KOP úloha 2

Jorge Zuňiga
zunigjor
2022

Executive summary

Úvod

Práce se zabývá řešením instancí **problému MWSAT** pomocí **simulovaného ochlazování**.

Implementovaný algoritmus a nasazená heuristika jsou otestovány během experimentálních white box a black box fází. Z výsledků těchto experimentů jsou nakonec vyvozeny závěry.

MWSAT

Vstup: vektor X o n proměnných ($x_1 \dots x_n$), $x_i \in 0,1$, dále Booleova formule těchto proměnných v konjunktivní normální formě o mm klauzulích (součtových termech), dále pak pro každou proměnnou x váha w(x).

Výstup: ohodnocení Y proměnných X takové, že F(Y)=1 a součet vah proměnných, ohodnocených 1, je maximální. [2]

Sady instancí

Sady použité k experimentům pochází z [1]. Jedná se o instance SATu obohacené o váhy literálů.

Whitebox fáze byla spuštěna nad sadami **wuf20-91, wuf50-218, wuf75-325**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **10 instancí** a každá byla spuštěna **5 krát**.

Blackbox fáze byla spuštěna nad sadami **wuf20-91 a wuf50-218**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **100 instancí** a každá byla spuštěna **10 krát.**

White box fáze

Cílem white box experimentů bylo najít ohodnocení vstupních parametrů algoritmu simulovaného ochlazování. Každý z experimentů byl definován funkcí nastavující parametry a cenovou funkcí.

Celkem bylo spuštěno **dvanáct white box experimentů**. Výsledkem tohoto experimentu bylo **vylepšení cenové funkce a odladění parametrů pro blackbox**.

Black box fáze

Parametry pro black box fázi byly získány během experimentů z white box fáze.

temperature	math.pow(10, len(instance_data.weights)/10)
equilibrium	len(instance_data.weights) * 4
min_tepmerature	0.0001
cooling_coef	0.9
cost_coef	0.00001
cost funkce	$\frac{wsum \cdot coef + sat \cdot (1 - coef)}{twsum \cdot coef + tclaus \cdot (1 - coef)}$ Kde **coef** je koeficient, **wsum** je součet vah literálů nastavených na 1, **twsum** je celkový součet vah všech literálů, **sat** je počet splněných klauzulí, **tclaus** je celkový počet klauzulí.

Výsledky

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
blackbox	wuf20-91-M	1000	1000	1	1000	**1**	966	**0.966**
blackbox	wuf20-91-N	1000	1000	1	1000	**1**	973	**0.973**
blackbox	wuf20-91-Q	1000	999.989011	0.999989011	999	**0.999**	973	**0.973**
blackbox	wuf20-91-R	1000	999.5604396	0.9995604396	960	**0.96**	930	**0.93**
blackbox	wuf50-218-M	1000	998.5733945	0.9985733945	736	0.736	56	0.056
blackbox	wuf50-218-N	1000	998.5688073	0.9985688073	730	0.73	48	0.048
blackbox	wuf50-218-Q	1000	998.5688073	0.9985688073	719	0.719	76	0.076
blackbox	wuf50-218-R	1000	998.0458716	0.9980458716	636	0.636	58	0.058

Závěr

V této práci byl implementován **algoritmus simulovaného ochlazování** pro řešení **problému MWSAT**.

Během práce bylo provedeno **dvanáct white box experimentů**, které sloužily k úpravě parametrů pro black box fázi experimentu.

Díky experimentům **byla nasazena lepší heuristika**, která kvalitněji vyhodnocovala lepší stavy a **výrazně zlepšila výsledky**. Změna této vyhodnocovací funkce ale měla být doprovázena dalšími white box testy upravující nejen parametry teploty a ekvilibria.

Algoritmus je schopný s téměř stoprocentní úspěšností najít optimální řešení pro instance obsahující 20 literálů. Těžší instance už jsou problematičtější, úspěšnost nalezení řešení klesá na 70% a nalezení optima je velmi nepravděpodobné.

Nízká čísla nalezených optim naopak mohou naznačovat moc vysoké nastavení hodnoty ekvilibria, kdy se kvůli velkému počtu zkontrolovaných sousedů častěji stane, že algoritmus odskočí do horšího stavu.

Úvod

Práce se zabývá řešením instancí problému MWSAT pomocí simulovaného ochlazování. Implementovaný algoritmus a nasazená heuristika jsou otestovány během experimentálních white box a black box fází. Z výsledků těchto experimentů jsou nakonec vyvozeny závěry.

MWSAT

Problém MWSAT je rozšířením problému SAT o vážené literály. Problém patří do třídy NPO, tedy jedná se o optimalizační problém. Cílem je najít takové ohodnocení literálů, aby formule byla splněna a zároveň součet vah literálů bude maximální.

Vstup: vektor X o n proměnných ($x_1 \dots x_n$), $x_i \in 0,1$, dále Booleova formule těchto proměnných v konjunktivní normální formě o mm klauzulích (součtových termech), dále pak pro každou proměnnou x váha w(x).

Výstup: ohodnocení Y proměnných X takové, že F(Y)=1 a součet vah proměnných, ohodnocených 1, je maximální. [2]

Sady instancí

Sady použité k experimentům pochází z [1]. Jedná se o instance SATu obohacené o váhy literálů.

Whitebox fáze byla spuštěna nad sadami **wuf20-91, wuf50-218, wuf75-325**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **10 instancí** a každá byla spuštěna **5 krát**.

Blackbox fáze byla spuštěna nad sadami **wuf20-91 a wuf50-218**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **100 instancí** a každá byla spuštěna **10 krát.**

Algoritmus

Algoritmus simulovaného ochlazování je implementován v programovacím jazyce Python.

Simulované ochlazování začíná vytvořením náhodného stavu. Stavem v tomto kontextu je vektor ohodnocení proměnných. Stavy jsou implementovány pomocí třídy State, která obsahuje další atributy a metody. Klíčová je metoda get_cost(), které se věnuji dále v textu.

```
class State:
    def __init__(self, bitmap, instance_data, cost_coef, cost_function):
        self.bitmap = bitmap
        self.instance_data: InstanceData = instance_data
        self.cost_coef = cost_coef
        self.cost_function = cost_function
        self.max_weight = max(self.instance_data.weights)
        self.weight_sum = self.get_weight_sum()
        self.total_clauses = len(self.instance_data.clauses)
        self.satisifed_clauses = self.get_satisfied_clauses()
        self.cost = self.get_cost()
```

Algoritmus má ve své podstatě dvě smyčky. Vnější, která se řídí dle parametrů **počáteční teploty**, **minimální teploty** a **ochlazovacího koeficientu**. V této smyčce dochází k postupnému ochlazování, které končí ve chvíli kdy teplota dosáhne určeného minima.

Vnitřní smyčka prohledává sousední stavy a provádí **ekvilibrium** iterací. Sousedním stavem chápeme stav, který má jinak ohodnocenou pouze jednu proměnnou.

Prohledávání tedy probíhá vytvořením nového stavu, porovnáním jeho ceny se stavem současným a v případě lepšího výsledku přesunutí se do tohoto stavu. Po dosažení minimální teploty algoritmus končí a vrací nejlepší stav.

```
def simulated annealing (init_state, equilibrium, init_temperature,
                         cooling_coefficient, min_temperature ):
    state = init_state
    temperature = init_temperature
    best_state = init_state
    while not frozen(temperature, min_temperature):
        for e in range (equilibrium):
            if is_better(state , best_state):
                best_state = state
            state = get_neighbour(state, temperature )
        temperature = cool(temperature, cooling_coefficient)
    return best_state
```

Jak bylo popsáno dříve, sousedem je stav, který má jinak ohodnocen právě jeden literál. Algoritmus nejdříve vybírá literály z nesplněných klauzulí. V případě, že jsou splněny všechny klauzule, vybírá literál náhodně.

Pro vymanění se z lokálních optim může algoritmus v 10% případů přejít i do stavu horšího. Dalším způsobem jak se vymanit z lokálních optim, je možnost přijmout řešení, které není o moc horší.

```
def get_neighbour(state, temperature):
    new_state = get_random_neighbour(state)
    if is_better(new_state, state) or random.randint(1, 100) < 10:
        return new_state
    delta = how_much_worse(new_state, state)
    if 0.6 < math.exp(-delta/Decimal(temperature)):
        return new_state
    return state
```

Pro další testování byl celý běh simulovaného ochlazování parametrizován dvěma funkcemi. První, která nastavovala proměnné a druhou, která sloužila jako cost funkce stavu. Toto je dále rozebráno v následující kapitole.

White box fáze

Sledované metriky

Jednotlivé běhy byly hodnoceny na základě následujících metrik:

Průměrná chyba Vyjadřuje poměr nesplněných klauzulí neboli:

$$1 - \frac{počet\ splněných\ klauzulí}{počet\ všech\ klauzulí}$$

Poměr nalezených řešení Poměr případů kdy algoritmus našel řešení, tedy vektor splňující formulí, ku všem běhům.

Poměr optimálních řešení Poměr případů kdy algoritmus našel optimální řešení ku všem řešením.

Datové sady

Jak bylo zmíněno dříve, white box experimenty byly spuštěny na sadách **wuf20-91, wuf50-218, wuf75-325**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **10 instancí** a každá byla spuštěna **5 krát**.

Experimenty

Cílem experimentů bylo najít ohodnocení vstupních parametrů algoritmu simulovaného ochlazování. Každý z experimentů byl definován funkcí nastavující parametry a cenovou funkcí.

Celkem bylo spuštěno **dvanáct white box experimentů**.

Hledání hodnot parametrů

Nastavení parametrů je definováno pomocí funkcí v souboru box.py. Tyto funkce vypadají zhruba takto:

```
def whitebox_0(instance_data):
    temperature = 10000
    equilibrium = 20
    min_tepmerature = 0.0001
    cooling_coef = 0.9
    cost_coef = 3
    return temperature, equilibrium, min_tepmerature, cooling_coef, cost_coef
```

Vstupním parametrem těchto funkcí jsou informace o instanci, tedy třída InstanceData ze souboru instance_parser.py.

Jak jednotlivé funkce nastavují proměnné je ukázáno v následující tabulce:

name	**temperature**	**equilibrium**	**min. temp.**	**cool. coef.**	**cost coef.**
whitebox_0	10 000	20	0.0001	0.9	3
whitebox_1	100 000	20	0.0001	0.9	3
whitebox_2	10 000	50	0.0001	0.9	3
whitebox_3	sum(instance_data.weights)	len(instance_data.weights)	0.0001	0.9	3
whitebox_4	sum(instance_data.weights)	len(instance_data.weights) * 2	0.0001	0.9	3
whitebox_5	sum(instance_data.weights)	len(instance_data.weights) * 2	0.0001	0.9	1
whitebox_6	sum(instance_data.weights)	len(instance_data.weights) * 2	0.0001	0.9	5
whitebox_7	len(instance_data.weights) * max(instance_data.weights)	len(instance_data.weights) * 4	0.0001	0.9	5
whitebox_8	math.pow(10, len(instance_data.weights)/10)	len(instance_data.weights) * 4	0.0001	0.9	5
whitebox_9	math.pow(10, len(instance_data.weights)/10)	len(instance_data.weights) * 4	0.0001	0.9	0.1
whitebox_10	math.pow(10, len(instance_data.weights)/10)	len(instance_data.weights) * 4	0.0001	0.9	0.001
whitebox_11	math.pow(10, len(instance_data.weights)/10)	len(instance_data.weights) * 4	0.0001	0.9	0.00001

Tyto hodnoty nejsou náhodné, jednotlivé úpravy hodnot parametrů byly převážně určeny pomocí průběžných výsledků experimentů. Šedé zvýraznění naznačuje změnu oproti předchozí hodnotě.

První změny se týkaly samotných parametrů **temperature** a **equilibrium**, velmi brzo bylo jasné, že tyto parametry bude vhodé škálovat s velikostí instance.

Změna minimální teploty a ochlazovacího koeficientu mají podobný efekt jako zvyšování teploty, tedy vyšší počet opakování vnější smyčky algoritmu. Pro zakotvení výsledků jsem s těmito parametry nehýbal.

Ve white boxech 5, 6, 7 byl pokus o zlepšení výsledků pomocí koeficientu cost funkce, zde se ukázalo, že vyšší koeficient u whitebox_7 společně se zvednutou teplotou jemně pomohli výsledkům.

Whitebox_8 byl posledním pokusem s první cenovou funkcí, kdy další zvýšení teploty pomohlo už jen velmi málo. V tuto chvíli bylo jasné, že s touto cost funkcí nebudou výsledky o moc lepší.

Velmi výrazná změna koeficientu cost funkce od whitebox_9 dál, je kvůli **změně cenové funkce**, která je rozebrána v další sekci.

Cenová funkce

Prvních devět white box experimentů bylo spuštěno s následující cost funkcí:

```
def cost_function_0(state):
    return state.weight_sum - (state.cost_coef * state.max_weight) *
(state.total_clauses - state.satisifed_clauses)
```

Myšlenka za touto funkcí byla prostá. Cena stavu se odvíjela od součtu vah literálů nastavených na 1. Tento součet byl penalizován podle počtu nesplněných klauzulí krát nejvyšší váha pronásobená koeficientem.

Tato funkce se ale ukázala být nevhodná. Algoritmus se soustředil příliš na optimalizaci vah a ne na samotné splnění formule. Výsledky posloužily dostatečně dobře pro prvotní nastavení parametrů.

Zbylé white box experimenty pracovaly s následující cenovou funkcí:

```
def cost_function_1(state):
    return Decimal(
        (state.weight_sum * state.cost_coef) + (state.satisifed_clauses * (1 -
state.cost_coef))
    ) / Decimal(
        (state.instance_data.max_w_sum * state.cost_coef) +
(len(state.instance_data.clauses) * (1 - state.cost_coef))
    )
```

Tato funkce už je poměrně sofistikovanější. Koeficient slouží převážně k určení zda má být výsledek funkce více ovlivněn váhami nebo splněnými klauzulemi. V kódu je přesný předpis poměrně nečitelný, matematický předpis této funkce je:

$$\frac{wsum \cdot coef + sat \cdot (1 - coef)}{twsum \cdot coef + tclaus \cdot (1 - coef)}$$

Kde **coef** je koeficient, **wsum** je součet vah literálů nastavených na 1, **twsum** je celkový součet vah všech literálů, **sat** je počet splněných klauzulí, **tclaus** je celkový počet klauzulí.

Na tomto předpisu je dobře vidět jak koeficient ovlivňuje chování. Tedy čím menší hodnota koeficientu, tím více funkce odměňuje splněné klauzule.

Výsledky white box experimentu

wuf20-91-M

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf20-91-M	50	49.87912088	0.9975824176	39	0.78	23	0.46
whitebox_1	wuf20-91-M	50	49.95604396	0.9991208791	46	0.92	26	0.52
whitebox_2	wuf20-91-M	50	49.96703297	0.9993406593	47	0.94	33	0.66
whitebox_3	wuf20-91-M	50	49.86813187	0.9973626374	39	0.78	23	0.46
whitebox_4	wuf20-91-M	50	49.96703297	0.9993406593	47	0.94	37	0.74
whitebox_5	wuf20-91-M	50	49.85714286	0.9971428571	41	0.82	30	0.6
whitebox_6	wuf20-91-M	50	49.92307692	0.9984615385	43	0.86	25	0.5
whitebox_7	wuf20-91-M	50	49.96703297	0.9993406593	47	0.94	41	0.82
whitebox_8	wuf20-91-M	50	49.95604396	0.9991208791	46	0.92	36	0.72
whitebox_9	wuf20-91-M	50	49.10989011	0.9821978022	31	0.62	31	0.62
whitebox_10	wuf20-91-M	50	50	1	50	1	50	1
whitebox_11	wuf20-91-M	50	50	1	50	1	50	1

wuf20-91-N

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf20-91-N	50	49.75824176	0.9951648352	30	0.6	15	0.3
whitebox_1	wuf20-91-N	50	49.89010989	0.9978021978	41	0.82	25	0.5
whitebox_2	wuf20-91-N	50	49.94505495	0.9989010989	45	0.9	33	0.66
whitebox_3	wuf20-91-N	50	49.91208791	0.9982417582	43	0.86	24	0.48
whitebox_4	wuf20-91-N	50	49.92307692	0.9984615385	44	0.88	30	0.6
whitebox_5	wuf20-91-N	50	49.87912088	0.9975824176	43	0.86	36	0.72
whitebox_6	wuf20-91-N	50	49.96703297	0.9993406593	47	0.94	32	0.64
whitebox_7	wuf20-91-N	50	49.98901099	0.9997802198	49	0.98	45	0.9
whitebox_8	wuf20-91-N	50	49.96703297	0.9993406593	47	0.94	37	0.74
whitebox_9	wuf20-91-N	50	49.13186813	0.9826373626	30	0.6	30	0.6
whitebox_10	wuf20-91-N	50	49.8021978	0.996043956	39	0.78	38	0.76
whitebox_11	wuf20-91-N	50	50	1	50	1	48	0.96

wuf20-91-Q

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf20-91-Q	50	49.64835165	0.992967033	22	0.44	15	0.3
whitebox_1	wuf20-91-Q	50	49.56043956	0.9912087912	13	0.26	8	0.16
whitebox_2	wuf20-91-Q	50	49.63736264	0.9927472527	18	0.36	18	0.36
whitebox_3	wuf20-91-Q	50	49.51648352	0.9903296703	15	0.3	10	0.2
whitebox_4	wuf20-91-Q	50	49.63736264	0.9927472527	22	0.44	21	0.42
whitebox_5	wuf20-91-Q	50	49.0989011	0.981978022	2	0.04	2	0.04
whitebox_6	wuf20-91-Q	50	49.84615385	0.9969230769	36	0.72	27	0.54
whitebox_7	wuf20-91-Q	50	49.81318681	0.9962637363	33	0.66	30	0.6
whitebox_8	wuf20-91-Q	50	49.81318681	0.9962637363	35	0.7	31	0.62
whitebox_9	wuf20-91-Q	50	46.24175824	0.9248351648	0	0	0	0
whitebox_10	wuf20-91-Q	50	49.15384615	0.9830769231	7	0.14	7	0.14
whitebox_11	wuf20-91-Q	50	50	1	50	1	49	0.98

wuf20-91-R

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf20-91-R	50	49.64835165	0.992967033	22	0.44	14	0.28
whitebox_1	wuf20-91-R	50	49.6043956	0.9920879121	19	0.38	16	0.32
whitebox_2	wuf20-91-R	50	49.68131868	0.9936263736	23	0.46	22	0.44
whitebox_3	wuf20-91-R	50	49.53846154	0.9907692308	14	0.28	8	0.16
whitebox_4	wuf20-91-R	50	49.62637363	0.9925274725	20	0.4	18	0.36
whitebox_5	wuf20-91-R	50	49.04395604	0.9808791209	3	0.06	3	0.06
whitebox_6	wuf20-91-R	50	49.78021978	0.9956043956	30	0.6	20	0.4
whitebox_7	wuf20-91-R	50	49.83516484	0.9967032967	35	0.7	34	0.68
whitebox_8	wuf20-91-R	50	49.79120879	0.9958241758	32	0.64	25	0.5
whitebox_9	wuf20-91-R	50	46.2967033	0.9259340659	0	0	0	0
whitebox_10	wuf20-91-R	50	46.54945055	0.930989011	0	0	0	0
whitebox_11	wuf20-91-R	50	50	1	50	1	49	0.98

wuf50-218-M

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf50-218-M	50	49.32110092	0.9864220183	4	0.08	0	0
whitebox_1	wuf50-218-M	50	49.38073394	0.9876146789	5	0.1	0	0
whitebox_2	wuf50-218-M	50	49.50917431	0.9901834862	8	0.16	0	0
whitebox_3	wuf50-218-M	50	49.63761468	0.9927522936	11	0.22	1	0.02
whitebox_4	wuf50-218-M	50	49.70183486	0.9940366972	15	0.3	0	0
whitebox_5	wuf50-218-M	50	49.65137615	0.9930275229	16	0.32	1	0.02
whitebox_6	wuf50-218-M	50	49.73394495	0.9946788991	17	0.34	0	0
whitebox_7	wuf50-218-M	50	49.80733945	0.996146789	19	0.38	0	0
whitebox_8	wuf50-218-M	50	49.84862385	0.9969724771	24	0.48	0	0
whitebox_9	wuf50-218-M	50	49.24311927	0.9848623853	11	0.22	2	0.04
whitebox_10	wuf50-218-M	50	49.84862385	0.9969724771	29	0.58	2	0.04
whitebox_11	wuf50-218-M	50	49.90825688	0.9981651376	32	0.64	2	0.04

wuf50-218-N

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf50-218-N	50	49.31651376	0.9863302752	5	0.1	0	0
whitebox_1	wuf50-218-N	50	49.3853211	0.987706422	4	0.08	0	0
whitebox_2	wuf50-218-N	50	49.5	0.99	10	0.2	0	0
whitebox_3	wuf50-218-N	50	49.58715596	0.9917431193	8	0.16	0	0
whitebox_5	wuf50-218-N	50	49.57798165	0.991559633	13	0.26	0	0
whitebox_4	wuf50-218-N	50	49.74770642	0.9949541284	14	0.28	0	0
whitebox_6	wuf50-218-N	50	49.66972477	0.9933944954	9	0.18	1	0.02
whitebox_7	wuf50-218-N	50	49.78899083	0.9957798165	18	0.36	0	0
whitebox_8	wuf50-218-N	50	49.79816514	0.9959633028	17	0.34	0	0
whitebox_9	wuf50-218-N	50	49.28440367	0.9856880734	7	0.14	0	0
whitebox_10	wuf50-218-N	50	49.64220183	0.9928440367	17	0.34	2	0.04
whitebox_11	wuf50-218-N	50	49.93119266	0.9986238532	37	0.74	3	0.06

wuf50-218-Q

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf50-218-Q	50	49.32110092	0.9864220183	4	0.08	0	0
whitebox_1	wuf50-218-Q	50	49.45412844	0.9890825688	6	0.12	0	0
whitebox_2	wuf50-218-Q	50	49.47706422	0.9895412844	5	0.1	0	0
whitebox_3	wuf50-218-Q	50	49.56880734	0.9913761468	9	0.18	0	0
whitebox_4	wuf50-218-Q	50	49.5412844	0.9908256881	9	0.18	0	0
whitebox_5	wuf50-218-Q	50	49.17431193	0.9834862385	1	0.02	0	0
whitebox_6	wuf50-218-Q	50	49.70183486	0.9940366972	13	0.26	1	0.02
whitebox_7	wuf50-218-Q	50	49.76146789	0.9952293578	14	0.28	0	0
whitebox_8	wuf50-218-Q	50	49.77981651	0.9955963303	17	0.34	0	0
whitebox_9	wuf50-218-Q	50	46.7293578	0.934587156	0	0	0	0
whitebox_10	wuf50-218-Q	50	49.28899083	0.9857798165	4	0.08	0	0
whitebox_11	wuf50-218-Q	50	49.89449541	0.9978899083	31	0.62	0	0

wuf50-218-R

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf50-218-R	50	49.2293578	0.984587156	3	0.06	0	0
whitebox_1	wuf50-218-R	50	49.30733945	0.986146789	4	0.08	0	0
whitebox_2	wuf50-218-R	50	49.48623853	0.9897247706	5	0.1	0	0
whitebox_3	wuf50-218-R	50	49.56422018	0.9912844037	6	0.12	0	0
whitebox_4	wuf50-218-R	50	49.60091743	0.9920183486	8	0.16	0	0
whitebox_5	wuf50-218-R	50	49.16513761	0.9833027523	2	0.04	0	0
whitebox_6	wuf50-218-R	50	49.6559633	0.9931192661	11	0.22	0	0
whitebox_7	wuf50-218-R	50	49.79357798	0.9958715596	15	0.3	1	0.02
whitebox_8	wuf50-218-R	50	49.74770642	0.9949541284	12	0.24	0	0
whitebox_9	wuf50-218-R	50	46.49082569	0.9298165138	0	0	0	0
whitebox_10	wuf50-218-R	50	47.25229358	0.9450458716	0	0	0	0
whitebox_11	wuf50-218-R	50	49.88990826	0.9977981651	28	0.56	1	0.02

wuf75-325-M

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf75-325-M	50	49.01230769	0.9802461538	0	0	0	0
whitebox_1	wuf75-325-M	50	49.10153846	0.9820307692	2	0.04	0	0
whitebox_2	wuf75-325-M	50	49.24923077	0.9849846154	0	0	0	0
whitebox_3	wuf75-325-M	50	49.37230769	0.9874461538	3	0.06	0	0
whitebox_4	wuf75-325-M	50	49.48923077	0.9897846154	4	0.08	0	0
whitebox_5	wuf75-325-M	50	49.31076923	0.9862153846	4	0.08	0	0
whitebox_6	wuf75-325-M	50	49.48615385	0.9897230769	3	0.06	0	0
whitebox_7	wuf75-325-M	50	49.59076923	0.9918153846	8	0.16	0	0
whitebox_8	wuf75-325-M	50	49.70461538	0.9940923077	9	0.18	0	0
whitebox_9	wuf75-325-M	50	48.67384615	0.9734769231	0	0	0	0
whitebox_10	wuf75-325-M	50	49.64	0.9928	13	0.26	0	0
whitebox_11	wuf75-325-M	50	49.81538462	0.9963076923	19	0.38	1	0.02

wuf75-325-N

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf75-325-N	50	48.93846154	0.9787692308	1	0.02	0	0
whitebox_1	wuf75-325-N	50	49.05846154	0.9811692308	0	0	0	0
whitebox_2	wuf75-325-N	50	49.24307692	0.9848615385	4	0.08	0	0
whitebox_3	wuf75-325-N	50	49.43076923	0.9886153846	3	0.06	0	0
whitebox_4	wuf75-325-N	50	49.45846154	0.9891692308	3	0.06	0	0
whitebox_5	wuf75-325-N	50	49.33230769	0.9866461538	4	0.08	0	0
whitebox_6	wuf75-325-N	50	49.51076923	0.9902153846	4	0.08	0	0
whitebox_7	wuf75-325-N	50	49.59692308	0.9919384615	9	0.18	0	0
whitebox_8	wuf75-325-N	50	49.69846154	0.9939692308	12	0.24	0	0
whitebox_9	wuf75-325-N	50	48.90461538	0.9780923077	0	0	0	0
whitebox_10	wuf75-325-N	50	49.19384615	0.9838769231	3	0.06	0	0
whitebox_11	wuf75-325-N	50	49.77846154	0.9955692308	14	0.28	0	0

wuf75-325-Q

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf75-325-Q	50	49.01230769	0.9802461538	1	0.02	0	0
whitebox_1	wuf75-325-Q	50	49.09230769	0.9818461538	1	0.02	0	0
whitebox_2	wuf75-325-Q	50	49.22153846	0.9844307692	0	0	0	0
whitebox_3	wuf75-325-Q	50	49.31384615	0.9862769231	1	0.02	0	0
whitebox_4	wuf75-325-Q	50	49.44615385	0.9889230769	2	0.04	0	0
whitebox_5	wuf75-325-Q	50	49.05538462	0.9811076923	0	0	0	0
whitebox_6	wuf75-325-Q	50	49.47076923	0.9894153846	4	0.08	0	0
whitebox_7	wuf75-325-Q	50	49.53846154	0.9907692308	3	0.06	0	0
whitebox_8	wuf75-325-Q	50	49.65230769	0.9930461538	7	0.14	0	0
whitebox_9	wuf75-325-Q	50	47.01230769	0.9402461538	0	0	0	0
whitebox_10	wuf75-325-Q	50	49.44923077	0.9889846154	1	0.02	0	0
whitebox_11	wuf75-325-Q	50	49.77538462	0.9955076923	16	0.32	0	0

wuf75-325-R

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
whitebox_0	wuf75-325-R	50	48.96307692	0.9792615385	1	0.02	0	0
whitebox_1	wuf75-325-R	50	49.10153846	0.9820307692	0	0	0	0
whitebox_2	wuf75-325-R	50	49.25230769	0.9850461538	1	0.02	0	0
whitebox_3	wuf75-325-R	50	49.33230769	0.9866461538	0	0	0	0
whitebox_4	wuf75-325-R	50	49.44	0.9888	0	0	0	0
whitebox_5	wuf75-325-R	50	48.97230769	0.9794461538	0	0	0	0
whitebox_6	wuf75-325-R	50	49.46769231	0.9893538462	2	0.04	0	0
whitebox_7	wuf75-325-R	50	49.64	0.9928	8	0.16	0	0
whitebox_8	wuf75-325-R	50	49.67076923	0.9934153846	9	0.18	0	0
whitebox_9	wuf75-325-R	50	46.94153846	0.9388307692	0	0	0	0
whitebox_10	wuf75-325-R	50	47.56307692	0.9512615385	0	0	0	0
whitebox_11	wuf75-325-R	50	49.78769231	0.9957538462	15	0.3	0	0

Závěr white box experimentů

Zvyšování teploty a ekvilibria podle velikosti instance pomáhá k lepším výsledkům. Zvyšování teploty ale nevede lineárně k lepším výsledkům a proto je potřeba nasadit lepší heuristiku pomocí sofistikovanější cost funkce.

Celkově si nejlépe vedl whitebox_11, který byl nakonec vybrán jako základ pro black box experiment.

Black box fáze

Nastavení parametrů

Finální nastavení black box fáze je určeno pomocí funkce blackbox a cost funkce:

```
def blackbox(instance_data):
    temperature = math.pow(10, len(instance_data.weights)/10)
    equilibrium = len(instance_data.weights) * 4
    min_tepmerature = 0.0001
    cooling_coef = 0.9
    cost_coef = 0.00001
    return temperature, equilibrium, min_tepmerature, cooling_coef, cost_coef


def cost_function_1(state):
    return Decimal(
        (state.weight_sum * state.cost_coef) + (state.satisifed_clauses * (1 -
state.cost_coef))
    ) / Decimal(
        (state.instance_data.max_w_sum * state.cost_coef) +
(len(state.instance_data.clauses) * (1 - state.cost_coef))
    )
```

Respektive takto:

temperature	math.pow(10, len(instance_data.weights)/10)
equilibrium	len(instance_data.weights) * 4
min_tepmerature	0.0001
cooling_coef	0.9
cost_coef	0.00001
cost funkce	$\frac{wsum \cdot coef + sat \cdot (1 - coef)}{twsum \cdot coef + tclaus \cdot (1 - coef)}$

Instance

Blackbox fáze byla spuštěna nad sadami **wuf20-91 a wuf50-218**. Z těchto sad byly vybrány všechny sady **-M, -N, -Q, -R,** z každé bylo vybráno **100 instancí** a každá byla spuštěna **10 krát.**

Výsledky a závěry

name	instances	runs	satisfied clauses	satisfied clauses rate	solutions found	solutions found rate	optimums reached	optimums reached rate
blackbox	wuf20-91-M	1000	1000	1	1000	1	966	0.966
blackbox	wuf20-91-N	1000	1000	1	1000	1	973	0.973
blackbox	wuf20-91-Q	1000	999.989011	0.999989011	999	0.999	973	0.973
blackbox	wuf20-91-R	1000	999.5604396	0.9995604396	960	0.96	930	0.93
blackbox	wuf50-218-M	1000	998.5733945	0.9985733945	736	0.736	56	0.056
blackbox	wuf50-218-N	1000	998.5688073	0.9985688073	730	0.73	48	0.048
blackbox	wuf50-218-Q	1000	998.5688073	0.9985688073	719	0.719	76	0.076
blackbox	wuf50-218-R	1000	998.0458716	0.9980458716	636	0.636	58	0.058

Sady wuf20-91-M, wuf20-91-N, wuf20-91-Q, wuf20-91-R

Výsledky pro instance z těchto sad byly dobré. Převážně u sad -M a -N se algoritmu podařilo vždy splnit všechny klauzule a tedy najít řešení. Toto řešení bylo optimální pouze v 96.6% a 97.3% případů, není to špatný výsledek ale mohl by být lepší. O něco málo horší jsou výsledky u sad -Q a -R.

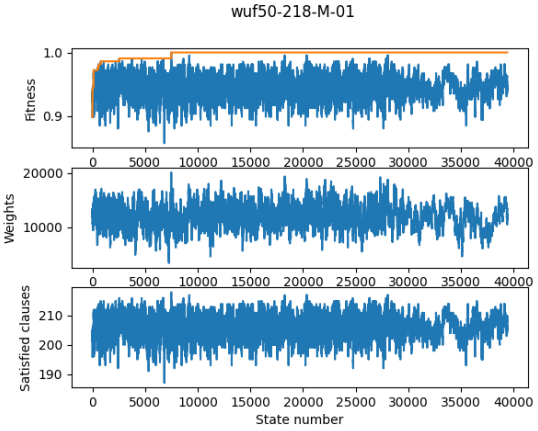
Sady wuf50-218-M, wuf50-218-N, wuf50-218-Q, wuf50-218-R

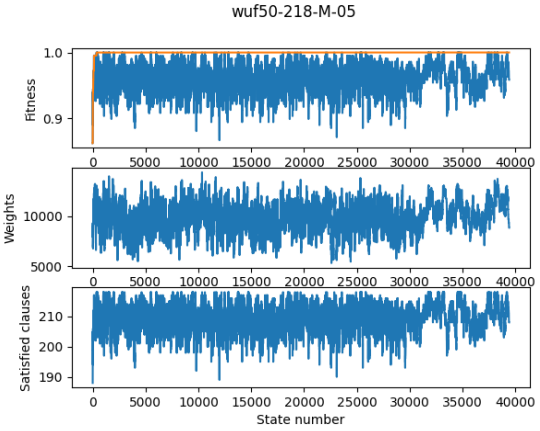
Výsledky pro instance z těchto sad už byly horší. Ačkoli se čísla **splněných klauzulí** držela vysoko, samotné počty nalezených řešení a dosažených optim byly podstatně nižší.

Podle vysokých čísel splněných klauzulí očekávám, že mírné zvýšení ochlazovacího koeficientu, tedy zvýšení počtu vnějších cyklu algoritmu by mohlo vést k lepším výsledkům.

Nízká čísla nalezených optim naopak mohou naznačovat moc vysoké nastavení hodnoty ekvilibria, kdy se kvůli velkému počtu zkontrolovaných sousedů častěji stane, že algoritmus odskočí do horšího stavu. Toto mohlo být odchyceno ještě ve white box fázi.

Podíváme-li se na grafy běhů, kdy bylo nalezeno optimum (wuf50-218-M-01 nebo wuf50-218-M-01), je vidět, že tohoto optima bylo dosaženo poměrně brzo a algoritmus poté přecházel do horších stavů.





Závěr

V této práci byl implementován **algoritmus simulovaného ochlazování** pro řešení **problému MWSAT**.

Během práce bylo provedeno **dvanáct white box experimentů**, které sloužily k úpravě parametrů pro black box fázi experimentu.

Díky experimentům **byla nasazena lepší heuristika**, která kvalitněji vyhodnocovala lepší stavy a **výrazně zlepšila výsledky**. Změna této vyhodnocovací funkce ale měla být doprovázena dalšími white box testy upravující nejen parametry teploty a ekvilibria.

Algoritmus je schopný s téměř stoprocentní úspěšností najít optimální řešení pro instance obsahující 20 literálů. Těžší instance už jsou problematičtější, úspěšnost nalezení řešení klesá na 70% a nalezení optima je velmi nepravděpodobné.

Zdroje

[1] Jan Schmidt. Data a programy.
url: https://courses.fit.cvut.cz/NI- KOP/download/index.html#mwsatinst.

[2] Jan Schmidt. Splnitelnost Booleovy formule.
url: https://courses.fit.cvut.cz/NI -KOP/problems/sat.html#mwsat.