1 Zadání úlohy

Celé zadání se skládá ze tří částí:

1. Zvolte si heuristiku, kterou budete řešit problém vážené splnitelnosti booleovské formule.
2. Tuto heuristiku použijte pro řešení problému batohu.
3. Hlavním cílem domácí práce je seznámit se s danou heuristikou, zejména se způsobem, jakým se nastavují její parametry a modifikace.

2 Rozbor možných variant řešení

Pro řešení problému batohu mám na výběr z následujících heuristik:

1. simulované ochlazování,
2. genetický algoritmus,
3. tabu vyhledávání.

3 Popis postupu řešení

Z výše zmíněných variant jsem si vybral genetický algoritmus. Nejdříve jsem si zvolil jakým způsobem chci implementovat jednotlivé částí genetického algoritmu. Poté jsem algoritmus naimplementoval a nakonec jsem provedl systematickou sadu měření, pomocí kterých jsem odhalil, za jakých vstupních parametrů jsou výstupy z mého genetického algoritmu nejvýhodnější (poměr chyba/čas).

3.1 Algoritmus

Genetický algoritmus je založen na principu populace, kde každá člen představuje nějakou konfiguraci batohu. V průběhu generací se tato populace mění, ať už křížením nebo postupnou mutací jejich členů.

V první řadě se náhodně vygeneruje vstupní populace. Pravděpodobnost, že člen vstupní populace má i-tý gen nastavený na 1 je 50 %. Následně proběhne stanovený počet generací.

V každé generaci se nejdříve vyberou elitní jedinci (jejich počet je dán parametrem **ELITISM**), kteří okamžitě postupují do další generace. Ze zbylých členů populace se selekcí vyberou jedinci. Každý z těchto jedinců se rozhodne zdali se chce podílet na křížení (pravděpodobnost, že se bude chtít podílet na křížení je dána parametrem **CROSSOVER_PROBABILITY**). Křížením se se vytvoří noví jedinci. Tito noví jedinci společně s členy populace, kteří prošli selekcí, ale již se nepodíleli na křížení musí projít částečnou mutací.

Jakmile doběhne poslední generace, metoda konverguje k řešení, které je rovna nejzdatnějšímu členu populace.

3.2 Stanovení parametrů a modifikací

3.2.1 Selekce

Selekce probíhá formou turnaje, kdy je z populace náhodně vybrán určitý (parametr **TOURNAMENT_SIZE**) počet členů, ze kterých se vybere ten nejzdatnější. Tento postup se opakuje dokud nezískáme stejný počet členů, jako byl počet členů populace bez elitních jedinců.

3.2.2 Křížení

Ke každému jedinci, který se rozhodl, že bude součástí procesu křížení, se náhodně vybere jiný jedinec. Vygeneruje se uniformní list, který nám řekne zdali má potomek sdílet i-tý gen po 1. (hodnota 1) nebo 2. (hodnota 0) rodiči. Výstupem jsou všichni potomci společně s jedinci, kterým nebyl nalezen partner pro křížení.

3.2.3 Mutace

Abychom neuvízli v lokálním minimu, používáme binární mutaci. Každý gen jedince, který prochází mutací má určitou pravděpodobnost (**MUTATION_PROBABILITY**) za které se daný gen změní.

3.3 Určování vstupních parametrů

Aby algoritmus fungovat co možná nejlépe je zapotřebí správně nastavit jeho vstupní parametry (například selekční tlak apod.). Rozhodl jsem se systematicky prověřit všechny možné parametry. Pevně jsem stanovil všechny parametry až na jeden a ten jsem postupně měnil a sledoval jak se mění rychlost výpočtu a relativní chyba řešení.

4 Kostra algotritmu

Program přijímá tyto vstupní paramentry:

- TYPE – typ algoritmu (nyní ga)
- COUNT_TIME – má program vypočítat dobu běhu algoritmu
- INTERATIONS – v případě, že má program vypočítat dobu běhu algoritmu, kolikrát má daný algoritmus spustit a hodnoty zprůměrovat
- GENERATION – kolika generacemi populace projde
- POPULATION – velikost populace
- TOURNAMENT_SIZE – kolik jedinců se bude spolu utkávat v turnaji
- CROSSOVER_PROBABILITY – jaká je pravděpodobnost, že bude jedinec křížen

- MUTATION_PROBABILITY – jaká je pravděpodobnost, že se daný gen jedince zmení
- ELITISM – kolik bude jedinců, kteří bez selekce a mutace postoupí rovnou do další generace

4.1 Implementace pravděpodobnosti

Metoda nextDouble() třídy Random generuje náhodně číslo v rozmezí od 0 do 1. Máme tedy desetinné číslo udávající pravděpodobnost, se kterou se něco stane, stačí zjistit zdali náhodně vygenerované číslo je menší rovno naší pravděpodobnosti. Pokud ano, tak jev nastal.

4.2 Výpočet zdatnosti

Zdatnost se počítá ve chvíli vytváření nového člena populace nebo jeho mutací. Při řešení problému batohu je zdatností cena položek v batohu. Jestliže se konfigurace nevejde do batohu je zdatnost členu nulová.

4.3 Výpočet relativní chyby

Pro měření relativní chyby jsem vytvořil separátní modul, který má na vstupu dva parametry:

1. soubor s naměřenými výsledky,
2. soubor se správnými výsledky.

Algoritmus přečte oba soubory a vytvoří pole relativních chyb ($\frac{opt(n)-apx(n)}{opt(n)}$) veškerých instancí obsažených v souboru. Následně vypočítá průměrnou relativní chybu, maximální relativní chybu a tyto hodnoty zapíše do souboru. Pro spouštění modulu se všemi soubory jsem vytvořil skript.

4.4 Měření času

Abych co možná nejlépe změřil čas, který algoritmus stráví na CPU, použil jsem java třídu ThreadMXBean. Pokud chci změřit čas, dám modulu na vstupu parametr kolikrát chci daný algoritmus zopakovat. Po každé iteraci změřený čas zapíšu do pole a po skončení všech iterací hodnoty v poli zprůměruji. Výsledek zapíši do souboru.

5 Naměřené výsledky

Pro implementaci jsem využil programovací jazyk Java. Testy běžely na operačním systému macOS.

Pro získání optimálního řešení jsem spouštěl stejný program akorát s jiným parametrem, který spustí algoritmus dynamického programovaní při dekompozici podle podle váhy. Vstupní generované instance byly velikosti 40.

Pomocí skriptu jsem systematicky spouštěl algoritmus se statickými hodnotami všech parametrů až na jeden, který se postupně měnil. Po skončení jsem vybral nejlepší hodnotu tohoto parametru a spustil jsem algoritmus nyní s jiným dynamickým parametrem. Není-li uvedeno jinak hodnoty parametrů odpovídaly hodnotám v tabulce 6

5.1 Počet generací

Se zvyšujícím se počtem generací relativní chyba klesá nepatrně 1, zato doba běhu výpočtu razantně roste. Za přijatelnou hodnotu jsem zvolil 90.

5.2 Velikost populace

Čím více členů populace má, tím je chyba menší a zároveň doba běhu větší 2. Za přijatelnou hodnotu jsem zvolil 4 (4x větší populace něž je velikost instance. Například pokud je 40 věcí v batohu, pak je populace velikosti 160).

5.3 Počet jedinců v turnaji

Se zvyšujícím se počtem jedinců v turnaji rostě relativní chyba společně s dobou běhu algoritmu 3. Čím je větší počet jedinců, kteří se spolu utkávají v turnají (vysoký selekční tlak), tím populace více degeneruje. Za přijatelnou hodnotu jsem zvolil 5.

5.4 Pravděpodobnost křížení jedince

Čím je větší pravděpodobnost, že se jedinec zkříží, tím roste doba běhu a klesá relativní chyba 4. Za přijatelnou hodnotu jsem zvolil 0.8 (80 procentní šanci na křižení).

5.5 Pravděpodobnost mutace genu jedince

Relativní chyba klesá zhruba do 0.6 a poté znovu roste 5. Doba běhu není ovlivněna tímto parametrem. Tomu by mohlo býti jinak pokud bych použil odchylku dvou po sobě jdoucích generacích pro ukončení algoritmu. Za přijatelnou hodnotu jsem zvolil 0.6 (60 procentní šanci na mutaci genu jedince).

5.6 Počet elitních jedinců v populaci

Výsledky měření jsou zrádné. Chvilkami relativní chyba 1 a doba běhu 2 klesá a chvilkami roste. Za přijatelnou hodnotu pro populaci o velikosti 160 jsem zvolil 16 elitních jedinců. Musím si dát pozor na degeneraci populace při vysokých hodnotách.

6 Závěr

Nastavení genetického algoritmu je při řešení určitého problému zásadní. Záleží na velikosti a typu problému. Zjistil jsem, že optimálním nastavením pro řešení problému batohu je 6.

7 Přílohy

Počet generací	Relativní chyba	Doba běhu
30	1,29 %	1,10511 s
60	1,05 %	2,10597 s
90	1,05 %	2,8826 s
120	1,05 %	3,71902 s
150	0,91 %	4,50071 s
180	0,91 %	5,34401 s

Table 1: Závislost relativní chyby a doby běhu na počtu generací

Velikost populace	Relativní chyba	Doba běhu
1	9,85 %	0,70186 s
2	3,61 %	1,43412 s
3	1,71 %	2,16101 s
4	0,99 %	2,81358 s
5	0,69 %	3,47951 s
6	0,42 %	4,17196 s

Table 2: Závislost relativní chyby a doby běhu na velikosti populace

Počet jedinců v turnaji	Relativní chyba	Doba běhu
5	0,99 %	3,04827 s
10	2,54 %	3,23669 s
15	5,70 %	3,28249 s
20	7,54 %	3,75568 s
25	9,56 %	3,80405 s
30	10,52 %	4,46779 s

Table 3: Závislost relativní chyby a doby běhu na počtu jedinců v turnaji

Pravděpodobnost křížení jedince	Relativní chyba	Doba běhu
20 %	7,14 %	2,61107 s
40 %	3,42 %	2,76676 s
50 %	2,31 %	2,80032 s
60 %	1,85 %	2,80311 s
80 %	0,94 %	2,91062 s
90 %	0,79 %	2,99595 s

Table 4: Závislost relativní chyby a doby běhu na pravděpodobnosti křížení jedince

Pravděpodobnost mutace jedince	Relativní chyba	Doba běhu
20 %	1,46 %	2,90644 s
40 %	1,15 %	2,83408 s
50 %	1,07 %	2,91802 s
60 %	0,87 %	2,89247 s
80 %	1,11 %	2,93014 s
90 %	1,21 %	2,97302 s

Table 5: Závislost relativní chyby a doby běhu na pravděpodobnosti mutace jedince

Vstupní parametr	Hodnota
Počet generací	90
Velikost populace	$4 * velikost\_instance$
Počet jedinců v turnaji	5
Pravděpodobnost křížení jedince	0.8
Pravděpodobnost mutace jedince	0.6
Počet elitních jedinců v populaci	16

Table 6: Přijatelné hodnoty vstupních parametrů

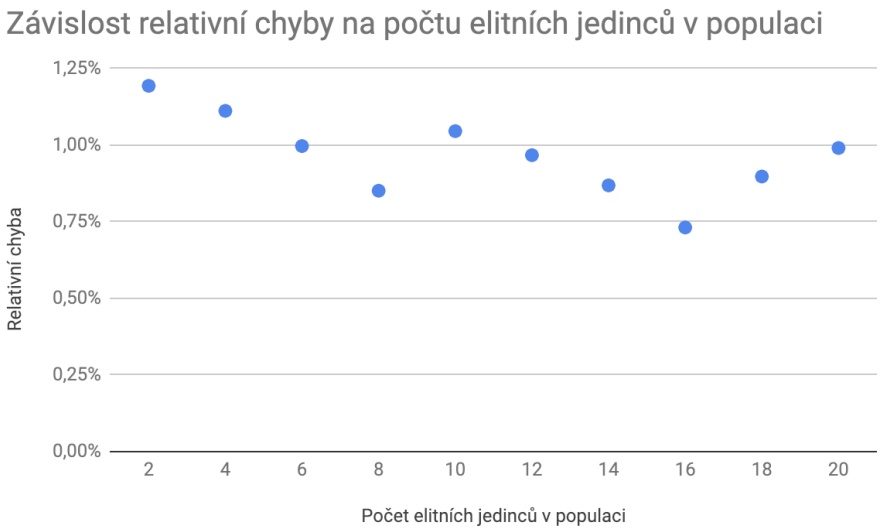

Figure 1: Závislost relativní chyby na počtu elitních jedinců v populaci

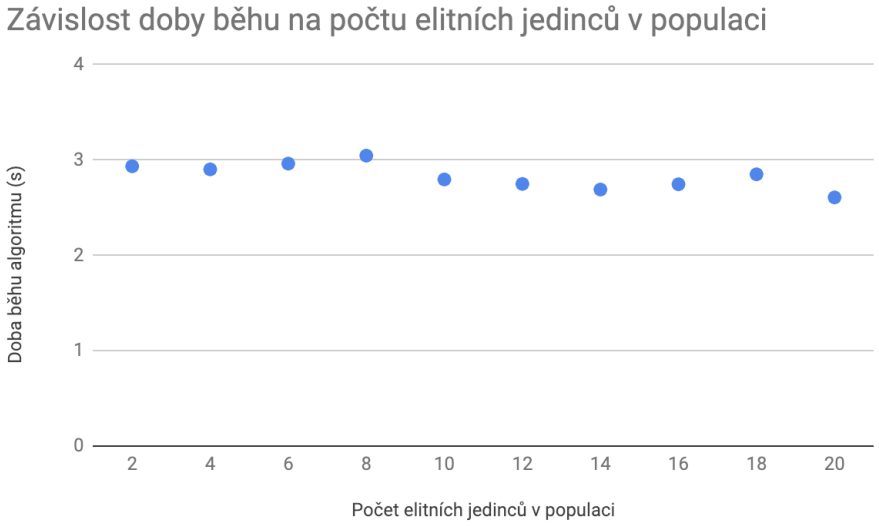

Figure 2: Závislost doby běhu na počtu elitních jedinců v populaci