Dokumentacja do projektu PSZT.

Dawid Brzozowski

Robert Ostoja – Lniski

Skrót projektu: TM.AE.4

Decyzje projektowe:

1) Projekt realizowany jest w języku Java, w technologii JavaFX.

1.1) Projekt korzysta ze wzorca projektowego MVC.

1.2) Dane są wizualizowane przy użyciu klasy dziedziczącej po klasie BubbleChart pochodzącej z pakietu javafx.scene.chart.

2) Rozwiązanie zadania oparte jest na algorytmie 1+1.

2.1) Początkowo generowane są dwie populacje (jedna poszukująca minimum, druga maksimum) równomiernie rozłożone na obszarze [ A , B ] x [ C , D ]. Każda kolejna generacja zbliża program do odnalezienia ekstremów lokalnych funkcji.

2.2) Poszukiwanie minimum funkcji z = f( x , y ) sprowadzane jest do poszukiwania maksimum funkcji  
 z = -f( x , y ).

2.2) Realizacja generacji kolejnych pokoleń :

* Dla każdego osobnika populacji, generujemy potomka o losowej ( według rozkładu Gaussa) odległości od rodzica.
* Z tych dwóch osobników ( rodzica i potomka ), wybieramy tego, któremu odpowiada większa wartość funkcji.
* Wprowadzamy paramter fitness jako proporcję liczby wyborów potomka, do liczby liczby wszystkich dotychczasowych generacji.
* W zależności od φ co M generacji zmniejszamy bądź zwiększamy promień poszukiwań. Dla fitness < φ, promień poszukiwać jest mnożony przez stałą C1, dla fitness > φ, mnożony jest przez stałą C2.
* Po zadanej ilości generacji, kończymy program i wizualizujemy rezultat.
* Przyjęte wartości parametrów to: φ = 0.2, M = 10, C1 = 0.82, C2 = 1.2 ( zgodnie z treścią wykładu zapewniają one najlepsze rezultaty).

3) Opis struktury programu.

3.1) Program jest podzielony na klasy:

* Evolver.java - klasa odpowiedzialna za realizację algorytmu 1+1.
* Function.java - klasa obliczająca wartość funkcji w punkcie, za pomocą parametru można wybrać jedną z dostępnych funkcji.
* Point.java - klasa przechowująca informacje o położeniu punktu oraz promieniu poszukiwań.
* Individual.java - klasa dziedzicząca po klasie Point zawierająca dodatkowo wartość funkcji w punkcie i parametr fitness.
* PopulationChart.java - klasa dziedzicząca po klasie BubbleChart. Otrzymuje wektor przechowujący populację i przedstawia ją na wykresie z rozróżnieniem minimum i maksimum.
* PopulationHandler.java - klasa zarządzająca całą populacją. Implementuje metodę run(), która jest odpowiedzialna za generowanie nowych osobników.

4) Wykorzystywane narzędzia i biblioteki

* JavaFX
* Java.util.Random
* MVC
* CSS

5) Korzystanie z programu

5.1) Użytkownich po urochomieniu programu wybiera:

* Ilość generacji w celu odnalezienia ektremów
* Wartości parametrów A , B , C , D
* Jedną z czterech podanych funkcji

5.2) Po naciśnięciu przycisku z funkcją wyświetla się w oddzielnym oknie wykres z naniesonymi wszystkimi osobnikami z populacji. Za pomocą odpowiednich kolorów są rozróżnione populację wyszukujące minimum i maksimum.

5.3) Parametry określające działanie algorytmu ewolucyjnego są zdefiniowane jako zmienne typu private static final double i zapisane są w klasie Evolver.java

5.4) Parametry określające przedział poszukiwań, dokładność poszukiwań, liczbę osobników znajdują się w funkcji Main.

6) Podział obowiązków

6.1) Podczas tworzenia projektu korzystaliśmy ze wspólnego repozytorium na stronie https://github.com/robertostoja-lniski/AI\_PSZT

6.2) W trakcie pracy nad projektem, początkowo korzystaliśmy z algorytmu ( λ + μ). Rozwiązanie odparte na nim znajduje się pod linkiem <https://github.com/robertostoja-lniski/AI_PSZT/commit/33969054e09998100194ab7b309688ea2edc06f6>.

6.2) Dawid Brzozowski obowiązki

* MVC, interfejs aplikacji

6.3) Robert Ostoja – Lniski

* Wizualnizacja danych

6.4) Decyzje projektowe tj. schemat klas, wybór algorytmu, realacje między klasami zostały uzgadniane wspólnie. Nad klasami związanymi ze sztuczną inteligencją i przechowywaniem danych praca była naprzemienna. Dowodem na to jest nasza regularna praca nad wspólnym repozytorium.