Konrad Opaliński
Wojciech Sitek

Dokumentacja projektu 1 [PSZT]

1. **Interpretacja treści zadania (doprecyzowanie, dodatkowe założenia)**
    a. 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)).

    b. Interpretacja treści:
    Celem zadania jest implementacja klasycznego algorytmu ewolucyjnego powiększonego o możliwość działania w zmodyfikowany sposób przedstawiony w treści zadania. Następnym etapem jest zbadanie, jak owa modyfikacja oraz prawdopodobieństwo mutacji wpływa na wyniki działania algorytmu. Powinna zostać udostępniona możliwość wykonania algorytmu bez oraz z opcją ślubu.

    Ślub jest wykonywany na początku każdego kroku ewolucji. Składa się on z następujących etapów:
    - Obliczenie funkcji przystosowania wszystkich osobników,
    - Losowe połączenie w pary osobników (wszystkich lub prawie wszystkich w przypadku nieparzystej liczby osobników. Algorytm opisany w punkcie 3.4.1.),
    - Ustawienie flagi świadczącej o tym, że osobnik jest już po ślubie. 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,
    - Przypisanie do wartości funkcji przystosowania średniej wartości funkcji przystosowania dwóch osobników z pary.

2. **Wkład poszczególnych autorów**
    a. Implementacja algorytmu ewolucyjnego z modyfikacją (śluby) – praca wspólna,
    b. Implementacja funkcji oceny – Wojciech Sitek,
    c. Implementacja testowania – Konrad Opaliński,
    d. Dokumentacja - praca wspólna,
    e. Podział wkładu pracy jest ogólny, ponieważ wszystkie zagadnienia były omawiane wspólnie i często udoskonalane przez drugą osobę.

3. **Ważne decyzje projektowe**
    a. 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 funkcji oceny CEC. Język Java wydawał się idealny do tego typu przedsięwzięcia z powodu subiektywnej łatwości implementacji.

b. Typ algorytmu ewolucyjnego – μ + λ
Opis algorytmu w punkcie 4.a.

c. Operatory ewolucyjne - mutacja
Opis mutacji w punkcie 4.b.

4. Opis algorytmów

a. Algorytm ewolucyjny (μ + λ):
   - Stworzenie losowej populacji μ osobników (wektorów liczb rzeczywistych),
   - Ślub (opcjonalnie),
   - Utworzenie tymczasowej populacji potomnej (liczebności λ osobników),
   - Mutacje na populacji tymczasowej,
   - Ponowne obliczenie funkcji przystosowania dla tymczasowej populacji potomnej,
   - Połączenie populacji rodzicielskiej i tymczasowej potomnej, wybór μ osobników o najlepszej funkcji przystosowania.

b. Mutacja:
   - Natężenie rozmywania
     $\sigma_i^R = \sigma_i^- exp(\tau N(0,1) + \tau' N_i(0,1))$,
     gdzie $\tau = \frac{1}{\sqrt{2d}}$, $\tau' = \frac{1}{\sqrt{2\sqrt{d}}}$ oraz d = dim(x).

   - Rozwiązanie
     $x_i^R = x_i^- + \sigma_i^R N_i(0,1)$

c. Łączenie w pary osobników:
   - Użycie funkcji Collections.shuffle() w celu otrzymania losowej permutacji osobników w populacji,
   - Dla całej populacji:
     - Poszukiwanie kolejnych dwóch osobników, którzy nie są jeszcze małżonkami,
     - Ślub tych osobników.

d. Utworzenie tymczasowej populacji potomnej (liczebności λ osobników):
Skorzystaliśmy z metody koła ruletki. Polega ona na losowym wybieraniu λ osobników z populacji rodzicielskiej z prawdopodobieństwem wybrania odwrotnie proporcjonalnym do wartości funkcji celu (zgodnie ze wzorem) w porównaniu z wartościami funkcji celu całej populacji rodzicielskiej.

$$p(i) = \frac{\frac{1}{f(i)}}{\sum_{n=1}^{count} \frac{1}{f(n)}}$$

e. Wybór populacji wynikowej z populacji rodzicielskiej i potomnej:
   - Połączenie populacji rodzicielskiej i potomnej w jedną,
   - Posortowanie populacji względem wartości funkcji przystosowania od najlepszego do najgorszego,
   - Usunięcie z populacji osobników (z końca listy), tak aby zostało μ osobników.

5. Lista wykorzystanych narzędzi i bibliotek

Wykorzystano standardowe biblioteki i narzędzia Javy.

6. Instrukcja pozwalająca na odtworzenie uzyskanych wyników (obsługa narzędzia testującego)

Wersja użytej Javy - Java SDK 11.

a. Sposób korzystający z terminala:
   - Należy przejść do katalogu src projektu oraz wpisać komendę:
     `$ javac evolutionary/*`
     To powinno skompilować projekt.
   - Następnie odpalamy projekt, a dokładnie narzędzie testujące poprzez wpisanie:
     `$ java evolutionary/Main`
     Po odpaleniu tej komendy powinny zacząć się testy. Wyniki testów zostaną zapisane w pliku test_info.txt w aktualnym katalogu.

b. Innym sposobem jest skorzystanie z VS Code, IntelliJ, czy Eclipse.

7. Funkcje oceny jakości

Korzystano z funkcji oceny jakości CEC2014, z tym zastrzeżeniem, że nie wykonywano operacji *rotation* oraz *shifting* w funkcjach kompozycji.

a. Funkcja Weierstrass'a (CEC2014 nr 1),

b. Funkcja kompozycji 1 (na podstawie funkcji CEC2014 numer 23 - bez rotacji i shiftingu):
   Składa się ona z następujących funkcji:
   - Rosenbrock's Function (CEC2014 nr 4)
   - High Conditioned Elliptic Function (CEC2014 nr 1)
   - Bent Cigar Function (CEC2014 nr 2)
   - Discus Function (CEC2014 nr 3)
   - High Conditioned Elliptic Function (CEC2014 nr 1)

c. Funkcja kompozycji 2 (na podstawie funkcji CEC2014 numer 27 - bez rotacji i shiftingu):
Składa się ona z następujących funkcji:
   - HGBat Function (CEC2014 nr 12)
   - Rastrigin's Function (CEC2014 nr 8)
   - Modified Schwefel's Function (CEC2014 nr 9)
   - Weierstrass Function (CEC2014 nr 6)
   - High Conditioned Elliptic Function (CEC 2014 nr 1)

8. **Cele i tezy przeprowadzonych badań**
Celem eksperymentu jest zbadanie, czy ślub oraz prawdopodobieństwo mutacji w algorytmie ewolucyjnym wpływa na jego optymalizację. Tezą badania jest, iż ślub nieznacznie zmienia wyniki działania algorytmu, niezależnie od wartości prawdopodobieństwa mutacji. Eksperymenty mają na celu potwierdzenie lub obalenie niniejszej tezy.

9. **Wyniki eksperymentów w postaci czytelnych tabel**
Dla każdej funkcji, każdego prawdopodobieństwa mutacji (spośród 0.1, 0.3, 0.5, 0.7) utworzono 25 losowych populacji o ilości osobników 1000, wielkości populacji potomnej 1500, wymiarze 8 wektora x, ilości wykonywania ewolucji 75.

Uznano, że te wielkości będą reprezentatywne po wykonanej wstępnej analizie i eksperymentach. Próbowano także uzyskać wyniki dla większej liczby wymiarów, lecz są one mniej przejrzyste oraz dla większej liczby powtórzeń algorytmu, ale wtedy algorytm znajduje to samo minimum, niezależnie czy wykonamy ślub, czy nie.

Tabela 1. Średnia wartości funkcji celu najlepszych osobników 25 populacji

	Weierstrass	**Composition 1**	**Composition 2**
mp=0.1, initial	8.47	310.66	419.88
mp=0.1, wedding	8.06	225.33	400.40
mp=0.1, no-wedding	7.11	225.45	400.63
mp=0.3, initial	8.64	287.16	422.21
mp=0.3, wedding	6.74	210.47	400.13
mp=0.3, no-wedding	6.61	210.10	400.18
mp=0.5, initial	8.78	299.89	421.83
mp=0.5, wedding	8.78	201.81	400.13
mp=0.5, no-wedding	8.78	203.74	400.11
mp=0.7, initial	8.70	297.14	422.92

mp=0.7, wedding	5.43	201.95	400.13
mp=0.7, no-wedding	6.13	200.59	400.13

Tabela 2. Najlepszy / Najgorszy osobnik z 25 populacji o danych parametrach

	Weierstrass	**Composition 1**	**Composition 2**
mp=0.1, initial	6.91 / 24.21	250.43 / 6.64E9	404.42 / 486517.77
mp=0.1, wedding	6.60 / 8.99	120.95 / 310.17	400.03 / 402.59
mp=0.1, no-wedding	5.40 / 7.98	119.11 / 298.55	400.01 / 403.91
mp=0.3, initial	7.41 / 24.56	215.37 / 7.08E9	404.55 / 497885.00
mp=0.3, wedding	4.10 / 8.08	112.92 / 289.57	400.01 / 401.00
mp=0.3, no-wedding	5.33 / 7.54	108.04 / 289.38	400.01 / 401.00
mp=0.5, initial	7.01 / 24.47	231.50 / 1.33E10	404.18 / 466854.81
mp=0.5, wedding	7.01 / 24.47	137.88 / 257.01	400.01 / 400.77
mp=0.5, no-wedding	7.01 / 24.47	137.87 / 257.02	400.01 / 400.48
mp=0.7, initial	6.71 / 25.63	250.84 / 1.14E10	404.67 / 458966.61
mp=0.7, wedding	1.44 / 8.06	111.02 / 272.49	400.00 / 400.58
mp=0.7, no-wedding	4.92 / 7.17	110.25 / 272.48	400.00 / 400.55

10. Omówienie wyników eksperymentów

Dla wykonanych eksperymentów zauważono następujące zależności:

a. Dla większych wartości prawdopodobieństwa mutacji, zazwyczaj algorytm znajduje lepsze minima,
b. Nie ma znaczących rozbieżności pomiędzy eksperymentami na tych samych populacjach bez ślubu i ze ślubem, w niektórych przypadkach nawet eksperymenty ze ślubem wskazują nieco większą funkcję celu,
c. Niektóre wektory najlepszych osobników wynikowej populacji bez ślubu i ze ślubem znajdują minimum o bardzo podobnej wartości, ale zupełnie inaczej usytuowane w hiperprzestrzeni (odbiegające od siebie składowe kolejnych wymiarów wektora x),
d. Dla funkcji Weierstrass'a ze ślubem jest większa różnica pomiędzy elementami o najmniejszej i największej wartości funkcji celu.

11. Wnioski

Dla wykonanych eksperymentów wnioski są następujące:

a. Wykonanie algorytmu ze ślubem daje lepsze wyniki tylko w niektórych, specyficznych sytuacjach. Zazwyczaj algorytm daje bardzo podobne rezultaty z i bez ślubu,
b. Na przebieg algorytmu ma bardzo duży wpływ prawdopodobieństwo mutacji. Najlepsze wyniki uzyskujemy dla prawdopodobieństwa mutacji równej 0.3 oraz 0.7.