AEM - Zadanie nr 3

Bartosz Sobkowiak 125342 Joanna Świda 138675

27.04.2020

1 Opis zadania

Rozważany problem to zmodyfikowana wersja problemu komiwojażera. Dany jest zbiór wierzchołków i macierz symetrycznych odległości między nimi. Zadanie polega na implementacji lokalnego przeszukiwania w wersji stromej (steepest) z ruchem wymiany krawędzi.Stosujemy dwa mechanizmy: Wykorzystanie ocen ruchów z poprzednich iteracji z uporządkowaną listą ruchów; Ruchy kandydackie. Celem jest poprawa efektywności czasowej lokalnego przeszukiwania.

2 Pseudokod

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie S
wyznacz elementy (wierzchołki) które nie znajdują się w rozwiązaniu
while *dopóki generowane są ruchy kandydackie* **do**
  for *dla par: każdego punktu w rozwiązaniu S i każdego punktu poza rozwiązaniem* **do**
    oblicz deltę po operacji podmiany ww. punktów
    IF delta $<$ 0:
    dodaj tą podmianę (ruch) do listy candidates$_{m}oves_{l}ist(listyruchwkandydackich)$
  end
  for *dla par: każdej krawędzi w rozwiązaniu i najbliższych punktów krawędzi* **do**
    zbiór wszystkich mozliwych wymian krawędzi ograniczamy tylko do takich krawędzi w których jeden wierzchołek jest najblizszym sasiadem drugiego
    IF punkt w rozwiązaniu S:
    oblicz deltę dla zamienionych krawędzi
    IF delta $<$ 0:
    dodaj tą podmianę (ruch) do listy candidatesMovesList (listy ruchów kandydackich)
  end
  posortuj candidatesMovesList
  jeśli znaleziono lepszą podmianę to zamień krawędzie wskazane przez tą podmianę
end

Algorithm 1: Local search with candidates moves

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie
wyznacz elementy (wierzchołki) które nie znajdują się w rozwiązaniu
wyznacz listę możliwych ruchów
while *lista ruchów jest niepusta* **do**
  for *dla każdego ruchu z listy* **do**
    Zawsze sprawdź czy własnie wykonane ruchy dają poprawę, jesli tak to wykonuj podmianę, jeśli to tylko dodawaj do listy movesToRemove Jeśli zamiana wierzchołków: zamień wierzchołki (punkty) z rozwiazania z tymi spoza rozwiązania
    dodaj ten ruch do movesToRemove
    Jeśli zamiana krawędzi: zamień krawędzie jeśli jest to możliwe (tj. jeśli nie są sąsiednie i można je wymienić)
    dodaj ten ruch do movesToRemove
  end
  usuń ruchy z listy movesToRemove
end

Algorithm 2: Local search with cache

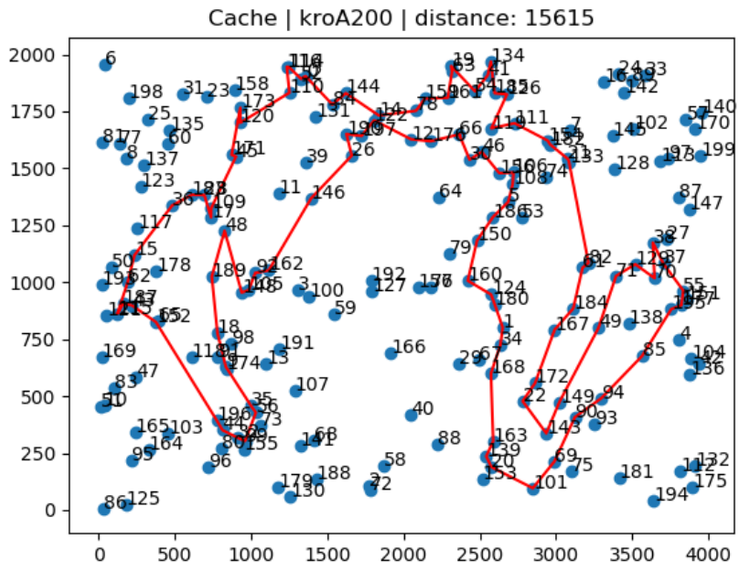
3 Wyniki obliczeń i wizualizacje

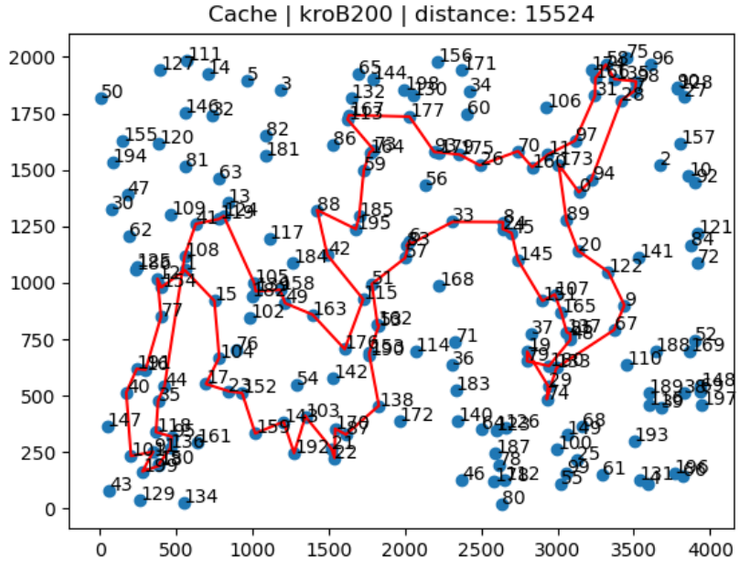
Zbiór	Wersja	Typ	Min	Avg	Max
kroA_{200}	LocalSearch	Normal	15315	17084	19228
kroA_{200}	LocalSearch	Cache	15615	17783	20138
kroA_{200}	LocalSearch	CandidateMoves	16156	17995	21605
kroB_{200}	LocalSearch	Normal	14265	17731	22737
kroB_{200}	LocalSearch	Cache	15524	18007	20788
kroB_{200}	LocalSearch	CandidateMoves	15582	18092	21931

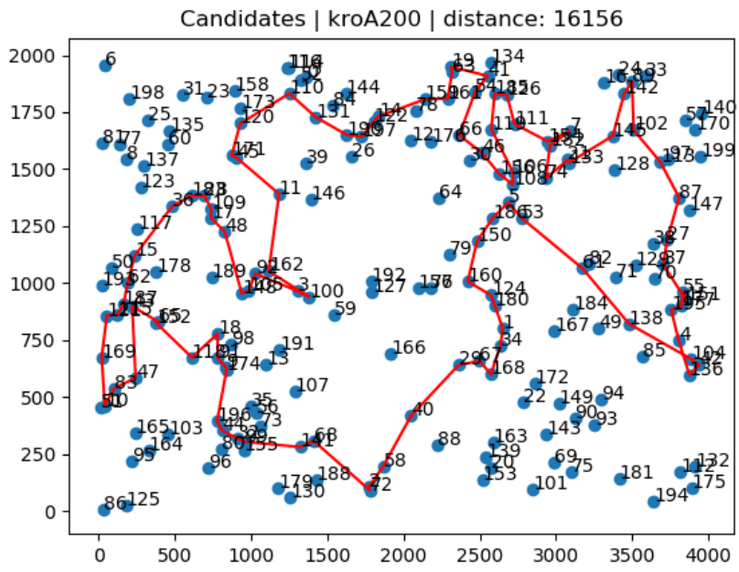
Tabela 1: Wartości rozwiązań

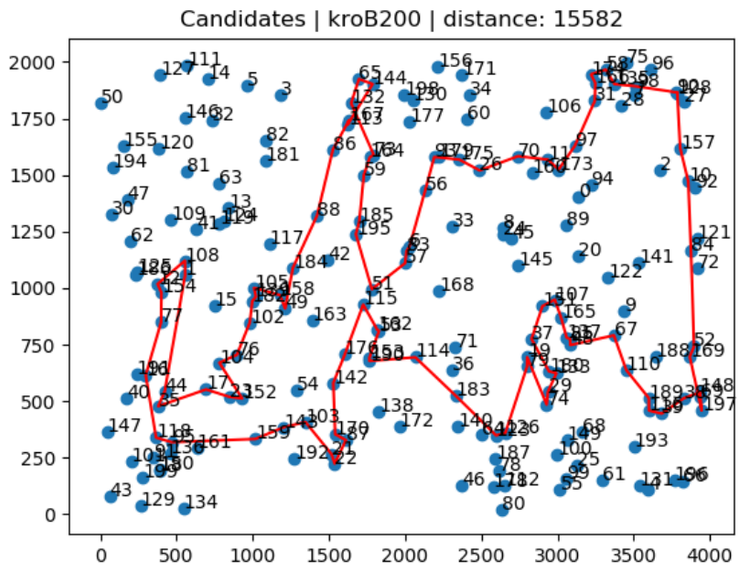
Zbiór	Wersja	Typ	Min	Avg	Max
kroA_{200}	LocalSearch	Normal	1.9963	3.0339	5.2151
kroA_{200}	LocalSearch	Cache	1.4389	1.8165	3.2346
kroA_{200}	LocalSearch	CandidateMoves	1.8991	2.8971	5.9477
kroB_{200}	LocalSearch	Normal	2.0824	3.2177	5.2551
kroB_{200}	LocalSearch	Cache	1.5047	1.8342	3.0707
kroB_{200}	LocalSearch	CandidateMoves	1.8912	2.8222	4.9185

Tabela 2: Czasy trwania









4 Wnioski

Wykorzystanie oceny ruchów z poprzednich iteracji, czyli cache'owania znacząco przyspieszania działanie algorytmu - czasy iteracji są zmniejszone niemal dwukrotnie, co jest znaczącym wynikiem. Jakość rozwiązania jest porównywalna, czego niestety nie można powiedzieć o ruchach kandydackich, gdzie pogorszyło się ono o kilkanaście procent. Niemniej, cel został osiągnięty, gdyż czas wykonania jednej iteracji jest zauważalnie niższy. Prawdopodobnie jeszcze lepszy czas można by osiągnąć po optymalizacji kodu. Zatem cache'owanie rozwiązań w naszym przypadku jest najlepszą metodą rozwiązania tego zadania.

5 Kod programu

Repozytorium z kodem algorytmów dostępne jest pod: `https://github.com/bbbrtk/aem`