Konrad Opaliński

Wojciech Sitek

Dokumentacja projektu 1 PSZT

# Interpretacja treści zadania (doprecyzowanie, dodatkowe założenia)

## Treść zadania

Modyfikujemy klasyczny algorytm ewolucyjny, losowo łącząc osobniki w pary (aż do śmierci). Niech funkcja oceny zwraca średnią funkcji oceny pary: q(i) = q(j) = mean(q(i), q(j)).

## Interpretacja treści

Celem zadania jest implementacja klasycznego algorytmu ewolucyjnego w jednym z języków programowania, a następnie dodanie funkcji jego działania w trybie zmodyfikowanym. Następnie głównym celem zadania jest zbadanie, jak modyfikacja oraz prawdopodobieństwo mutacji wpływa na wyniki działania algorytmu. Powinna zostać udostępniona możliwość wykonania algorytmu bez oraz z opcją ślubu.

### Ślub

Ślub jest wykonywany na początku każdego kroku ewolucji. Składa się on z następujących kroków:

* Obliczenie funkcji przystosowania wszystkich osobników
* Losowe połączenie w pary osobników[[1]](#footnote-1)
  + Ustawienie flagi zaznaczającej, że osobnik jest małżonkiem[[2]](#footnote-2)
  + Przypisanie do wartości funkcji przystosowania średniej wartości funkcji przystosowania dwóch osobników z pary

# Wkład poszczególnych autorów

* Implementacja algorytmu ewolucyjnego z modyfikacją (śluby) – praca wspólna
* Implementacja funkcji oceny – Wojciech Sitek
* Implementacja testowania – Konrad Opaliński
* Dokumentacja (punkty 1-5) – Wojciech Sitek
* Dokumentacja (punkty 6-10) – Konrad Opaliński

Podział wkładu pracy jest ogólny, ponieważ wszystkie zagadnienia były omawiane wspólnie i często udoskonalane przez drugą osobę.

# Ważne decyzje projektowe

1. Język programowania – Java

Celem wyboru języka Java do zaimplementowania algorytmu była chęć całkowicie **samodzielnego** (bez korzystania z gotowych narzędzi, bibliotek i kodów) zaimplementowania algorytmu ewolucyjnego, jego modyfikacji (ślubu) oraz przede wszystkim funkcji oceny CEC. Język Java wydawał się idealny do tego typu przedsięwzięcia z powodu subiektywnej łatwości implementacji.

1. Typ algorytmu ewolucyjnego – mi + lambda
2. Operatory ewolucyjne – wyłącznie mutacja (bez krzyżowania)

## 3.4. Opis algorytmów

1. Algorytm ewolucji:
   1. Stworzenie losowej populacji mi osobników (wektorów liczb rzeczywistych)
   2. Ślub (opcjonalnie)
   3. Utworzenie tymczasowej populacji potomnej lambda osobników
   4. Mutacje z rozkładu normalnego
   5. Ponowne obliczenie funkcji przystosowania dla tymczasowej populacji potomnej
   6. Połączenie populacji rodzicielskiej i tymczasowej potomnej i wybór mi osobników o najlepszej funkcji przystosowania
2. Łączenie w pary osobników
   1. Użycie funkcji Collections.shuffle() w celu otrzymania losowej permutacji osobników w populacji
   2. Dla całej populacji:
      1. Poszukiwanie kolejnych dwóch osobników, którzy nie są jeszcze małżonkami
      2. Wykonanie ślubu tych osobników
3. Uworzenie tymczasowej populacji potomnej lambda osobników

Korzystano z metody koła ruletki. Polega ona na losowym wybieraniu lambda osobników z populacji rodzicielskiej z prawdopodobieństwem wybrania proporcjonalnym do wartości funkcji celu w porównaniu z wartościami funkcji celu całej populacji rodzicielskiej.

1. Mutacja

Jest ona przeprowadzana zgodnie za pomocą wzorów pobranych z wykładu przedmiotu PSZT.

1. Wybór populacji wynikowej z populacji rodzicielskiej i potomnej
   1. Połączenie populacji rodzicielskiej i potomnej w jedną
   2. Posortowanie populacji względem wartości funkcji przystosowania od najlepszego do najgorszego
   3. Usunięcie z populacji ostatnich najgorszych osobników, tak aby zostało mi osobników.

# Lista wykorzystanych narzędzi i bibliotek

Wykorzystano język Java oraz standardowe biblioteki i narzędzia Javy, takie jak między innymi:

* ArrayList
* Java.util.Math

# Instrukcja pozwalającą na odtworzenie uzyskanych wyników

* Cele i tezy przeprowadzonych badań, np. celem tego eksperymentu jest zbadanie wpływu zmiany rozmiaru populacji na wynik optymalizacji. Zakładamy, że wraz ze zwiększeniem rozmiaru populacji rosła będzie jakość uzyskanych wyników.
* Wyniki eksperymentów w postaci czytelnych tabel i/lub wykresów.
* Omówienie wyników eksperymentów. (co widać)
* Wnioski

1. Wszystkich lub prawie wszystkich w przypadku nieparzystej liczby osobników. Algorytm opisany w punkcie 3.4.1. [↑](#footnote-ref-1)
2. Flaga dla tego osobnika nigdy nie zostaje usunięta (aby spełnić warunek połączenia aż do śmierci). Dopiero osobniki potomne od niego pochodzące mają usuniętą tę flagę. [↑](#footnote-ref-2)