Teoria algorytmów i obliczeń – Projekt – Etap 2

Błażej Bobko, Jakub Gocławski, Patryk Kujawski, Radosław Kutkowski

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska

1 Szczegóły implementacyjne

Rozwiązanie zostało zaimplementowane w języku *C#* z wykorzystaniem technologii *WPF* do stworzenia środowiska graficznego (*GUI*). Nie były wykorzystywane żadne dodatkowe, zewnętrzne biblioteki.

1.1 Moduły

Projekt został podzielony na 4 moduły opisane poniżej.

UserInterface Moduł zawiera implementację *GUI* w technologii *WPF* oraz klasy pomocnicze wspomagające interakcję z użytkownikiem, m.in. w celu wczytania automatu z pliku lub wprowadzenia parametrów obliczeń.

TestGenerator Moduł zawiera klasę *TestSets*, służącą do przechowywania danych zbiorów: treningowego oraz testowego. Klasa ta potrafi także generować te zbiory, a także zapisywać je do pliku i wczytywać z pliku. Na podstawie danych zawartych w tej klasie można odpowiedzieć na pytanie, czy dwa słowa są w relacji.

PSO Moduł zawiera klasę *MachinePSO* zawierającą główną pętlę algorytmu *PSO* oraz klasę *Particle* będącą reprezentacją pojedynczej cząsteczki roju wykorzystywanego przez *MachinePSO*.

LanguageProcessor Moduł zawiera klasę *Machine* będącą reprezentacją odtwarzanego automatu, klasę *Alphabet* zawierającą dostępne litery alfabetu wraz z funkcjami ułatwiającymi konwersję ich formatu oraz klasę pomocniczą *Extensions*.

2 Interfejs użytkownika

Po uruchomieniu programu możliwa jest modyfikacja głównych parametrów obliczeń. Oprócz podstawowych parametrów dokładniej opisanych w dokumencie z I etapu obecne są także dwa dodatkowe:

- Szansa śmierci cząsteczki – cząsteczka z podanym prawdopodobieństwem może umrzeć w czasie każdego wykonywanego kroku i zostać zastąpioną nową, losową cząsteczką
- Cząsteczki przekazane do następnej iteracji – liczba cząsteczek z n-tej iteracji, które zostaną przekazane do $n+1$-szej iteracji, gdzie n, to liczba stanów poszukiwanego automatu

Parametry należy zatwierdzić przyciskiem *Zatwierdź Parametry*. Następnie należy kliknąć przycisk *Wczytaj automat*. Pokaże się okno wyboru pliku, w którym należy wskazać plik w zdefiniowanym w zadaniu formacie. Po wczytaniu pliku należy kliknąć przycisk *Generuj zbiory*.

Pojawi się nowe okno umożliwiające albo wczytanie wygenerowanych wcześniej testów z pliku albo wygenerowanie nowych zbiorów testowych i treningowych.

Podczas generowania zbiorów testowych można ustalić parametry:

- Losowe Testy – liczba losowych słów dłuższych niż parametr *Dokładne próby*
- Dokładne próby – zostaną wygenerowane wszystkie słowa o długości <= od podanego parametru
- Grupa Kontrolna – rozmiar zbioru par dłuższych słów, nie bioracych udziału w procesie optymalizacji w trakcie trwanis PSO

Po wygenerowaniu zbiorów lub wczytaniu ich z pliku, należy nacisnąć przycisk *Akceptuj*. Po powrocie do głównego okna programu, aktywny staje się przycisk *Rozpocznij PSO*; jego kliknięcie powoduje pojawienie się nowego okna, w którym można rozpocząć obliczenia. Aby to zrobić, należy kliknąć przycisk *Start*. W trakcie trwania obliczeń można śledzić ich postęp. Po zakończeniu działania algorytmu pojawi się okno z komunikatem. Następnie można zapisać logi z obliczeń za pomocą przycisku *Stwórz log*.

3 Dodatkowe funkcjonalności

3.1 Interfejs graficzny

Co prawda element ten nie był jeszcze wymagany na tym etapie, jednak został już zaimplementowany, co ułatwia uruchamianie obliczeń. Interfejs jest w pełni responsywny w trakcie trwania obliczeń i pozwala na bieżąco śledzić postępy w poszukiwaniu rozwiązania.

3.2 Zapisywanie i wczytywanie zbiorów testowych

Raz wygenerowane pozwalają znacznie przyspieszyć porównywanie wydajności programu dla różnych parametrów oraz zapewniają, że próby odtwarzania będą dotyczyć dokładnie tego samego automatu.

3.3 Równoległe wykonywanie obliczeń

Główna pętla programu została zaimplementowana z wykorzystaniem wątków za pomocą dostępnej w $C\#$ klasy $Task$.

3.4 Zapisywanie logów

Po wykonaniu obliczeń możliwe jest zapisanie do pliku pełnego logu zawierającego wyniki dla każdej testowanej pary słów.

4 Zmiany względem I etapu

4.1 Reprezentacja automatu w trakcie działania PSO

W części opisującej działanie algorytmu, w dokumencie z poprzedniego etapu, planowaliśmy reprezentować automat jako macierz zmiennopozycyjnych liczb. Macierz ta miała mieć zaokrąglane wartości do liczb całkowitych i na jej podstawie miał być konstruowany automat, za pomocą którego obliczana byłaby wartość funkcji błędu.

Zdecydowaliśmy się jednak wprowadzić usprawnienie polegające na tym, że cząsteczka posiada automat reprezentowany przez macierz z liczbami zmiennopozycyjnymi, a są one zaokrąglanie do liczb całkowitych tylko w momencie przeprowadzania obliczenia na automacie. Dzięki temu uniknęliśmy kosztownego tworzenia nowych obiektów automatów, a w szczególności kopiowania macierzy z wartościami. Teraz obliczenia odbywają się „w miejscu” z perspektywy pamięci.