# Teoria algorytmów i obliczeń – Projekt – Etap 3

Błażej Bobko, Jakub Gocławski, Patryk Kujawski, Radosław Kutkowski

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska

## 1 Dokumentacja algorytmu

## 2 Dokumentacja techniczna rozwiązania

### 2.1 Szczegóły implementacyjne

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

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*.

**LanguageProcessorTests** Moduł zawiera testy jednostkowe najważniejszych modułów rozwiązania.

### 2.2 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:

- Liczba cząsteczek – parametr PSO, liczba cząsteczek w roju
- Waga prędkosci – parametr PSO, wpływ poprzedniej prędkosci podczas ustalania kolejnej
- Waga lokalna – parametr PSO, wpływ najlepszego rozwiązania danej cząsteczki podczas ustalania nowej prędkosci
- Waga globalna – parametr PSO, wpływ najlepszego rozwiązania z całego roju podczas ustalania nowej prędkosci
- 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

– Maksymalna liczba stanow – maksymalna licbza stanów rekonstruowanego 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:

– Maksymalna długoć „krótkiego" słowa – zostaną wygenerowane wszystkie pary słów o długości niewiększej od podanego parametru
– Liczba długich słów (zbiór treningowy) – liczba par losowych słów dłuższych niż parametr okreslony powyżej
– Rozmiar zbioru testowego – liczba par dłuższych słów, nie bioracych udziału w procesie optymalizacji w trakcie trwania 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*. Można także podejrzeć różnice pomiędzy znalezionym, a poszukiwanym automatem, za pomocą przycisku *Porównaj automaty*.

### 2.3 Dodatkowe funkcjonalności

**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.

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

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

### 2.4 Zmiany względem I etapu – TODO!

**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.

## 3 Raport z testów

### 3.1 Parametry obliczeń

– Liczba liter w alfabecie: 5
– Liczba cząsteczek: 20 (liczba wystarczająca, aby otrzymać interesujące wyniki, ale nie nazbyt duża, by nie spowolnić obliczeń)

- Waga prędkości (inertia weight): 0,729 (jedna z wartosci polecanej dla PSO)
- Waga lokalna: (cognitive weight): 1,49445 (jedna z wartosci polecanej dla PSO)
- Waga globalna: (social weight): 1,49445 (jedna z wartosci polecanej dla PSO)
- Szansa śmierci cząsteczki: 0,01 (wybrana przez nas wartość, wieksza zaburzy algorytm, a mniejsza spowoduje znikome znaczenie tego usprawnienia)
- Cząsteczki przekazane do następnej iteracji: 4 (wybrana przez nas wartość)
- Maksymalna liczba stanów: (różne wartosci w częsci A oraz B)

### 3.2 Testy dla stałej c=4

Obliczenia były wkonywane dla stałej $c = 5$. A zatem:

- Liczba wszystkich słów krótkich: 780
- Liczba wszystkich permutacji słów krótkich: 303 810
- A zatem rozmiar zbioru treningowego: 607 620
- Rozmiar zbioru testowego: 607 620

### 3.3 Testy dla stałej c=5

Nie udało się wykonać obliczeń dla stałej $c = 5$, gdyż taka wartość ma nastepujące konsekwencje:

- Liczba wszystkich słów krótkich: 3 905
- Liczba wszystkich permutacji słów krótkich: 7 622 560
- A zatem rozmiar zbioru treningowego: 15 245 120
- Rozmiar zbioru testowego: 15 245 120

Powoduje to problemy z pamięcią oraz wydajnoscią. Sam proces generowania zbiorów o takiej licznosci wymaga w szczytowym momencie 10,5 GB pamięci RAM. Wymaga to bardzo mocnego komputera oraz wykonywania w rodowisku 64-bitowym. Niestety nawet po wygenerowaniu tak ogromnych zbiorów obliczenia trwają bardzo długo. Próbowalismy uruchomić obliczenia dla Automatu nr 2 z podpunktu A, klasa 5-stanowa. Po 25 minutach obliczeń (z wykorzystaniem 11 GB pamięci RAM), przetwarzany był ciągle automat o 2 stanach, z błędem 45%.

Przerwalimy obliczenia, nie widząc perspektyw na ich ukończenie, a wolelimy skupić się na dokładniejszych testach dla stałej $c = 4$.

### 3.4 A. Rekonstrukcja automatów

### 3.5 B. Aproksymacja automatów