# Řešení problému vážené splnitelnosti booleovské formule pokročilou iterativní metodou

## Specifikace problému

Je dána booleovská formule F o n proměnných, X=(x1, x2, ... , xn) v konjunktivní normální formě (CNF). Dále jsou dány celočíselné kladné váhy těchto n proměnných W=(w1, w2, ... , wn). Nalezněte přiřazení Y=(y1, y2, ... , yn) proměnných x1, x2, ... , xn, takové, že F(Y)=1 a součet S vah proměnných ohodnocených jedničkou je maximální.

Omezte se na vážený 3-SAT problém, kde každá klauzule má právě 3 proměnné. Složitost problému je stejná, implementace a klasifikace instancí jsou jednodušší.

## Rozbor možností řešení

### Hrubá síla

### Metoda větví a hranic

### Dynamické programování

### Simulované ochlazování

### Tabu Search

### Genetický algoritmus

## Popis genetického algoritmu

Genetický algoritmus (GA) patří do skupiny algoritmů inspirovaných evolučními procesy v přírodě. Vychází z Darwinovi evoluční teorie, která tvrdí, že v přírodě přežijí nejsilnější jedinci. Základním prvkem algoritmu je populace, která se skládá z jedinců, kde každý jedinec reprezentuje řešení problému. S každou novou generací se algoritmus snaží najít lepší řešení. Kvalita řešení je určena fitness funkcí (zdatnost jedince). GA je iterativní algoritmus, který se skládá ze čtyř primárních kroků.

### Selekce

Prvním krokem je výběr nejsilnějších jedinců, kteří se budou v dalším kroku křížit. Výběr jedinců je založen na hodnotě fitness funkce. Existuje několik strategií, jak selekci dělat.

#### Zkrácený výběr

Jednou ze strategií je vybrat *N* nejzdatnější jedinců z populace. Tato strategie se nazývá zkrácený výběr, její nevýhodou je, že pokud budeme vybírat stále nejsilnější jedince, tak populace může zdegenerovat. Tomuto jevu se říká silný selekční tlak. Selekční tlak vyjadřuje, s jakou pravděpodobností vybíráme nejlepší jedince.

#### Ruletový výběr

Ruletový výběr probíhá tak, že naneseme na ruletu jednotlivé fitness hodnoty jedinců, kde ti silnější mají větší pravděpodobnost, že budou vybráni. Vzniká zde nebezpečí degenerace populace, jelikož existuje přímá úměra mezi zdatností jedince a pravděpodobností výběru. Je tedy nutné nadržovat slabším.

#### Turnajový výběr

Náhodně se vybere *N* jedinců (velikost turnaje) a z nich nejlepší. Vybíraní se opakuje, dokud se nenaplní populace. Selekční tlak je řízen velikostí turnaje.

### Křížení

Po selekci následuje křížení vybraných jedinců. Po křížení dochází ke vzniku nové populace. Křížení lze provést různými způsoby. Nejobvyklejšími metodami jsou jednobodové, dvoubodové nebo permutované křížení. Je možné do algoritmu zavést více metod a pravděpodobnostně vybírat, které bude použito.

### Mutace

Při mutaci dochází k náhodné změně genetické informace. Mutace je jedním ze způsobů, jak řídit selekční tlak.

### Řízení populace

Řízení populace říká, jakým způsobem budeme vytvářet populaci po křížení. Existuje několik způsobů, jak po křížení vytvořit novou populaci. Například můžeme kompletně nahradit starou generaci novou generací, nebo můžeme zachovat určitý počet jedinců ze staré generace.

## Popis implementace genetického algoritmu

### Počáteční populace

Počáteční populaci vybírám náhodně. Náhodně vygeneruji bitové vektory, dokud nedosáhnu požadované velikosti populace. Velikost populace lze parametricky řídit. Zda věc bude v batohu zahrnuta je rozhodnuto s pravděpodobností 50 %.

### Selekce

Zvolil jsem selekci turnajového výběru. Nejprve náhodně vyberu *N* jedinců, a poté toho nejsilnějšího. Opakuji, dokud nenaplním populaci. Velikost turnaje lze parametricky řídit.

### Křížení

Algoritmus používá jednobodové křížení a dvoubodové křížení. Jaké křížení se použije je dáno pravděpodobnostním výběrem.

### Mutace

Každý gen jedince zmutuje s pravděpodobností, kterou lze parametricky řídit.

#### Řízení populace

Populace je řízena metodou elitismus. Parametricky lze určit počet nejlepších jedinců, kteří budou zachování do nové generace.

## Platforma a HW konfigurace

* CPU Intel Core i5-8350U
* RAM 32GB
* OS Windows 10 64-bit
* Platforma .NET Core a programovací jazyk C#

## Hledání optimální konfigurace parametrů

### Postup měření

Cílem měření je najít optimální konfiguraci parametrů GA. Toho lze dosáhnout experimentálním měřením a sledováním různých konfigurací algoritmu. Postup měření je takový, že se v každé iteraci zafixují jednotlivé parametry a vybere se jeden pro který se zkouší různé hodnoty. Pro každou hodnotu se změří průměrný výpočetní čas a relativní chyba. Po vyzkoušení všech parametrů algoritmu dostaneme optimální konfiguraci.

Výchozí konfigurace pro měření vychází z předchozího úkolu, kde byl algoritmus použit pro řešení problému batohu. Tato konfigurace vyšla jako optimální pro problém batohu o velikosti instance 35.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost populace | 100 |
| Počet generací | 180 |
| Velikost turnaje | 3 |
| Pravděpodobnost mutace | 0.07 |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 1 - výchozí konfigurace

### Instance

Pro hledání optimální konfigurace byly použity instance z připravených sad wuf20-78-M1 a wuf50-201-R1. Z každé sady bylo vybráno 50 instancí.

### Fitness funkce

Implementoval jsem dvě fitness funkce. První fitness funkce přiřazovala nesplnitelným konfiguracím hodnotu 0. Druhá funkce přiřazovala nesplnitelným konfiguracím zápornou zdatnost, která se rovnala počtu nesplněných klauzulí. První funkce nedokázala najít žádné řešení pro instance o velikosti 50 proměnných. Druhá funkce dokázala najít řešení pro tyto instance. Vybral jsem tedy druhou funkci, pomocí které lze říci, která nesplnitelná konfigurace je lepší.

### Počet generací

Počet generací určuje, kolik iterací algoritmus udělá. V ideálním případě by počet generací měl být takový, aby se algoritmus přiblížil k optimálnímu řešení a zároveň nebyl počet generací zbytečně velký a nezpomaloval běh algoritmu.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost populace | 100 |
| **Počet generací** | **20, 40, 60, 80, 100, 120, 140, 160 180, 200, 300, 400, 500, 600, 700, 800, 900, 1000** |
| Velikost turnaje | 3 |
| Pravděpodobnost mutace | 0.07 |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 2 – konfigurace

Obrázek 1 - závislost výpočetního času na počtu generací wuf20-78-M1

Obrázek - závislost výpočetního času na počtu generací wuf50-201-R1

Výpočetní čas roste lineárně se zvyšujícím se počtem generací.

Obrázek 3 - závislost relativní chyby na počtu generací

Relativní chyba klesá s přibývajícím počtem generací. Nicméně, na některých intervalech je vidět zhoršení relativní chyby. To může být dáno tím, že spoustu kroků algoritmu je randomizovaných a algoritmus může dávat při každém spuštění jiný výsledek.

Ustálení nastalo u generace o velikosti 500, kde relativní chyba je 3.2 % a výpočetní čas je 132 milisekund. Další zajímavá velikost generace je 180. Tato velikost generace má sice vyšší relativní chybu 4 %, ale mnohem lepší výpočetní čas 44 milisekund. V poměru kvalita/rychlost vychází nejlépe.

### Velikost populace

Velikost populace určuje, kolik řešení je uchováváno.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| **Velikost populace** | **20, 40, 50, 60, 70, 80, 90, 100, 120, 140, 150** |
| Počet generací | 180 |
| Velikost turnaje | 3 |
| Pravděpodobnost mutace | 0.07 |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 3 - konfigurace

Obrázek 4 - závislost výpočetního času na velikosti populace

Obrázek - závislost výpočetního času na velikosti populace

Výpočetní čas roste se zvětšující se populací.

Obrázek 6 - závislost relativní chyby na velikosti populace

Relativní chyba klesá se zvětšující se populací. Nejlepší relativní chyba vyšla pro velikost populace 100.

### Velikost turnaje

Velikost turnaje určuje z kolika jedinců se bude vybírat nejlepší jedinec. Pomocí velikosti turnaje lze řídit selekční tlak. Čím větší velikost turnaje, tím větší selekční tlak. Turnaj s jedním jedincem znamená žádný selekční tlak.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost populace | 100 |
| Počet generací | 180 |
| **Velikost turnaje** | **1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20** |
| Pravděpodobnost mutace | 0.07 |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 4 - konfigurace

Obrázek 7 - závislost výpočetního času na velikosti turnaje

Obrázek - závislost výpočetní času na velikosti turnaje

S větší velikostí turnaje roste lineárně výpočetní čas. To je očekáváné chování, platí zde totiž přímá úměra, algoritmus musí s narůstající velikostí turnaje vybírat více účastníků.

Obrázek 9 - závislost relativní chyby na velikosti turnaje

Nejhorší relativní chyba se objevila u velikosti turnaje 1 a 100. U velikosti turnaje 1 není žádný selekční tlak a následkem toho algoritmus nekonverguje k optimálnímu řešení. U velikosti turnaje 100 je do turnaje vybraná celá populace. Selekční tlak je tady velmi velký a dochází k degeneraci populace.

Ideální velikostí turnaje se zdá být hodnota 3. Relativní chyba je ze všech hodnot nejnižší 6.8 % a stejně tak výpočetní čas vychází nejlépe.

### Pravděpodobnost mutace

Pravděpodobnost mutace určuje, s jakou pravděpodobností dojde k mutaci genu. Pravděpodobnost mutace je další parametr, pomocí kterého lze řídit selekční tlak.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost populace | 100 |
| Počet generací | 180 |
| Velikost turnaje | 3 |
| **Pravděpodobnost mutace** | **0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09** |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 5 – konfigurace

Obrázek 10 - závislost relativní chyby na pravděpodobnosti mutace

Nejlépe možná trochu překvapivě vychází hodnota 0.08 neboli 8 %. To je poměrně velká pravděpodobnost.

Počet nejlepších jedinců z předchozí generace

Tento parametr určuje, kolik nejlepších jedinců z předchozí generace, bude zachováno do další generace. Jedná se o způsob, jakým řídit vývoj populace. Při této metodě řízení populace může docházet k degeneraci populace. Pokud nastavíme hodnotu na nulu bude se jednat o náhradu en bloc, při které nová generace nahradí starou kompletně.

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost populace | 100 |
| Počet generací | 180 |
| Velikost turnaje | 3 |
| Pravděpodobnost mutace | 0.07 |
| **Počet nejlepších jedinců z předchozí generace** | **1, 2, 3, 4, 5, 6, 7, 8, 9, 10** |

Tabulka 6 – konfigurace

Výpočetní čas roste se zvyšujícím se počtem jedinců z předchozí generace.

Obrázek 11 - závislost relativní chyby na počtu nejlepších jedinců z předchozí populace

Jako nejlepší možnost se ukázalo zachovávat čtyři nejlepší jedince z předchozí generace.

## Výsledná nejlepší konfigurace

|  |  |
| --- | --- |
| **Parametr** | **Hodnota** |
| Velikost instance | 35 |
| Velikost populace | 100 |
| Počet generací | 180, 500 |
| Velikost turnaje | 3 |
| Pravděpodobnost mutace | 0.7 |
| Počet nejlepších jedinců z předchozí generace | 1 |

Tabulka 7 - výsledná kofigurace

Do výsledné konfigurace jsem uvedl u počtu generací dvě hodnoty. Hodnota 180 nabízí dobrý poměr kvalita/výpočetní čas a hodnota 500 poskytuje nejkvalitnější řešení.

Velikost generace 180 má relativní chybu 6 % a řešení dokáže poskytnout za 54 milisekund. Velikost generace 500 má relativní chybu 3 % a výpočetní čas 153 milisekund.

## Závěr

Výpočetní čas roste s velikostí instance, velikostí populace a počtem generací lineárně. Genetický algoritmus je tedy použitelný i na velkých instancích.

Relativní chyba klesá s počtem generací do určitého bodu, kde se už řešení nezlepšuje. Stejně tak klesá relativní chyba i s velikostí populace. Parametr velikost turnaje byl důležitým prvkem při řízení selekčního tlaku. Optimální velikost turnaje se ukázala být hodnota 3.

Algoritmus se ukázal být citlivý na parametr pravděpodobnost mutace. Trochu překvapivě vyšla nejlépe vyšší hodnota 7 %.

Velmi zajímavým parametrem se ukázal být počet nejlepších jedinců z přechozí generace. Při uchovávání nulového počtu nejlepších jedinců z předchozí generace byla relativní chyba největší ze všech testovaných hodnot.