Algorytmy ewolucyjne

Piotr Lipiński

Lista zadań nr 0 – wprowadzenie

Zadanie 1. *(2 punkty)*
Zapoznaj się z biblioteką TSPLib [1], w której zgromadzono popularne instancje problemu TSP wykorzystywane do ewaluacji algorytmów optymalizacji.

1. Wybierz jeden z problemów TSP rozmiaru co najmniej 25, na przykład Berlin52, narysuj graf, narysuj optymalną trasę, narysuj kilka losowych tras (losowanych według własnego pomysłu).

2. Wygeneruj milion losowych tras (losowanych według własnego pomysłu). Oblicz ich koszty. Porównaj z kosztem trasy optymalnej. Jakie były różnice kosztów między trasami losowymi a trasą optymalną? Ile z losowych tras było gorszych o mniej niż 10% od trasy optymalnej? A o 20%? A o 30? itd. Zrób rysunek prezentujący wyniki (według własnego pomysłu).

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Zadanie 2. *(4 punkty)*
1. Napisz prosty algorytm Hill Climbing dla problemu TSP:

   (A) ustaw rozwiązanie początkowe p - losową permutację długości n, gdzie n jest rozmiarem problemu TSP
   (B) rozpatrz sąsiedztwo permutacji p – wszystkie permutacje q różniące się od p na nie więcej niż m pozycjach (m jest parametrem algorytmu, zazwyczaj dość małym ze względów obliczeniowych) – spośród rozpatrywanych permutacji q wybierz tę o najniższym koszcie
   (C) jeśli p <> q, to p := q i przejdź do kroku (B)
   (D) zwróć permutację p jako wynik działania algorytmu

2. Zauważ, że algorytm Hill Climbing jest nie jest deterministyczny. Jaki wpływ na jego działanie ma wybór początkowego rozwiązania p? Wygeneruj milion permutacji używając algorytmu Hill Climbing (albo mniej, jeśli obliczenia miałyby być bardzo czasochłonne). Oblicz ich koszty. Porównaj z kosztem trasy optymalnej i kosztami tras z Zadania 1.2. Jakie były różnice kosztów między trasami losowymi a trasami z Hill Climbing? Zrób rysunek prezentujący wyniki (według własnego pomysłu).

3. Przyjrzyj się działaniu algorytmu. Zrób wykres kosztu rozwiązania p w kolejnych iteracjach algorytmu. Czy warto algorytm Hill Climbing uruchamiać wielokrotnie? Zrób wykres liczby iteracji po których algorytm kończy działanie (dla miliona uruchomień). Porównaj z liczbę iteracji działania algorytmu z kosztami otrzymywanych rozwiązań. Co zrobić, żeby algorytm nie kończył działania, jeśli znalezione rozwiązanie jest słabe? Spróbuj udoskonalić algorytm Hill Climbing.