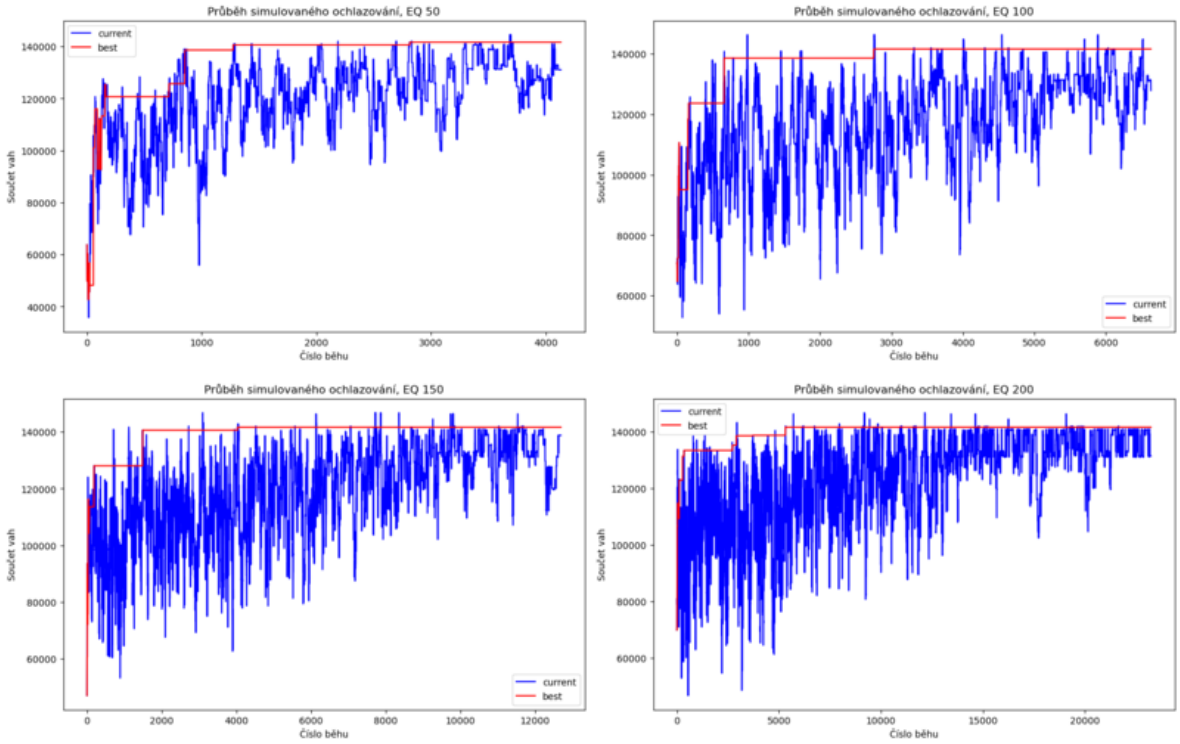

Figure 8: Srovnání vlivu parametru meze equilibria na součet vah v závislosti na čísle iterace algoritmu

3.5 Základní řešení

Finální měření jsem provedla na 100 prvních instancí z každé ze sad `wuf20-91` a `wuf50-218`, měření byla provedena desetkrát a výsledky byly zprůměrovány. Použila jsem parametry, které byly vybrány v rámci ladění parametrů v sekcích výše.

Vybrané nastavení:

- **Počáteční stav** jsem volila tak, že jsem ohodnotila jednotlivé proměnné náhodně.
- **Přechodovou funkci** jsem nastavila tak, že vybere náhodně takový sousední stav, který se od momentálního stavu liší v ohodnocení maximálně jedné proměnné.
- **Počáteční teplotu** jsem nastavila jako průměrný součet vah všech zadaných proměnných z formule. Tento způsob byl zvolen, neboť je citlivý na odlišné rozsahy vah mezi jednotlivými instancemi.
- **Koeficient chlazení** jsem nastavila na hodnotu **0,99**.
- **Bod mrazu** jsem definovala tak, že ke zmražení dojde v tom případě, když momentální teplota klesne pod mez, která je definovaná jako startovní teplota krát koeficient koncové teploty.
- **Koeficient koncové teploty** jsem nastavila na **0,1**.
- **Ekvilibrium** jsem nastavila na pevný počet kroků **200**.
- Hodnotu `take rate` jsem nastavila na **0,01**.

Výsledky měření dat po ladění parametrů zobrazuje tabulka 3.5.

Sada	Průměrná relativní chyba sum vah	Průměrná relativní chyba počtu splněných klauzulí	Průměrný naměřený čas [s]
wuf20-91-M	0,056	0	0,820
wuf20-91-N	0,076	0	0,986
wuf20-91-Q	0,095	0	0,988
wuf20-91-R	0,103	0	1,034
wuf50-218-M	0,117	0,007	2,433
wuf50-218-N	0,119	0,014	2,670
wuf50-218-Q	0,130	0,020	2,730
wuf50-218-R	0,125	0,021	2,686

4 Blackbox

Nyní přejdeme k závěrečnému ohodnocení vyhodnocení heuristiky. Pro testování byly vybrány všechny dostupné sady, které obsahovaly optima. Jedná se o sady `wuf20-91`, `wuf50-218` a `wuf75-325`, počítala jsem se všemi instancemi. S ohledem na randomizovanou povahu algoritmu jsem provedla deset měření a výsledky jsem zprůměrovala. Byly použity parametry, které jsem vybrala v rámci whitebox fáze v sekci výše. Tabulka 4 zobrazuje naměřené výsledky pro jednotlivé datové sady. Z výsledků je patrné, že heuristika dokáže najít u instancí s n=20, vždy takové ohodnocení, které danou formuli splňuje. U dalších datových sad vidíme mírné zhoršení průměrné relativní chyby počtu splněných klauzulí. Průměrná relativní chyba sum vah je u sady s n=20 pod hranicí 10 %, u n=50 se pak pohybujeme průměrné chyby kolem 12,5 %, resp. zhruba 13,6 % u sady s instancemi n=75.

Sada	Průměrná relativní chyba sum vah	Průměrná relativní chyba počtu splněných klauzulí	Průměrný čas
wuf20-91	0,083	0	0,970
wuf50-218	0,125	0,017	2,612
wuf75-325	0,131	0,025	7,065

5 Závěr

V práci byla implementována pokročilá iterativní metoda simulovaného ochlazování pro optimalizační problém vážené splnitelnosti boolovské formule. Úlohu jsem vypracoval v jazyce Python. V průběhu whitebox fáze jsem provedla řadu experimentů na základě jejichž výsledků jsem vybírala patřičné parametry heuristiky. Po provedení ladění parametrů na sadách s menším počtem proměnných bylo provedeno testování na sadách wuf20-91, wuf50-218 a wuf75-325. Byly změřeny výsledky průměrného času výpočtu a průměrné ralativní chyby sumy vah i počtu splněných klauzulí.