# NI-KOP úkol 2

Leoš Tobolka

ZS 2022/23

# Obsah

# 1 Úvod

## 1.1 Zadání

Cílem úkolu je nasadit vybraný algoritmus pro řešení problému maximální vážené splnitelnosti booleovské formule (MWSAT):

**Dáno:** vektor $X$ $n$ 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 $c$ váha $w(c)$.

**Sestrojit:** ohodnocení $Y$ proměnných $X$ takové, že $F(Y) = 1$ a $w(Y) = \sum_{i=1}^{n} y_i w(i)$ je maximální.

Heuristika musí zvádat instance rozdílných vlastností (zejména velikosti) bez interaktivních zásahů.

## 1.2 Zvolený algoritmus

Pro řešení problému jsem zvolil heuristiku **simulováné ochlazování**.

# 2 Simulované ochlazování

## 2.1 Stavový prostor

Pro implementaci simulovaného ochlazování je potřeba definovat stavový prostor, se kterým se bude pracovat.

**Stav** reprezentuje vektor bitů o délce n (počet vstupních pro). Hodnota i-tého bitu vektoru značí, že i-tá vstupní proměnná je nastavena na true, pokud je bit 1. Opačná hodnota bitu značí hodnotu false.

**Operátor** je změnění hodnoty náhodného bitu ve vektoru bitů.

Optimalizační kritérium je splnitelnost formule a maximální součet vah kladných bitů (viz. zadání 1.1).

Stavový prostor obsahuje i takové stavy, které nesplňují formuli. Kdyby tomu tak nebylo, operátor by byl mnohem komplexnější – musel by rozhodovat i o splnitelnosti sousedů; to by zapříčinilo jeho zpomalení a zvýšilo by se riziko uváznutí v lokálním extrému.

## 2.2 Implementace

Heuristiku jsem implementoval v jazyce C++, který pomůže v časové rychlosti při řešení problémů, na rozdíl třeba od některých dynamických jazyků. Čas není metrika která se bude brát v potaz ve vyhodnocování efektivity heuristiky, ale výrazně pomůže a zrychlí vyhodnocování všech běhů.

### 2.2.1 Základ algoritmu

Základní struktura simulovaného ochlazování jsou dva vnořené cykly v sobě.

Obrázek 1: pseudokód základní struktury algoritmu

```
while (!frozen(T)) {
    initInner();

    while (equilibrium()) {
        state = tryState(T, state);

        int newWeight = getStateCost(state);

        if (newWeight > bestStateWeight) {
            bestState = state;
            bestStateWeight = newWeight;
        }

    }
    T = cool(T);
}
```

### 2.2.2 Výpočet ceny stavu

Cena a váha stavu spolu úzce souvisí. Váha odpovídá součtu jednotlivých vah pozitivních bitů. Cena je váha, která v případě že stav nesplňuje formuli, je penalizována vynásobením zlomkem *počet splněných klauzulí / počet všech klauzulí* a dodatečně vynásobena konstantou *1/2*.

Tento způsob přiřazuje splnitelným stavům větší cenu, jinak by heuristika hledala stavy s nejvyšším počtem kladných bitů, bez ohledu na cokoliv jiného.

Obrázek 2: Pseudokód výpočtu ceny

```
function getStateCost(state) {
    satClauses = stateSpace.getSatisfiedClauses(state);
    weight = stateSpace.getStateWeight(state);

    if(stateSpace.isSat(state)){
        return weight;
    }

    return weight * satClauses / instance.clausesCount / 2;
}
```

### 2.2.3 Počateční teplota

Výpočet počáteční teploty je prováděno dle postupu z přednášky. Počáteční teplota závisí na pravděpodobnosti s jakou na začátku, po zvolení iniciálního stavu, bude přijat horší sousední stav. Pravděpodobnost přijetí horšího stavu je nastavena na 50 %. Teplota se stále zvedá, dokud se nedosáhne požadované procentuální meze. Po dosáhnutí meze je získaná teplota vybrána jako počáteční pro heuristiku.

### 2.2.4 Parametry heuristiky

Počáteční teplota se počítá automaticky. Stále ale zbývá několik parametrů které lze nastavit manuálně; seznam parametrů je následující:

| parametry heuristiky | |
|---|---|
| $T_{min}$ | konečná teplota |
| $\alpha$ | koeficient chlazení |
| $N$ | počet kroků equilibria |

# 3 White box fáze

V této fázi se budu soustředit na testování heuristiky na malých sadách instancí. Heuristika má různé parametry, pokusím se najít takové, aby po jejich odladění heuristika zvládla najít řešení pro různé složitosti instancí. V případě že heuristika nebude pracovat efektivně s žádnou kombinací parametrů, zkusím upravit kód programu.

Základní měřená metrika heuristiky je počet kroků (počet iterací provedených ve vnitřním cyklu algoritmu).

## 3.1 Prvotní test heuristiky

K ladění heuristiky jsou k dispozici 3 hlavní parametry: konečná teplota ($T_{final}$), koeficient chlazení $\alpha$ a počet kroků equilibria $N$. Počáteční teplota je nastavována dynamicky. Proto další parametr který má velký vliv na efektivitu je $N$. Pro začátek, abych ověřil, zda je heuristika efektivní, jsem zkusil spustit algoritmus na 10 náhodných instancích z každé sady wuf20-71R pro různé hodnoty $N$. Parametr $\alpha$ jsem nastavil na 0.995 pro pomalejší ochlazování, aby byl algoritmus stabilní. $T_{final}$ jsem nechal na hodnotě 1, aby nedošlo k předčasnému ukončení.

| wuf20-71R 10 instancí, každá 3 běhy $N = 10$, $\alpha = 0.995$ , $T_{min} = 1$ | | | | |
|---|---|---|---|---|
| sada | M | N | Q | R |
| úspěch | 1 | 1 | 0.3 | 0.24 |
| avg kroky | 15510 | 19928 | 15315 | 20364 |
| awg fined | 15510 | 19928 | 33227 | 39585 |

Uspěšnost sady Q a R je malá, proto zkusím $N$ zvýšit.

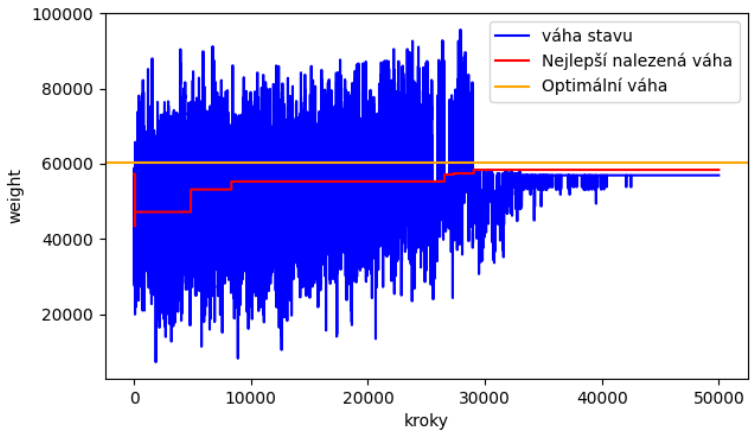
| wuf20-71R 10 instancí, každá 3 běhy $N = 50$, $\alpha = 0.995$ , $T_{min} = 1$ | | |
|---|---|---|
| sada | Q | R |
| úspěch | 0.63 | 0.7 |
| avg kroky | 84706 | 115101 |
| awg fined | 360610 | 418391 |

Tento výsledek je už lepší. Vyzkouším sadu s 91 klauzulemi.

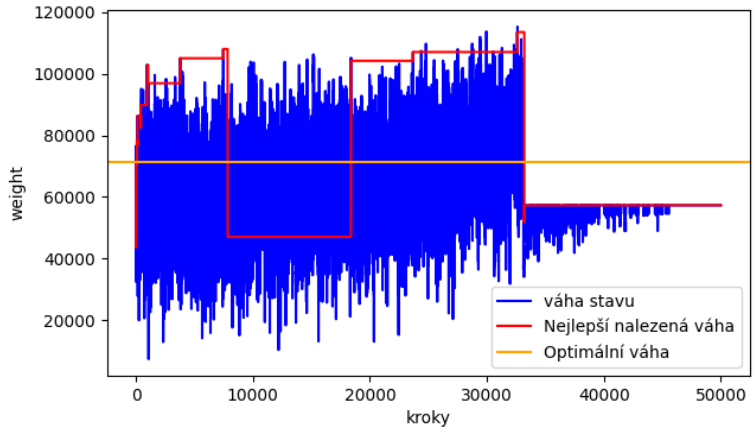
| wuf20-91R 10 instancí, každá 3 běhy $N = 50$, $\alpha = 0.995$ , $T_{min} = 1$ | | | | |
|---|---|---|---|---|
| sada | M | N | Q | R |
| úspěch | 0.77 | 0.83 | 0 | 0 |
| avg kroky | 84572 | 105201 | 84574 | 105462 |
| awg fined | 264162 | 258654 | 845740 | 1054627 |

Z tabulek je vidět že heuristika pro sadu M a N nejmenší instance nalezne řešení bez problému. Pro sadu Q a R se to stejné tvrdit nedá; úspěšnost v těchto případech je velmi malá (pouhých 30% a 24%). Zvýšením kroků equilibria (viz. tabulka 2) se výsledek zlepšil. Průměrný počet kroků je stále velký pro takovouto malou sadu a nelze říct že je výsledek uspokojivý. V třetí tabulce je vidět běh heuristiky na sadě 20-91R, ale jak šlo z předchozích výsledků očekávat, výsledky nejsou o nic lepší; pro sadu Q a R je úspěšnost nulová.

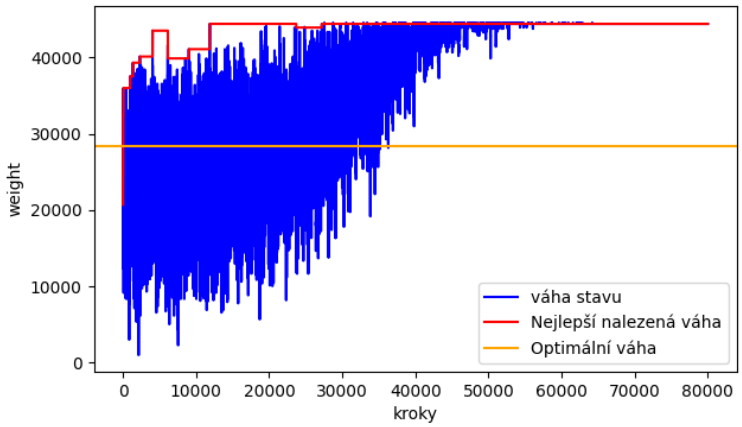
Ze statistik z tabulek není zřejmé jaký je problém. Proto, pro jednotlivé instance jsem vytvořil graf průběhu hledání optimální váhy. Vzal jsem problematické instance ze sady R; výsledek je následující:




Obrázek 3: uf20-71R/uf20-0134 $N = 10$




Obrázek 4: uf20-71R/uf20-0134 $N = 50$




Obrázek 5: uf20-71R/uf20-0134 $N = 50$

Na všech grafech lze vypozorovat že algoritmus má dlouhou fázi diverzifikace. To je pravděpodobně zapříčiněno náhodou volbou sousedních stavů. Nejdůležitějším faktem, který lze vyčíst, je že algoritmus se pokaždé zasekl v lokálním maximu. Na posledním grafu algoritmus zvolil nejlepší stav takový, který váhu mnohem vyšší než je optimální, to je z důvodu nevodného počítání ceny stavu. Algoritmus upřednostnil nesplnitelný stav s vyšším počtem kladných bitů před optimálním stavem s více zápornými bity.

Bez další analýzy, je už z těchto okrajových testů zřejmé, že algoritmus je potřeba upravit, aby se lépe vyvaroval lokálním maximům a upřednostňoval

více splnitelné stavy před nesplnitelnými s vyšší cenou.

## 3.2 Úprava implementace

Z prvotního testu vyplynulo, že je potřeba upravit počítání ceny a vylepšit výběr sousedních stavů. Obě tyto části algoritmu byly implementovány naivně a bez jejich úpravy by se i při sebelepším nastavení parametrů nedosáhlo uspokojivé úspěšnosti.

### 3.2.1 Úprava výpočtu ceny

První způsob výpočtu ceny, který spočíval v penalizaci zlomkem *počet splněných klauzulí / počet všech klauzulí*, nebyl příliš úspěšný. Je potřeba zavést vyšší penalizaci, aby se upřednostnili splnitelné stavy.

Pokusil jsem se pokračovat v konceptu penalizace na základu zlomku splněných klauzulí. Přidal jsem další koeficient, kterým se váha dělí; tento koeficient se počítal jako odmocnina z průměrné váhy proměnné. Po opakování stejných testů jako v kapitole 3.1, vyplynulo, že tento způsob je sice nepatrně lepší, ale stále není uspokojivý. Na těžších instancích s velkými rozdíly vah, měl algoritmus stále tendenci volit nesplnitelné ohodnocení s vyššími váhami než splnitelné ohodnocení.

Po tomto neúspěšném pokusu jsem se pokusil docílit výběru ceny takové, aby cena nesplnitelného ohodnocení nikdy nemohla přesáhnout cenu splnitelného stavu. Další způsob výpočtu ceny vypadá v pseudokódu následovně:

```
function getStateCost(state) {
    weight = stateSpace.getStateWeight(state);

    if(stateSpace.isSat(state)){
        return weight;
    }

    return weight - instance.sumWeight;

}
```

Obrázek 6: Pseudokód nového výpočtu ceny

### 3.2.2 Úprava volby sousedních stavů

Metodu pro volbu sousedních stavů jsem upravil tak, že pokud stav není splnitelný, změní se hodnota náhodné proměnné v náhodné nesplnitelné klauzuli. Tento způsob nasměruje volbu stavů směrem ke splnitelným a zároveň nezatíží příliš logiku operátoru, který by příliš zpomaloval čas výpočtu.

### 3.2.3 Úprava metody frozen

Metodu *frozen*, která ukončuje vnější cyklus, pokud se teplota sníží pod hranici $T_{min}$ jsem se také rozhodl upravit. Pokud algoritmus nalezne optimální stav příliš brzy, heuristika běží poté zbytečně dlouho. Proto jsem implementoval způsob z přednášky.

Pokud je procento nových přijatých stavů za jeden vnitřní cyklus menší než 0.1, algoritmus se ukončí. Díky tomuto způsobu nemá parametr $T_{min}$ tak důležitou roli při nastavování parametrů.

## 3.3 Test heuristiky po úpravě

Po úpravě implementace jsem spustil test znovu. Parametr $\alpha$ jsem nechal stejný jako minule. $N$ jsem pro nejmenší instanci zvolil 50, jako tomu bylo v minulém testu, kde se tato hodnota ukázala jako uspokojivá. Parametr $T_{final}$ nyní můžu bez ohledu nechat na hodnotě 1, vzhledem k úpravě metody $frozen$.

| wuf20-71R 10 instancí, každá 3 běhy<br>$N = 50$, $\alpha = 0.995$ , $T_{min} = 1$ | | | | |
|---|---|---|---|---|
| sada | M | N | Q | R |
| úspěch | 1 | 1 | 0.93 | 0.83 |
| avg kroky | 51023 | 44728 | 44037 | 37989 |
| awg fined | 51023 | 44728 | 68631 | 93820 |

Pro sadu Q i R se úspěšnost zlepšila. Pro další sadu nechám N stejné.

| wuf20-91R 10 instancí, každá 3 běhy $N = 50$, $\alpha = 0.995$ , $T_{min} = 1$ | | | | |
|---|---|---|---|---|
| sada | M | N | Q | R |
| úspěch | 1 | 0.9 | 0.9 | 0.63 |
| avg kroky | 25264 | 27160 | 35743 | 37093 |
| awg fined | 25264 | 50798 | 42653 | 163174 |

I pro tuto sadu je procentuální úspěšnost lepší než tomu bylo minule. Tyto výsledky vypadají více uspokojivě, proto jsem zkusil i následující sadu s 50 vstupními proměnnými. Více vstupních rozšiřuje výrazně velikost stavového prostoru, proto jsem parametr N nastavil na 100.

| wuf50-218R 10 instancí, každá 3 běhy $N = 100$, $\alpha = 0.995$ , $T_{min} = 1$ | | | | |
|---|---|---|---|---|
| sada | M | N | Q | R |
| úspěch | 0.56 | 0.55 | 0.15 | 0.1 |
| avg kroky | 55941 | 68521 | 85532 | 90772 |
| awg fined | 339873 | 413694 | 795322 | 895729 |

Pro sadu M a N se úspěšnost pohybuje kolem 55%. Pro sadu Q a R je úspěšnost malá, ale předpokládám že zvýšením N by se procentuální úspěšnost zvedla.

Parametr N má velký vliv na úspěšnost a nelze najít jeho jednu hodnotu, která by byla optimální pro všechny složitostí instancí. Proto je potřeba aby se jeho hodnota determinovala ze složitosti instance.

## 3.4 Nastavení parametru N

Jak je z předchozích testů vidět, parametr N by bylo vhodné nastavit dle obtížnosti instance. Na obtížnost instancí má velký vliv počet jejich proměnných, proto se pokusím nastavit N podle toho.

N se bude nastavovat podle následujícího vzorečku:

$$N = I_{prom} N_k$$

Parametr $I_{prom}$ je počet proměnných instance a parametr $N_k$ je koeficient, kterým se $I_{prom}$ bude násobit. Tímto způsobem dosáhnu nastavení N takového, aby se škáloval podle složitosti instance.

## 3.5 Hledání parametrů heuristiky

Vzhledem k způsobu implementace jsem se zbavil potřeby nastavovat parametr $T_0$, $N$, $T_{min}$; všechny tyto parametry dokáže algoritmus určit sám ze zadané instance. Zbývají parametry $N_k$ a $\alpha$.

Nastavení těchto dvou parametrů je potřeba vyvážit, aby byl algoritmus efektivní. Oba tyto parametry ovlivňují počet kroků heuristiky, je nevhodné je tedy nastavovat jednotlivě. K nalezení optimálního nastavení použiji faktorový návrh.

### 3.5.1 Faktorový návrh

Na začátek je potřeba zvolit rozsah parametrů. Rozsah pro $N_k$ zvolím tak, aby minimum pro instanci s 20 proměnnými byla hodnota $N = 20$, protože jak bylo z druhého testu po úpravě algoritmu vidět, $N = 10$ je už příliš málo. Těžší instance ze sady wuf50-218R jsem spouštěl v počátečním testu s $N = 100$, stále tam ale byl prostor pro zlepšení, proto zvolím $N_k$ taky, aby $N$ bylo alespoň 375.

Volba $N_k$ je tedy následující: 1.5, 3 ,4.5 ,6 ,7.5; rovnoměrně rozdělená mezi hodnoty 1.5 a 7.5.

Parametr $\alpha$ jsem v úvodních testech spouštěl s $\alpha = 0.995$, to bylo zvoleno, aby bylo ochlazování pomalé a dosáhlo se lepších výsledků nehledě na počet kroků. Pro celkové optimálnější výsledky pro faktorový návrh zvolím tedy hodnoty rovnoměrně mezi 0.99 a 0.80, aby počet kroků nebyl příliš velký a zároveň abych vyzkoušel i funkčnost heuristiky při rychlejším ochlazování.

Faktorový návrh vyšel následovně:

| Všechny 3 sady - z každé 6 instancí, každá 2 běhy | | | | | | |
|---|---|---|---|---|---|---|
| $\alpha$ | metrika | $N_k$ | | | | |
| | | 1.5 | 3 | 4.5 | 6 | 7.5 |
| 0.99 | úspěch % | 39 | 63 | 68.1 | 70.1 | 75.7 |
| | avg steps | 11348 | 35882 | 47717 | 69234 | 86342 |
| | awg fined | 84892 | 211823 | 246054 | 372787 | 428389 |
| 0.95 | úspěch % | 22 | 48 | 54 | 64.9 | 64.2 |
| | avg steps | 2530 | 7617 | 10656 | 15633 | 18471 |
| | awg fined | 21445 | 50955 | 66248 | 86187 | 104520 |
| 0.90 | úspěch % | 25 | 51 | 49 | 56 | 54 |
| | avg steps | 1366 | 4271 | 5446 | 8367 | 9415 |
| | awg fined | 10832 | 27777 | 36153 | 50573 | 56850 |
| 0.85 | úspěch % | | 35 | 47 | 48 | 49 |
| | avg steps | X | 2864 | 3777 | 5681 | 6519 |
| | awg fined | | 21596 | 25805 | 39164 | 42784 |
| 0.80 | úspěch % | | | 39 | 46 | 50 |
| | avg steps | X | X | 2896 | 4414 | 4871 |
| | awg fined | | | 21173 | 29835 | 32048 |

Hned z prvního pohledu na tabulku je vidět, že kombinace parametrů jsou nejlepší v pravém horním rohu. To jsou kombinace s nejpomalejším ochlazováním a zároveň největším počtem vnitřních kroků $N$. Faktorový návrh tedy nebyl vhodně rozvrhnut, optimální hodnoty jsou pouze v rohu. Také lze vypozorovat že ochlazovací faktor $\alpha = 0.95$ má úspěšnost nepříliš pozadu od $\alpha = 0.99$ pro $N_k = 4.5$ a $N_k = 6$, ale rozdíl v jejich počtu kroků je veliký. Z toho důvodu jsem se rozhodl udělat dodatečný faktorový návrh s novou hodnotou $N_k = 9$ a $\alpha = 0.97$, zda se nenajde ještě optimálnější výsledek.

| Všechny 3 sady - z každé 6 instancí, každá 2 běhy | | | | |
|---|---|---|---|---|
| $\alpha$ | metrika | $N_k$ | | |
| | | 6 | 7.5 | 9 |
| 0.99 | úspěch % | 70 | 75.7 | 74.3 |
| | avg steps | 69234 | 86342 | 111137 |
| | awg fined | 372787 | 428389 | 561583 |
| 0.97 | úspěch % | 64.6 | 69.1 | 66.7 |
| | avg steps | 25856 | 28965 | 40575 |
| | awg fined | 143942 | 157276 | 231825 |
| 0.95 | úspěch % | 65 | 64 | 64.58 |
| | avg steps | 15633 | 18471 | 23808 |
| | awg fined | 86187 | 104520 | 134569 |

Z další tabulky je vidět že pro kombinaci $\alpha = 0.97$ a $N_k = 7.5$ má algoritmus skoro o třetinu menší počet kroků než pro $\alpha = 0.99$ a $N_k = 6$, ale podobnou úspěšnost. Proto s ohledem na rychlost a úspěšnost volím jako optimální kombinaci parametrů $\alpha = 0.97$ a $N_k = 7.5$.

### 3.5.2 Zvolené parametry

Z faktorového návrhu jsem jako finální parametry pro nastavení heuristiky vybral $\alpha = 0.97$ a $N_k = 7.5$.

# 4 Black box fáze

Test jsem provedl jednotlivě pro sady $wuf20-71R$, $wuf20-91R$, $wuf50-218$.

Pro každou instanci z každé sady jsem naměřil 10 běhů. To je 4000 běhů pro každou sadu a 12000 běhů dohromady. Naměřené metriky jsou: počet kroků běhu, vážený počet kroků běhu, úspěch běhu a počet splněných klauzulí.

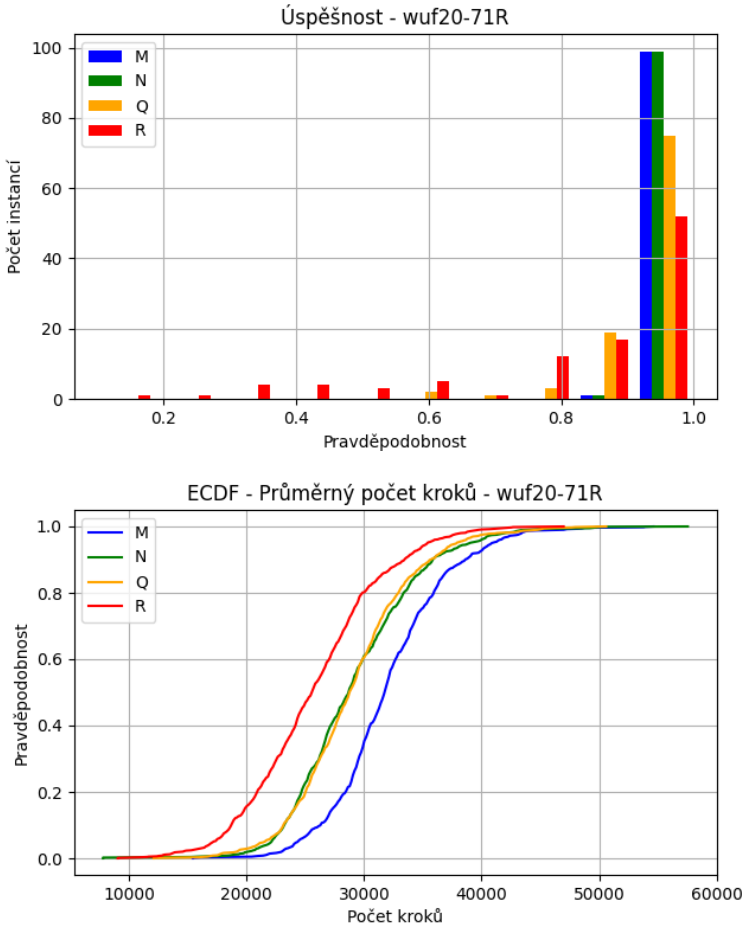
Zvolené parametry heuristiky jsou $\alpha = 0.97$ a $N_k = 7.5$.

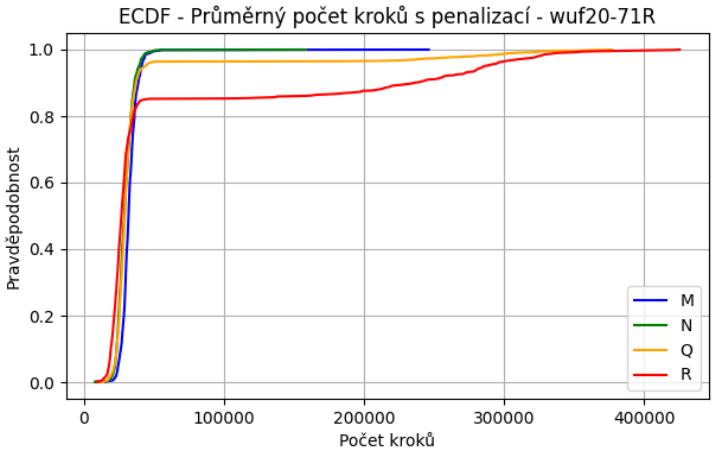
## 4.1 Naměřené data

Všechny grafy a zdroje ze kterých byly vygenerovány lze najít v příloze. Histogram úspěšnosti znázorňuje procentuální úspěšnost deseti běhů v jed-

notlivých podsadách M, N, Q a R. Histogram splněných klauzulí ukazuje kolik splněných klauzulí měly nalezené řešení. Na ECDF grafech lze vidět empirické distribuční funkce průměrných počtů iterací (kroků) a vážených počtu iterací algoritmů, kde neúspěšný běh se počítá jako 10 násobek počtu iterací.

### 4.1.1 Sada wuf20-71R



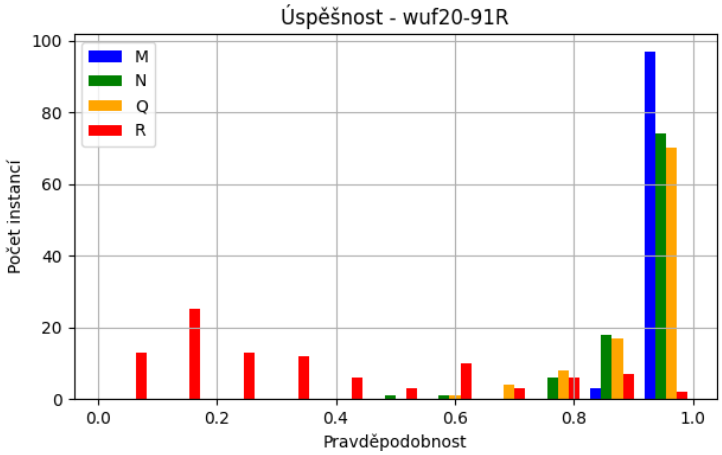


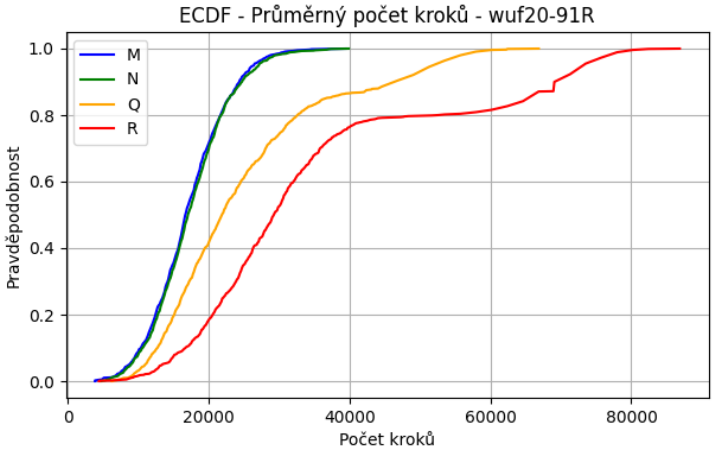

Z histogramu lze vypozorovat že pro všechny instance ze sady M a N heuristika našla správné řešení téměř v každém případě (Kromě 2–4 běhů). Pro sadu Q se pro přibližně 25% případů stalo, že pro jeden až tři běhy z deseti se pro každou instanci nenašlo správné řešení. Sada R je nejproblémovější; zde mělo 100% úspěšnost přibližně jen 50% instancí, avšak není žádná instance, pro kterou by se nepodařilo najít řešení alespoň jednou po deseti opakováních běhu.

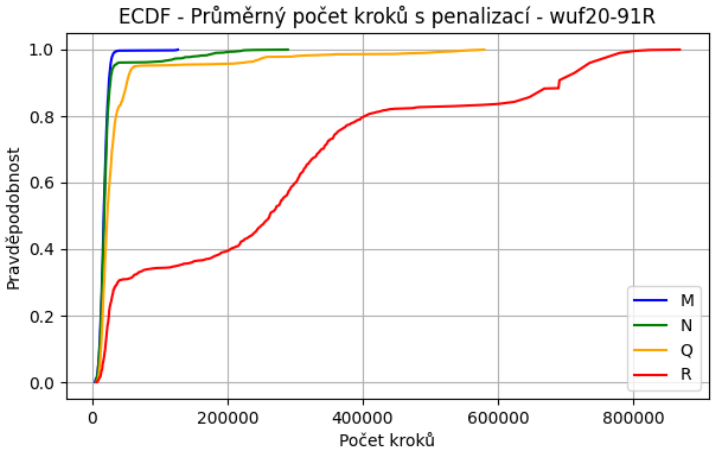
ECDF funkce s průměrným penalizovaným počtem kroků poukazují na to, že některé instance ze sady Q a R se nepodařilo v nějakých bězích vyřešit.

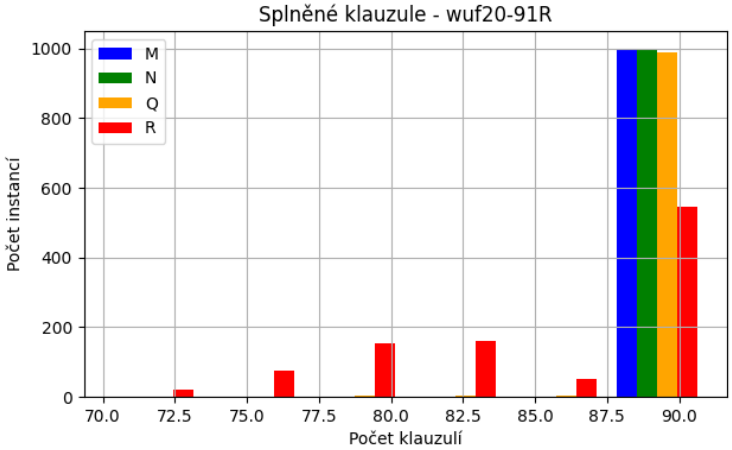
Celkový výsledek je dobrý, i pro sadu R bylo převážné množství běhů úspěšné.

### 4.1.2 Sada wuf20-91R



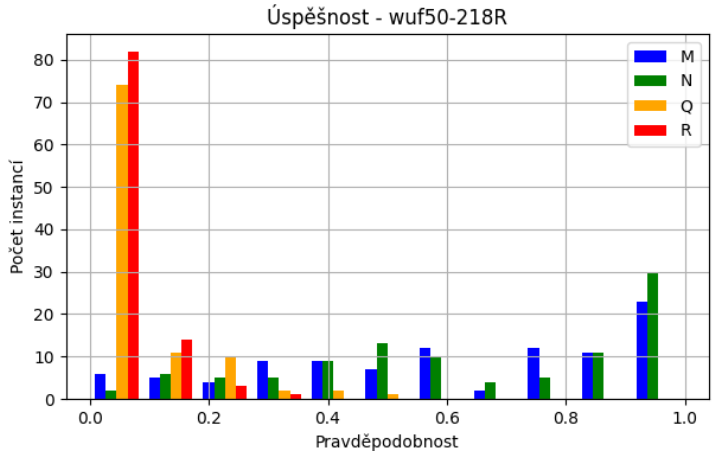



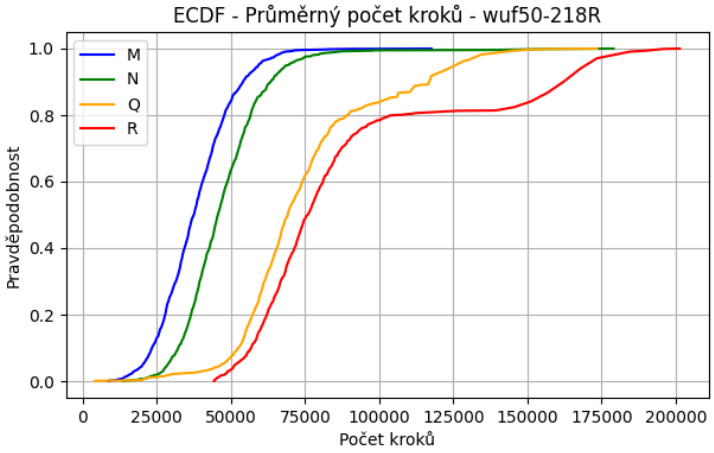


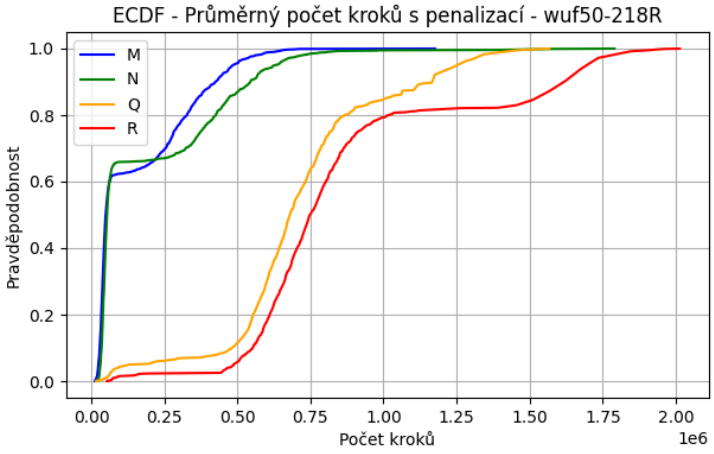

Podsada M má stále stejnou úspěšnost jako v minulé sadě. Podsada N a Q se už vzájemně liší. V tomto případě je jejich úspěšnost podobná, jen ne pro všechny běhy každé instance se v každém případě našel správný výsledek, ale stále, pro převážnou většinu instancí je úspěšnost nad 80%. Pro podsadu R úspěšnost výrazně klesla, tuto skutečnost odráží i ECDF graf vážených počtů kroků, kde je křivka výrazně pod ostatními.

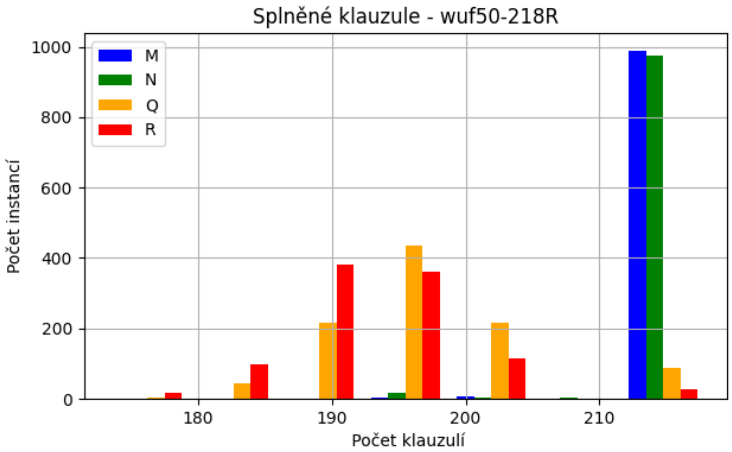
Na grafu splněných klauzulí je vidět že téměř pro polovinu běhů podsady R běh nenašel splnitelné řešení. Vzhledem k implementaci počítání váhy to značí že se algoritmu nepodařilo od počátečního stavu dostat do stavu splnitelného a prohledával stále nesplnitelné stavy.

### 4.1.3 Sada wuf50-218






V podsadě M a N se stále pro většinu instancí podařilo najít řešení alespoň jednou v opakovaných bězích pro každou instanci, avšak celková úspěšnost už je výrazně menší. Pro sadu Q a R se to stejné říct nedá, úspěšnost je v jejich případě mizivá. Pro většinu instancí se nepodařilo najít žádný úspěšný běh. Tento fakt odráží i vážený ECDF graf, kde jdou jejich dvě křivky hluboko pod předchozíma dvěma. Při pohledu na histogram splněných klauzulí je vidět, že i téměř ve všech případech algoritmus nenašel splnitelné řešení, natož správné.

# 5 Závěr

Implementoval jsem heuristiku simulovaného ochlazování pro pro optimalizační problém MAXSAT. Algoritmus byl implementován v jazyce C++. Prvotní naivní řešení jsem modifikoval o několik funkcionalit: dřívější zastavení v případě, že heuristika se dostala do lokálního nebo globálního minima; chytřejší hledání sousedních stavů; výpočet ceny stavu s prioritizací splnitelných stavů a závislost počtu vnitřních stavů na složitosti instance.

Pro parametry heuristiky jsem provedl ladění pomocí faktorového návrhu. Díky tomuto způsobu jsem byl schopen vybrat nastavení, které nevyžaduje velký počet kroků pro uspokojivou úspěšnost.

Výkon heuristiky byl testován na instancích v rozsahu 20–50 proměnných. Na sadách M a N se podařilo dosáhnout poměrně uspokojivých výsledků; řešení pro tyto sady byl algoritmus schopen ve převážné většině najít správné řešení. Pro podsadu Q a R v sadách o 20 proměnných také nebyl neúspěšný;

ve většině případů se po opakovaném puštění algoritmu na každé instanci povedlo v nenulovém procentu najít správné řešení. Pro podsadu Q a R v sadě $50 - 218R$ byl algoritmus neuspokojivý, ve velkém množství případů se nepodařilo najít splnitelné řešení, natož řešení se správnou vahou.

Experiment dokázal, že implementovaný algoritmus heuristiky je pro menší instance problému o 20 proměnných spolehlivý. Pro větší instance už algoritmus tak úspěšný není. V sadě s 50 proměnnými je procento celkové úspěšnosti pod 50%. Celkově hodnotím algoritmus jako nepříliš spolehlivý, celková úspěšnost na testovaných datech se pohybuje kolem 69%, přičemž 2/3 dat tvoří sady s 20 proměnnými. Věřím ale, že úspěšnost by se dala nepatrně zlepšit dalšími úpravami algoritmu.