Algorytmy genetyczne

2018/19 – projekt

Wykonał: Tomasz Kąkol

Cel:

„Mamy do dyspozycji pewną liczbę prętów stalowych o różnych długościach. Układamy je na taśmie produkcyjnej. Robot pobiera z taśmy trzy kolejne pręty i próbuje zespawać z nich trójkąt. Jeśli jest to możliwe, robi to, a jeśli nie, wyrzuca te trzy pręty i bierze kolejne. Celem projektu jest napisanie algorytmu genetycznego, który ustali taką kolejność prętów na taśmie, aby robot skonstruował jak najwięcej trójkątów i jednocześnie aby odchylenie standardowe ich pól powierzchni od średniego pola powierzchni wszystkich skonstruowanych trójkątów było jak najmniejsze.”

Wykonanie:

W ramach wykonania projektu wykorzystałem elementy biblioteki GALib. Do rozwiązania zadanego problemu użyłem interfejsu ‘GA1DArrayGenome’, przeznaczonego dla genomu reprezentowanego przez wektor (1D). Użyłem ‘prostego’ AG ‘GASimpleGA’

Inicjalizacja

Tworzenie populacji bazowej wykonałem w następujący sposób:

- Tworzę osobnika zawierającego zachowaną kolejność wszystkich prętów zgodnie z danymi początkowymi, zamieszczonymi w pliku prety.txt.
- Zamieniam miejscami w kolejce dwa losowe pręty. Zamian takich wykonujemy w zależności od rozmiaru osobnika.

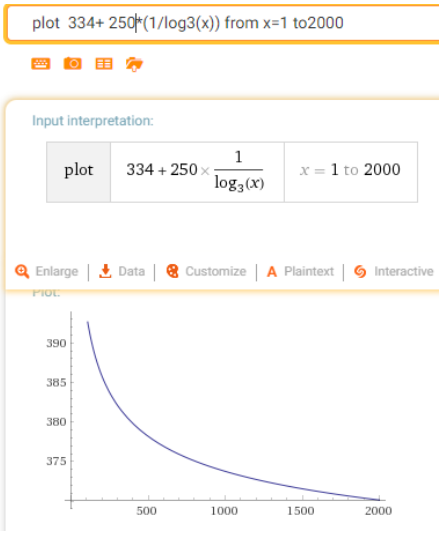
Funkcja dopasowania

Ocena przystosowania chromosomów w populacji polega na obliczeniu wartości funkcji przystosowania dla każdego chromosomu z tej populacji. Im większa jest wartość funkcji, tym lepsza ”jakość” chromosomów. Postać funkcji przystosowania zależy od rodzaju rozwiązywanego problemu. W naszym przypadku zdecydowałem się na zastosowanie takiej funkcji dostosowania, aby uwzględniała ona oba kryteria – maksymalizację ilości trójkątów i minimalizację odchylenia standardowego od średniej powierzchni tych trójkątów. Moją definicję cechuje użycie logarytmu o podstawie 3 (ponieważ trójkąt ma 3 boki), jak również uwzględnienie maksymalnej liczby trójkątów, które możemy teoretycznie otrzymać na podstawie ilości danych w pliku prenty.txt. Wzór uwzględnia również aktualną ilość trójkątów.

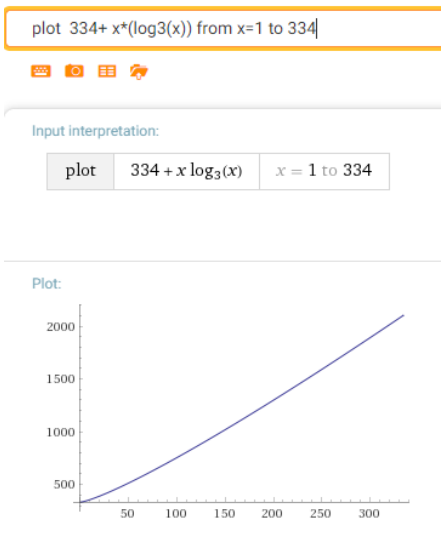
Starałem się tak dobrać współczynniki, aby zbiory o ‘mniejszej’ ilości danych promowała relacja ‘więcej trójkątów, ale godzę się na większe odchylenie’, natomiast ‘większe’ zbiory powinna cechować relacja ‘wolę mniejsze odchylenie, ale godzę się na nie uzyskanie maksymalnej ilości trójkątów’.

Takie podeście powinno sprzyjać odrzuceniu skrajnych rezultatów z dużymi wartościami odchyleń standardowych w przypadku większego rozmiaru genomu.

Przykładowa próba uwzględnia dwóch kryteriów w AG:



Rysunek 1 Uwzględnienie odchylenia standardowego



Rysunek 2 Uwzględnienie ilość trójkątów

Opis zastosowanych metod krzyżowania, mutacji oraz selekcji

Przyjąłem następujące wartości stałych:

- wielkość populacji: 250
- liczba generacji: 250
- metoda selekcji: Rank
- metoda krzyżowania: ‘PartialMatchCrossover’
- prawdopodobieństwo mutacji: 0.01
- prawdopodobieństwo krzyżowania: 0.9

Wielkość populacji i ilość generacji wyznaczyłem w sposób empiryczny. Co prawda, m.in. przy zastosowanej metodzie selekcji zwiększenie wartości tych parametrów może okazać się korzystne, to jednak czas wykonywanych obliczeń wzrasta. Jest to niekorzystna cecha, a chciałem uniknąć relatywnie ‘długiego’ czasu wykonywanych obliczeń. Przechowanie wartości zmiennoprzecinkowymi (typu float) długości prętów i powierzchni trójkątów umożliwiło stworzenie specjalnych wektorów.

W klasycznym algorytmie genetycznym stosuje się dwa podstawowe operatory genetyczne: operator krzyżowania oraz operator mutacji. Operator mutacji odgrywa zdecydowanie drugoplanową rolę do operatora krzyżowania.

Krzyżowanie występuje z dużym prawdopodobieństwem (ja wybrałem prawdopodobieństwo 0.9), podczas gdy mutacja relatywnie rzadko (prawdopodobieństwo 0.01). Taki wybór jest oparty w analogii do świata organizmów żywych, gdzie mutacje zachodzą niezwykle rzadko. W AG mutacja chromosomu może być dokonywana na populacji rodziców przed operacją krzyżowania lub na populacji potomków utworzonych w wyniku krzyżowania. Ja wybrałem pierwszą opcję. Jako metody krzyżowania użyłem ‘PartialMatchCrossover’. W naszym przypadku ważne jest zachowanie ilości wystąpień poszczególnych wartości (każda wartość może wystąpić tylko raz, nie można 2 razy użyć tego samego pręta, a inny odrzucić). Również genom ‘rodzic’ i genom ‘potomny’ muszą być tej samej wielkości. Użyta metoda gwarantuje nam spełnienie tych warunków.

Selekcja chromosomów polega na wybraniu na podstawie obliczonych wartości funkcji przystosowania, tych chromosomów, które będą brały udział w tworzeniu potomków do następnego

pokolenia, czyli następnej generacji. Wybór ten odbywa się zgodnie z zasadą naturalnej selekcji, tzn. największe szanse na udział w tworzeniu nowych osobników mają chromosomy o największej wartości funkcji przystosowania (w każdej kolejnej iteracji oblicza się wartość funkcji przystosowania każdego z chromosomów tej populacji. W naszym AG cała poprzednia populacja chromosomów zastępowana jest przez tak samo liczną nową populację potomków). Użyta metoda selekcji 'GARankSelector' wybiera po prostu najlepszego osobnika z populacji.

Sprawdzanie warunku zatrzymania

Nie zastosowałem specjalnego warunku zatrzymania algorytmu genetycznego. Algorytm jest zatrzymywany po wykonaniu zdefiniowanej ilości iteracji.

Testy

W celu przeprowadzenia testów wykonałem sobie automatyczną generację pliku prenty.txt z wartościami na zakresie 1-50 i długością do 1002 (czyli tak jak w zdefiniowanych warunkach projektu). Na podstawie wygenerowanych plików wypisywałem w konsoli ilości 'wyprodukowanych' trójkątów i wartość odchylenia standardowego. Proces testowania w znacznej mierze przeprowadziłem w sposób manualny. Etap testowania wykonanego przeze mnie projektu jest jego słabą częścią. Nie uśredniałem wyników dla 'najlepszej parametryzacji' algorytmu. Mimo tych wad, na podstawie wykonanych badań empirycznych i posiadanego doświadczenia zdecydowałem się na akceptacje rezultatów, ponieważ otrzymywane wyniki były dla mnie satysfakcjonuje.