Zadanie 2. Lokalne przeszukiwanie

Oskar Kiliańczyk 151863 & Wojciech Kot 151876

1 Opis zadania

Zadanie polega na implementacji lokalnego przeszukiwania w wersjach stromej (steepest) i zachłannej (greedy), z dwoma różnym rodzajami sąsiedztwa, startując albo z rozwiązań losowych, albo z rozwiązań uzyskanych za pomocą jednej z heurystyk opracowanych w ramach poprzedniego zadania. W sumie 8 kombinacji — wersji lokalnego przeszukiwania. Jako punkt odniesienia należy zaimplementować algorytm losowego błądzenia, który w każdej iteracji wykonuje losowo wybrany ruch (niezależnie od jego oceny) i zwraca najlepsze znalezione w ten sposób rozwiązanie. Algorytm ten powinien działać w takim samym czasie jak średnio najwolniejsza z wersji lokalnego przeszukiwania.

1.1 Sąsiedztwa

W przypadku rozważanego problemu potrzebne będą dwa typy ruchów:

- ruchy zmieniające zbiory wierzchołków tworzące dwa cykle,
- ruchy wewnątrztrasowe, które jedynie zmieniają kolejność wierzchołków na trasie.

Stosujemy dwa rodzaje ruchów wewnątrztrasowych (jeden albo drugi, stąd dwa rodzaje sąsiedztwa). jeden to wymiana dwóch wierzchołków wchodzących w skład trasy, drugi to wymiana dwóch krawędzi.

1.2 Randomizacja kolejności przeglądania dla algorytmów zachłannych

W obu wersjach algorytmów zachłannych stosujemy randomizację wyboru. Dla ruchów wewnątrz cykli wykonujemy lokalne zamiany elementów w obrębie jednej ścieżki. Randomizacja polega na losowym przetasowaniu indeksów w obrębie dostępnych miast, co zmienia kolejność, w jakiej będą przetwarzane. W przypadku algorytmu wykorzystującego zmiany pomiędzy ścieżkami zmiana odbywa się pomiędzy dwoma cyklami. Tworzymy listę możliwych par punktów do zamiany między dwoma cyklami. Losowo przetasowujemy te możliwe pary, dzięki czemu kolejność przetwarzania nie jest deterministyczna. Losowo wybieramy jedną parę do wymiany (w której obliczenie zmiany w funkcji celu daje poprawę) i przeprowadzamy zamianę.

2 Opisy algorytmów

2.1 Zmiany wewnątrz cyklu

2.1.1 Zachłanny

2.1.2 Steepest

2.2 Zmiany pomiędzy cyklami

2.2.1 Zachłanny

2.2.2 Steepest

3 Wyniki

3.1 Tabela wynikowa

3.2 Wizualizacja wyników

Rysunek 1: Instancja kroA200 - zmiany wewnątrz cyklu - zachłanny, start losowy

Rysunek 2: Instancja kroA200 - zmiany wewnątrz cyklu - steepest, start heurystyczny

Rysunek 3: Instancja kroA200 - zmiany pomiędzy cyklami - zachłanny, start losowy

Rysunek 4: Instancja kroA200 - zmiany pomiędzy cyklami - steepest, start heurystyczny

Rysunek 5: Instancja kroA200 - losowa zmiana

Rysunek 6: Instancja kroA200 - zmiany wewnątrz cyklu - zachłanny, start heurystyczny

Rysunek 7: Instancja kroA200 - zmiany wewnątrz cyklu - steepest, start losowy

Rysunek 8: Instancja kroA200 - zmiany pomiędzy cyklami - zachłanny, start heurystyczny

4 Link do repozytorium

Kod źródłowy w repozytorium GitHub dostępny pod linkiem:
Repozytorium TSP Heuristics.