Kamil Gołofit
Michał Toporowski

EINIS - DOKUMENTACJA PROJEKTU

21.06.2015

1. Temat projektu

Implementacja algorytmu genetycznego dla zadanego problemu optymalizacji.

2. Sprecyzowane tematu projektu

W ramach tematu projektu przedstawionego w punkcie 1. został zaimplementowany standardowy algorytm ewolucyjny wykorzystujący procesy krzyżowania i mutacji do rozwoju populacji.

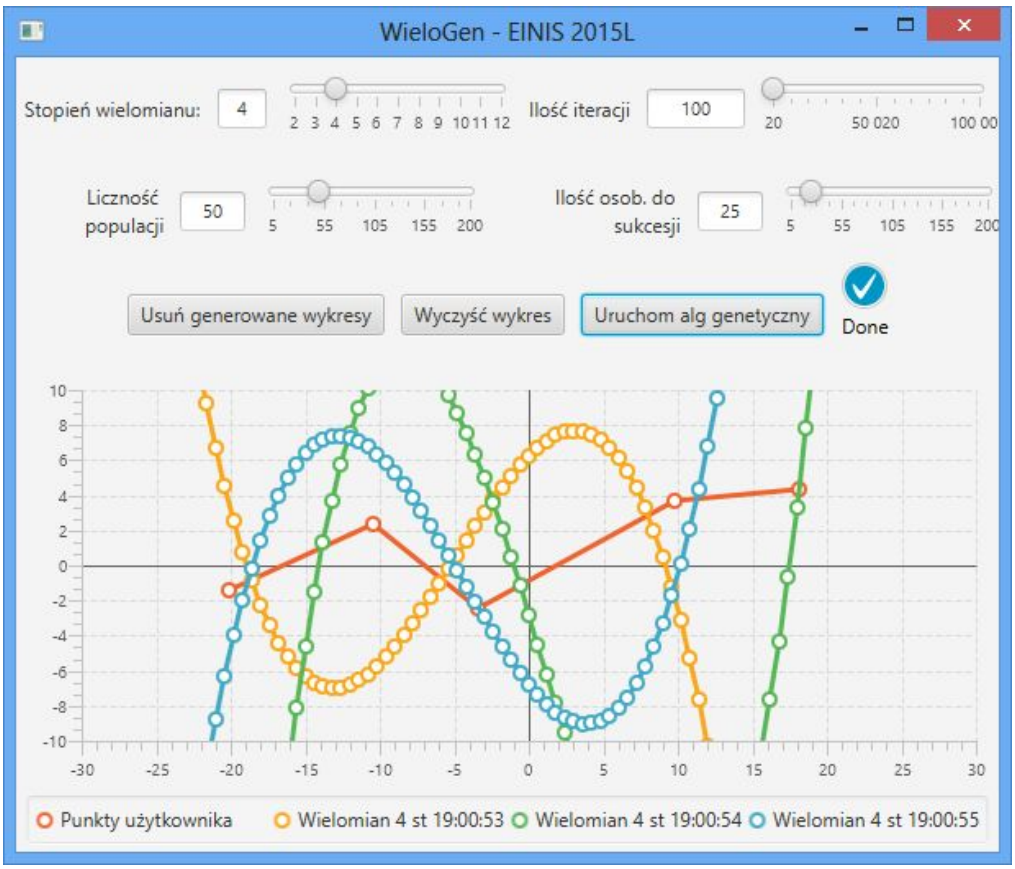
Jako problem optymalizacji zaproponowano dobór współczynników wielomianu *n*-tego stopnia tak, aby odległość w osi odciętych wielomianu do zadanych przez użytkownika punktów była jak najmniejsza.

3. Funkcjonalność aplikacji

Aplikacja będąca celem niniejszego projektu ma posiadać graficzny interfejs użytkownika, który pozwoli na:

- Zaznaczenie punktów przez użytkownika na wykresie,
- ustawienie parametrów problemu optymalizacji oraz algorytmu ewolucyjnego:
  - stopnia wielomianu,
  - ilości iteracji alg. ewolucyjnego,
  - liczności populacji,
  - ilości osobników do sukcesji w każdym kroku.
- uruchomienie algorytmu dla zadanych parametrów,
- wyświetlenie wyniku w formie wykresu.

Aplikacja powstała w wyniku projektu spełnia wymagania jej postawione. Jej okno wygląda następująco:



Powstała aplikacja jako pojedynczego osobnika algorytmu ewolucyjnego traktuje wektor współczynników $a_n,...,a_0$ kolejnych wyrazów wielomianu $f(x)=a_n*x^n + a_{n-1}*x^{n-1} + \ldots + a_1x + a_0$. Sam algorytm ewolucyjny w każdej iteracji:

- dla każdej pary osobników krzyżuje je z prawdopodobieństwem p_k losowanym z przedziału [0;1] z rozkładem normalnym. Krzyżowanie polega na zamianie współczynników na odpowiadających im miejscach z prawdopodobieństwem p_k.
- dla każdego osobnika populacji mutuje go z prawdopodobieństwem p_m losowanym z przedziału [0;1] z rozkładem normalnym. Operacja mutacji polega na dodaniu do każdego współczynnika wielomianu losowej wartości rozkładu Gaussa ze średnią wartością równą 0 i wariancją równą 1,
- ze zutowanych i skrzyżowanych osobników wybiera *i* najlepszych osobników, które przechodzą do następnej iteracji.

Warunkiem stopu przyjętym w projekcie jest zadana przez użytkownika liczba iteracji po wykonaniu których aplikacja przerywa ewolucję i prezentuje aktualny najlepszy wynik. Przykładowe wyniki załączone są w punkcie 5.

4. Użyte biblioteki i technologie

Aplikacja została napisana w języku Java oraz Scala, korzysta z następujących bibliotek:

- JUnit - biblioteka wspomagająca testowanie jednostkowe,
- Scala-library - biblioteka pozwalająca na połączenie kodu napisanego w języku Scala z kodem napisanym w języku Java oraz uruchomienie aplikacji na wirtualnej maszynie Javy,
- Log4J - biblioteka wspomagająca logowanie (tworzenie logów aplikacji),
- JavaFX - framework wspomagający tworzenie graficznego interfejsu użytkownika w oparciu o język skryptowy FXML.
- Watchmaker - framework wspomagający algorytm ewolucyjny. Dostarcza on cały schemat działania algorytmu ewolucyjnego, dzięki czemu zadaniem programisty staje się zdefiniowanie operatorów mutacji, krzyżowania, selekcji, funkcji oceny osobników, metody losowania populacji początkowej bez konieczności implementacji własnego “silnika” algorytmu ewolucyjnego.

Do kompilacji aplikacji oraz przy pobraniu i dołączeniu do programu wymienionych bibliotek został użyty Maven - narzędzie do automatyzacji budowy oprogramowania na platformę Java.

5. Testy aplikacji

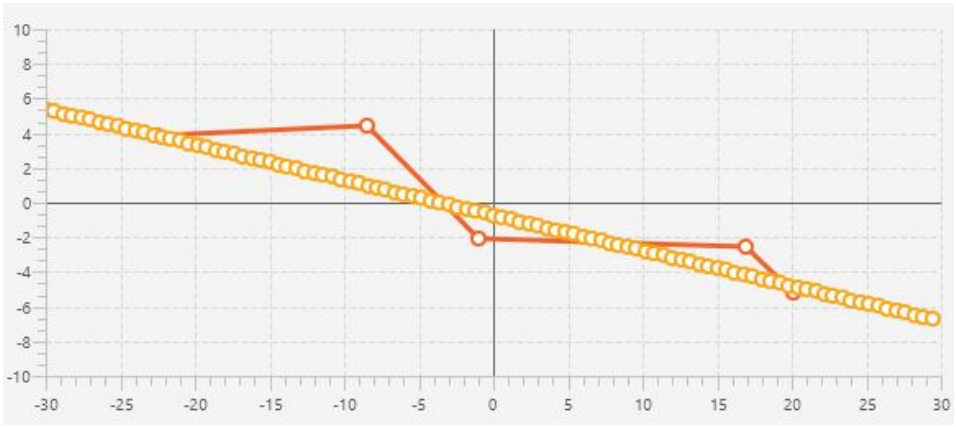
W ramach testów dla różnych parametrów wejściowych został uruchomiony zaimplementowany algorytm ewolucyjny. Przykładowe wyniki w formie graficznej dla wielomianów różnych stopni, oraz różnych punktach wprowadzonych przez użytkownika przedstawione są poniżej. Kolorem pomarańczowym oznaczone są zadane przez użytkownika punkty. Kolorami: żółtym, zielonym, niebieskim oznaczone są wykresy wielomianu wyznaczonego przez algorytm ewolucyjny.

Wielomian postaci $f(x) = a_1x + a_0$ dla 100 iteracji algorytmu ewolucyjnego dał wynik:

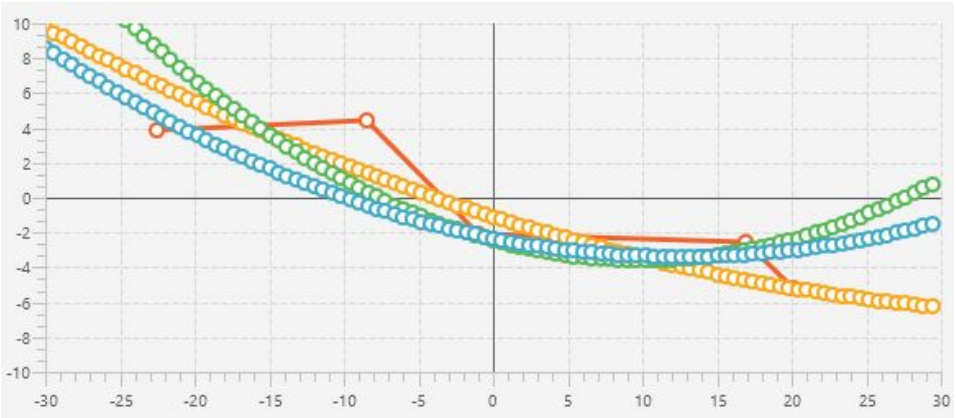
Kamil Gołofit
Michał Toporowski

EINIS - DOKUMENTACJA PROJEKTU

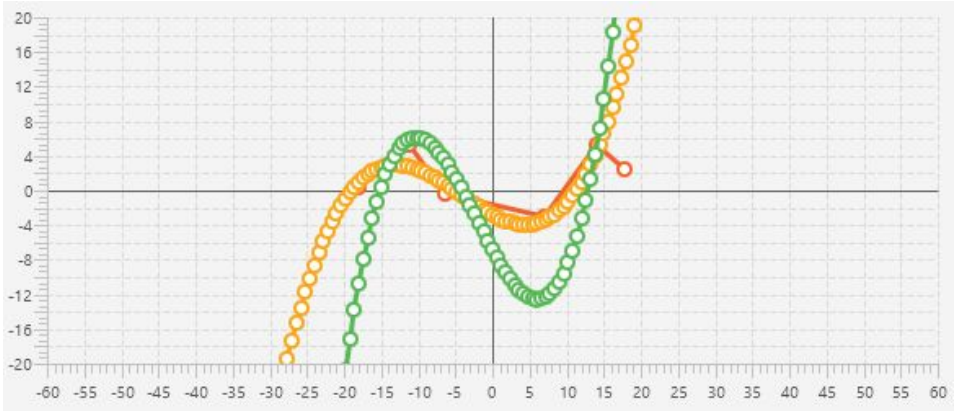
21.06.2015



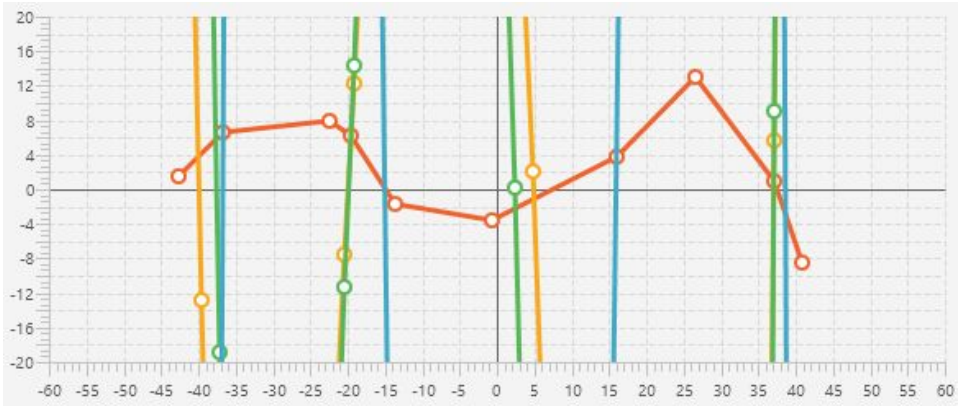
Wielomian postaci $f(x) = a_2x^2 + a_1x + a_0$ dla 100 iteracji algorytmu ewolucyjnego:



Wielomian postaci $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ dla 1000 iteracji algorytmu ewolucyjnego:



Wielomian postaci $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ dla 20000 iteracji:



6. Wnioski

Algorytm ewolucyjny sprawdził się dość dobrze dla zadanego problemu optymalizacji współczynników wielomianu. Każde uruchomienie daje nieco inne rezultaty, co jest oczywiście konsekwencją faktu, iż populacja inicjalna jest za każdym razem losowana.
Dla funkcji liniowej, kwadratowej, sześciennej wykresy wielomianu są bardzo “intuicyjne” - algorytm ewolucyjny w 100-1000 iteracjach odnajduje współczynniki wielomianu dające bardzo dobre efekty. Przy zwiększonej liczbie iteracji wynikowy wykres jeszcze dokładniej pokrywa zadane przez użytkownika punkty. Im większa liczba iteracji, tym bardziej “dobre” rozwiązania są faworyzowane, wskutek czego wynikowe wykresy są bliższe zadanym punktom.

Dla wielomianu stopnia 4go i wyżej algorytm zazwyczaj nie dopasowuje intuicyjnych współczynników wielomianu - otrzymane wykresy wykraczają poza prezentowany obszar, liczne punkty przegięcia posiadają bardzo duże lub bardzo małe wartości, przez co wykres wydaje się być fragmentami bardzo stromych funkcji. Taki stan rzeczy bierze się prawdopodobnie z losowania początkowych współczynników z zakresu [-5;5] z rozkładem liniowym, przez co wartości początkowe są zbyt zbliżone do siebie, i nawet przy ilości iteracji rzędu 100000 nie zdążą wyewoluować na tyle, by dać wielomian o wykresie mającym “intuicyjny” przebieg.

Algorytm ewolucyjny jest bardzo ciekawą metodą optymalizacji, która dzięki niedeterministycznemu podejściu pozwala czasem bardzo szybko dojść do rozwiązania o pożądanej jakości. Niestety ten sam niedeterminizm prowadzi czasami do “obracania się” algorytmu w tym samym miejscu, przez co dobre wyniki możliwe są jedynie przy wielokrotnym rozwiązaniu problemu optymalizacji i wybraniu najlepszego spośród wyników.