Jan Bienias, 238201

PD-1-genalg – sprawozdanie
Wybrany problem: 3-SAT

1. Wprowadzenie i omówienie

Chromosom 3-SAT
Chromosom to wektor zero-jedynkowy, w którym przechowywana jest wartość logiczna dla każdej unikalnej zmiennej danej formuły 3-SAT. Element *i* w chromosomie odpowiada wartości zmiennej x_i.

Funkcja fitness 3-SAT

```
10  var_location_array <- c()
11
12  fitness3SAT <- function(chromosome){
13    counter <- 0
14    for(i in seq(1, length(var_location_array), by = 3)) {
15      trio <- c(0,0,0)
16      for(j in 0:2) {
17        if (var_location_array[i+j] < 0) {
18          trio[j+1] <- negate(chromosome[abs(var_location_array[i+j])])
19        } else {
20          trio[j+1] <- chromosome[abs(var_location_array[i+j])]
21        }
22      }
23      if(1 %in% trio) {
24        counter <- counter+1
25      }
26    }
27    return(-counter)
28  }
```

`(linia 10)` Funkcja fitness korzysta ze zmiennej globalnej, w której przechowywana jest formuła 3SAT w postaci wektora liczbowego (całkowitego). Jako jedyny argument przyjmuje chromosom.
`(linia 14)` Pętla `for` iteruje po klauzulach formuły (co trzy elementy), tworząc tymczasowy pusty wektor trójelementowy.
`(linia 16)` Druga, zagnieżdżona pętla `for` iteruje przez trzy elementy kolejnej klauzuli.
`(linia 17)` Najpierw zostaje sprawdzony znak negacji przy elemencie, a następnie w zależności od jego istnienia wykonuje się jedna z dwóch akcji:
brak znaku negacji → do wektora tymczasowego zapisujemy wynikającą wartość logiczną danego elementu, która została wyczytana z chromosomu z miejsca odpowiadającego wartości rozpatrywanego elementu;
znak istnieje → podobnie jak wyżej, jendak do wektora wpisujemy wartość zanegowaną.
`(linia 23)` Po przepisaniu elementów klauzuli sprawdzony jest warunek istnienia co najmniej jednej wartości '1' w wektorze tymczasowym. Jeśli tak, inkrementujemy wynik o 1.
`(linia 27)` Po przeiterowaniu przez wszystkie elementy formuły, funkcja zwraca ilość spełnialnych klauzul dla danego chromosomu.

Inny pomysł na funkcję fitness
Nie zwracać liczby spełnionych klauzul, a procent spełnionych klauzul.

Przykład działania – obrazowanie na znanym chromosomie
Formuła: (¬x3 ∨ x2 ∨ ¬x5) ∧ (x1 ∨ ¬x4 ∨ ¬x3) która na potrzeby programu jest zakodowana w formie: -3 2 -5 1 -4 -3

Chromosom:

Indeks	1	2	3	4	5
Gen	0	1	1	0	1

Działanie:

(linia 14) Pętla for przyjmuje pierwszą klauzulę w formie części wektora liczbowego: -3 2 5
(linia 16) Druga, zagnieżdżona pętla for iteruje przez każdy z elementów; badanie elementu -3
(linia 17) -3 jest liczbą ujemną, więc
(linia 18) Do wektora tymczasowego wpisujemy zanegowaną wartość logiczną widniejącą pod indeksem 3 w chromosomie, czyli 0.
Aktualny stan wektora to [0, 0, 0]
(linia 16) Badanie elementu 2
(linia 17) 2 jest liczbą dodatnią, więc
(linia 18) Do wektora tymczasowego wpisujemy wartość logiczną widniejącą pod indeksem 2 w chromosomie, czyli 1.
Aktualny stan wektora to [0, 1, 0]
(linia 16) Badanie elementu 5
(linia 17) 5 jest liczbą dodatnią, więc
(linia 18) Do wektora tymczasowego wpisujemy wartość logiczną widniejącą pod indeksem 5 w chromosomie, czyli 1.
Aktualny stan wektora to [0, 1, 1]
(linia 23) Czy klauzula jest spełnialna? TAK, +1 punkt

(linia 14) Pętla for przyjmuje pierwszą klauzulę w formie części wektora liczbowego: 1 -4 -3
(linia 16) Badanie elementu 1
(linia 17) 1 jest liczbą dodatnią, więc
(linia 18) Do wektora tymczasowego wpisujemy wartość logiczną widniejącą pod indeksem 1 w chromosomie, czyli 0.
Aktualny stan wektora to [0, 0, 0]
(linia 16) Badanie elementu -4
(linia 17) -4 jest liczbą ujemną, więc
(linia 18) Do wektora tymczasowego wpisujemy zanegowaną wartość logiczną widniejącą pod indeksem 4 w chromosomie, czyli 1.
Aktualny stan wektora to [0, 1, 0]
(linia 16) Badanie elementu -3
(linia 17) -3 jest liczbą ujemną, więc
(linia 18) Do wektora tymczasowego wpisujemy zanegowaną wartość logiczną widniejącą pod indeksem 3 w chromosomie, czyli 0.
Aktualny stan wektora to [0, 1, 0]
(linia 23) Czy klauzula jest spełnialna? TAK, +1 punkt

(linia 27) Funkcja zwraca ilość punktów: 2.

2. Ocena szybkości i efektywności działania algorytmu genetycznego w zależności od jego parametrów

Bazą do tej części zadania był zestaw 91 klauzul dostępnych pod adresem https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html w paczce *uf20-91*. Został on wpuszczony do funkcji, która za pomocą trzech pętli for zmieniała (iterując) kolejne parametry algorytmu: szansę mutacji, wielkość populacji oraz elityzm. W sumie przeanalizowano ponad 800 zestawów danych.

Najlepszy wynik pod względem czasu – 0.461s osiągnięto dla parametrów:

- szansa na mutację: 0%
- rozmiar populacji: 10
- elityzm: true
- ilość spełnionych klauzul: 85/91

Szczegółowe wyniki dostępne są w pliku *data_czasyRosnaco.csv*

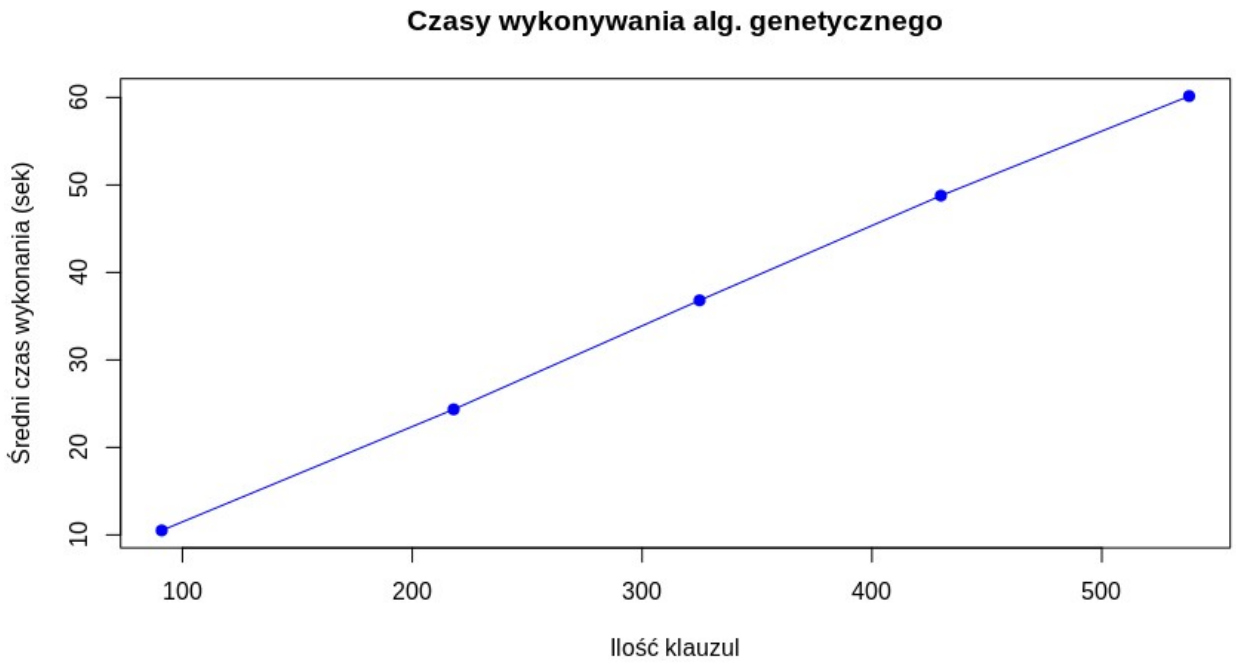
Najlepszy wynik pod względem efektywności – 91/91 osiągnięto dla 50 zestawów parametrów.
Oto 10 z nich, dodatkowo posortowanych względem czasu wykonania:

Wynik funkcji Fitness	Czas	Szansa Na mutację	Wielkość Populacji	Elityzm
91	0.486999	0.05	10	1
91	0.503999	0.25	10	1
91	0.512999	0.15	10	1
91	0.512999	0.35	10	1
91	0.528999	0.2	10	1
91	0.532000	0.55	10	1
91	0.543999	0.05	10	0
91	0.561999	0.2	10	0
91	1.025000	0.05	20	1
91	1.029999	0.15	20	1

Szczegółowe wyniki dostępne są w pliku *data_wynikiFitnessMalejaco.csv*

3. Ocena czasu działania algorytmu względem rozmiaru problemu

Bazą do tej części zadania były zestawy kolejno 91, 218, 325, 430, 538 klauzul dostępnych pod adresem https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html w paczkach kolejno *uf20-91*, *uf50-218*, *uf75-325*, *uf100-430*, *uf125-538*.



Czas wykonywania algorytmu rośnie liniowo. Możliwe, że czas wykonywania algorytmu w ogólności rośnie liniowo, a jeśli występują odchylenia, to są spowodowane dużą ilością unikalnych zmiennych co zmienia wielkość populacji, która wpływa w dużej mierze na czas działania algorytmu z uwagi na jego budowę.