Řešení MWSAT pokročilou iterativní metodou

Lukáš Zima, zimaluk1

Prosinec 2025

1 Úvod

V této zprávě bude dokumentován můj postup při implementaci řešiče pro úlohu maximální vážené splnitelnosti booleovské formule (MWSAT). Zpráva se bude zaměřovat na návrh, nastavení a nasazení heuristiky a na její experimentální vyhodnocení. Konkrétněji se zpráva bude skládat ze dvou dílčích částí:

- White-box fáze, která bude pojednávat o implementaci a nastavení heuristiky.
- Black-box fáze, ve které proběhne závěrečné vyhodnocení heuristiky.

2 Zadání

Problém vážené splnitelnosti Booleovy CNF formule s maximem jedniček

Dáno: vektor X n proměnných

$$X = (x_1, \ldots, x_n), \quad x_i \in \{0, 1\},$$

Booleova formule $F(X)$ těchto proměnných v konjunktivní normální formě o m klauzulích a dále pro každou proměnnou x_i váha $w(x_i)$.
Sestrojit: ohodnocení

$$Y = (y_1, \ldots, y_n), \quad y_i \in \{0, 1\},$$

proměnných X takové, že

$$F(Y) = 1$$

a součet vah proměnných ohodnocených jedničkou

$$\sum_{i:y_i=1} w(x_i)$$

je maximální. [3]

3 White-box fáze

3.1 Základní implementace

Pro tento problém jsem se rozhodl použít metodu simulovaného ochlazování. Jako implementační jazyk jsem si zvolil C++ a metodu jsem implementoval převážně podle přednášky. Pro alternativní výklad jsem využil platformu Baeldung. [1, 4]

Pseudokód mé implementace:

$s \leftarrow$ náhodné počáteční řešení;
$E(s) \leftarrow \textsc{Energy}(s)$;
$s_{\text{best}} \leftarrow s$;
$E_{\text{best}} \leftarrow E(s)$;
$T \leftarrow T_{\text{start}}$;
while $\neg$ *FROZEN*(T) **do**
  $iter \leftarrow 0$;
  while $\neg$ *EQUILIBRIUM*$(iter)$ **do**
    $s' \leftarrow \textsc{Neighbour}(s)$;
    $\Delta E \leftarrow \textsc{Energy}(s') - E(s)$;
    if $\Delta E \leq 0$ ***or*** *RANDOM*$(0, 1) < e^{-\Delta E/T}$ **then**
      $s \leftarrow s'$; $E(s) \leftarrow E(s')$;
      if $E(s) < E_{best}$ **then**
        $s_{\text{best}} \leftarrow s$; $E_{\text{best}} \leftarrow E(s)$;
      end
    end
    $iter \leftarrow iter + 1$;
  end
  $T \leftarrow \textsc{cool}(T)$;
end
return s_{best};

Algorithm 1: Simulované ochlazování

- s je aktuální řešení,
- s_{best} je nejlepší dosud nalezené řešení,
- $E(s)$ je energie řešení s, kterou metoda minimalizuje,
- T je aktuální teplota, T_{start} počáteční teplota,
- $\textsc{cool}(T)$ je chladicí funkce, která snižuje teplotu ($T \leftarrow \alpha \cdot T$),
- $\textsc{frozen}(T)$ určuje, zda je systém „zmrzlý" ($T \leq T_{\text{min}}$),
- $\textsc{equilibrium}(iter)$ určuje, zda už byl na dané teplotě proveden dostatek iterací,
- $\textsc{Neighbour}(s)$ vrací náhodného souseda aktuálního řešení,

3.2 Energie a penalizace

Z pseudokódu je vidět, že důležitou roli zastává funkce ENERGY. Tato funkce musí být schopna co nejefektivněji "ohodnotit" jednotlivé stavy, aby heuristika měla šanci konvergovat k dobrému (ideálně optimálnímu) řešení. Ze zadání úlohy plyne, že chceme zároveň:

- Najít ohodnocení, které **řeší** vstupní formuli a
- jeho součet vah proměnných ohodnocených 1 je **maximální**.

Funkce ENERGY by tedy měla brát v potaz obě podmínky. Navrhl jsem ji tak, aby upřednostňovala splnitelnost formule a teprve potom maximalizovala součet vah proměnných nastavených na 1. Konkrétněji:

$$ENERGY(s) = \text{unsat}(s) \cdot \text{penalty} - \text{weight}(s),$$

kde $\text{unsat}(s)$ je počet nesplněných klauzulí pro řešení s a $\text{weight}(s)$ je součet vah proměnných nastavených na 1. Penalty penalizuje za nesplněné klauzule a díky tomu, že chci prioritizovat splnění formule, jsem ji navrhl poměrně přísně. Konkrétně:

$$\text{penalty} = \sum_i w_i$$

Jedná se tedy o součet všech vah. Tímto docílím nezáporné energie pro všechny ohodnocení, které neřeší formuli. Jako důsledek mohou při porovnání dvou řešení s_1 a s_2 nastat následující situace:

- Obě řešení formuli neřeší: upřednostní se řešení s menším počtem nesplněných klauzulí, a pokud je tento počet stejný, tak to s větším součtem vah.
- Jedno řešení formuli řeší a druhé ne: splňující řešení má vždy nižší energii než to nesplňující, takže se vybere to.
- Obě řešení formuli řeší: rozhodne už jen součet vah proměnných nastavených na 1, upřednostní se to s vyšším součtem.

3.3 Počáteční nastavení a měření

Po implementaci výše zmíněné heuristiky přišla řada na otestování a vyhodnocení účinnosti. Rozhodl jsem se řešič otestovat na všech verzích (M, N, Q, R) jedné instance wuf20-01. Řešič jsem na každou verzi spustil 1000-krát s tímto počátečním nastavením:

- Počáteční teplota $T_{\text{start}} = 20$,
- činitel ochlazování $\alpha = 0{,}95$,
- minimální teplota $T_{\text{min}} = 10^{-5}$,

- počet iterací na teplotu $itersPerTemp = 100$.

Table 1: Výsledky simulovaného ochlazování pro verze instance wuf20-01

	M	N	Q	R
Podíl uspokojivých řešení vůči všem (%)	48,3	46	11,1	9,1
Podíl optimálních řešení vůči všem (%)	8,4	4,2	5,3	4,8
Průměrný počet neuspokojených klauzulí	0,74	0,87	1,72	1,78
Průměrná ztráta váhy uspokojivého vůči optimu (%)	9,83	11,09	39,04	35,33

Jak je patrné z tabulky, řešič není moc efektivní. Očekávaná prioritizace nalezení nějakého uspokojivého řešení se dostavila pouze v omezené míře, a pro těžší instance Q a R dosáhl řešič opravdu slabých výsledků. Jedno pozitivum z tabulky vyčíst lze - počet neuspokojených klauzulí je u neuspokojivých řešení velice malý, tudíž se vždy dostaneme alespoň "blízko".

3.4 Pokusy o vylepšení

K vylepšení tohoto výsledku mě intuitivně napadlo zpřísnit penalizaci vynásobením nějakou konstantou, tedy:

$$\text{penalty} = k \sum_i w_i$$

Po otestování řešiče stejným postupem (pro $k = 2, 5, 10$) jsem ale zjistil, že tento krok nevedl k žádnému podstatnému zlepšení (ani zhoršení) na dané instanci. Výsledky se ani podstatně nelišily při měnění vstupních parametrů jako počáteční teplota a počet iterací na teplotu. Tento fakt mě překvapil a donutil se více zaměřit na jednotlivé části heuristiky a jejich optimalizaci.

3.4.1 Získání sousedního stavu

Momentálně funkce NEIGHBOUR funguje velmi naivně - u aktuálního stavu náhodně vybere jednu proměnnou a její hodnotu prohodí. Tento přístup se z principu nesnaží záměrně přiblížit lepšímu řešení, protože ignoruje strukturu nesplněných klauzulí.

Alternativní přístup, inspirovaný heuristikou probSAT, je omezit výběr pouze na proměnné, které se vyskytují v některé z nesplněných klauzulí. K dalšímu potlačení náhody se z těchto proměnných náhodně vybere až n kandidátů na flip, u kterých se spočítá, jak by se změnil počet splněných klauzulí. [2] Konkrétně:

1. Nechť $U(s)$ je množina nesplněných klauzulí pro stav s a $\text{sat}(s)$ je počet splněných klauzulí.
2. Náhodně vybereme klauzuli $C \in U(s)$.

3. Z klauzule C náhodně vybereme $K = \min(n, |C|)$ různých proměnných $x_{i_1}, \ldots, x_{i_K}$.

4. Pro každého kandidáta x_{i_k} vytvoříme stav $s^{(k)}$, který se od s liší pouze flipem dané proměnné, a spočítáme $\text{sat}(s^{(k)})$.

5. Vybereme kandidáta, pro kterého je $\text{sat}(s^{(k)})$ největší, a ten ve stavu s skutečně flipneme.

Původní přístup náhodného flipu jsem ponechal pro případy, kdy už jsou všechny klauzule splněné. Hodnotu parametru n jsem vybíral experimentálně mezi 2, 3 - klauzule z datasetů mají maximálně 3 literály. Zkusil jsem znovu otestovat na instanci wuf20-01:

Pro $n = 2$:

	M	Q
Podíl uspokojivých řešení (%)	31,9	16,1
Podíl optimálních řešení (%)	9,9	8,6
Průměrný počet neuspokojených klauzulí	1,3	1,73

Pro $n = 3$:

	M	Q
Podíl uspokojivých řešení (%)	35,2	18,8
Podíl optimálních řešení (%)	12,4	8,2
Průměrný počet neuspokojených klauzulí	1,22	1,72

Z výsledků je vidět, že jsme si pohoršili u verze M, ale polepšili u Q. Obecně se tato změna jeví jako zhoršení - heuristika se pravděpodobně častěji zasekává na nějakém lokálním extrému. Intuitivně by ale fungovat měla, takže tuto myšlenku prozatím zahazovat nebudu - například ji do budoucna zkusím spustit s nějakou pravděpodobností nebo úpravou.

Rozhodl jsem se zvizualizovat jeden celý běh heuristiky na dané instanci, aby mi pomohl pochopit, kde se děje chyba:

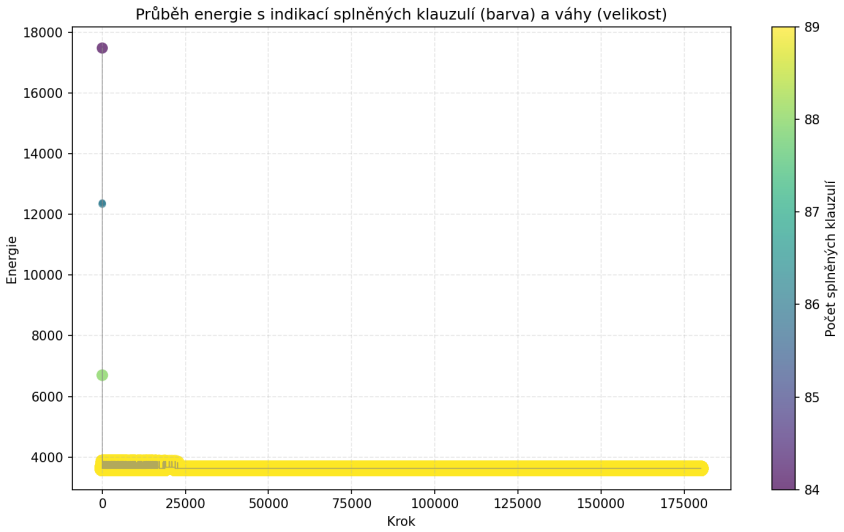

Figure 1: Vizualizace běhu heuristiky na instanci wuf20-01-M

Z grafu je zřetelně vidět, že je penalizace extrémně přísná. Heuristice skoro vůbec neumožňuje fázi diverzifikace (prozkoumávání stavového prostoru). Je tedy třeba lépe zaručit, aby heuristika zkoušela více stavů a je tedy nutné přehodnotit výpočet penalizace a energie.

3.4.2 Další pokusy

Po předchozím zjištění jsem zkoušel několik přístupů. Jedním z nich bylo určit penalizaci jako průměr vah či jejich maximum. Dalším bylo změnit výpočet energie na:

$$ENERGY(s) = \text{unsat}(s) \cdot \text{penalty} + (\sum_i w_i - \text{weight}(s)),$$

který pouze docílil toho, že cílenou hodnotou minimalizace je 0. Tuto úpravu jsem se rozhodl ponechat, jelikož práce s kladnou hodnotou reprezentující vzdálenost od optima je přirozenější a lépe interpretovatelná.

Všechny tyto pokusy vedly na porovnatelné či horší výsledky. Heuristiky se pořád brzo v běhu zasekávaly a neprozkoumávaly dostatek stavů. Proto jsem se rozhodl zavést úplný restart heuristiky v případě, kdy se zasekne na dostatečně dlouhou dobu. Tuto dobu jsem vypočítal jako podíl všech kroků, které heuristika provede. Konkrétněji vím, že:

$$T_{\text{min}} \approx T_{\text{start}} \cdot \alpha^K.$$

Po úpravě lze dostat:

$$K = \left\lceil \frac{\ln T_{\min} - \ln T_{\text{start}}}{\ln \alpha} \right\rceil .$$

Každé teplotě odpovídá pevný počet iterací `itersPerTemp`, takže celkový počet kroků jedním průchodem heuristiky je:

$$N = K \cdot \texttt{itersPerTemp}.$$

Nyní stačilo experimentálně vyhodnotit, jaký podíl $\frac{N}{P}$ je vhodný. Například $P = 2$ znamená, že se heuristika restartuje maximálně jednou a to pokud se nic nezmění po dobu poloviny běhu. Provedl jsem stejný experiment jako v předešlých případech pro $P = 2, 5, 10$.

Pro $P = 2$:

	M	N	Q	R
Podíl uspokojivých řešení (%)	67,2	57,7	1,5	1,3
Podíl optimálních řešení (%)	20	14,2	0	0

Pro $P = 5$:

	M	N	Q	R
Podíl uspokojivých řešení (%)	75,4	67,3	0	0,3
Podíl optimálních řešení (%)	34,1	26,6	0	0

Pro $P = 10$:

	M	N	Q	R
Podíl uspokojivých řešení (%)	82	76,5	0	0
Podíl optimálních řešení (%)	44,6	39,5	0	0

Z tabulek je vidět, že čím vyšší je P, tím lepší výsledky má heuristika u "lehčí" verze M a N a tím horší výsledky má u "těžší" Q a R. Celkově bych tedy tento přístup hodnotil negativně a pravděpodobně ho v této podobě nevyužiju, jelikož dosahuje horších výsledků u některých verzí. Je možné, že v kombinaci s jinými mechanizmy začne dávat smysl.

Jako další přístup mě napadlo, že by se běh řešiče rozdělil na 2 fáze:

1. Hledání nějakého splnitelného řešení
2. Maximalizace váhy napříč splnitelnými řešeními

Upravil jsem řešič tak, aby se řídil touto logikou. Konkrétně v první fázi jako energii používá pouze počet nesplněných klauzulí a v druhé potom váhu, přičemž už nepřipouští žádné nesplňující ohodnocení. Po otestování na zmíněné instanci

se řešič jevil nadějně - vždy našel nějaké splňující ohodnocení a poměr optimálních se pohyboval okolo 20-40%. Po vizualizaci běhu mi ale bylo jasné, že tudy cesta nepovede:

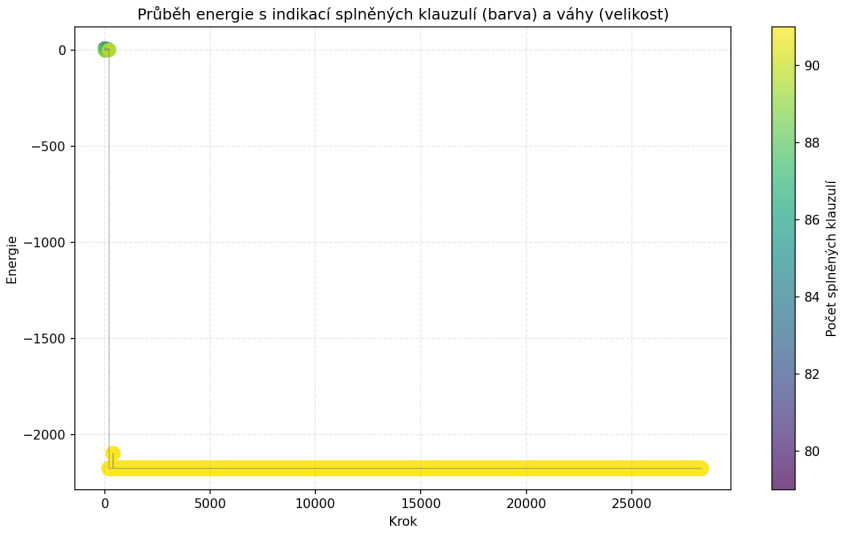

Figure 2: Vizualizace běhu dvou fázové heuristiky na instanci wuf20-01-M

Na vizualizaci je vidět, kdy heuristika přepne z fáze 1 do fáze 2. Bohužel ve fázi dvě se rychle uklidní a neprohledává dále stavový prostor. To je způsobeno tím, že se ve druhé fázi řádně přijme soused jedině, pokud sám splňuje formuli a má lepší ohodnocení vah. Toto omezení je moc striktní a zabraňuje spolehlivému hledání optimálních vah.

3.4.3 Úspěšné úpravy

Z vizualizací běhu algoritmu jsem si uvědomil, že se pořád dokola točím okolo toho samého problému – prakticky nikdy nedocházelo k uplatnění stochastické části simulovaného ochlazování. Ačkoliv algoritmus obsahoval podmínku pro přijetí horšího řešení s pravděpodobností $P = e^{-\frac{\Delta E}{T}}$, v praxi tato větev nebyla téměř nikdy vybrána.

Příčinou byla velikost rozdílu energií ΔE. Jelikož váhy proměnných nabývají hodnot v různých řádech (stovky, tisíce…), i malá změna v ohodnocení může způsobit obrovské ΔE. Konkrétně pro $T = 20$ byl exponent $-\frac{\Delta E}{T}$ příliš malé číslo a tím pádem se pravděpodobnost blížila nule. Algoritmus se tak jednoduše zasekl v lokálním extrému.

Kromě tohoto klíčového zjištění jsem se navrátil k mé dosavadní implementaci NEIGHBOUR a uvědomil si, že je také moc omezující, jelikož se ne vždy vyplatí vybrat tu zdánlivě nejlepší možnost. Implementaci jsem následně

předělal podle vzoru algoritmu WalkSAT, který tento nedostatek řeší přidáním elementu náhody. [5]

Normalizace energie Aby byl algoritmus odolný vůči různým rozsahům vah v instancích, zavedl jsem normalizaci energetické funkce. Původní vypočtená energie je nyní dělena průměrnou váhou všech proměnných $\bar{w}$:

$$ENERGY_{norm}(s) = \frac{ENERGY_{raw}(s)}{\bar{w}}$$

Výpočet energie jsem zanechal stejný a `penalty` jsem ponechal jako $\sum_i w_i$. Díky této úpravě se rozdíly energií ΔE pohybují v rozumných číslech. Jako důsledek se může můj algoritmus doopravdy chovat jako simulované ochlazování a tím pádem někdy přijímat i dočasně horší stavy.

Vylepšení metody generování sousedních stavů Provedl jsem pár úprav v metodě generování sousedů podle vzoru WalkSAT. Metoda náhodně zvolí jednu nesplněnou klauzuli a v ní vybere proměnnou k otočení na základě parametru p:

- S pravděpodobností p vybere literál v klauzuli zcela náhodně, což vede k diverzifikaci. Tento krok jsem v předešlé verzi neměl.
- S pravděpodobností $1-p$ vybere literál, jehož otočení minimalizuje celkový počet nesplněných klauzulí, což vede k intenzifikaci.

Tato úprava by měla zajistit, že se řešič pořád umí zaměřit na úpravu nesplněných klauzulí, ale nově by měl být schopnější unikat z lokálních optim. Parametr p jsem vybíral experimentálně z $\{0,2; 0,4; 0,5\}$ a výsledky pro instanci wuf20-01 jsou:

Pro $p = 0,2$:

	`M`	`N`	`Q`	`R`
Podíl uspokojivých řešení (%)	88,8	89,8	86,6	87,8
Podíl optimálních řešení (%)	45,8	45,4	50,4	48,4

Pro $p = 0,4$:

	`M`	`N`	`Q`	`R`
Podíl uspokojivých řešení (%)	96,6	96,7	96,1	96,2
Podíl optimálních řešení (%)	54,8	55,9	60,1	59,5

Pro $p = 0,5$:

	M	N	Q	R
Podíl uspokojivých řešení (%)	97,5	98,2	99,2	98,6
Podíl optimálních řešení (%)	62,0	61,2	63,4	59,3

Vypadá to, že se důsledkem úprav výsledky řešiče značně zlepšily, obzvlášť při $p = 0,5$. Abych získal komplexnější přehled o kvalitě heuristiky před tím, než začnu upravovat vstupní parametry jako je počáteční teplota, jsem se rozhodl provést robustnější měření - řešič jsem spustil na všech 100 instancích (a všech jejich verzích - M, N, Q, R) sady wuf-20-91R. Na každou z dohromady 400 instancí byl řešič spuštěn 100 krát.

Table 2: Výsledky simulovaného ochlazování pro dataset wuf-20-91R

	M	N	Q	R
Podíl uspokojivých řešení (%)	87	86,9	83,2	83,5
Podíl optimálních řešení (%)	78,3	78,6	69,2	69,2
Průměrný počet neuspokojených klauzulí	0,14	0,14	0,18	0,19
Průměrná ztráta váhy uspokojivého vůči optimu (%)	1,38	1,35	4,71	4,84

3.5 Nastavení počáteční teploty

Základní podobu počáteční teploty T_{start} jsem odvodil následovně:

Vím, že:

$$P_0 = e^{-\frac{\Delta E}{T_{start}}}$$

Úpravou rovnice vyjádřím teplotu jako:

$$\ln(P_0) = -\frac{\Delta E}{T_{start}} \implies T_{start} = -\frac{\Delta E}{\ln(P_0)}$$

Z definice $ENERGY_{norm}$ plyne, že při přechodu z lepšího stavu do horšího o jednu proměnnou, lze odhadnout rozdíl energií jako:

$$\Delta E \approx n + 1,$$

Jelikož při zhoršení o 1 proměnnou platí:

$$\Delta E \approx \frac{\overbrace{1 \cdot \sum_{i=1}^{n} w_i}^{\text{změna penalizace}} + \overbrace{\bar{w}}^{\text{změna váhy}}}{\bar{w}} = \frac{n \cdot \bar{w} + \bar{w}}{\bar{w}} = n + 1$$

Z toho tedy plyne:

$$T_{start} = -\frac{n+1}{\ln(P_0)}$$

K určení parametru P_0 jsem provedl stejné robustní měření jako minule na sadě wuf-20-91R. Uvažoval jsem hodnoty $P_0 = \{0,5; 0,6; 0,7; 0,8; 0,9\}$. Výsledky jednotlivých verzí M, N, Q, R jsem pro přehlednější vizualizaci zprůměroval, jelikož se lišily pouze o desetiny procent.

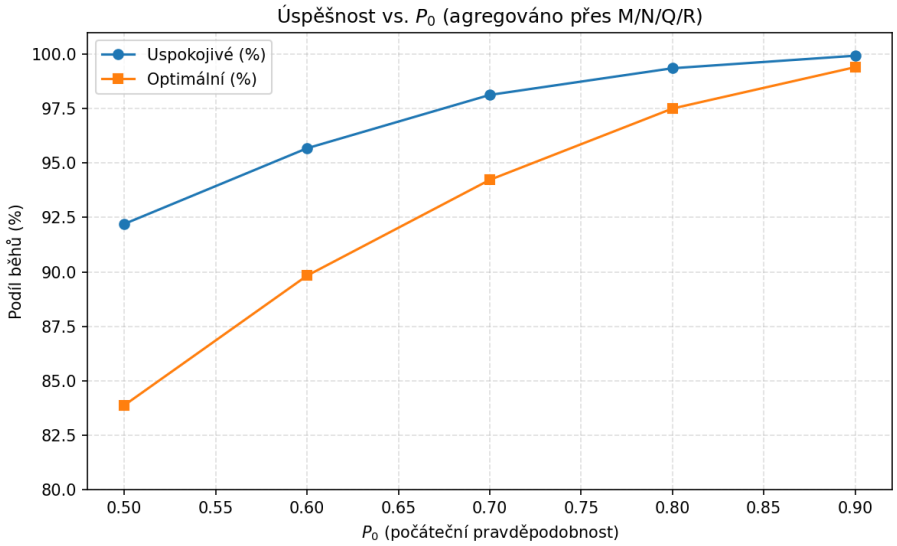

Figure 3: Měření hodnoty P_0

Jak je vidět, nejlepších výsledků dosahovala heuristika s velice vysokým $P_0 = 0,9$. Za příčinu této poněkud překvapivé skutečnosti dávám fakt, že můj odhad typického ΔE byl pravděpodobně podhodnocený a v realitě je vyšší, tudíž jej vyšší hodnoty P_0 kompenzují.

3.6 Nastavení konečné teploty

K určení hodnoty konečné (minimální) teploty T_{min} lze využít stejné odvození jako při nastavování počáteční teploty - jenom místo počáteční pravděpodobnosti P_0 nyní experimentálně určím konečnou pravděpodobnost přijetí horšího stavu P_{end}.

Pro konečnou pravděpodobnost jsem bral v potaz hodnoty $P_{end} = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Řešič jsem spustil na všechny instance v datasetech wuf20-91R-Q a wuf50-218R-M, přičemž pro každou instanci byl řešič spuštěn 100 krát.

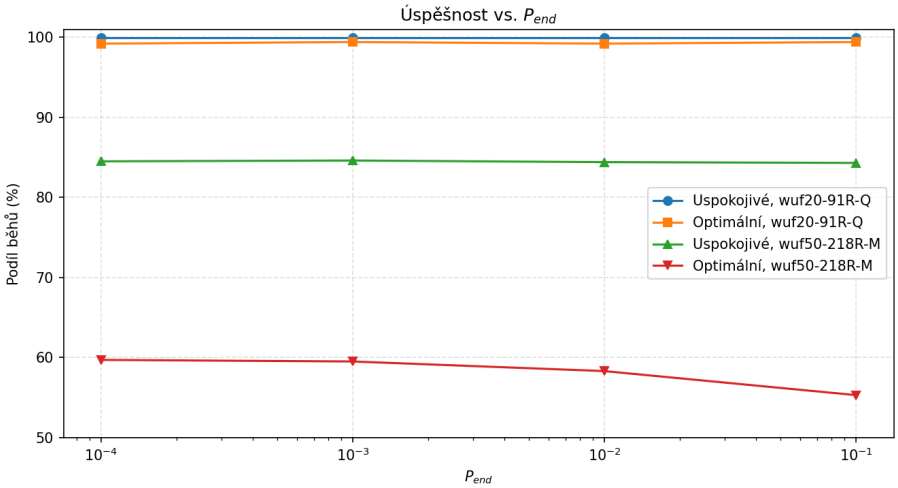

Figure 4: Měření hodnoty P_{end}

U instancí o 20 proměnných byl řešič prakticky stejně efektivní pro všechny z hodnot P_{end}. U instancí s 50 proměnnými je ale vidět zhoršující trend s rostoucí konečnou pravděpodobností. U instancí s ještě větším počtem proměnných očekávám to samé chování. Proto jsem se rozhodl nastavit konečnou pravděpodobnost na nejnižší zkoumanou hodnotu $P_{end} = 10^{-4}$.

3.7 Nastavení ekvilibria a koeficientu chlazení

Z přednášky vyplývá, že délka ekvilibria N a koeficient chlazení α spolu úzce souvisí. Pro fixní celkový počet kroků platí, že změna N vyžaduje adekvátní úpravu α tak, aby bylo dosaženo požadované teploty v konkrétním kroku. Zároveň je uvedeno, že v tzv. „rozumném rozmezí“ má manipulace s délkou ekvilibria stejný efekt na průběh ochlazování jako změna koeficientu α. [4]

Ačkoliv by tedy teoreticky stačilo zafixovat délku ekvilibria (např. $N = n$) a ladit pouze α, což jsem původně prováděl, tak jsem se později rozhodl pro úplnost experimentu zkoumat vliv obou parametrů.

Délku ekvilibria volím úměrně velikosti instance:

$$N = k \cdot n, \quad \text{kde } k \in \{1, 2, 3\} \text{ a } n \text{ je počet proměnných.}$$

Pro koeficient chlazení α jsem zvolil sadu hodnot:

$$\alpha \in \{0,80; 0,90; 0,92; 0,95; 0,98; 0,99\}$$

Experiment jsem provedl na stejných datech jako ten předchozí a zajímala mě primárně optimální úspěšnost a počet kroků.

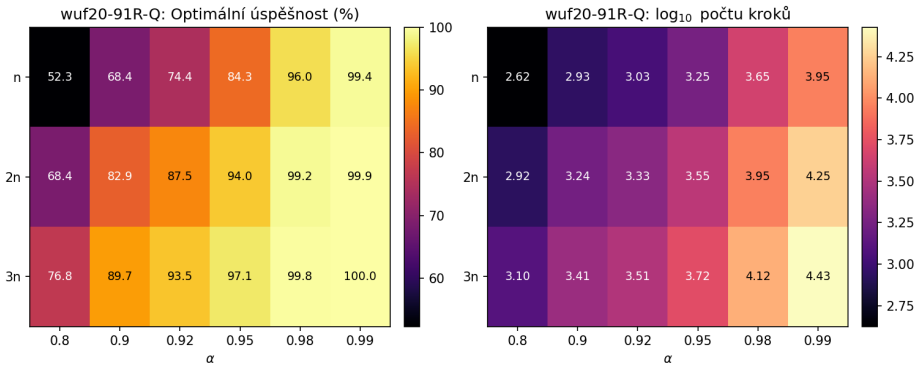

Figure 5: Heatmapa pro sadu wuf20-91R-Q

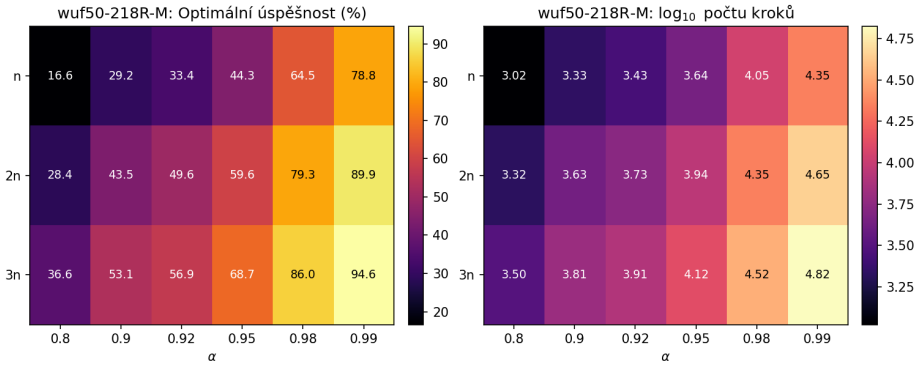

Figure 6: Heatmapa pro sadu wuf50-218R-M

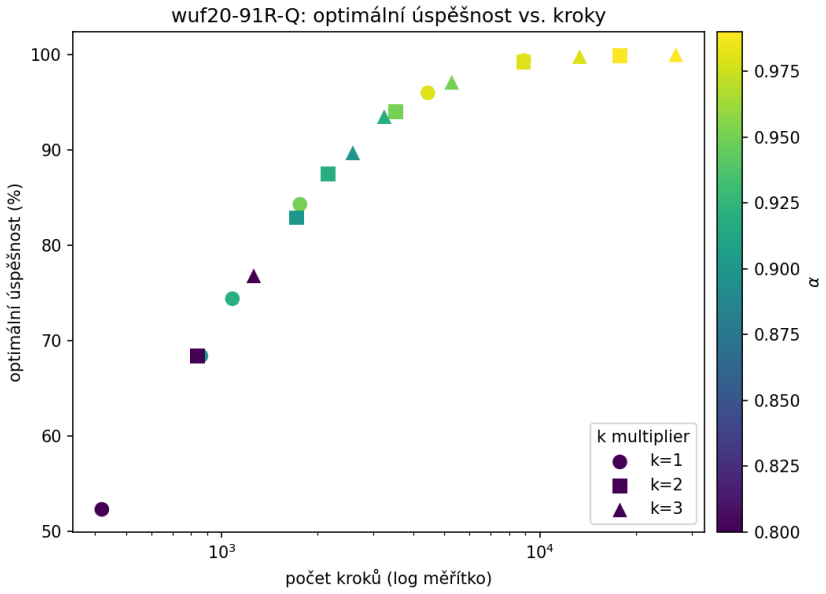

Figure 7: Porovnání opt. úspěšnosti s kroky pro wuf20-91R-Q

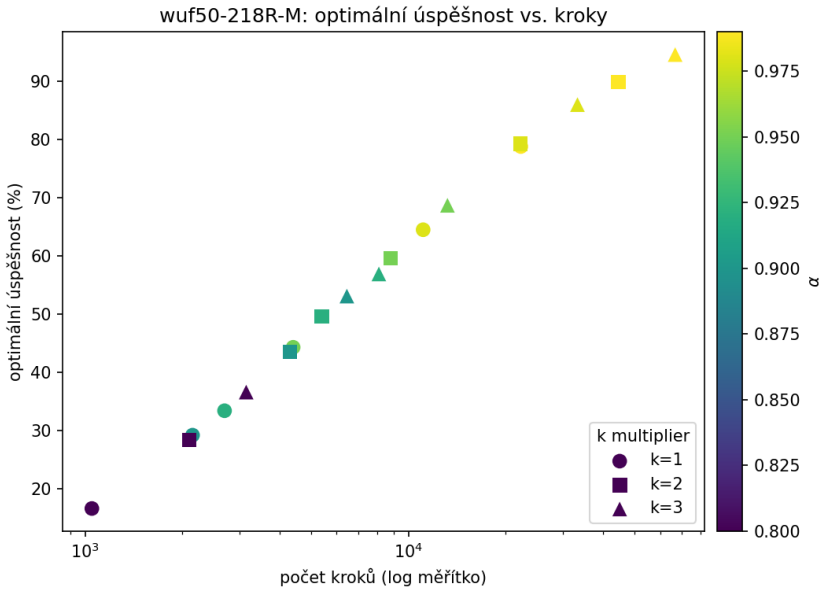

Figure 8: Porovnání opt. úspěšnosti s kroky pro wuf50-218R-M

Z výsledků je patrné, že se zvyšujícím počtem kroků roste úspěšnost heuristiky. Zároveň je nutné podotknout, že pro nejvyšší (a nejúspěšnější) $N = 3n$ a $\alpha = 0,99$ běžel experiment relativně dlouhou dobu. Na druhou stranu je skok v nalezených optimálních řešeních pro sadu s 50 proměnnými natolik výrazný, že jsem se rozhodl dané parametry takto nastavit. Pro kontext: Jeden běh řešiče na instanci o 50 proměnných s $N = 3n$ a $\alpha = 0,99$ trval $\approx 0,12s$, což považuji za přijatelné. Zároveň věřím, že pro instance s ještě více proměnnými bude takto nastavený řešič škálovat lépe - tuto domněnku si částečně ověřím v následující sekci kontrolního měření.

3.8 Kontrolní měření

Ještě než takto nastavený řešič využiji pro black-box fázi, rozhodl jsem se provést poslední kontrolní měření, kterým bych případně mohl odhalit nedostatky řešiče, primárně pro složitější instance, na kterých jsem řešič doposud tolik netestoval.

Řešič jsem spustil na prvních 100 instancí z datových sad wuf20-91-N, wuf20-91-R, wuf50-218-N, wuf50-218-R, wuf75-325-N a wuf75-325-R. Pro každou instanci běžel řešič 20 krát a zkoumal jsem podíl uspokojivých řešení, podíl optimálních řešení a průměrnou ztrátu váhy uspokojivého vůči optimu.

Table 3: Kontrolní měření

	`Uspokojivé (%)`	`Optimální (%)`	`Ztráta váhy (%)`
Sada wuf20-91-N	100	100	0
Sada wuf20-91-R	100	99,9	0,02
Sada wuf50-218-N	97,3	91,3	0,1
Sada wuf50-218-R	97,1	86,3	2,06
Sada wuf75-325-N	93,5	62,4	1,22
Sada wuf75-325-R	93,7	56	6,88

Nechal jsem si také vizualizovat běh řešiče pro instanci wuf75-01 ze sady wuf75-325-M.

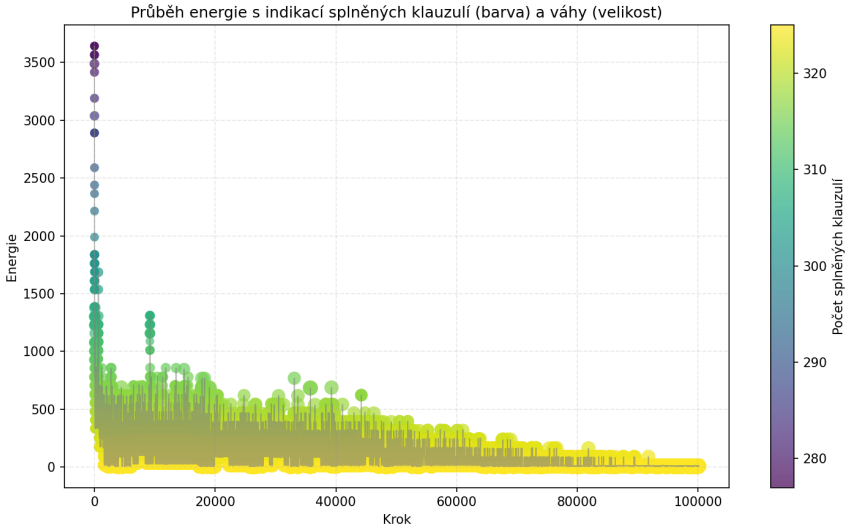

Figure 9: Vizualizace běhu heuristiky na instanci wuf75-01

Výsledky měření považuji za uspokojivé a takto nastavený řešič ponechám. Pro instance s menším počtem proměnných než 75 se řešič zdá být poměrně efektivní. Pro instance s více proměnnými pak poměr především optimálních řešení razantně klesá. Na druhou stranu se i přesto drží nad 50% případů a průměrný rozdíl vah (ztráta vůči optimálnímu řešení) se drží v řádu jednotek procent, tedy řešič není obvykle moc "daleko".

Vizualizace běhu heuristiky vypadá poměrně standardně a ilustruje plynulý přechod od počáteční fáze diverzifikace, kde je vysoký rozptyl hodnot způsoben přijímáním horších stavů, k závěrečné fázi intenzifikace, kde se řešič postupně stabilizuje.

4 Black-box fáze

4.1 Nastavení heuristiky

Pro závěrečnou Black-box fázi testování byl na základě poznatků a experimentů z White-box fáze sestaven finální řešič s následující konfigurací a parametry:

1. **Metoda a energetická funkce:** Řešič využívá metodu simulovaného ochlazování. Energetická funkce je normalizovaná průměrnou váhou ($\overline{w}$) a konstruovaná tak, aby prioritizovala splnitelnost formule (penalizace rovna součtu všech vah) a následně maximalizovala váhu splněných proměnných:

$$ENERGY(s) = \frac{\text{unsat}(s) \cdot \sum w_i + (\sum w_i - \text{weight}(s))}{\overline{w}}$$

2. **Generování sousedů:** Byla implementována strategie inspirovaná algoritmem WalkSAT. V každém kroku je náhodně vybrána nesplněná klauzule. V rámci ní se s pravděpodobností $p = 0{,}5$ vybere literál k otočení zcela náhodně (diverzifikace) a s pravděpodobností $1 - p$ se vybere literál minimalizující počet nesplněných klauzulí (intenzifikace). Pokud je řešení splnitelné, volí se náhodný flip.

3. **Počáteční teplota (T_{start}):** Teplota je vypočítávána dynamicky pro každou instanci na základě odhadovaného rozdílu energií ($\Delta E \approx n + 1$) a požadované počáteční pravděpodobnosti přijetí horšího stavu P_0. Na základě experimentů byla tato pravděpodobnost nastavena na $P_0 = 0{,}9$.

$$T_{start} = -\frac{n+1}{\ln(0{,}9)}$$

4. **Parametry ochlazování:** Byl zvolen rozvrh ochlazování s koeficientem $\alpha = 0{,}99$. Délka ekvilibria (počet kroků na jedné teplotní hladině) je nastavena lineárně v závislosti na počtu proměnných instance (n) jako $N = 3 \cdot n$. Tato kombinace se v testech ukázala jako nejúspěšnější pro hledání optimálních řešení, byť za cenu vyššího výpočetního času.

5. **Minimální teplota (T_{min}):** Algoritmus se zastaví, když teplota klesne pod úroveň, která odpovídá velmi malé pravděpodobnosti přijetí zhoršujícího stavu. Tato koncová pravděpodobnost byla experimentálně stanovena na $P_{end} = 10^{-4}$.

4.2 Výsledky

Řešič jsem spustil na všechny verze (M, N, Q, R) všech datových sad s existujícím přiloženým řešením (.dat soubory). Jedná se o sady wuf20-91, wuf36-157, wuf50-218 a wuf75-325. Na každou instanci byl řešič spuštěn 100 krát a výsledné metriky prezentované v tabulce vznikly dvoustupňovou agregací. Nejprve byly pro každou instanci zprůměrovány hodnoty ze všech jejích běhů a následně byl vypočítán aritmetický průměr přes všechny instance v dané verzi datové sady. Měřil jsem následující metriky:

- **Úspěšnost (%):**
  - *Splnitelné:* Podíl běhů, kdy řešič nalezl nějaké platné řešení splňující všechny klauzule.
  - *Optimální:* Podíl běhů, kdy řešič nalezl optimální řešení dle přiloženého .dat souboru.
- **Ztráta váhy (%):** Metrika vyjadřující, jak daleko od optima se pohybují nalezená řešení (počítáno pouze pro splnitelné stavy).
  - *Průměrná:* Průměrná procentuální odchylka nalezené váhy od optima.
  - *Maximální:* Nejhorší zaznamenaná odchylka (pro nejméně kvalitní splnitelné řešení).
- **Počet kroků:** Počet kroků (iterací) heuristiky před ukončením běhu. Tato hodnota indikuje časovou náročnost pro danou velikost instance a v případě mého řešiče se odvíjí pouze od velikosti instance.

Table 4: Souhrnné výsledky black-box testování

	Úspěšnost (%)		**Ztráta váhy (%)**		
Sada instancí	*Splnitelné*	*Optimální*	*Průměrná*	*Maximální*	**Počet kroků**
wuf20-91-M	99,99	99,99	0,00	40,00	26 700
wuf20-91-N	100,0	99,99	0,00	28,21	26 700
wuf20-91-Q	99,99	99,86	0,05	91,96	26 700
wuf20-91-R	99,99	99,89	0,05	57,98	26 700
wuf36-157-M	99,64	99,15	0,03	64,86	48 060
wuf36-157-N	99,63	99,15	0,03	64,86	48 060
wuf36-157-Q	99,54	96,51	0,77	98,96	48 060
wuf36-157-R	99,57	96,58	0,73	99,81	48 060
wuf50-218-M	98,41	93,15	0,25	61,03	66 750
wuf50-218-N	98,43	93,12	0,26	61,09	66 750
wuf50-218-Q	98,25	85,52	2,40	99,29	66 750
wuf50-218-R	98,27	85,52	2,39	99,92	66 750
wuf75-325-M	94,11	60,56	1,34	41,78	100 125
wuf75-325-N	94,04	60,98	1,28	41,81	100 125
wuf75-325-Q	93,76	56,00	6,99	81,54	100 125
wuf75-325-R	93,54	55,46	7,23	77,53	100 125

4.3 Diskuse

Z naměřených výsledků plyne, že vytvořený řešič založený na metodě simulovaného ochlazování je pro řešení problému MWSAT efektivní a robustní. Na lehčích (M, N) i těžších (Q, R) instancích dosahuje algoritmus obecně velmi dobrých výsledků a to napříč všemi zkoumanými velikostmi instancí. Z naměřených dat lze vyvodit několik klíčových pozorování:

- **Vysoká úspěšnost u malých a středních instancí:** U sad wuf20 a wuf36 dosahuje řešič téměř perfektních výsledků. U nejmenších instancí (20 proměnných) je spolehlivost nalezení optima prakticky 100 %, u středních (36 proměnných) se pohybuje nad 96 %. I u instancí o 50 proměnných dosahuje řešič velice silných, ač o něco horších, výsledků.

- **Škálovatelnost a degradace výkonu:** S rostoucí velikostí instance (od 50 do 75 proměnných) je patrný očekávaný pokles především v úspěšnosti nalezení globálního optima. Zatímco u wuf50 se optimální úspěšnost drží mezi 85-93 %, u wuf75 klesá k hranici 55-60 %. Tento trend naznačuje, že pro větší instance by bylo vhodné prozkoumat a nasadit vhodné adaptační mechanizmy, které nyní chybí.

- **Priorita splnitelnosti:** I přes pokles schopnosti nalézt optimální řešení u velkých instancí si řešič zachovává vysokou schopnost nalézt alespoň splnitelné řešení. U nejtěžší sady wuf75 je úspěšnost nalezení platného řešení stále nad 93 %. To ukazuje, že penalizační funkce správně navádí

řešič do oblasti platných řešení, přičemž ale penalizace není tak silná, aby zabraňovala diverzifikaci.

- **Vliv typu instance (M/N vs. Q/R):** Výsledky potvrzují, že verze instancí Q a R jsou pro heuristiku obtížnější než verze M a N. Například u sady wuf75 je rozdíl v optimální úspěšnosti mezi verzí N (60,98 %) a R (55,46 %) přibližně 5 procentních bodů. Ještě výraznější je rozdíl v průměrné ztrátě váhy, kde verze Q a R vykazují výrazně vyšší odchylku od optima (cca 7 %) oproti verzím M a N (cca 1,3 %). Dovolím si tvrdit, že řešič si obecně se zavádějícími verzemi poradil velmi dobře, jelikož rozdíl pár procentních bodů není tak významný. Může to být způsobeno poctivou fází diverzifikace, při které se heuristika je schopna dostat z lokálních extrémů a zavádějící úlohu vyřešit.

- **Kvalita suboptimálních řešení:** Ačkoliv u sady wuf75 řešič najde optimum jen v polovině případů, průměrná ztráta váhy se drží na přijatelné úrovni v řádu jednotek procent. U lehčích variant (M, N) je průměrná odchylka od optima pouze okolo 1,3 % a u těžších okolo 7 %. Řešič se tedy zpravidla pohybuje velice blízko optimu.

- **Maximální ztráta a stabilita:** Zatímco průměrná ztráta váhy je velmi nízká, hodnoty maximální odchylky ukazují, že v ojedinělých případech může chyba dosáhnout vysokých hodnot (u těžších sad Q a R i přes 90 %). To naznačuje, že ačkoliv je řešič statisticky spolehlivý, existuje riziko uvíznutí v hlubokém a nekvalitním lokálním optimu. Tento nedostatek by pravděpodobně také šel řešit nasazením vhodného mechanizmu, například restartu, který jsem v rámci práce zkoumal, ale do výsledného řešiče nakonec nezařadil.

References

[1] Baeldung. Simulated annealing explained. `https://www.baeldung.com/cs/simulated-annealing`, 2025. Accessed: 2025-11-29.

[2] FIT CTU. Experimentální vyhodnocení algoritmu. `https://courses.fit.cvut.cz/NI-KOP/homeworks/files/task1.html`, 2025. Accessed: 2025-11-11.

[3] FIT CTU. Splnitelnost booleovy formule. `https://courses.fit.cvut.cz/NI-KOP/problems/sat.html#mwsat`, 2025. Accessed: 2025-11-11.

[4] Jan Schmidt. PokroČilÉ heuristiky simulovanÉ ochlazovÁnÍ. `https://courses.fit.cvut.cz/NI-KOP/lectures/files/2023/KOP08%20SimulovanÃĺ%20ochlazovÃąnÃŋ.pdf`, 2025. Accessed: 2025-11-29.

[5] Wikipedia. Walksat. `https://en.wikipedia.org/wiki/WalkSAT`, 2025. Accessed: 2025-11-29.