Solving MWSAT using Genetic Algorithm

Lukáš Tomáš Petrželka

20.1.2024

Contents

1 Úvod **3**
- 1.1 Zadání . . . 3
- 1.2 Data . . . 3
- 1.3 Implementace . . . 3
- 1.4 Základní nastavení genetického algoritmu . . . 4
  - 1.4.1 Genotyp . . . 4
  - 1.4.2 Inicializace populace . . . 4
  - 1.4.3 Selekce . . . 4
  - 1.4.4 Křížení . . . 4
  - 1.4.5 Mutace . . . 4

2 White box fáze **5**
- 2.1 Iterace 1 . . . 5
- 2.2 Iterace 2 . . . 7
- 2.3 Iterace 3 . . . 8
- 2.4 Iterace 4 . . . 8
- 2.5 Iterace 5 . . . 9

3 Black box fáze **12**
- 3.1 Úvod . . . 12
- 3.2 Metody a materiály . . . 12
- 3.3 Výsledky . . . 12

4 Závěr **13**

1 Úvod

1.1 Zadání

Problém MWSAT (**M**aximum **W**eighted **SAT**isfiability problem) je optimalizační verzí problému booleovské splnitelnosti (SAT). V tomto problému je dána booleovská formule, obvykle v konjunktivní normální formě (CNF), spolu s váhou přiřazenou každé klauzuli. Cílem je určit pravdivostní přiřazení proměnných, které maximalizuje celkovou váhu splněných klauzulí.

Cílem této práce je řešit tento problém pomocí genetického algoritmu a ověřit kvalitu zvolené heuristiky experimentálním vyhodnocením.

1.2 Data

Data byly staženy ze stránky předmětu. Konkrétní vybrané sady instancí použité v tomto projektu jsou v tabulce 1.

Název	Počet proměnných	Počet klauzulí	Obsahuje řešení
wuf20-71	20	71	Ano
wuf36-157	36	157	Ano
wuf50-218	50	218	Ano
wuf75-325	75	325	Ano
wuf100-430	100	430	Ne

Table 1: Dataset

1.3 Implementace

Genetický algoritmus byl naprogramován v jazyku C++ z důvodu požadavku na optimalizaci. Experimenty a jejich vyhodnocení byly dále provedeny s pomocí jazyku Python. Samotné vizualizace a zkoumané metriky se nacházejí v jupyter notebooku. Celý kód je dostupný zde.

Níže je uveden výstup nápovědy programu pro řešení MWSAT instance pomocí Genetického Algoritmu:

```
.Usage: ./ga_solve [--help] [--version] [--seed VAR] [--max-generations VAR] [--population-size VAR] [--elitism VAR]
                   [--mutation-probability VAR] [--mutation-method VAR] [--fitness-alpha VAR] [--fitness-beta VAR]
                   [--crossover VAR] cnf_file

Genetic algorithm for solving MWSAT problem.

Positional arguments:
  cnf_file                Path to the CNF file.

Optional arguments:
  -h, --help              shows help message and exits
  -v, --version           prints version information and exits
  -s, --seed              Seed for random number generator. [nargs=0..1] [default: 0]
  --max-generations       Maximum number of generations. [nargs=0..1] [default: 400]
  --population-size       Population size. If population size is set -1, it will be computed adaptively. [nargs=0..1] [default: 1
  --elitism               Ratio of best individuals to select for the next generation. [nargs=0..1] [default: 0.5]
  --mutation-probability  Probability of mutation. [nargs=0..1] [default: 0.8]
  --mutation-method       Mutation method, possible values: fixed linear-adaptive  [nargs=0..1] [default: "fixed"]
  --fitness-alpha         Alpha parameter for fitness function. [nargs=0..1] [default: 0.5]
  --fitness-beta          Beta parameter for fitness function. [nargs=0..1] [default: -1]
  --crossover             Crossover method, possible values: one-point two-point uniform  [nargs=0..1] [default: "one-point"]
```

1.4 Základní nastavení genetického algoritmu

Toto nastavení genetického algoritmu bylo společné pro veškeré iterace white box fáze a i pro black box fázi.

1.4.1 Genotyp

Genotypem (reprezentace jedince) je binární řetězec o délce počtu proměnných problému, kde jednotlivé bity představují ohodnocení konkrétních proměnných.

1.4.2 Inicializace populace

Populace (množina jedinců) z počáteční generace je inicializována náhodně. Tedy každý jedinec je náhodným binárním řetězcem. Hyperparametr *population size* je laděn ve white-box fázi.

1.4.3 Selekce

Krok selekce v genetickém algoritmu určuje, kteří jedinci budou vybráni ke křížení. Byla zvolena *metoda ruletového výběru*. Ta vybere jedince s pravděpodobností proporcionální k jeho fitness hodnotě. Jedinci s vyšší fitness funkcí mají větší šanci na výběr. Jeden jedinec může být vybrán opakovaně.

1.4.4 Křížení

Bylo zvoleno *uniformní křížení*, které pro každý bit genotypu výsledného jedince náhodně vybere jednoho z rodičů, od kterého bude tento bit vybrán.

1.4.5 Mutace

Mutace jedinců je provedena tak, že každý bit genotypu je s pravděpodobností *mutation probability* invertován. Hodnota tohoto hyperparametr je také laděna ve white-box fázi.

2 White box fáze

Tato fáze slouží k ladění heuristiky s přístupem k algoritmu. Použil jsem v ní převážně datasety s instacemi menší velikosti, a které také mají výsledky: *wuf20-71* a *wuf36-157*. Kromě poslední (5.) fáze, kde jsem použil ještě *wuf50-218*. V každé fázi jsem zvolil kromě sady M také Q, kde váhy vytvářejí (mírně) zavádějící úlohu. Konkrétně jsem vybral z každé sady 10 instancí.

Pro reprodukovatelnost výsledků jsem zvolil pevně daný seed 123, který jsem použil k pseudonáhodnému generování několika dalších seedů. Každá instance byla následně spuštěna s jiným seedem. Tímto způsobem do určité míry eliminuji vliv náhodnosti na výsledky algoritmu. Konkrétně jsem v této fázi vygeneroval 5 různých seed hodnot a to z důvodu nedostatku výpočetních prostředků.

U výsledků sleduji několik metrik. Nejdříve jsem pro každou metriku každou instanci spočítal průměr přes všechny různé seed hodnoty. Následně jsem spočítal průměr přes všechy instance.

- **Poměr splněných klauzulí**
- **Poměr splnění** - počet běhů instance, kde bylo nalezeno řešení, kde ohodnocení dělá formuli splněnou
- **Relativní chyba** - $\frac{|S_{\text{optimal}} - S|}{S_{\text{optimal}}}$ - umožňuje srovnávat různé instance díky normalizaci
- **Hodnota fitness funkce** - pouze u první iterace

2.1 Iterace 1

V první iteraci jsem se snažil vymyslet a otestovat fitness funkci.

Table 2: Hyperparametry Genetického Algoritmu nastavené v iteraci 1

Parametr	**Hodnota**
Max generations	1000
Population size	300
Elitism	0.5
Mutation Probability	0.3
Crossover	Uniform
Fitness-alpha	hledaná

Nejdřive jsem vymyslel tuto první fitness funkci, u které jsem dále zkoumal, jak nastavit hodnotu α:

$$\text{fitness(chromosome)} = \alpha \cdot \frac{C}{\Sigma_C} + (1 - \alpha) \cdot \frac{W}{\Sigma_W}$$

kde jednotlivé proměnné jsou:

- C - počet splněných klauzulí
- Σ_C - počet všech klauzulí
- W - suma vah proměnných nastavených na 1

- Σ_W - suma vah všech proměnných
- $\alpha\ in \langle 0, 1 \rangle$ - váhový koeficient určující důležitost maximalizace splněných klauzulí (C) oproti maximalizaci váhy proměnných (W).

Vzhledem k normalizaci výrazů $\frac{C}{\Sigma_C}$ a $\frac{W}{\Sigma_W}$ je možné porovnávat fitness hodnoty řešení na instancích různé velikosti.

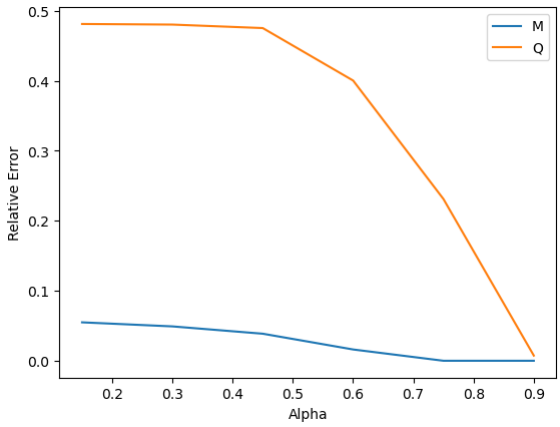
Table 3: Výsledky testování 1. fitness funkce pro různé hodnoty parametru α u podstady N.

	Parametr α					
	0.15	0.30	0.45	0.60	0.75	0.90
Poměr splněných klauzulí	0.939	0.959	0.977	0.993	1.000	1.000
Poměr splnění SAT	0.300	0.300	0.300	0.500	1.000	1.000
Relativní chyba	0.055	0.049	0.039	0.016	0.000	0.000
Hodnota Fitness	0.987	0.980	0.98	0.980	0.986	0.994

Table 4: Výsledky testování 1. fitness funkce pro různé hodnoty parametru α u podstady Q.

	Parametr α					
	0.15	0.30	0.45	0.60	0.75	0.90
Poměr splněných klauzulí	0.871	0.873	0.877	0.918	0.968	1.000
Poměr splnění SAT	0.000	0.000	0.000	0.000	0.100	1.000
Relativní chyba	0.481	0.480	0.475	0.400	0.231	0.007
Hodnota Fitness	0.981	0.961	0.942	0.930	0.935	0.969

Následující graf ukazuje, jaký vliv má parametr α na výslednou relativní chybu u obou podsad. Můžeme vidět, že čím větší α tím menší relativní chyba. Jinými slovy, čím více upřednostňuji splnitelnost formule nad maximalizaci vah, tím menší je relativní chyba.



2.2 Iterace 2

V této fázi jsem použil stejné parametry jako v předchozí fázi až na jinou fitness funkci, kde jsem se snažil ladit její parametry a zjistit, zda je tato nová fitness funkce lepší než ta v předchozí iteraci.

Table 5: Hyperparametry Genetického Algoritmu nastavené v iteraci 2

Parametr	**Hodnota**
Max generations	1000
Population size	300
Elitism	0.5
Mutation Probability	0.3
Crossover	Uniform
Fitness-beta	hledaná
Fitness-alpha	hledaná

Upravená fitness funkce je následující:

$$\text{fitness(chromosome)} = \alpha \cdot \frac{C}{\Sigma_C} + (1-\alpha) \cdot \frac{W}{\Sigma_W} - \beta \cdot \frac{\Sigma_C - C}{\Sigma_C}$$

Přidaná část je míněna jako penalizace za nesplněné klauzule. Míru penalizace je možné ovlivnit nastavením parametru β.

Table 6: Relativní chyba pro různé kombinace parametrů α, β u podsady N.

		Parametr α			
		0.60	0.70	0.80	0.90
Parametr β	0.10	0.014	0.000	0.000	0.000
	0.20	0.011	0.000	0.000	0.000
	0.30	0.008	0.000	0.000	0.000
	0.40	0.004	0.000	0.000	0.000

Table 7: Relativní chyba pro různé kombinace parametrů α, β u podsady Q.

		Parametr α			
		0.60	0.70	0.80	0.90
Parametr β	0.10	0.385	0269	0.141	0.009
	0.20	0.368	0.231	0.109	0.000
	0.30	0.309	0.193	0.087	0.000
	0.40	0.286	0.161	0.050	0.004

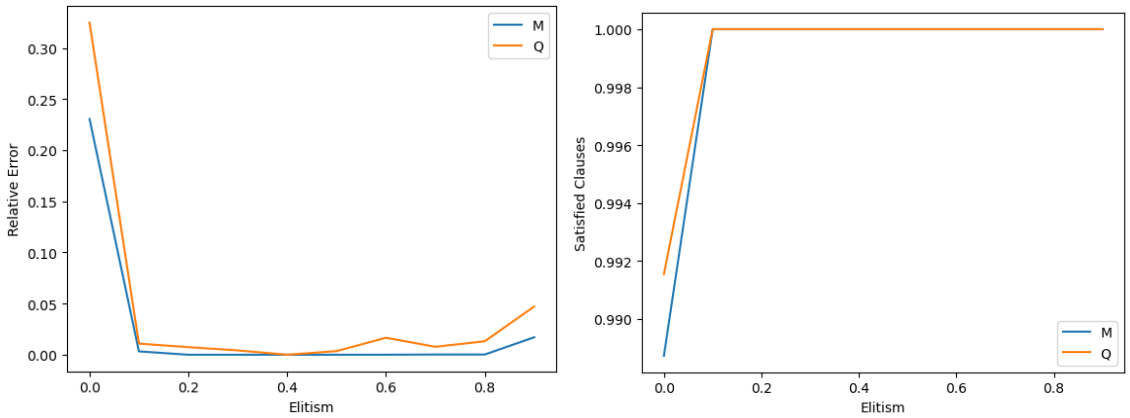
Vidíme, že relativní chyba u podsady Q pro $\alpha = 0.9, \beta = 0.3$ je dokonce 0. To je lepší výsledek, než relativní chyba 0.007 u minulé fitness funkce. Z toho důvodu vybírám tuto fitness funkci i pro další iterace a to právě s parametry $\alpha = 0.9, \beta = 0.3$.

2.3 Iterace 3

V této iteraci jsem zkoumal, jaký vliv má elitismus (počet nejlepší jedinců ponechaných do další generace bez jakýchkoliv změn) na relativní chybu a poměr splněných klauzulí. Měření jsem provedl na datasetu *wuf20-71* - podsada *M* a *Q*.

Table 8: Hyperparametry Genetického Algoritmu nastavené v iteraci 3

Parametr	**Hodnota**
Max generations	1000
Population size	300
Elitism	hledaná
Mutation Probability	0.3
Crossover	Uniform
Fitness-beta	0.9
Fitness-alpha	0.4



Z grafu můžeme vidět, že nejelpších hodnot algoritmus dosáhl při použití elitismu=0.4 (40% nejlepších jedinců se ponechá do další generace beze změn. Proto jsem pro testování v dalších iteracích zvolil tuto hodnotu tohoto hyperparametru. Testoval jsem to již na těžších instancích, konkrétně na datasetu *wuf36-157*.

2.4 Iterace 4

V této iteraci butu zkoumat optimální nastavení hyperparametru *mutation rate*. Porovnávám fixně nastavenou hodnotu s adaptivní strategií nastavování tohoto hyperparametru.

Table 9: Hyperparametry Genetického Algoritmu nastavené v iteraci 4

Parametr	**Hodnota**
Max generations	1000
Population size	300
Elitism	0.4
Mutation Probability	variable
Crossover	Uniform
Fitness-beta	0.9
Fitness-alpha	0.4

Moje navrhnutá lineární adaptivní stratigie pro nastavování pravděpodobnosti mutace:

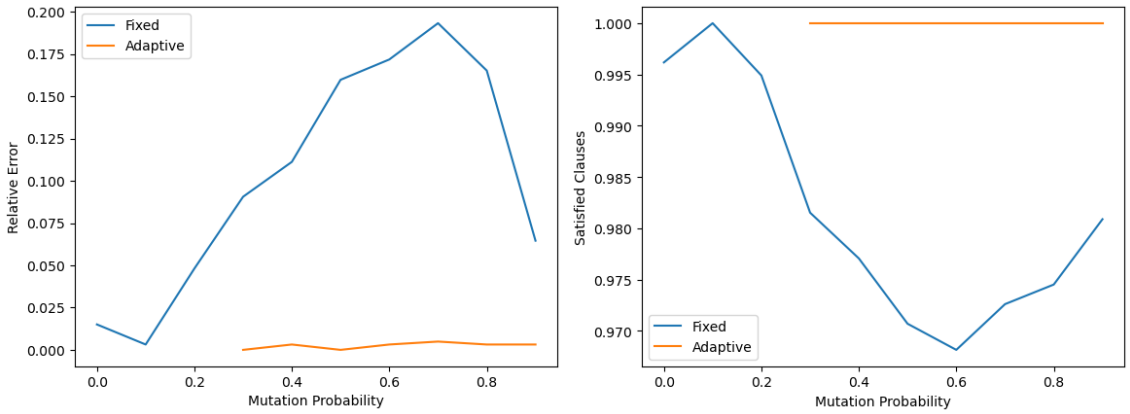
$$P_{mut}(g) = P_{mut}^{max} - g \cdot \frac{P_{mut}^{max} - P_{mut}^{min}}{G}$$

Kde:

- g - číslo aktuální generace
- G - číslo maximální generace
- P_{mut}^{max} - maximální pravděpodobnost mutace
- P_{mut}^{min} - minimální pravděpodobnost mutace (nastaveno na 0.0)

Následující graf porovnává fixní strategii a adaptivní. V případě adaptivní strategie hodnota "Mutation Probability" udává startovací pravděpodobnost mutace.

Figure 1: Relativní chyba pro fixní a adaptivní strategii pro pravděpodobnost mutace - podsada M.



Jelikož adaptivní strategie byla pro různé hodnoty startovací pravděpodobnosti mutace stabilnější, rozhodnul jsem se vybrat tuto strategii.

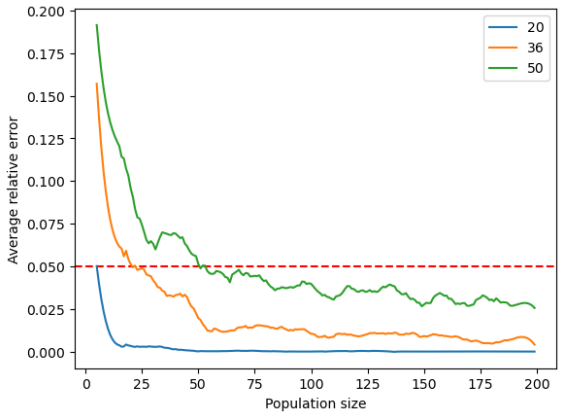
2.5 Iterace 5

V poslední páté iteraci jsem se snažil navrhnout funkci **pro počítání vhodné velikosti populace v závislosti na velikosti problému** (počet proměnných).

Table 10: Hyperparametry Genetického Algoritmu nastavené v iteraci 5

Parametr	**Hodnota**
Max generations	500
Population size	laděna
Elitism	0.4
Mutation method	Linear adaptive
Starting Mutation Probability	0.7
Crossover	Uniform
Fitness-beta	0.9
Fitness-alpha	0.4

Nejprve jsem zjistil průměrné relativní chybu pro sady *wuf20-71*, *wuf36-157* a *wuf50-218* (typ M) v rozmezí hodnot parametru *Population size* od 5 do 200. Jelikož křivky měly velké kratkodobé odchylky, vyhladil jsem je pomocí Savitzky-Golay filtru (používající konvoluci). Vyhlazené křivky jsou vidět na následujícím obrázku 2.5.



Dále jsem pro každou datovou sadu zjistil nejmenší hodnotu velikosti populace, aby relativní chyba byla menší než stanovený treshold. Zvolil jsem **treshold 0.05** jako kompromis mezi přesností a časovými náklady na výpočet. Dostal jsem následující hodnoty:

Počet proměnných	**Postačující velikost populace**
20	6
36	21
50	51

Jelikož v předchozí tabulce nevidím linearitu, rozhodnul jsem se zvolit polynomiální funkci pro počítání vhodné velikosti populace: $P = a \cdot V^b$, kde a, b jsou koeficienty, V je počet proměnných.

Tu jsem nejdříve zlogaritmoval, abych koeficienty mohl odhadnout pomocí lineární regrese (tedy jsem předtím musel zlogaritmovat i hodnoty vstupující do linerání regrese). Dostal jsem hodnoty $a = 0.005737$ a $b = 2.3115$. Tedy výsledná funkce pro výpočet vhodného počtu jedinců v populaci v závislosti na velikosti problému vypadala takto:

$$P = 0.005737 \cdot V^{2.3115}$$

Je však potřeba poznamenat, že jsem pro trénování použil pouze tři datové body. Tento model tedy nemusí být příliš robustní. Nicméně pro účely přibližné aproximace je dostatečný.

Následující tabulka ukazuje spočítaný vhodnou velikost populace pro různý počet proměnných problému:

Počet proměnných	**Velikost populace**
20	6
36	23
50	49
75	124
100	241

3 Black box fáze

3.1 Úvod

Tato fáze nasazení slouží k otestování funkčnosti heuristiky se stanovenými parametry. Experiment bude prováděn bez možnosti porozumění vnitřního fungování algoritmu a tím i parametrů heuristiky.

3.2 Metody a materiály

Použil jsem datasety *wuf36-157*, *wuf50-218*, *wuf75-325* a *wuf100-430*, pro každý možnosti M, N a Q. Dataset *wuf100-430* neobsahoval výsledky, takže jsem u něho nemohl počítat relativní chybu. Podobně jako u white box fáze jsem zvolil počet seed hodnot 10 a 100 instancí. Dohromady tedy pro každou datovou sadu proběhlo $10 \cdot 100 = 1000$ běhů. Hardware, na kterém byly experimenty provedeny, neumožňoval instanci spustit vícekrát v rozumném časovém úseku. Celkem tedy proběhlo $1000 \cdot 3 \cdot 4 = 12000$ běhů.

Použil jsem takové hodnoty hyperparametrů genetického algoritmu, které jsem zjistil ve white box fázi.

Parameter	**Value**
Max generations	500
Population size	$P = 0.005737 \cdot V^{2.3115}$
Elitism	0.4
Mutation method	Linear adaptive
Starting Mutation Probability	0.7
Crossover	Uniform
Fitness-beta	0.9
Fitness-alpha	0.4

3.3 Výsledky

Z tabulky výsledků 11 můžeme vidět, že navrhnutá heuristika si umí mnohem lépe poradit s podsadami M a N, zatímco u podsad Q, kde váhy vytvářejí (mírně) zavádějící úlohu, jsou všechny statistiky horší. Průměrná relativní chyba přes všechny datasety navrhnuté heuristiky je 2.7131. Nicméně u podsad M, N je průměrná relativní chyba 0.0358, což je již výrazně menší.

U všech datasetů je průměrný poměr splněných klauzulí větší než $\sim 96\%$.

Table 11: Výsledky testovaní v Black box fázi

Dataset	**Splněných klauzule**	**Splněných formule**	**Relativní chyba**
36-157-M	0.9964 ± 0.0048	0.5700	0.0255 ± 0.0356
36-157-N	0.9964 ± 0.0047	0.5600	0.0261 ± 0.0365
36-157-Q	0.9825 ± 0.0076	0.0000	12.2529 ± 81.1545
50-218-M	0.9949 ± 0.0049	0.3400	0.0293 ± 0.0286
50-218-N	0.9956 ± 0.0046	0.3900	0.0296 ± 0.0290
50-218-Q	0.9811 ± 0.0077	0.0000	8.3406 ± 37.6638
75-325-M	0.9875 ± 0.0050	0.0100	0.0514 ± 0.0358
75-325-N	0.9886 ± 0.0050	0.0300	0.0529 ± 0.0368
75-325-Q	0.9732 ± 0.0074	0.0000	3.6097 ± 4.8980
100-430-M	0.9821 ± 0.0047	0.0000	–
100-430-N	0.9823 ± 0.0048	0.0000	–
100-430-Q	0.9667 ± 0.0067	0.0000	–
Průměr (M,N)	$\mathbf{0.9905 \pm 0.0048}$	**0.2375**	$\mathbf{0.0358 \pm 0.0337}$
Průměr (vše)	$\mathbf{0.9856 \pm 0.0057}$	**0.1583**	$\mathbf{2.7131 \pm 2.7117}$

4 Závěr

Implementoval jsem a odladil heuristiku genetického algoritmu pro řešení problému maximální vážené splnitelnosti boolovské formule (MWSAT).

Nasazená heuristika byla testována na 1200 instancích a celkově proběhlo 12000 běhů. Celková relativní chyba pro lehčí sady (M, N) je 0.0358 a pro těžší (Q) je 2.7131. To hodnotím jako uspokojivý výsledek, ale určitě existují i lepší heuristiky, který by problém řešily s vyšší úspěšností.