MWSAT

1 Úvod

Cílem této práce je vyřešit problém maximální vážené splnitelnosti booleovské formule (MWSAT) pomocí pokročilé iterativní metody, konkrétně simulovaného ochlazování. Heuristika musí být schopná zpracovat instance s různými vlastnostmi bez potřeby interaktivních zásahů.

V této práci jsou vlastnosti heuristiky ověřeny experimentálním vyhodnocením, které ukazuje, jaké typy instancí je heuristika schopna efektivně zpracovat. Jsou popsány obě fáze implementace heuristiky: fáze nastavování parametrů (tzv. white box fáze) a fáze konečného hodnocení heuristiky (tzv. black box fáze).

2 Metody

2.1 MWSAT

Problém vážené splnitelnosti Booleovy CNF formule s maximem jedniček (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 klauzuli x váha *w(x)*.

Sestrojit: ohodnocení Y proměnných X takové, že *F(Y) = 1* a součet vah proměnných, ohodnocených 1, je maximální.

2.2 Heuristika

Pro řešení tohoto problému byla zvolena metoda simulovaného ochlazování (simulated annealing). Tato metoda je pokročilá heuristická technika inspirovaná procesem fyzikálního ochlazování materiálů, při kterém se systém postupně dostává do stavu s minimální energií. V oblasti optimalizace se tato analogie využívá k nalezení globálního optima v případech, kdy je prostor řešení složitý a obsahuje mnoho lokálních optim.

Algoritmus simulovaného ochlazování funguje na principu iterativního zlepšování řešení. V každém kroku se generuje nové řešení, které se přijímá buď na základě zlepšení cílové funkce, nebo s určitým pravděpodobnostním mechanismem, který umožňuje překonávat lokální minima. Pravděpodobnost přijetí horšího řešení postupně klesá s tím, jak teplota systému klesá. Tento proces ochlazování řídí tzv. chladicí plán (cooling schedule), který určuje rychlost snižování teploty.

2.3 Implementace

Pro implementaci simulovaného ochlazování byly navrženy a implementovány následující funkce:

- **cool** - funkce pro výpočet nové teploty během simulovaného ochlazování. Sníží aktuální teplotu t podle koeficientu ochlazování *alpha*.

```
double cool(double t, double alpha) const {
    return t * alpha;
}
```

- **frozen** - funkce pro kontrolu, zda proces simulovaného ochlazování skončil. Kontroluje, zda teplota klesla pod konečnou hodnotu *t_fin* nebo zda byl dosažen maximální počet iterací *itr_max*.

```
bool frozen(double t, double t_fin, int iter, int itr_max) const {
    return t < t_fin || iter > itr_max;
}
```

- **equilibrium** - funkce, která určuje, zda se dosažený počet vnitřních iterací překročil stanovený limit.

```
bool equilibrium(int inner_max, int inner, int last_imp, int thr_imp) const {
    return inner - last_imp >= thr_imp || inner >= inner_max;
}
```

- **cost** - funkce vypočítává náklady na základě počtu splněných klauzulí a jejich vah. Pokud jsou všechny klauzule splněny, vrací hodnotu, která zohledňuje poměr váhy splněných klauzulí k maximální váze, jinak je vrácen poměr splněných klauzulí k celkovému počtu klauzulí.

```
double cost(int satisfied, int weight, int totalWeight) const {
    if (satisfied == numClauses) {
        return 1 + weight / totalWeight;
    }
    return satisfied / numClauses;
}
```

- **howMuchWorse** - funkce vypočítá, jak moc je nový stav horší než předchozí. Rozdíl mezi náklady na nový stav a předchozí stav je vypočítán pomocí funkce *cost*.

```
double howMuchWorse(int prevSatisfied, int prevWeight, int newSatisfied,
                    int newWeight, int totalWeight) const
{
    return cost(newSatisfied, newWeight, totalWeight) -
           cost(prevSatisfied, prevWeight, totalWeight);
}
```

- **chooseLiteral** - funkce vybírá literál na základě počtu splněných klauzulí. Pokud jsou všechny klauzule splněny, náhodně generuje literál, jinak vybírá literál z nevyřešené klauzule, a větší pravděpodobnost má výběr false literálu.

```
int chooseLiteral(int prevSatisfied) const {
    int literal;
    if (prevSatisfied == numClauses) {
        literal = dist(gen);
    } else {
        literal = formula.getLiteralFromUnsatisfiedClause();
    }
    return literal;
}
```

2.4 Parametry

Parametry jsou následující:

```
struct SAParameters {
    double t0;     /* initial temperature */
    double t_fin;  /* final temperature */
    double cool;   /* cooling coefficient. */
    int inner_max; /* inner loop count */
    int itr_max;   /* total iterations max */
    int thr_imp;   /* max iterations without improvement */
}

struct ActualParameters {
    double temp;          /* actual temperature */
    int inner;            /* inner loop count */
    int iter;             /* total iterations */
    int last_imp;         /* last improvement iteration */
    int satisfied;        /* current no. of satisfied clauses */
    int best;             /* max no. of satisfied clauses */
    int best_weight;      /* best weight achieved when all clauses are satisfied */
    int num_clauses;      /* total number of clauses */
    vector<int> best_sol; /* best solution so far based on the best_weight */
}
```

Parametry v `SAParameters` se používají k nastavení počáteční teploty, koncové teploty, koeficientu ochlazování a počtu iterací. Parametry ve `ActualParameters` sledují aktuální stav procesu, včetně počtu splněných klauzulí, teploty, nejlepšího řešení a počtu iterací.

3 White box

Pro fázi white-box jsem náhodně vybral **10 instancí** z každé z následujících sad: wuf20-71R-M, wuf20-71R-N, wuf20-71R-Q, wuf20-71R-R, wuf20-91R-M, wuf20-91R-N, wuf20-91R-Q, wuf20-91R-R, wuf50-218R-M, wuf50-218R-N, wuf50-218R-Q, wuf50-218R-R, a každou instanci jsem spustil **50krát**.

3.1 White box

Nejprve jsem si udělal přehled o možných hodnotách funkce `cost()`, která se pohybuje v rozmezí od 0 do 2, a podle toho jsem zvolil příslušné parametry pro první experiment.

Parametr	**Hodnota**
t0	100
t_fin	0.1
cool	0.8
inner_max	50
thr_imp	25
itr_max	10000

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	17.20	827
wuf20-71R-N	100.00	13.20	827
wuf20-71R-Q	100.00	29.80	827
wuf20-71R-R	100.00	32.20	826
wuf20-91R-M	78.60	45.40	817
wuf20-91R-N	71.40	44.80	812
wuf20-91R-Q	72.00	50.00	807
wuf20-91R-R	90.60	68.60	813
wuf50-218R-M	1.60	0.20	829
wuf50-218R-N	9.80	0.00	833
wuf50-218R-Q	9.20	0.40	835
wuf50-218R-R	1.40	0.20	835

Průměrný počet kroků je velmi malý, program se rychle ochlazuje. Počet úspěšných nalezení řešení je u instancí 20-71 a 20-91 uspokojivý, ale úspěšnost nalezení optimálního řešení je nízká. Instance 50-218 nejsou řešitelné. Nejprve je třeba zvýšit koeficient *cool* pro pomalejší ochlazování.

3.2 White box

Parametr	**Hodnota**
t0	100
t_fin	0.1
cool	0.95
inner_max	50
thr_imp	25
itr_max	10000

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	54.20	3442
wuf20-71R-N	100.00	46.40	3444
wuf20-71R-Q	100.00	73.20	3437
wuf20-71R-R	100.00	71.00	3440
wuf20-91R-M	95.40	87.00	3425
wuf20-91R-N	91.80	76.20	3419
wuf20-91R-Q	91.40	74.60	3413
wuf20-91R-R	99.60	95.60	3417
wuf50-218R-M	9.20	1.40	3442
wuf50-218R-N	26.60	1.40	3449
wuf50-218R-Q	24.40	0.80	3446
wuf50-218R-R	6.60	0.80	3445

Výsledky se zlepšily, ale rozdíl mezi nalezením řešení a nalezením optimálního řešení je stále velmi velký. To může být způsobeno tím, že hodnota parametru *inner_max* byla původně nastavena příliš nízko. Nižší hodnota *inner_max* znamená, že v rámci jedné teplotní úrovně je prováděno méně iterací, což vede k rychlejší změně teploty a menšímu prozkoumání stavového prostoru. To může vést k tomu, že algoritmus uvízne v lokálních extrémech. Proto je vhodné zvýšit hodnoty *inner_max* a *thr_imp*, aby měl algoritmus více příležitostí zlepšit řešení před dalším ochlazením.

3.3 White box

Parametr	**Hodnota**
t0	100
t_fin	0.1
cool	0.95
inner_max	150
thr_imp	75
itr_max	10000

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	88.10	10037
wuf20-71R-N	100.00	82.80	10037
wuf20-71R-Q	100.00	94.30	10038
wuf20-71R-R	100.00	92.10	10038
wuf20-91R-M	98.60	98.20	10036
wuf20-91R-N	95.20	91.00	10039
wuf20-91R-Q	96.20	84.40	10039
wuf20-91R-R	100.00	99.80	10038
wuf50-218R-M	23.20	1.80	10038
wuf50-218R-N	35.80	3.00	10037
wuf50-218R-Q	43.40	3.80	10037
wuf50-218R-R	17.40	3.80	10039

Výsledky se výrazně zlepšily, což naznačuje, že přidání více iterací na jedné teplotní úrovni vedlo k lepšímu prozkoumání stavového prostoru. Rozdíl mezi instancemi 20-71 a 20-91 je zanedbatelný, protože mají stejné množství literálů a malý rozdíl v počtu klauzulí. Na druhou stranu, instance 50-218 se řeší velmi špatně a téměř bez optimálních řešení. Proto je potřeba přizpůsobit počet iterací podle počtu klauzulí a literálů, aby instance 50-218 měly více iterací na konkrétních úrovních. Mírně zvýším hodnoty *inner_max* a *thr_imp*, protože je zde ještě prostor pro zlepšení v nalezení optimálního řešení. Je také nutné zvýšit *itr_max*, protože současná řešení evidentně narazila na limit 10000.

3.4 White box

Parametr	**Hodnota**
sum	počet literálů + počet klauzulí
t0	100
t_fin	0.1
cool	0.95
inner_max	2.5 × sum
thr_imp	2.0 × sum
itr_max	10 × počet literálů × počet klauzulí

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	92.00	14297
wuf20-71R-N	100.00	86.80	14300
wuf20-71R-Q	100.00	94.20	14296
wuf20-71R-R	100.00	95.40	14296
wuf20-91R-M	99.80	99.80	18308
wuf20-91R-N	97.60	94.20	18299
wuf20-91R-Q	98.20	86.60	18291
wuf20-91R-R	100.00	100.00	18295
wuf50-218R-M	53.60	17.40	72793
wuf50-218R-N	48.60	21.00	72857
wuf50-218R-Q	72.20	17.60	72813
wuf50-218R-R	46.60	18.40	72745

Výsledky se zlepšily, zejména u větších instancí jako wuf50-218R, díky zvýšení počtu kroků. Pokrok je sice viditelný, ale výsledky stále nejsou dostatečně optimalizované, zejména u těchto větších instancí. Je nutné upravit parametry tak, aby lépe odpovídaly velikosti problému. Počáteční teplotu je také třeba nastavit v závislosti na velikosti instance, stejně tak je nutné zvýšit maximální počet iterací. Je potřeba upravit parametr *inner_max*, aby se postupně zvyšoval s tím, jak algoritmus pokračuje v běhu, což umožní algoritmu lépe prozkoumávat prostor řešení.

3.5 White box

Parametr	**Hodnota**
sum	počet literálů + počet klauzulí
t0	sum
t_fin	0.1
cool	0.95
inner_max	(2 + 5 × (iter / itr_max)) × sum
thr_imp	2.5 × sum
itr_max	2 × sum × sum

Výsledek:

Instance	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	91.80	16753
wuf20-71R-N	100.00	88.40	16753
wuf20-71R-Q	100.00	95.80	16753
wuf20-71R-R	100.00	95.60	16753
wuf20-91R-M	100.00	99.80	24875
wuf20-91R-N	98.80	97.00	24875
wuf20-91R-Q	98.80	88.00	24875
wuf20-91R-R	100.00	100.00	24875
wuf50-218R-M	59.00	20.80	103046
wuf50-218R-N	54.00	21.20	103046
wuf50-218R-Q	81.80	20.00	103046
wuf50-218R-R	50.80	21.80	103046

Nepodstatná zlepšení ve srovnání s předchozí konfigurací. Z výsledků jsem zjistil, že při řešení větších instancí obvykle chybí k nalezení řešení 1–3 klauzule z celkových 218. To znamená, že řešení se blíží k extrému vzhledem k pravdivým klauzulím, ale nedokáže ho dosáhnout. Proto by stálo za to snížit konečnou teplotu, aby algoritmus v určitém bodě přestal přijímat horší řešení a soustředil se na hledání extrému. Zároveň, aby tento proces netrval příliš mnoho iterací, by bylo vhodné také snížit

chladicí koeficient. Také je vidět, že všechny instance narážejí právě na omezení počtu iterací, takže jeho hodnotu také trochu zvýším.

3.6 White box

Parametr	**Hodnota**
sum	počet literálů + počet klauzulí
t0	sum
t_fin	0.001
cool	0.91
inner_max	(2 + 5 × (iter / itr_max)) × sum
thr_imp	2.5 × sum
itr_max	3 × sum × sum

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	100.00	24925
wuf20-71R-N	100.00	99.60	24925
wuf20-71R-Q	100.00	98.60	24925
wuf20-71R-R	100.00	99.80	24925
wuf20-91R-M	100.00	100.00	34293
wuf20-91R-N	99.80	98.40	34293
wuf20-91R-Q	99.80	89.60	34293
wuf20-91R-R	100.00	100.00	34293
wuf50-218R-M	93.40	63.60	88976
wuf50-218R-N	84.00	75.40	88976
wuf50-218R-Q	98.20	56.00	88976
wuf50-218R-R	88.20	68.80	88976

Uspokojivé výsledky ukázala tato konfigurace. Tu budu i nadále používat v black-box fázi. Pro instance wuf20-71R je průměrný procentní podíl nalezení řešení 100 %, nalezení optimálního řešení 99,5 %. Pro wuf20-91R je průměrný podíl nalezení řešení 99,9 %, nalezení optimálního řešení 97 %. Pro wuf50-218R je průměrný podíl nalezení řešení 90,95 %, nalezení optimálního řešení 65,95 %.

4 Black box

Pro fázi black-box jsem vybral všechny instance z následujících sad: wuf20-71R-M, wuf20-71R-N, wuf20-71R-Q, wuf20-71R-R, wuf20-91R-M, wuf20-91R-N, wuf20-91R-Q, wuf20-91R-R, wuf50-218R-M, wuf50-218R-N, wuf50-218R-Q, wuf50-218R-R. V každé sadě se nachází **100 instancí**, a každou instanci jsem spustil **100krát**.

Parametr	**Hodnota**
sum	počet literálů + počet klauzulí
t0	sum
t_fin	0.001
cool	0.91
inner_max	(2 + 5 × (iter / itr_max)) × sum
thr_imp	2.5 × sum
itr_max	3 × sum × sum

Výsledek:

Instances	Success Rate (%)	Optimal Rate (%)	Average Steps
wuf20-71R-M	100.00	99.91	24925
wuf20-71R-N	100.00	99.88	24925
wuf20-71R-Q	100.00	99.37	24925
wuf20-71R-R	100.00	99.46	24925
wuf20-91R-M	99.77	99.52	34293
wuf20-91R-N	99.79	99.49	34293
wuf20-91R-Q	99.76	99.50	34293
wuf20-91R-R	99.77	99.38	34293
wuf50-218R-M	89.11	64.98	88976
wuf50-218R-N	88.71	64.94	88976
wuf50-218R-Q	88.50	57.80	88976
wuf50-218R-R	89.28	58.60	88976

Pro instance wuf20-71R je průměrný procentní podíl nalezení řešení 100 %, nalezení optimálního řešení 99,655 %. Pro wuf20-91R je průměrný podíl nalezení řešení 99,7725 %, nalezení optimálního řešení 99.4725 %. Pro wuf50-218R je průměrný podíl nalezení řešení 88,9 %, nalezení optimálního řešení 61,58 %.

Výsledky fáze black-box byly u sad wuf20-71R stejné jako v případě výsledků fáze white-box s touto konfigurací, což ukazuje na stabilitu a konzistenci dosažených výsledků. U sad wuf20-91R byla mírně lepší úspěšnost nalezení optimálního řešení, a u větších instancí se výsledky zhoršily o několik procent. Tento výsledek je však v souladu s očekáváním. Pro menší instance byly dosaženy vynikající výsledky, s vysokým procentem nalezených řešení a optimálních řešení. U větších instancí se algoritmus stále dobře vyrovnal s nalezením řešení, ale schopnost najít optimální řešení byla nižší. Počet kroků je považován za vhodný pro daný rozsah instancí. I když výsledky black-box fáze splnily očekávání, je patrné, že u větších instancí stále existuje prostor pro zlepšení, zejména při hledání optimálních řešení.

Závěr

V této práci byla implementována heuristika pro řešení problému maximální vážené splnitelnosti booleovské formule (MWSAT) pomocí pokročilé metody simulovaného ochlazování. Heuristika byla navržena tak, aby dokázala efektivně zpracovávat instance s různými vlastnostmi a velikostmi, aniž by vyžadovala interaktivní zásahy.

Výsledky experimentálního vyhodnocení ukázaly, že heuristika dosahuje velmi dobrých výsledků na menších instancích (20 literálů, 71–91 klauzulí), kde je schopna nalézt optimální řešení ve vysokém procentu případů (+99 %). U větších instancí (50 literálů, 218 klauzulí) bylo dosaženo uspokojivých výsledků, algoritmus byl schopen nalézt řešení ve většině případů (+88 %), ale s menší úspěšností v hledání optimálních řešení (+61 %).