NI-KOP 2. Úloha

Matouš Kovář

1 Stručný souhrn

Tato semestrální práce se zabývá řešením problému maximální vážené splnitelnosti booleovské formule (MWSAT) s využitím simulovaného ochlazování. Cílem práce byl návrh a implementace robustního algoritmu, který je schopen efektivně nalézt validní a kvalitní řešení bez nutnosti interaktivních zásahů do konfigurace.

Postup řešení je rozdělen do dvou fází. V první, tzv. *Whitebox fázi*, proběhlo ladění hyperparametrů na redukované sadě instancí. Druhá, *Blackbox fáze*, se zaměřila na finální experimentální vyhodnocení na nezávislých sadách instancí o velikosti 20, 50 a 75 proměnných s různým rozložením vah. Dosažené výsledky prokazují uspokojivou úspěšnost algoritmu (přes 98 % vyřešených instancí) a schopnost škálování i na výpočetně náročnější problémy.

2 Úvod

Cílem této semestrální práce je návrh, implementace a experimentální vyhodnocení řešení problému maximální vážené splnitelnosti booleovské formule (MWSAT). Jedná se o optimalizační rozšíření problému SAT, kde je každé proměnné přiřazena nezáporná váha a úkolem je nalézt takové ohodnocení proměnných, které maximalizuje součet vah proměnných nastavených na 1 a zároveň splňuje všechny klauzule. Vzhledem k tomu, že se jedná o problém patřící do třídy NPO, je pro hledání kvalitního řešení v rozumném čase využita pokročilá iterační heuristika – konkrétně **simulované ochlazování**.

Řešení musí být schopno zpracovávat instance rozdílných vlastností a velikostí bez nutnosti interaktivních zásahů do konfigurace. Algoritmus je navrhován tak, aby si poradil s instancemi, které jsou považovány za výpočetně náročné ve smyslu Selmanova kritéria fázového přechodu.

Tento dokument reflektuje postup řešení rozdělený do dvou klíčových fází:

- **Whitebox fáze** – popisuje proces hledání konfigurace, ladění parametrů a analýzu neúspěšných řešení.
- **Blackbox fáze** – předkládá finální experimentální vyhodnocení hotové heuristiky na nezávislé testovací sadě. V této části je ověřena škálovatelnost, stabilita a celková úspěšnost algoritmu.

3 Materiál

Celá práce je vypracována v programovacím jazyce **Python**. Pro načítání vstupních dat a ukládání výsledků byly implementovány specializované třídy `MWSATInstance` a `MWSATSolution`. Tyto třídy zajišťují nejen manipulaci s daty, ale umožňují i efektivní mapování a sledování postupu algoritmu během výpočtu. Samotný algoritmus je realizován pomocí funkce `simulated_annealing`.

Pro následnou vizualizaci naměřených dat a tvorbu grafů jsou využity knihovny **Matplotlib** a **Seaborn**. Veškeré experimenty byly spouštěny v prostředí Jupyter Notebook na stroji **MacBook Pro** s procesorem **M4**, přičemž pro urychlení výpočtů bylo využito 12 procesorových jader.

Pro experimentální vyhodnocení byly použity celkem tři velikostní kategorie datových sad. Tyto sady obsahují instance o **20, 50 a 75 proměnných**. Zatímco sady s 20 a 50 proměnnými obsahují 1 000 instancí, sada se 75 proměnnými je omezena na 100 instancí. Klíčovou vlastností použitých dat je jejich variabilita ve váhách: každá velikostní kategorie obsahuje **čtyři podsady**, které se liší způsobem generování vah jednotlivých proměnných. Toto rozdělení umožňuje ověřit robustnost heuristiky na instancích s odlišným charakterem váhové funkce.

Všechny testovací instance byly staženy z webových stránek předmětu NI-KOP[1]. Podle dostupné dokumentace se jedná o výpočetně náročné a neredukované sady.

[1] `https://courses.fit.cvut.cz/NI-KOP/download/index.html#mwsatinst`

4 Popis algoritmu a implementace

Základem implementovaného řešení je heuristika simulovaného ochlazování. Architektura řešení je rozdělena do dvou hlavních tříd: `MWSATInstance`, která zajišťuje načítání a předzpracování dat, a `MWSATSolution`, která reprezentuje aktuální stav v prohledávacím prostoru a zapouzdřuje operace nad ohodnocením řešení. Samotná heuristika je implementována jako funkce v souboru `simulated_annealing.py`

4.1 Reprezentace dat a vyhodnocovací funkce

Vstupní data instancí jsou v rámci předzpracování transformována. Klíčovým krokem je **lineární přeškálování vah** proměnných do intervalu $[0, 1]$. Tento krok je zásadní pro robustnost algoritmu, neboť umožňuje definovat hyperparametry (zejména počáteční teplotu a penalizace nesplněných klauzulí) nezávisle na absolutní velikosti vah v konkrétní instanci.

Algoritmus využívá hierarchickou vyhodnocovací logiku, která striktně upřednostňuje validitu řešení před jeho váhou. Při porovnání dvou stavů platí(v implementaci k tomu slouží funkce `compare_states`):

1. **Kritérium splnění formule:** Pokud nový stav splňuje více klauzulí než starý, je považován za striktně lepší bez ohledu na součet vah. Tím je zajištěno směřování do oblasti přípustných řešení.
2. **Kritérium maximalizace součtu vah:** Pouze pokud je počet splněných klauzulí shodný, rozhoduje o kvalitě vyšší normalizovaný součet vah proměnných ohodnocených jako pravda.

Pro případy kdy je vygenerováno horší nové řešení využíváme fitness funkci, na které je závislá pravděpodobnost přijetí tohoto řešení. Tato funkce bude popsána v sekci 5.3.

4.2 Parametry algoritmu

Chování algoritmu je řízeno sadou hyperparametrů:

P0 (Počáteční pravděpodobnost přijetí): Určuje počáteční teplotu T_{start}. Teplota je nastavena tak, aby v úvodní fázi byla průměrná pravděpodobnost přijetí zhoršujícího stavu rovna hodnotě P0.

Cooling Coefficient: Koeficient ochlazování, kterým je po každém ekvilibriu násobena aktuální teplota.

Equilibrium Steps: Nastavuje délku vnitřního cyklu pro danou teplotní hladinu. Parametr je násoben počtem proměnných, tak aby přirozeně pro instance s více podmínkami prozkoumával déle.

Fitness Coefficient: Váha penalizace za nesplněnou klauzuli ve fitness funkci. Musí být nastavena tak, aby algoritmus nikdy neupřednostnil zisk váhy za cenu porušení logické platnosti formule.

Max Steps Without Improvement: Zastavovací kritérium. Pokud algoritmus nenalezne nové globální maximum po počtu kroků odpovídajícímu

$$max\_steps\_without\_improvement \cdot num\_clauses$$

, tak je ukončen.

Random Flip: Binární parametr určující strategii výběru proměnné pro změnu stavu (flip):

- *True:* Náhodný výběr jakékoliv proměnné.
- *False:* Pokud existují nesplněné klauzule, vybírá se proměnná z náhodně zvolené nesplněné klauzule. To cíleně opravuje validitu řešení.

5 Ladění parametrů (whitebox fáze)

Lazení parametrů bylo provedeno iterativně. Jelikož jsou parametry na sobě závislé, neexistuje nějaký obecný způsob v jakém pořadí by se měly ladit. Výchozím bodem experimentů byla baseline konfigurace - tedy nějaká konfigurace, která relativně dobře funguje na datech.

Co se týče dat, na kterých byly parametry lazeny, tak zde byla nejprve používal data s 50 proměnnými a váhami druhu M. V poslední části evaluace, kde byla ověřoval funkčnost parametrů na jiných sadách, bylo zjištěno, že původní nastavení nevykazuje dostatečnou robustnost na sadách R a Q. Proto byla whitebox fáze zopakována, tentokrát ale na datech z instance R. To budou výsledky při volbě parametrů v této práci - `wuf-50-218-R`. V celé whitebox fázi jsou při lazení parametry rozbíhány 4 krát na 10 různých instancích ze sady.

5.1 Volba baseline konfigurace

Výchozí konfigurace parametrů, shrnutá v Tabulce 1, byla zvolena následovně:

Tabulka 1: Výchozí konfigurace parametrů (Baseline)

Parametr (kód)	**Hodnota**	**Význam**
`P0`	0.9	Počáteční pravděpodobnost přijetí horšího stavu
`cooling_coefficient`	0.9	Faktor geometrického ochlazování teploty
`equilibrium_steps`	3	Násobek N_{vars} pro délku vnitřního cyklu
`fitness_coefficient`	300	Váha penalizace za nesplněné klauzule
`max_steps_without_improvement`	100	Násobek $N_{clauses}$ pro zastavení při stagnaci
`random_flip`	True	Strategie čistě náhodného výběru proměnné

Jak si vedla tato konfigurace na 10 bězích 10 různých instancí je možné vidět v tabulce 2. Klíčovým nedostatkem je nízký počet vyřešených instancí, který je velmi malý. Vyvstává tedy otázka: Jak zvýšit tlak na splnitelnost formule?

Tabulka 2: Výsledky experimentu - Baseline (úspěšnost z 10 běhů)

Instance	**Solved**	**Optimal**
wuf50-0323	6	2
wuf50-0366	5	5
wuf50-069	5	5
wuf50-094	0	0
wuf50-0861	2	2
wuf50-0824	3	2
wuf50-0657	10	3
wuf50-0286	0	0
wuf50-0612	8	7
wuf50-0939	3	2

5.2 Nastavení výběru proměnné

Parametr `random_flip` nastavený na False zajišťuje, že budou obraceny jen ty proměnné, které jsou obsaženy v nesplněných klauzulích. Tento krok se ukázal jako výhodný, což potvrzují výsledky v tabulce 3.

Tabulka 3: Výsledky vylepšené konfigurace (úspěšnost z 10 běhů)

Instance	**Solved**	**Optimal**
wuf50-0323	10	10
wuf50-0366	10	10
wuf50-069	10	10
wuf50-094	7	5
wuf50-0861	10	9
wuf50-0824	9	4
wuf50-0657	10	6
wuf50-0286	10	10
wuf50-0612	10	10
wuf50-0939	10	6

Z výsledků je patrný dramatický nárůst úspěšnosti v nalezení validního řešení (sloupec *Solved*). Zatímco baseline verze spoléhající na náhodu selhávala, zavedení heuristiky umožnilo algoritmu efektivněji směřovat k splnitelným oblastem stavového prostoru.

Nicméně, při bližším pohledu na sloupec *Optimal* (např. u instance `wuf50-094` nebo `wuf50-0824`) vidíme, že ačkoliv algoritmus nalezne validní řešení, často nedokáže konvergovat k optimálnímu řešení.

5.3 Nastavení fitness funkce

Nyní přejdeme k dalšímu klíčovému parametru a to fitness funkci. Fitness funkce přímo ovlivňuje pravděpodobnost s kterou je horší stav přijat při dané teplotě. Původně byla navržena fitness funkce následující:

$$f(s) = \text{Weight}_{norm}(s) - (\text{fitness\_coefficient} \cdot N_{unsat}(s))$$

kde Weight_{norm} je součet normalizovaných vah splněných proměnných, N_{unsat} je počet nesplněných klauzulí a `fitness_coefficient` je koeficient penalizace ovlivňující jak moc má vliv každá nesplněná klausule.

Tato fitness funkce fungovala poměrně dobře, nicméně při revizi kódu byl odhalen hlavní nedostateka to je absence jakékoliv škálovatelnosti na počet proměnných. Například pro instance s větším počtem proměnných bude první člen Weight_{norm} přirozeně větší zase naopak druhý člen bude pro instance s větším počtem klausulí nabývat vyšší absolutní hodnoty, jelikož může obsahovat větší počet nesplněných klauzulí.

Proto byla fitness funkce přepracována na následující finální podobu, která adresuje tyto nedostatky:

$$f(s) = \text{Weight}_{norm}(s) - (\text{fitness\_coefficient} \cdot \text{num\_variables} \cdot \text{unsat\_ratio})$$

kde `num_variables` je počet proměnných a `unsat_ratio` je poměr nesplněných klauzulí vůči všem klauzulím.

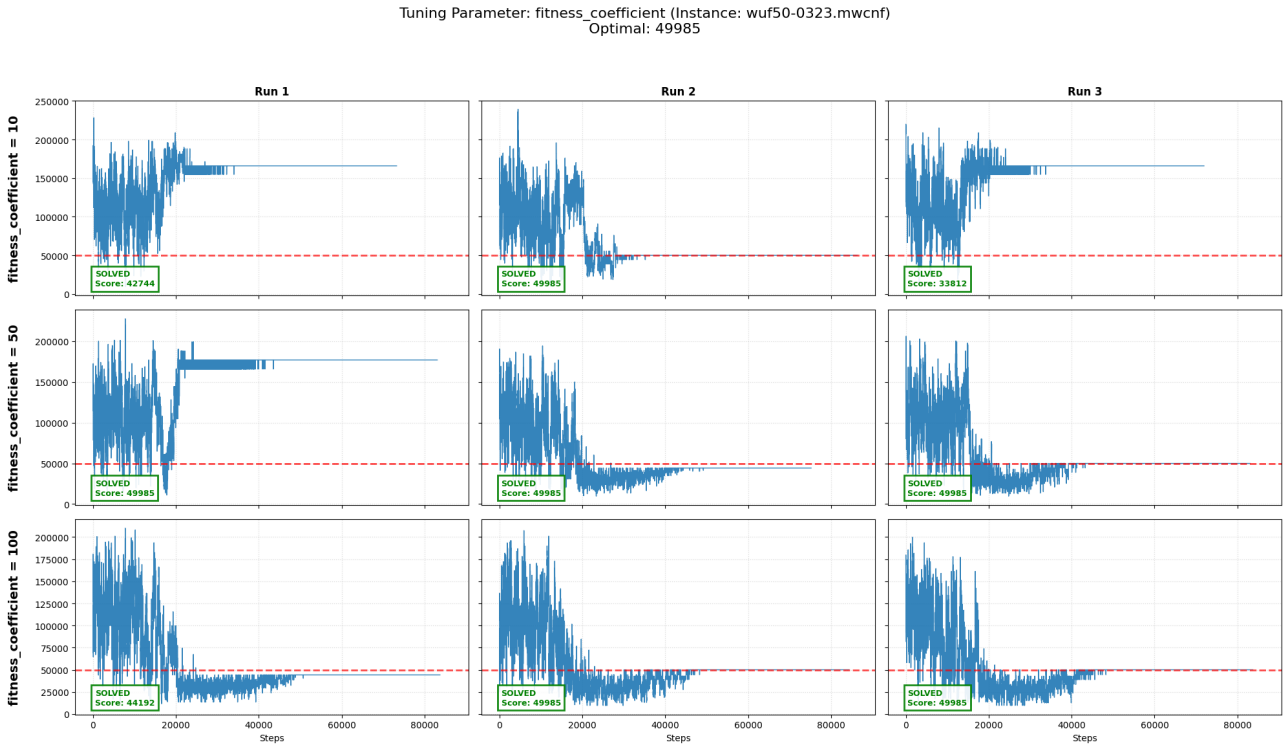
Hodnota `fitness_coefficient` 100 byla zvolena jako nejlepší kompromis, protože jako první dosáhla 100% validity (40/40) a zároveň si zachovala vysokou míru optimality.

Na tabulce 4 je možné sledovat počet vyřešených a optimálně vyřešených instancí. Ve sloupci Avg. % of Opt. je zapsáno jak velké hodnoty dosahuje neoptimální validní řešení v průměru.

Na grafu 1 je možné vidět běhy na jedné instanci s různými hodnotami. Pro nízké hodnoty je vidět jak algoritmus upřednostní řešení s větší váhou, protože nejsou dostatečně penalizovány nesplněné klauzule.

Tabulka 4: Vliv nastavení fitness koeficientu na kvalitu řešení (40 běhů)

fitness_coefficient	**Solved**	**Optimal**	**Avg. % Opt.(for solved)**	**Avg. steps**
10	32	16	90,30	78 136,6
20	36	24	92,12	79 624,5
50	39	34	97,57	80 376,6
100	40	30	96,31	79 395,6
150	38	30	95,58	80 164,1
200	39	31	97,41	79 461,0



Obrázek 1: Vliv parametru fitness coefficient na konvergenci. Zelené labely značí úspěšné nalezení splnitelného řešení, červené značí uvíznutí v neplatném stavu.

5.4 Inicializace teploty

Počáteční teplota je volena na základě vzorce z přednášky:

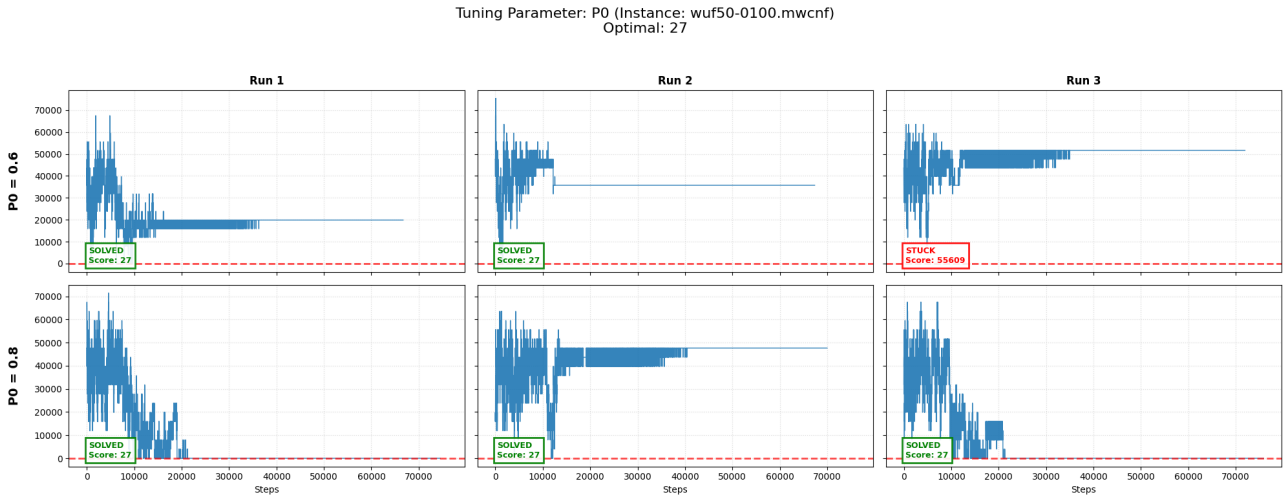
$$T_0 = \frac{\delta}{|\ln P_0|}$$

kde δ je průměrná velikost změny fitness při náhodném kroku. V praxi je vypočítána tak, že je spuštěn algoritmus na 3000 kroků s vysokou teplotou a nízkým `cooling_coefficient` — takže to v praxi odpovídá náhodné procházce po stavovém prostoru. V každém kroku odečtu fitness aktuálního a sousedního stavu a absolutní hodnoty zprůměruji. P_0 je pravděpodobnost úniku z tohoto minima.

Parametr, který je nalazen je tedy v tomto případě P_0. Vyšší hodnota odpovídá vyšší počáteční teplotě — vyšší pravděpodobnost přijetí zhoršení.

Tato metodu automatického nastavení teploty byla zvolena zvolil z důvodu, že teplotu automaticky škáluje na různé velikosti instance co do rozdílu mezi váhami a různé hloubky lokálních optim optim.

Hodnota byla nastavena na `P0 = 0.8`. Na grafu 2 je vidět jaký vliv mají 2 různé hodnoty tohoto parametru na 3 běžích na stejné instanci. Na tabulce 5.



Obrázek 2: Vliv parametru P0 na konvergenci

Tabulka 5: Vliv počáteční pravděpodobnosti P_0 na kvalitu řešení

P_0	**Solved**	**Optimal**	**Avg. % Opt.(for solved)**	**Avg. steps**
0,60	38	21	93,87	70 550,2
0,70	38	22	93,42	72 446,9
0,80	38	25	93,93	75 422,6
0,85	36	27	96,60	76 959,4
0,90	36	27	96,12	78 659,9

Hodnota $P_0 = 0.8$ byla zvolena jako nejvhodnější konfigurace na základě následujících kritérií:

- **Maximální robustnost:** Nastavení zachovává nejvyšší počet úspěšně vyřešených instancí (38), což je shodný výsledek jako u nižších hodnot (0.6 a 0.7). U vyšších hodnot (0.85 a 0.90) již dochází k poklesu spolehlivosti algoritmu (pouze 36 vyřešených).
- **Kvalita řešení:** Oproti konzervativnějším nastavením (0.6 a 0.7) dosahuje tato hodnota vyššího počtu nalezených globálních optim (25 oproti 21, resp. 22). To indikuje lepší schopnost prozkoumávat stavový prostor a unikat z lokálních minim.

5.5 Nastavení ekvilibria a chladícího koeficientu

Tyto dva parametry jsou spolu úzce spjaty, proto bude nejvhodnější hledat optimální hodnotu obou parametrů najednou. Optimalizace proběhla formou Grid Search nad definovaným prostorem parametrů. Spuštěny byly 3 krát na 10 různých instancích. Výsledky v tabulce 6, kde úspěšnost znamená kolik procent řešení spňuje formuli pro danou sadu parametrů.

Tabulka 6: Výsledky grid search pro parametry Cooling a Equilibrium Steps (průměr z 10 instancí)

Cooling	Steps	Úspěšnost	Prům. skóre	Prům. kroků
0,950	1	86,7 %	18 243,6	68 837
0,950	3	96,7 %	20 372,1	84 845
0,950	4	100,0 %	20 487,6	87 170
0,950	6	100,0 %	20 520,7	94 481
0,990	1	100,0 %	20 475,6	93 732
0,990	3	100,0 %	20 427,4	109 043
0,990	4	96,7 %	20 338,8	109 232
0,990	6	100,0 %	19 730,7	103 724
0,995	1	100,0 %	20 520,7	105 897
0,995	3	100,0 %	19 695,7	105 010
0,995	4	100,0 %	20 080,8	112 458
0,995	6	100,0 %	19 525,1	117 938
0,999	1	100,0 %	19 306,9	112 044
0,999	3	100,0 %	19 123,1	105 926
0,999	4	96,7 %	19 368,6	106 965
0,999	6	100,0 %	19 381,1	105 773

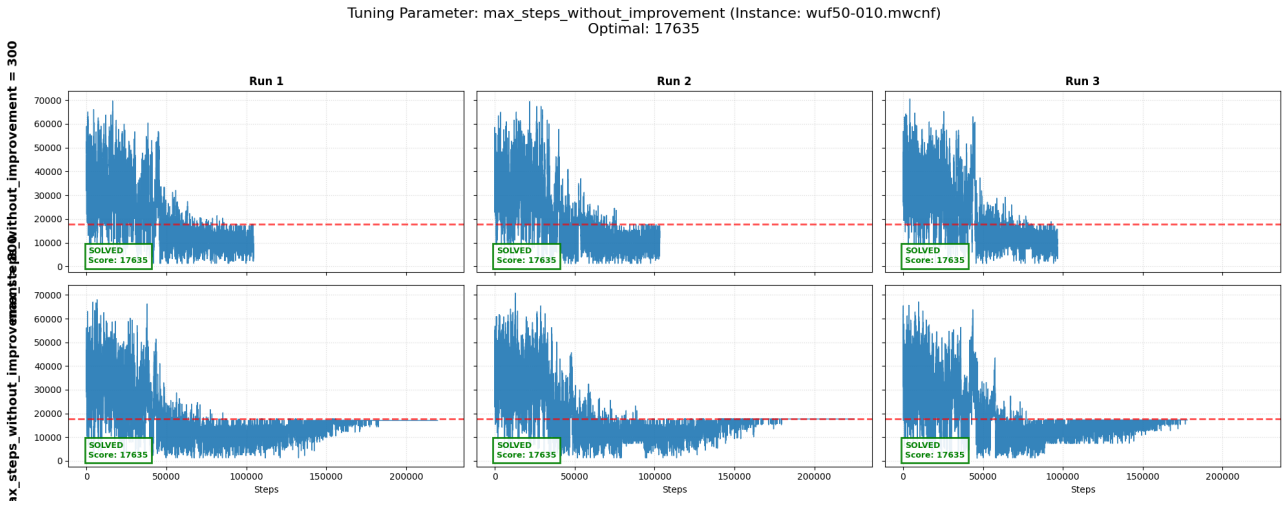
Na základě analýzy naměřených dat lze formulovat následující závěry ohledně vlivu parametrů na chování algoritmu:

- **Kvalita řešení:** Nejvyššího průměrného skóre (20 520, 7) dosáhly dvě konfigurace: kombinace `cooling_coefficient = 0.995` s (equilibrium_steps = 1) a kombinace `cooling_coefficient = 0.95` s `equilibrium_steps = 6`. Tyto sady parametrů se jeví jako globálně optimální pro testovanou množinu instancí.
- **Efektivita konvergence:** Při srovnání obou nejúspěšnějších konfigurací vykazuje varianta `cooling_coefficient = 0.95` / (equilibrium_steps = 6) vyšší výpočetní efektivitu. Ačkoliv obě nastavení vedou ke stejné kvalitě řešení, tato varianta konverguje průměrně v 94 000 krocích, zatímco varianta s `cooling_coefficient = 0.995` vyžaduje přibližně 106 000 kroků. Rychlejší chlazení kompenzované delším prohledáváním na dané teplotě se tak ukazuje jako výhodnější strategie.

5.6 Nastavení počtu kroků algoritmu bez zlepšení

Poslední parametr, který je potřeba nastavit, je parametr `max_steps_without_improvement`. Tento parametr je násoben počtem klauzulí a tím je získán počet kroků algoritmu který lze provést bez nalezení globálního maxima před ukončením algoritmu. Je potřeba ho nastavit tak aby neukončil algoritmus příliš brzy a zároveň aby ho nenechal běžet příliš dlouho a tak neplýtval výpočetními prostředky.

Graf 3 běhů na instanci pro hodnoty parametru 300 a 800 je zobrazen na obrázku 3. Kvantitativní výsledky jsou shrnuty v tabulce 7



Obrázek 3: Vliv parametru max_steps_without_improvement na konvergenci

Tabulka 7: Porovnání výsledků pro různé hodnoty parametru max_steps_without_improvement

Max Steps	Runs	Solved	Optimal	Avg % Opt	Avg Steps
200	40	40	35	98.40	72 888.3
300	40	40	37	99.15	94 928.1
600	40	40	39	99.85	158 268.0
800	40	40	40	100.00	205 552.2
1200	40	40	40	100.00	290 735.7

Na základě výsledků shrnutých v Tabulce 7 byla jako finální hodnota parametru `max_steps_without_improvement` zvolena hodnota 800.

Důvody pro tento výběr jsou následující:

- **Garance optimality:** Hodnota 800 je nejnižší testovanou konfigurací, která dosáhla 100% úspěšnosti v nalezení optimálního řešení (40/40). Nižší nastavení, vykazovala v několika běžích konvergenci pouze k lokálním optimům.
- **Stabilita vs. Rychlost:** Ačkoliv je nastavení 600 výpočetně méně náročné, riziko nenalezení globálního optima (úspěšnost 39/40) je pro účely této aplikace nepřijatelné. Hodnota 800 tak představuje nutnou investici výpočetního času pro zajištění maximální spolehlivosti a dokončení chlazení.
- **Šetření výpočetních prostředků:** Další navýšení parametru na 1200 sice udržuje 100% úspěšnost, ale nepřináší žádnou kvalitativní výhodu oproti hodnotě 800, přičemž neúměrně zvyšuje výpočetní čas (*Avg Steps*). Bod 800 tedy představuje hranici, kde se kvalita řešení stabilizuje.

Tímto jsme dokončili volbu parametrů. **Finální parametry** lze vidět v tabulce 8.

Tabulka 8: Finální kombinace parametrů

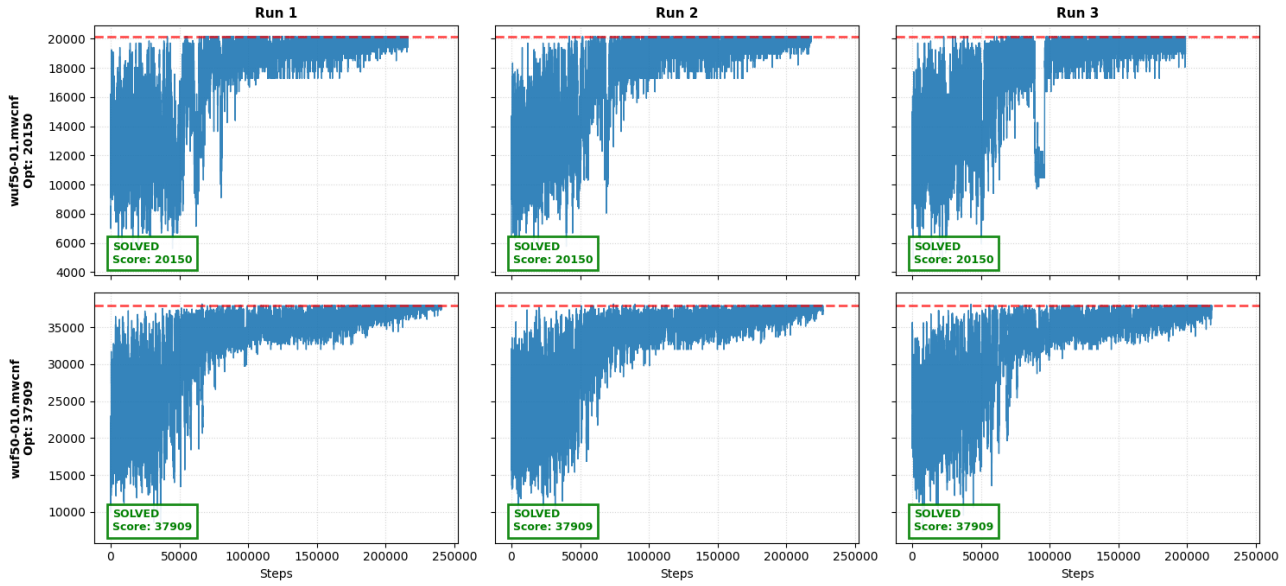
Parametr (kód)	Hodnota
`P0`	0.8
`cooling_coefficient`	0.95
`equilibrium_steps`	6
`fitness_coefficient`	100
`max_steps_without_improvement`	800
`random_flip`	False

5.7 Adaptace parametrů na odlišné instance

Závěrečným krokem White box fáze je ověření generalizačních schopností nalezené konfigurace. Je nutné potvrdit, že parametry optimalizované pro těžkou sadu R (50 proměnných) nejsou *přeucené* (overfitted) a že algoritmus dokáže efektivně škálovat i na instance jiné velikosti či struktury.

5.7.1 Test na odlišném rozložení vah (Sada M a Q)

První test proběhl na sadě `wuf-50-218-M`. Výsledky jsou zobrazeny na Obrázku 4 a v Tabulce 9. Z dat je patrné, že na této sadě dosahuje algoritmus excelentních výsledků (100 % optimalita ve většině běhů). To potvrzuje zajímavou vlastnost sady M: váhy proměnných zde pravděpodobně pozitivně korelují se splnitelností klauzulí. Algoritmus není nucen dělat složité kompromisy mezi validitou a váhou, což indikuje, že robustní nastavení pro sadu R nebrzdí výkon na sadách jednodušších.

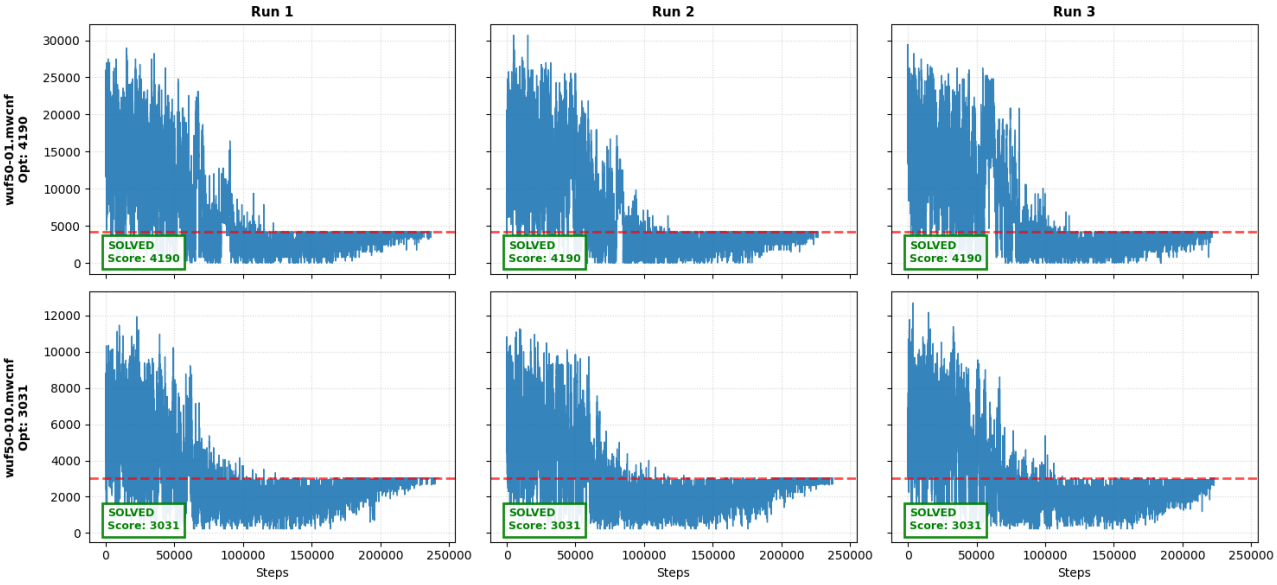


Obrázek 4: Průběh heuristiky na instancích wuf-50-218-M. Rychlá konvergence k optimu.

Tabulka 9: Výsledky pro instance wuf50-M (4 běhy)

Instance	Solved	Optimal	% Opt. (for solved)	Avg Steps
wuf50-0323	4	4	100.00 %	199 143
wuf50-0366	4	4	100.00 %	204 048
wuf50-069	4	4	100.00 %	196 527
wuf50-094	4	4	100.00 %	201 759
wuf50-0861	4	4	100.00 %	203 067
wuf50-0824	3	2	99.90 %	225 630
wuf50-0657	4	4	100.00 %	200 778
wuf50-0286	4	4	100.00 %	202 413
wuf50-0612	4	4	100.00 %	199 797
wuf50-0939	4	4	100.00 %	209 280

Oproti tomu sada `wuf-50-218-Q` (viz Obrázek 5 a Tabulka 10) vykazuje podobné charakteristiky jako trénovací sada R. Zde se projevuje fenomén "zlého prostoru", kdy pro splnění logické formule je nutné nastavit na 0 i proměnné s vysokou váhou. I přesto si heuristika vede velmi dobře – validita (Solved) zůstává 100 % a průměrná odchylka od optima je minimální (často pod 1 %).



Obrázek 5: Průběh heuristiky na instancích wuf-50-218-Q. Náročnější hledání optima oproti sadě M.

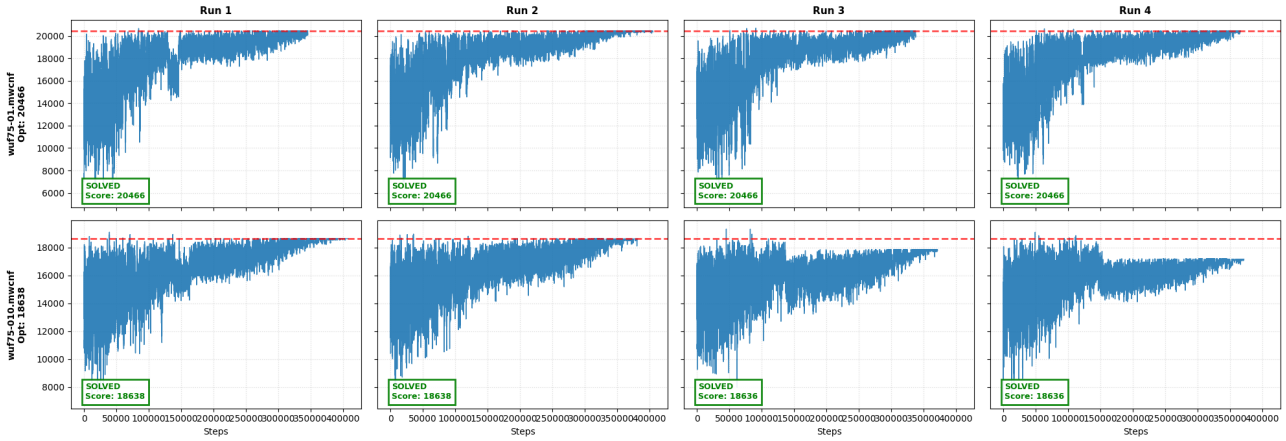
Tabulka 10: Výsledky pro instance wuf50-Q (4 běhy)

Instance	Solved	Optimal	% Opt. (for solved)	Avg Steps
wuf50-0323	4	4	100.00 %	195 219
wuf50-0366	4	4	100.00 %	203 067
wuf50-069	4	4	100.00 %	195 219
wuf50-094	4	3	97.20 %	190 314
wuf50-0861	4	4	100.00 %	194 892
wuf50-0824	4	3	98.97 %	206 337
wuf50-0657	4	3	98.87 %	210 261
wuf50-0286	4	4	100.00 %	207 645
wuf50-0612	4	4	100.00 %	196 527
wuf50-0939	4	4	100.00 %	217 455

5.7.2 Ověření škálovatelnosti (75 proměnných)

Poslední testovací sadou ve White box fázi byla `wuf-75-325-M` (Obrázek 6, Tabulka 11). Cílem bylo ověřit, zda nastavení parametrů odvozené od velikosti instance (např. $equilibrium\_steps \times N_{vars}$) funguje i při nárůstu dimenze problému o 50 %.

Výsledky jsou uspokojivé. Algoritmus spolehlivě nachází validní řešení (sloupec Solved) a velmi se blíží optimu. Nárůst průměrného počtu kroků (z cca 200 tisíc na 300+ tisíc) odpovídá nárůstu složitosti, avšak nedochází ke kolapsu konvergence. To potvrzuje, že adaptivní nastavení teploty a délky kroku je funkční.



Obrázek 6: Průběh heuristiky na větších instancích wuf-75-325-M.

Tabulka 11: Výsledky pro instance wuf75-M (4 běhy)

Instance	Solved	Optimal	% Opt. (for solved)	Avg Steps
wuf75-03	4	4	100.00 %	305 663
wuf75-059	4	4	100.00 %	317 850
wuf75-099	4	3	99.73 %	324 188
wuf75-021	4	4	100.00 %	342 713
wuf75-064	4	0	97.30 %	311 513
wuf75-040	3	2	99.69 %	312 000
wuf75-080	4	3	99.96 %	355 875
wuf75-038	4	4	100.00 %	318 825
wuf75-082	4	4	100.00 %	337 350
wuf75-042	4	4	100.00 %	364 163

Tímto je ověřena robustnost konfigurace na malém vzorku dat. Následuje **Black box fáze**, která podrobí heuristiku statisticky významnému testování na plném rozsahu datových sad (tisíce běhů) pro získání objektivního zhodnocení výkonu.

6 Evaluace (Black box fáze)

V této fázi jsou parametry heuristiky zafixovány na hodnotách určených ve White box fázi. Algoritmus je následně podroben rozsáhlému testování na kompletních datových sadách, které nebyly použity pro ladění.

Vzhledem ke stochastické povaze algoritmu (Simulated Annealing) a přirozené variabilitě obtížnosti jednotlivých instancí je nutné zajistit statistickou robustnost výsledků. Experimentální design proto zahrnuje dva mechanismy pro snížení rozptylu měření:

- **Potlačení randomizace algoritmu:** Každá jednotlivá instance je řešena opakovaně, aby se eliminoval vliv náhodnéhy při konkrétním běhu.
  - Pro sady s 20 a 50 proměnnými: **100 běhů** na instanci.
  - Pro sadu se 75 proměnnými: **50 běhů** na instanci (z důvodu vyšší časové náročnosti).
- **Potlačení variability instancí:** Testování probíhá na velkém vzorku dat, což vyhlazuje rozdíly mezi "lehkými"a "těžkými"instancemi v rámci jedné třídy.
  - Sady 20 a 50 proměnných: 1 000 instancí (celkem tedy 100 000 měření pro každou kategorii).
  - Sada 75 proměnných: 100 instancí (celkem 5 000 měření).

6.1 Sledované metriky

Pro kvantitativní vyhodnocení kvality heuristiky jsou sledovány následující metriky:

Solved (%): Úspěšnost validity. Procento běhů, které nalezly splnitelné řešení, bez ohledu na jeho váhu.

Opt. (%): Úspěšnost optimality. Procento běhů, které nalezly *globální optimum* (maximální možný součet vah při zachování validity).

Steps: Výpočetní náročnost udaná průměrným počtem iterací potřebných k nalezení nejlepšího řešení v daném běhu.

Time (s): Průměrný čistý výpočetní čas na vyřešení jedné instance. Slouží hlavně k porovnání výpočetní náročnosti napříč sadami — je závislé na platformě.

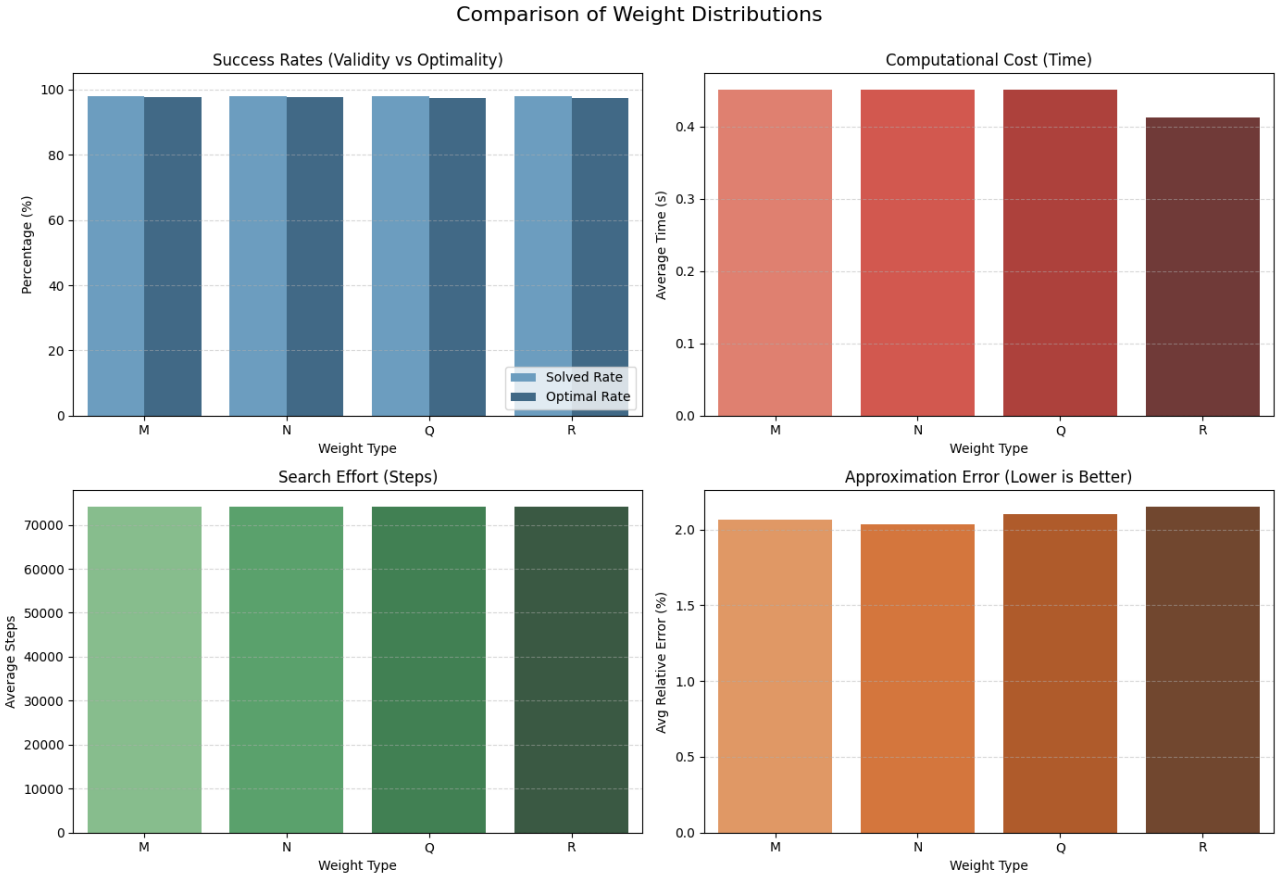
Err (%): Průměrná relativní chyba. Vyjadřuje, o kolik procent je nalezené validní řešení horší než optimum. (Počítá se pouze pro běhy, které našly validní řešení).

6.2 Instance s 20 proměnnými

Evaluaci začínáme nejmenšími instancemi o 20 proměnných. Vzhledem k malé velikosti stavového prostoru (2^{20}) představují tyto úlohy nejjednodušší testovací sadu. Výsledky měření jsou shrnuty v Tabulce 12 a vizualizovány na Obrázku 7.

Tabulka 12: Srovnání typů vah: Instance 20 proměnných (průměr ze 100 000 běhů)

Typ	Solved (%)	Opt. (%)	Time (s)	Steps	Err (%)
M	98.00	97.61	0.4507	74 120	2.06
N	98.03	97.66	0.4511	74 126	2.03
Q	98.07	97.53	0.4511	74 141	2.10
R	98.01	97.50	0.4119	74 144	2.15



Obrázek 7: Porovnání úspěšnosti heuristiky na jednotlivých sadách s 20 proměnnými.

Na těchto datech byly očekávány výsledky blížící se 100 % úspěšnosti. Dosažená úspěšnost okolo 98 % je velmi vysoká, avšak fakt, že algoritmus v cca 2 % případů nenašel optimum, naznačuje, že parametry laděné na složitější instance (50 proměnných) nejsou pro malé instance zcela optimální. Algoritmus pravděpodobně tráví zbytečně mnoho času v oblasti vysokých teplot, což u takto malého prostoru může vést k neefektivitě.

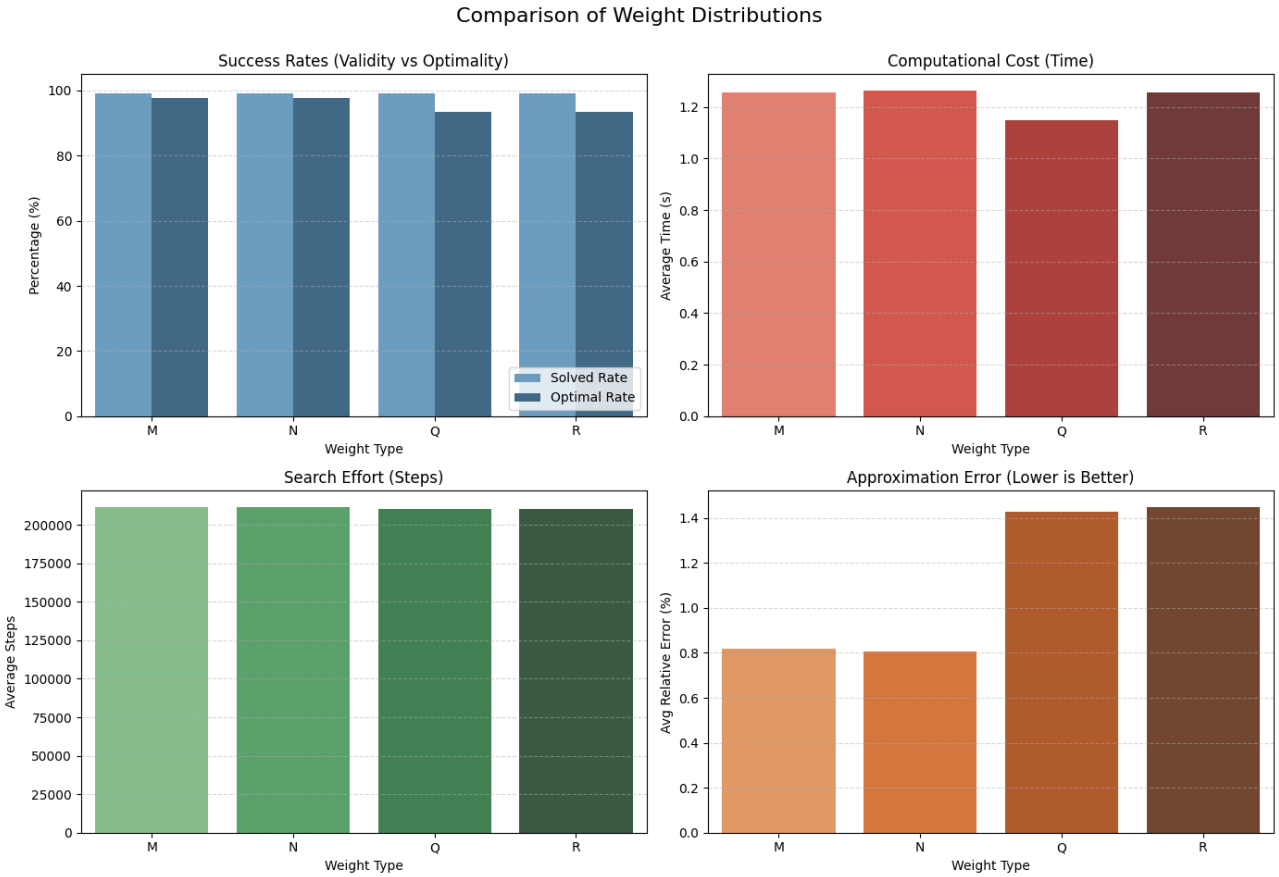
Z hlediska typů vah (M, N, Q, R) jsou rozdíly ve výsledcích zcela zanedbatelné. Algoritmus se na této velikosti chová konzistentně bez ohledu na charakter váhové funkce.

6.3 Instance s 50 proměnnými

Další část evaluace se zaměřuje na instance s 50 proměnnými. Vzhledem k tomu, že právě na této velikostní kategorii (konkrétně sadě R) probíhalo ladění parametrů ve White box fázi, lze očekávat nejvyšší efektivitu heuristiky. Kvantitativní výsledky jsou shrnuty v Tabulce 13 a na Obrázku 8.

Tabulka 13: Srovnání typů vah: Instance 50 proměnných (průměr ze 100 000 běhů)

Typ	Solved (%)	Opt. (%)	Time (s)	Steps	Err (%)
M	99.22	97.78	1.2577	211 560	0.82
N	99.23	97.75	1.2639	211 597	0.81
Q	99.20	93.38	1.1485	210 377	1.43
R	99.17	93.51	1.2573	210 363	1.45



Obrázek 8: Porovnání heuristiky na jednotlivých sadách s 50 proměnnými.

Výsledky potvrzují předpoklad z ladící fáze. Úspěšnost nalezení validního řešení (*Solved*) přesahuje 99 % ve všech kategoriích, což je dokonce lepší výsledek než u menších instancí s 20 proměnnými. To svědčí o tom, že nastavení teploty a délky kroku je pro tuto velikost instancí optimální.

Zajímavý je pohled na úspěšnost v nalézání globálního optima (*Opt.*). Zde se jasně profilují dvě skupiny:

- **Sady M a N:** Zde algoritmus dosahuje vysoké úspěšnosti (cca 97,8 %) a minimální chyby (0,8 %). Tyto instance jsou pro navrženou heuristiku "příznivé".

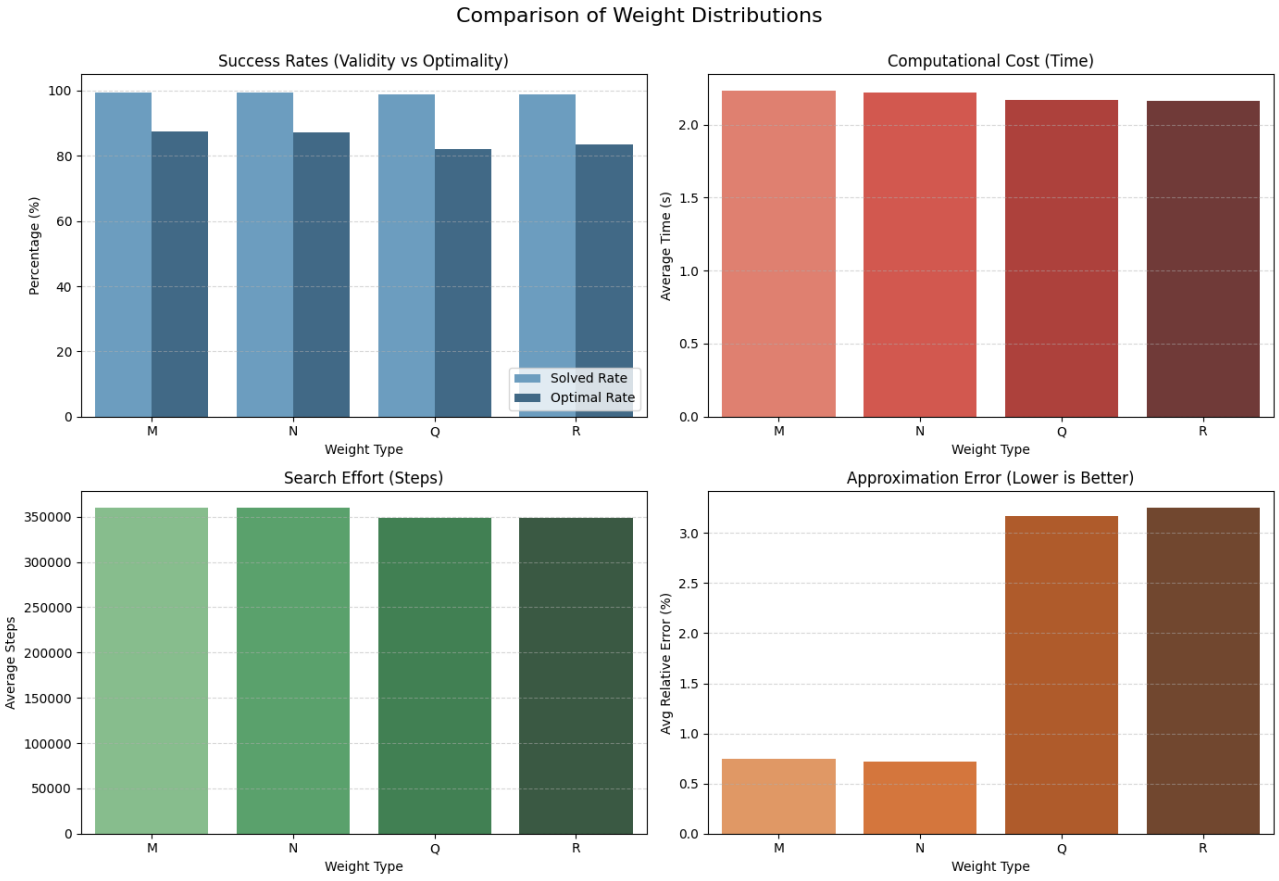
- **Sady Q a R:** Zde úspěšnost klesá k 93,5 % a chyba roste k 1,45 %. Tento pokles potvrzuje, že sady Q a R obsahují strukturálně náročnější rozložení vah, které ztěžuje konvergenci ke globálnímu maximu, přestože validní řešení je nalezeno spolehlivě.

6.4 Instance se 75 proměnnými

Závěrečná část evaluace proběhla na nejnáročnější sadě instancí se 75 proměnnými. Tyto úlohy slouží především k ověření škálovatelnosti algoritmu při dalším exponenciálním nárůstu stavového prostoru. Výsledky jsou zobrazeny na Obrázku 9 a v Tabulce 14.

Tabulka 14: Srovnání typů vah: Instance 75 proměnných (průměr z 5 000 běhů)

Typ	Solved (%)	Opt. (%)	Time (s)	Steps	Err (%)
M	99.42	87.36	2.2328	360 019	0.74
N	99.42	87.12	2.2201	359 912	0.72
R	98.70	83.36	2.1625	348 427	3.25
Q	98.70	82.18	2.1679	348 265	3.17



Obrázek 9: Porovnání heuristiky na jednotlivých sadách se 75 proměnnými.

U těchto instancí se již projevuje jejich vysoká obtížnost, přesto jsou výsledky uspokojivé. Algoritmus si zachovává excelentní schopnost nalézt validní řešení – úspěšnost (*Solved*) neklesá pod 98,7 %. To potvrzuje robustnost nastavení penalizační funkce i pro větší instance.

Znatelný pokles nastává v četnosti nalezení globálního optima (*Opt.*), která se pohybuje mezi 82 % a 87 %. To je u stochastické heuristiky na takto velkém prostoru očekávatelné. Důležité však je, že i když algoritmus nenalezne přesné globální optimum, nalezené řešení je velmi kvalitní. Průměrná relativní chyba (*Err*) se u lehčích sad (M, N) pohybuje pod 1 % a u těžších (Q, R) okolo 3 %. Algoritmus tedy i v případě "neúspěchu"konverguje do velmi hlubokého lokálního optima, které je pro praktické účely často dostatečné.

6.5 Finální zhodnocení

7 Diskuse

V této práci jsem si vyzkoušel navrhnout a nasadit heuristiku simulovaného ochlazování. Z procesu jsem si odesl několik zásadních znalostí: item: Nalezení ideální heuristiky je iterativní proces: mnohokrát se mi stalo, že jsem si myslel že jsem na samotném konci ale při testování bylo zjištěno, že cíl je ještě daleko item: Návrh robustní funkce pro lazení parametrů se vyplatí - původně byly parametry lazeny pouze na nějkolika instancích z nejjednodušší sady. Přirozeně tak nebyla testována heuristika v nejtěžších úkolech item: Pokud jsou data, tak je dobré při whitebox fázi více zapojit různé druhy instancí item: Škálovatelnost heuristiky je klíč - pevně nastavené multiplikativní konstanty často pohoří při změně charakteru instancí - váhy, počet klauzulí apod

V mé práci vnímám jako hlavní poučení důkladnější testování ve whitebox fázi, zejména testovat na různě velkých instancích.