AEM - Zadanie nr 2

Bartosz Sobkowiak 125342 Joanna Świda 138675

07.04.2020

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. Lokalne przeszukiwanie my by zaimplementowane w wersji stromej i zachłannej z dwoma różnym rodzajami sąsiedztwa.

2 Pseudokod

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie
wyznacz elementy (wierzchołki) które nie znajdują się w rozwiązaniu
while *dopóki znalezione rozwiązania są lepsze* **do**
  typu wyjściowego **for** *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    oblicz deltę po operacji opdmiany wirzechołków
    jeśli delta jest mniejsza od 0, wstaw do obecnego rozwiązania element z listy elementów nie znajdujących się w rozwiązaniu, wedle indeksów wyznaczonych przez parę
  end
  typu wejściowego **for** *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    oblicz deltę dla zamienionych wierzochołków
    jeśli mniejsza od zera, to zamień wierzchołki w obecnym rozwiązaniu
  end
  jeśli zmienione rozwiazanie jest lepsze: wybierz je jako najlepsze
end
dodaj do listy rozwiązań

Algorithm 1: Greedy - vertex

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie
wyznacz elementy (wierzchołki) które nie znajdują się w rozwiązaniu
while *dopóki znalezione rozwiązania są lepsze* **do**
  typu wyjściowego **for** *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    oblicz deltę po operacji opdmiany wirzechołków
    jeśli delta jest mniejsza od najlepszej jak dotąd znalezionej delty: przypisz nową wartość najlepszej delty, wstaw do obecnego rozwiązania element z listy elementów nie znajdujących się w rozwiązaniu, wedle indeksów wyznaczonych przez parę
  end
  typu wejściowego **for** *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    oblicz deltę dla zamienionych wierzochołków
    jeśli delta jest mniejsza od najlepszej jak dotąd znalezionej delty: przypisz nową wartość najlepszej delty, to zamień wierzchołki w obecnym rozwiązaniu
  end
  jeśli zmienione rozwiazanie jest lepsze: wybierz je jako najlepsze
end
dodaj do listy rozwiązań

Algorithm 2: Steepest - vertex

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie
while *dopóki znalezione rozwiązania są lepsze* **do**
  for *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    podmień krawędzie według indeksów
    oblicz deltę dla zamienionych krawędzi
    jeśli delta jest mniejsza od zera, to podmień w obecnym rozwiązaniu
  end
  jeśli zmienione rozwiazanie jest lepsze: wybierz je jako najlepsze
end
dodaj do listy rozwiązań

Algorithm 3: Greedy - edges

Data: zbiór wierchołków, macierz odległości pomiędzy wierzchołkami
Result: najlepsze rozwiązanie
wygeneruj losowe rozwiązanie
while *dopóki znalezione rozwiązania są lepsze* **do**
  for *dla każdej pary indeksów w zakresie 0-49 w losowej kolejności* **do**
    podmień krawędzie według indeksów
    oblicz deltę dla zamienionych krawędzi
    jeśli delta jest mniejsza od najlepszej jak dotąd znalezionej delty: przypisz nową wartość najlepszej delty, to podmień w obecnym rozwiązaniu
  end
  jeśli zmienione rozwiazanie jest lepsze: wybierz je jako najlepsze
end
dodaj do listy rozwiązań

Algorithm 4: Steepest - edges

3 Wyniki obliczeń i wizualizacje

Zbiór	Wersja	Sąsiedztwo	Min	Avg	Max
kroA_{100}	Greedy	Wierzchołki	14265	17731	22737
kroA_{100}	Greedy	Krawędzie	11657	12744	14762
kroB_{100}	Greedy	Wierzchołki	13968	17413	22414
kroB_{100}	Greedy	Krawędzie	11522	12705	14318

Tabela 1: Wartości rozwiązań dla algorytmu Greedy

Zbiór	Wersja	Sąsiedztwo	Min	Avg	Max
kroA_{100}	Steepest	Wierzchołki	0.1613	0.3202	0.4985
kroA_{100}	Steepest	Krawędzie	0.0924	0.1441	0.2291
kroB_{100}	Steepest	Wierzchołki	0.1839	0.3249	0.5958
kroB_{100}	Steepest	Krawędzie	0.0753	0.1329	0.1827

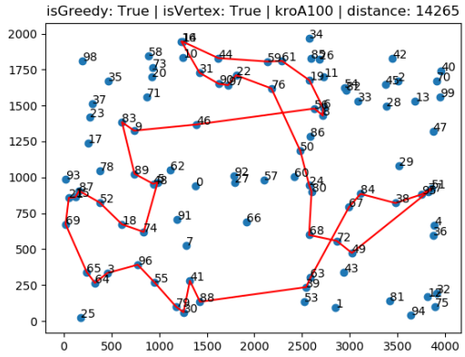
Tabela 2: Czasy trwania dla algorytmu Greedy

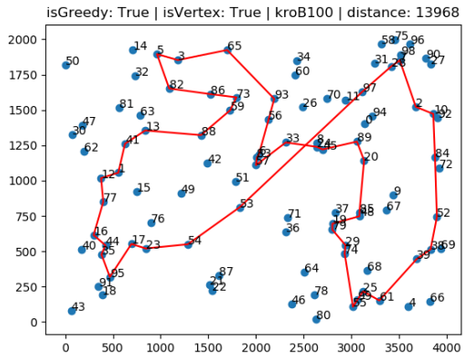
Zbiór	Wersja	Sąsiedztwo	Min	Avg	Max
kroA_{100}	Steepest	Wierzchołki	11845	17736	22527
kroA_{100}	Steepest	Krawędzie	11147	12676	15291
kroB_{100}	Steepest	Wierzchołki	15084	17793	21609
kroB_{100}	Steepest	Krawędzie	11322	12372	13434

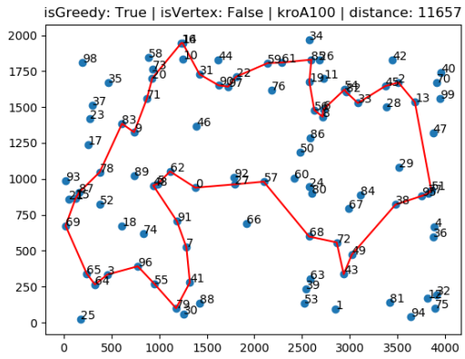
Tabela 3: Wartości rozwiązań dla algorytmu Steepest

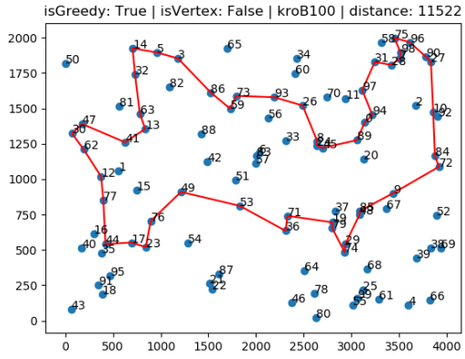
Zbiór	Wersja	Sąsiedztwo	Min	Avg	Max
kroA_{100}	Steepest	Wierzchołki	0.2281	0.3567	0.7047
kroA_{100}	Steepest	Krawędzie	0.1987	0.3554	1.3037
kroB_{100}	Steepest	Wierzchołki	0.2504	0.3887	0.69448
kroB_{100}	Steepest	Krawędzie	0.1697	0.3078	1.1066

Tabela 4: Czasy trwania dla algorytmu Steepest





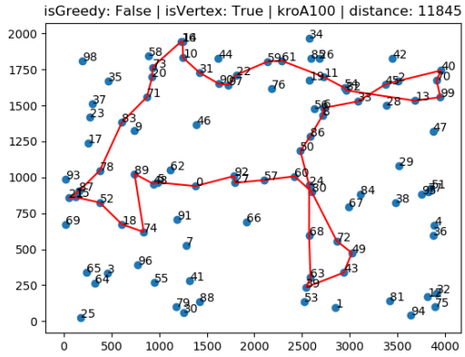


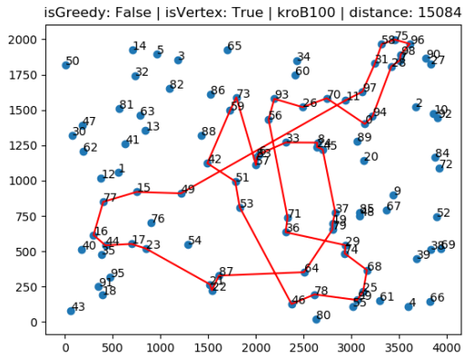


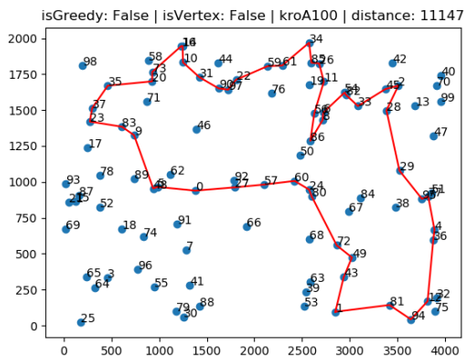
4 Wnioski

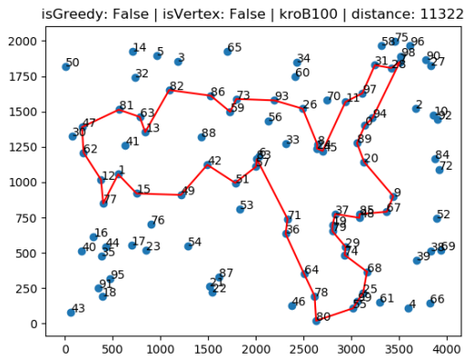
4.1 Randomizacja

Zgodnie z założeniami zadania zastosowano randomizacje kolejności przeszukiwania. Indeksy krawędzi oraz indeksy punktów były generowane w kolejności pseudolosowej z wykorzystaniem pakietu numpy. W wersji zachłannej krawędzie do wstawienia i indeksy miejsca wstawienia były również generowane w losowej kolejności i zestawiane były w pary, również w kolejności losowej. Algorytm iterował po tej liście par, dzięki czemu kolejność przeglądania za każdym razem była inna.









4.2 Wnioski do problemu

Zgodnie z wiedzą poznaną podczas analizy problemu, wymiana krawędzi w problemach podobnych do problemu komiwojażera przy generowaniu losowym daje wyniki lepsze niż wymiana wierzchołków. Przy zmianie wierzchołków następuje zazwyczaj więcej przecięć trasy, co wpływa na większą długość ścieżki. Oczywiście w wersji stromej, gdy szukane jest najlepsze rozwiązanie algorytm może niekiedy dawać wyniki zbliżone do sytuacji gdy rozważana jest wymiana krawędzi. Jednak jest to obarczone dłuższym czasem obliczeń. Przy wymianie krawędzi, dla algorytmu stromego i zachłannego wyniki można uznać za porównywalne. Dla wymiany wierzchołków algorytm stromy radzi sobie w ogólności nieco lepiej. Większy czas trwania obliczeń dla wersji stromej wynika z tego, że przeszukiwanie trwa znacznie dłużej, gdyż algorytm musi przeszukać więcej rozwiązań by znaleźć to najlepsze.

5 Kod programu

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