Zastosowanie algorytmu genetycznego do uzsykania rozwiązań problemów 3-SAT

Jan Iwaszkiewicz*

Uniwersytet Gdański

8 listopada 2018

Streszczenie

W poniższym opracowaniu, proponuję metodę określenia, bądź znalezienia jak najbardziej optymalnego wartościowania dla problemu spełnialności formuł w koniunkcyjnej postaci normalnej z klauzulami posiadającymi trzy literały (3-SAT). Przedstawione podejście zostało oparte o heurystyczne techniki algorytmów genetycznych, które czerpią swoją inspirację z ewolucji organicznej. Na podstawie przeprowadzonych testów oraz wyprowadzonych z nich wniosków, stwierdzona zostaje poprawność metody. Ostatecznie określona zostaje również najefektywniejsza dostępna konfiguracja dla zaimplemenotwanego rozwiązania.

I. Wstęp

3-SAT jest jednym z najbardziej rozpoznawalnych problemów należących do klasy złożoności obliczeniowej zwanej NP. Charakterystyczną dla tej grupy cechą jest możliwość sprawdzenia rozwiązania w czasie wielomianowym. Pomimo trywialności sprawdzenia rozwiązania, znalezienie takiego jest zadaniem trudniejszym. Zatem wyszukiwanie wartościowania spełniającego daną formułę logiczną w formie CNF powinno być równie czasochłonnym zajęciem. Czy na pewno? W poniższym artykule przedstawiona zostanie metoda wyszukiwania rozwiązań za pomocą binarnego algorytmu genetycznego. Cechujący się losowością, bardzo nieprzewidywalny algorytm, którego głównym zadaniem jest znalezienie optymalnego rozwiązania (niekoniecznie dokładnego) zostanie skonfrontowany z nietrywialną naturą problemu 3-SAT.

II. Chromosom pracujący

Inspiracją dla metody jest naturalne środowisko. Chromosomy jak żywe organizmy próbują dostosować się jak najlepiej do postawionych im warunków. Zatem właściwości chormosomu oraz jego metody przystosowywania się można opisać w następujący sposób:

- **Budowa chromosomu**
  Chromosom posiada swoją wartość (zwaną też genotypem) zapisaną w postaci ciągu bitów, gdzie każdy bit (gen) przyjmuje wartość 0 lub 1. Indeks bitu w ciągu jest odpowiadający indeksowi zmiennej w formule logicznej. Przykład:

Chromosom w postaci:
0110
dla formuły:
$$(\neg x_1 \vee \neg x_4 \vee x_2) \wedge (x_1 \vee \neg x_1 \vee \neg x_3)$$
tworzy następujące podstawienie:
$$(\neg 0 \vee \neg 0 \vee 1) \wedge (0 \vee \neg 0 \vee \neg 1) \rightarrow (1 \vee 1 \vee 1) \wedge (0 \vee 1 \vee 0)$$

*indeks: 238215, kontakt: jiwaszkiewicz6@gmail.com

- **Selekcja**
  Do ponownego zasiedlenia populacji wybiera się nowe osobniki w drodze selekcji. W przedstawionym rozwiązaniu porównujemy dwie metody:
  - **Metoda ruletki**, polegająca na losowym wybraniu dwóch osobników z dostępnej puli.
  - **Metoda turniejowa**, opierająca się na wybieraniu dwóch grup (o stałym rozmiarze podanym przy rozpoczęciu obliczeń) składającej się z losowych osobników. W każdej grupie wybiera się najlepiej przystosowanego z nich, który jest przekazywany do następnego pokolenia.

- **Krzyżowanie**
  Po selekcji dwóch osobników może nastąpić etap krzyżowania. Operacja powoduje przecięcie dwóch wybranych chromosomów w losowo wybranym miejscu. Powstają wtedy dwa nowe osobniki, wprowadzane są do nowej populacji. Operacja powtwarzana jest do momentu ponownego zapełnienia puli chromosomów. Szansa skrzyżowania osobników określana jest w przedziale *(0,1)*.
- **Mutacja**
  Następny nadchodzący proces to mutacja. Polega ona na zamianie bitu na przeciwny. Dzięki takiej operacji możemy spowodować przerwanie rozwoju populacji spowodowanego przez przeludnienie podobnymi osobnikami. Szansa mutacji określana jest w przedziale *(0,1)*.
- **Elityzm**
  Opcjonalny parametr zapewniający przeniesienie danej liczny najlepiej przystosowanych osobników do następnej generacji. Zapewniony zostaje wtedy brak regresji w jakości rozwiązania. Przesadny elityzm może jednak doprowadzić do zastoju w populacji, większość osobników jest bardzo podobna, zmniejsza się szansa na stworzenie nowego rozwiązania. Wartość musi znajdować się w przedziale *(0, rozmiar populacji)*.

III. Fitness

Dla wcześniej opisanych operacji kluczową dla działania algorytmu jest funckja fitness. Określna ona jak dobrze przystosowany jest w danym momencie chromosom. Na podstawie wyniku tej funkcji dokonuje się selekcji osobników. W przedstawionym rozwiązaniu istnieje możliwość zaprzestania działania algorytmu, gdy najlepszy przedstawiciel grupy osiąga pożądaną wartość zwróconą przez funckję fitness. Pozwala to na ograniczenie zbędnych iteracji algorytmu. Poniżej zostaje opisane działanie funkcji:

1. Ustaw licznik poprawnych klauzul równy 0.
2. Dla każdej klauzuli w tablicy formuły:
   (a) Pobierz klauzulę.
   (b) Ustaw wartość check na 0.
   (c) Dla każdego elementu w klauzuli:
      i. Znajdź bit w chromosomie odpowiadający za i-ty element klauzuli.
      ii. Jeśli element klauzuli jest ujemny, zaprzecz bitowi chromosomu.
      iii. Zwróć wartość bitu.
      iv. Sprawdź alternatywę bieżącego elementu z check.
   (d) Zwróć wartość check.
   (e) Jeśli wartość check jest równa 1, dodaj 1 do wartości licznika.
3. Zwróć wartość licznika jako nową funkcję oceny.

IV. Testy i wnioski

i. Selekcja naturalna

Podane testom formuły (wszystkie zawarte w folerze *datasets*) zostały sprawdzone przy pomocy dostępnego w internecie SAT-solvera[1] dla zapewnienia rozwiązywalności zadania.

[1] `https://msoos.github.io/cryptominisat_web/`

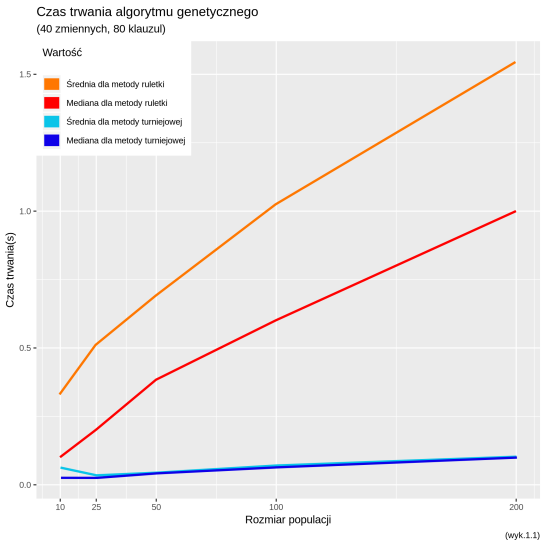
Tablica 1: *Ilość niedokładnych wyników zwróconych przez algorytm*

Selekcja	fails(40x 80k)	fails(50x 100k)
ruletka	1	58
turniejowa	0	3

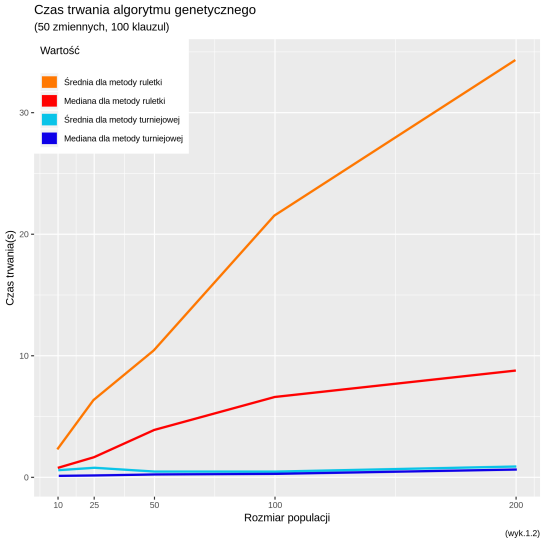
By uzyskać odpowiedź na pytanie, która z metod selekcji prowadzi do skuteczniejszego rozwiązywania przeprowadzone zostały testy dla dwóch instacji problemu w postaci 40 zmiennych w 80 klauzulach oraz 50 zmiennych w 100 klauzulach. Parametry takie jak maksymalna ilość iteracji *(10000)*, elityzm *(1)*, szansa krzyżowania *(0.3)* oraz mutacji *(0.1)* były stałe dla każdego testu. Dla każdej z 10 losowo tworzonych generacji uruchamiano pomiary na 5 unikalnych rozmiarów populacji. Dodatkowo by zapewnić wiarygodność rozwiązania, każda wygenerowana populacja początkowa była identyczna dla obu selekcji. Wykresy sporządzone na podstawie wyników *(Rysunek 1)* wyraźnie przedstawiają przewagę metody turniejowej. Związek ilości iteracji, a czasu wykonania jest również widoczny w przeprowadzonych badaniach[2]. Jest o kilka rzędów wielkości sprawniejsza od przeciętnego uruchomienia selekcji ruletki, pomimo obciążenia ciągłym sortowaniem grup turniejowych.

Efektywność metody to jednak nie tylko czas obliczeń, porównana została również stabilność. Różnica między medianą, a średnią arytmetyczną czasu działania algorytmu dla metody ruletki jest znacząca. Metoda nie jest stabilna przez wysoki stopień losowości w wyborze następnych kandydatów do przetrwania. Określony również został powód tak znaczącej różnicy. W tabeli *(Tablica 1)* wyraźnie widać ilość niedokładnych wyników zwróconych przez algorytm w przypadku niestabilnej metody. W przypadku funkcji ruletki dokładnie rozwiązanie nie zostało znalezione nawet w czasie stałej liczby 10000 iteracji. Dane pokazują jednak, że metoda ruletki potrafi również trafić w momencie lub w pierwszej iteracji w idealne rozwiązanie. Zdarzenie przypadku trafienia idealnego rozwiązania w momencie generacji występowało i będzie występować w obu wersjach selekcji, nie wypływa ono zatem na decyzję o wyborze jednej z przedstawionych metod.

Analizując wyniki badań oczywistą opcją przy uruchomieniu algorytmu staje się zastosowanie selekcji turniejowej, która zapewnia większą stabliność oraz jakość rozwiązania.



(a) *Rysunek 1.1*



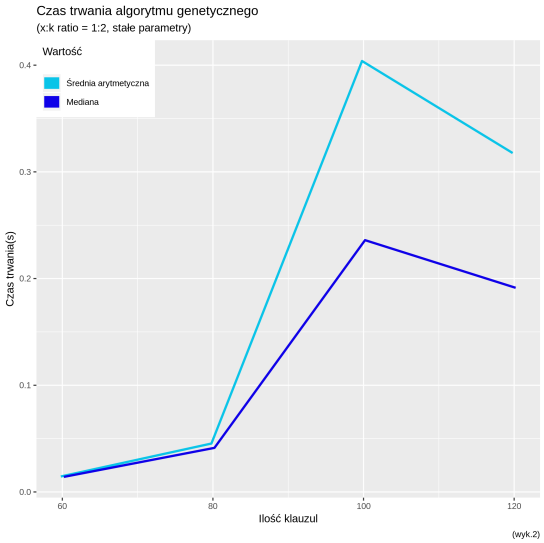
(b) *Rysunek 1.2*

Rysunek 1: *Porównanie czasu trwania obliczeń dla dwóch instancji 3-SAT*

[2]Dostępne w folderze datasets.

ii. Rozmiar formuł

Na wykresie *(Rysunek 2)* został przedstawiony czas działania algorytmu dla zmieniających się rozmiarów klauzul. Zgodnie z oczekiwaniami czas potrzebny na wykonanie programu zwiększa się razem z rozmiarem rozwiązywanego problemu. Program nadzwyczajnie dobrze radzi sobie z problemami do rozmiarów 40 zmiennych w 80 klauzulach. Następnie zauważalny jest gwałtowny spadek wydajności rozwiązania oraz większa rozbieżność czasu wykonania pomiędzy uruchomieniami algorytmu. Możliwym rozwiązaniem tych zachwiań może okazać się dobór bardziej optymalnych parametrów. Natomiast czynnik losowości zmian bądź inny zestaw chromosomów początkowych może zaważyć o czasie zakończenia algorytmu, dlatego nie jest możliwe przewidzenie optymalnych parametrów dla danej instancji problemu bez wyczerpujących testów.



Rysunek 2: *Selekcja turniejowa dla różnych rozmiarów problemu*

Literatura

[Zasoby online] *Algorytmy genetyczne* , dostęp online: `http://home.agh.edu.pl/~vlsi/AI/gen_t/`

[Zasoby online] *Introduction to Genetic Algorithm & their application in data science* , dostęp online: `https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/`

[Kazuo Sugihara] *Measures for Performance Evaluation of Genetic Algorithms* , dostęp online: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.8611&rep=rep1&type=pdf`