# Inteligencja Obliczeniowa

## Praca domowa nr 1 – Algorytm genetyczny – 3-SAT

Krzysztof Kulewski, 238149, grupa 1, 02.11.2018

## Opis badania

Zadanie polegało na rozwiązaniu problemu 3-SAT za pomocą algorytmu genetycznego, a następnie zbadaniu, jak modyfikacja jego parametrów wpływa na otrzymywane wyniki i czas działania.

Problem spełnialności to zagadnienie rachunku zdań, określające, czy dla danej formuły logicznej istnieje takie podstawienie, żeby formuła była prawdziwa. 3-SAT jest problemem NP-zupełnym, co uzasadnia zastosowanie metod heurystycznych, takich jak algorytmy genetyczne.

Do testów użyto danych benchmarkowych - formuł zakodowanych w formacie DIMACS CNF.

## Format danych

Specyfikacja formatu DIMACS CNF definiuje 3 linie komentarza, następnie linię zawierającą nazwę problemu, liczbę zmiennych atomowych oraz klauzul. Każdy kolejny wiersz składa się z trzech liczb reprezentujących zmienne, a każda z nich może być poprzedzona znakiem ‘-‘, który oznacza negację danej zmiennej. Wiersz kończy się znakiem ‘0’.

Przykładowa formuła:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee \neg x_5 \vee \neg x_1) \wedge (x_3 \vee x_4 \vee \neg x_2)$$

Sposób zakodowania:

```
c
c SAT instance in DIMACS CNF input format.
c
p cnf 5 3
-1 2 3
4 -5 -1
3 4 -2
```

Tak zakodowane dane zostają wprowadzone do listy wektorów, gdzie każdy wektor reprezentuje pojedynczą klauzulę, a jego elementy to numery zmiennych.

## Struktura chromosomu

Chromosom jest ciągiem bitów, w którym bit na danej pozycji określa podstawienie zmiennej o danym numerze: prawda (T) lub fałsz (F). Został on zamodelowany za pomocą wektora jedynek i zer, a więc (1, 0, 1, 1, 0) to podstawienie $x_1$ = T, $x_2$ = F, $x_3$ = T, $x_4$ = T, $x_5$ = F.

## Funkcja fitness

Zadaniem funkcji fitness jest ocena pojedynczego chromosomu, tj. określenie jak trafne było podstawienie dla danej instancji problemu 3-SAT.
Wynik funkcji fitness to liczba klauzul, które nie zostały spełnione w danym podstawieniu. Najwyższą oceną jest więc liczba 0, która oznacza ich brak; najniższą – całkowita ilość klauzul w danej instancji, czyli przypadek, gdy żadna z nich nie została spełniona.

Warto zauważyć, że dla wybranego problemu ciężko jest zdefiniować inną funkcję fitness, gdyż jedyny element podlegający ocenie to ilość spełnionych formuł.

Kod funkcji fitness wraz z wyjaśnieniem działania:

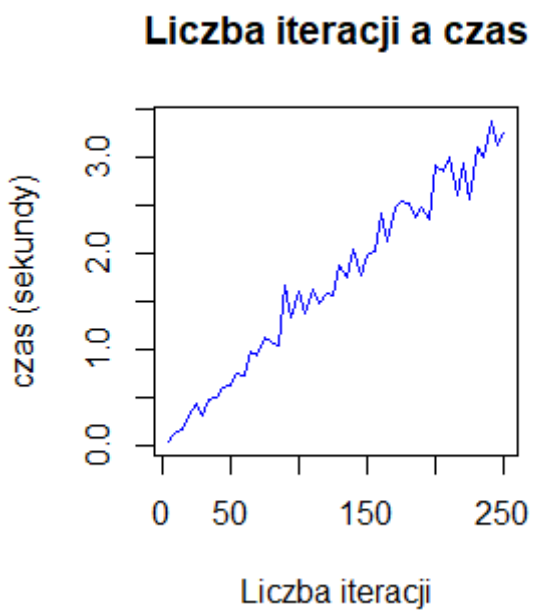
```
# argumenty to sparsowana formuła (lista wektorów) i chromosom (wektor 1 i 0)
CnfFitness = function(cnf, chromosome)
{
    clausesToSatisfy = length(cnf) # określamy całkowitą ilość klauzul do spełnienia

    for (clauseNumber in 1:length(cnf)) # iterujemy po wszystkich klauzulach
    {
        clauseSatisfied = FALSE # początkowo klauzula nie jest spełniona
        clause = cnf[[clauseNumber]] # pobieramy wektor reprezentujący daną klauzulę

        # iterujemy po długości wektora, czyli klauzuli: dla 3-SAT to zawsze 3
        for (variableNumber in 1:length(clause))
        {
            variable = clause[variableNumber] # pobieramy numer zmiennej na danej pozycji
            if (variable > 0) # numer dodatni, czyli zmienna bez negacji
            {
                # jeśli zmienna ma wartość 1 (T) – klauzula zostaje spełniona
                if (chromosome[variable] == 1) { clauseSatisfied = TRUE }
            }
            else # przypadek, gdy przed liczbą stoi minus – negacja zmiennej
            {
                # jeśli zanegowana zmienna ma wartość 0 (F) – klauzula spełniona
                if (chromosome[-variable] == 0) { clauseSatisfied = TRUE }
            }
        }

        if (clauseSatisfied == TRUE)
        {
            # klauzula została spełniona, więc redukujemy liczbę pozostałych
            clausesToSatisfy = clausesToSatisfy - 1
        }
    }

    return (clausesToSatisfy) # zwraca liczbę klauzul, które nie zostały spełnione
}
```

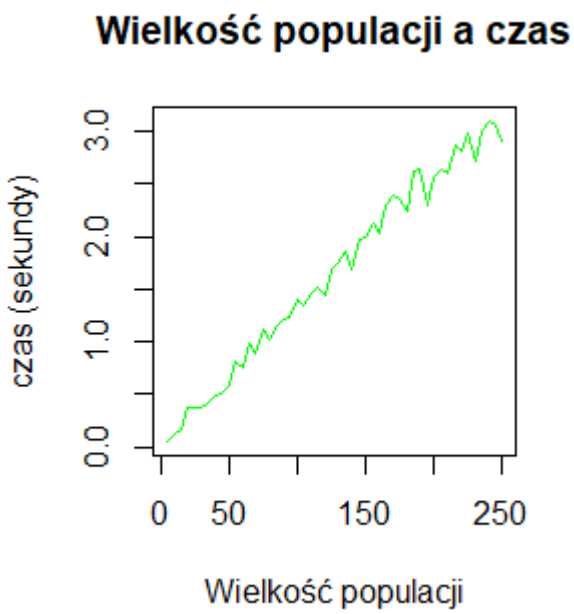
## Badanie 1 – wpływ wzrostu liczby iteracji i populacji na czas obliczeń

W pierwszym badaniu zmierzono jak zwiększanie liczby iteracji bądź wielkości populacji wpływa na czas obliczeń. Testowana instancja 3-SAT składała się ze:
- 100 zmiennych,
- 403 klauzul.

Parametry algorytmu:
- włączony elityzm,
- szansa mutacji: 2%.
- w przypadku 1., gdy zmienną była liczba iteracji (od 5 do 250), wielkość populacji wynosiła 25,
- w przypadku 2., gdy zmienną była wielkość populacji (od 5 do 250), liczba iteracji wynosiła 25.

Dla każdej wartości zmiennych przeprowadzono 10 prób, a uśredniony wynik naniesiono na wykres.



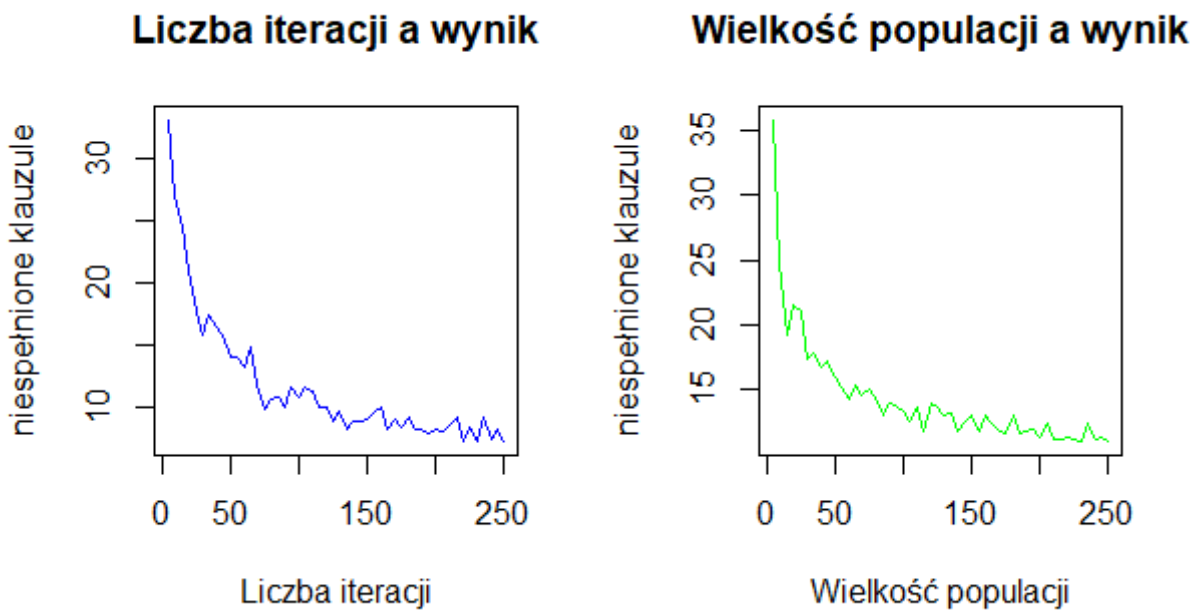

Jak widać na powyższych wykresach, czas obliczeń rośnie proporcjonalnie do liczby iteracji lub wielkości populacji.

Warto odnotować również fakt, iż wartości w punktach pomiarowych są bardzo zbliżone, tj. zwiększanie liczby iteracji i populacji w podobny sposób wpływa na czas obliczeń.

## Badanie 2 – wpływ wzrostu liczby iteracji i populacji na wynik

W drugim badaniu dokonano pomiaru wpływu liczby iteracji lub wielkości populacji na otrzymany wynik. Przypomnijmy, że im niższa wartość, tym lepiej – oznacza ona, że spełniono więcej klauzul.

Do badań użyto tej samej instancji problemu 3-SAT co w badaniu 1., identyczna jest również konfiguracja samego algorytmu genetycznego. Podobnie jak wcześniej, wyliczono średnią z 10 prób dla każdej wartości zmiennych.




Na pierwszy rzut oka – podobnie jak w badaniu pierwszym – wykresy są do siebie bardzo podobne. Podczas zwiększania liczby iteracji lub wielkości populacji od 1 do 50, możemy odnotować znaczną poprawę uzyskanych wyników. Efekt ten następnie spowalnia, a uzyskany wynik powoli zbliża się do asymptoty poziomej, wraz z dalszym wzrostem wartości zmiennych.

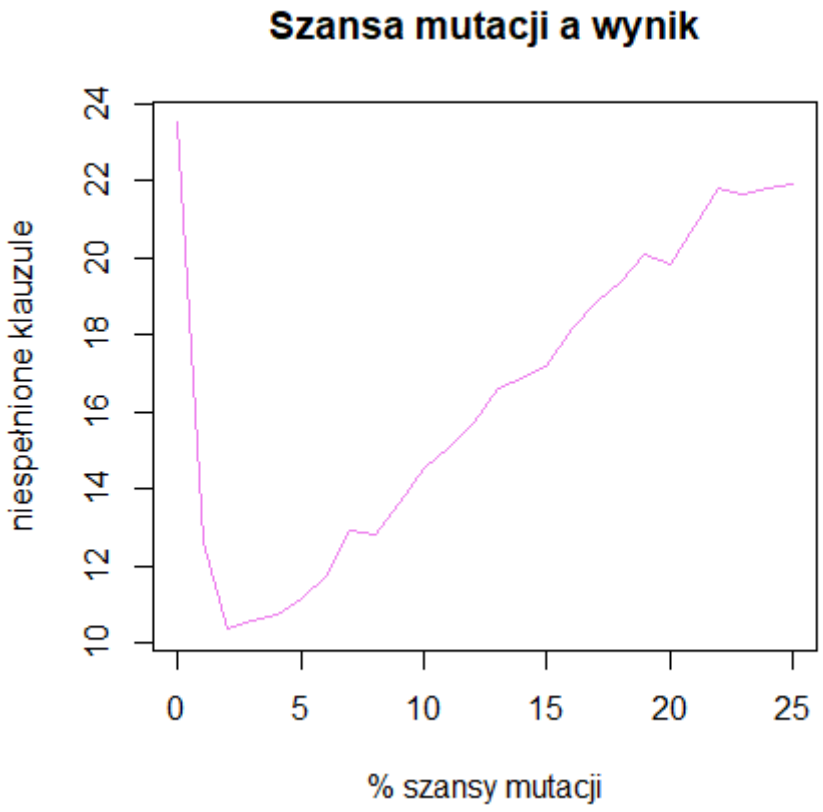
Zasadniczą różnicą jest jednak to, iż po 250 iteracjach, spełnione zostało średnio 94% klauzul, natomiast gdy to populacja jest zwiększana (również do 250), spełniono ich 88%.

Możemy więc wyciągnąć wniosek, że przy zastosowanej kombinacji parametrów, zwiększanie liczby iteracji jest skuteczniejsze.

## Badanie 3 – wpływ wzrostu szansy mutacji na wynik

Trzecie badanie to pomiar tego, jak  szansa mutacji w chromosomie wpływa na uzyskany wynik. Użyto tej samej instancji problemu 3-SAT co w badaniu 1. i 2., natomiast parametry algorytmu genetycznego były następujące:
- włączony elityzm
- szansa mutacji: od 0 do 25%
- wielkość populacji: 50
- liczba iteracji: 50

Dla każdej wartości szansy mutacji przeprowadzono 10 prób i wyliczono średnią wartość.




Wykres bardzo wyraźnie obrazuje fakt, iż istnieje pewien zakres „szansy mutacji", przy którym uzyskane wyniki są ponadprzeciętnie dobre. Dla zastosowanej kombinacji parametrów algorytmu, jest to zakres 2 – 4%.

Możemy zauważyć również, że dalszy wzrost szansy na mutację jest niekorzystny, tak samo jak jej całkowity brak.

Po wielu iteracjach, nasza populacja może stać się homogeniczna i zbiegać do jednego rozwiązania, wykluczając inne, potencjalnie lepsze. Mutacje wprowadzają element losowości, który może być w takim przypadku zbawienny.
Zbyt wysoka wartość sprawia natomiast, że algorytm staje się nadmiernie losowy, pogarszając wynik.

## Badanie 4 – wpływ wielkości instancji problemu na czas obliczeń

W czwartym badaniu pod lupę wzięto wpływ wielkości instancji (tj. liczby zmiennych) problemu 3-SAT na czas obliczeń. Zastosowane parametry algorytmu genetycznego:
- włączony elityzm
- szansa mutacji: 2%
- wielkość populacji: 50
- liczba iteracji: 50


Wykres pokazuje, że czas obliczeń jest proporcjonalny do liczby zmiennych w danej instancji problemu 3-SAT.

## Badanie 5 – wpływ elityzmu na wynik

Ostatnim badanym zagadnieniem był wpływ elityzmu na uzyskany wynik. Przy parametrach analogicznych do badania trzeciego, oraz tej samej instancji problemu, uzyskano następujące wyniki.

| Elityzm | Wynik (spełnione klauzule) |
|---|---|
| **wyłączony** | 88.35% |
| **włączony** | 89.40% |

Jak widać, elityzm pozytywnie wpływa na uzyskany wynik, jednak jego efekt nie jest zbyt imponujący.

## Podsumowanie

3-SAT to problem NP-zupełny, a więc wymagający użycia alternatywnych metod, takich jak algorytmy genetyczne. Uzyskane w ten sposób rozwiązania nie są idealne, jednak w wielu przypadkach – wystarczająco dobre.

Przeprowadzone badania pokazują, że odpowiednia kombinacja parametrów może znacząco poprawić wyniki. Kluczem do sukcesu okazuje się być niewielka, aczkolwiek występująca szansa mutacji, włączony mechanizm elityzmu oraz odpowiednio wysoka liczba iteracji i wielkości populacji.

Czas obliczeń rośnie liniowo wraz ze wzrostem liczby zmiennych w instancji problemu, jak również iteracji i wielkości populacji. Oznacza to, że metoda ta sprawdzi się również w przypadku formuł z bardzo dużą ilością zmiennych, gdyż zwróci wynik w rozsądnym czasie.

## Źródła

1. Dane benchmarkowe
https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

2. Dokumentacja biblioteki GenAlg
https://cran.r-project.org/web/packages/genalg/genalg.pdf

## Załączniki

1. Pliki źródłowe w języku R: fitness.r, parser.r, program.r
2. Formuły CNF w formacie DIMACS
3. Treść zadania (PDF)