AEM - Poprawa efektywności czasowej lokalnego przeszukiwania

Dawid Białek 131731, Bartosz Mila 131804

07 maja 2020

1. Krótki opis zadania.

Celem jest poprawa efektywności czasowej 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.

Mechanizmy te stosujemy oddzielnie, czyli implementujemy dwie różne wersje lokalnego przeszukiwania. Opcjonalnie można zaimplementować trzecią wersję łączącą oba te mechanizmy.

Jako kandydackie stosujemy ruchy wprowadzające do rozwiązania co najmniej jedną krawędź kandydacką. Krawędzie kandydackie definiujemy wyznaczając dla każdego wierzchołka 5 innych najbliższych wierzchołków.

Stosujemy większe instancje kroa200 i krob200.

Jako punkt odniesienia uruchamiamy też lokalne przeszukiwanie w wersji stromej bez powyższych mechanizmów.

Każdy z trzech algorytmów na każdej instancji uruchamiany 100 razy startując z rozwiązań losowych i raportujemy wartości min, max i średnią dla funkcji celu i czasu. Wszystkie wyniki zbieramy w jednej tabeli.

Poza tym zawartość sprawozdania podobnie jak poprzednio. Pseudokody algorytmów, wyniki liczbowe i wizualizacje najlepszych rozwiązań, wnioski, link do kodu.

2. Opis zaimplementowanych algorytmów w pseudokodzie.

Lokalne przeszukiwanie w wersji stromej z ruchem wymiany krawędzi **bez** wykorzystania ocen ruchów z poprzednich iteracji / ruchów kandydackich:

- Utwórz losową ścieżkę między 100 punktami.
- Dopóki znajdujesz rozwiązania zmniejszające długość ścieżki:
    - Dla każdej krawędzi ze ścieżki:

            - Dla każdej innej krawędzi ze ścieżki:
                - Sprawdź koszt wymiany tych dwóch krawędzi.
        - Dla każdego wierzchołka ze ścieżki:
            - Dla każdego wierzchołka spoza ścieżki:
                - Sprawdź koszt wymiany tych dwóch wierzchołków.
        - Wprowadź najlepsze znalezione rozwiązanie, tzn. zamień dwie odpowiednie krawędzie lub wymień dwa odpowiednie punkty.

Lokalne przeszukiwanie w wersji stromej z ruchem wymiany krawędzi z wykorzystaniem ocen ruchów z poprzednich iteracji:

- Utwórz losową ścieżkę między 100 punktami.
- Utwórz pustą listę typu SortedList i wypełnij ją wszystkimi możliwymi wymianami zmniejszającymi długość ścieżki (pomiędzy wszystkimi możliwymi krawędziami albo pomiędzy wszystkimi wierzchołkami ze ścieżki z tymi spoza niej).
- Dopóki znajdujesz rozwiązania zmniejszające długość ścieżki:
    - Weź najlepsze z rozwiązań i sprawdź jego poprawność / możliwość wdrożenia.
    - Jeśli rozwiązanie jest prawidłowe - wprowadź je, następnie usuń z listy wszystkie powiązane z wymienionymi elementami rozwiązania oraz dopisz wszystkie możliwe nowe (dla tych elementów).
    - Jeśli rozwiązanie posiada zapisany błędny zysk wymiany lecz jest możliwe do wdrożenia - zaktualizuj zysk. Jeśli jest ujemny / równy zero - usuń rozwiązanie z listy.
    - Jeśli rozwiązanie jest niemożliwe do wdrożenia - usuń z listy wszystkie powiązane rozwiązania (ze znalezionym włącznie) oraz dopisz wszystkie możliwe nowe (dla elementów tego rozwiązania).

Lokalne przeszukiwanie w wersji stromej z ruchem wymiany krawędzi z wykorzystaniem ruchów kandydackich:

- Utwórz losową ścieżkę między 100 punktami.
- Dopóki znajdujesz rozwiązania zmniejszające długość ścieżki:
    - Dla każdej krawędzi ze ścieżki:
        - Dla 5 krawędzi kandydackich (zbudowanych na 5 najbliższych wierzchołkach ze ścieżki):
            - Sprawdź koszt wymiany krawędzi.
    - Dla każdego wierzchołka ze ścieżki:
        - Dla 5 wierzchołków kandydackich (5 najbliższych wierzchołków spoza ścieżki):
            - Sprawdź koszt wymiany wierzchołków.
    - Wprowadź najlepsze znalezione rozwiązanie, tzn. zamień dwie odpowiednie krawędzie lub wymień dwa odpowiednie punkty.

3. Wyniki eksperymentu obliczeniowego.

Instancja A	Wersja podstawowa	Ruchy z poprzedniej iteracji	Ruchy kandydackie
Najkrótsza ścieżka	14985	20875	15981
Średnia ścieżka	16994	23750	17960
Najdłuższa ścieżka	19500	26744	20642
Minimalny czas [s]	6.760	0.571	1.992
Średni czas [s]	7.725	0.654	2.218
Maksymalny czas [s]	9.085	0.762	2.627

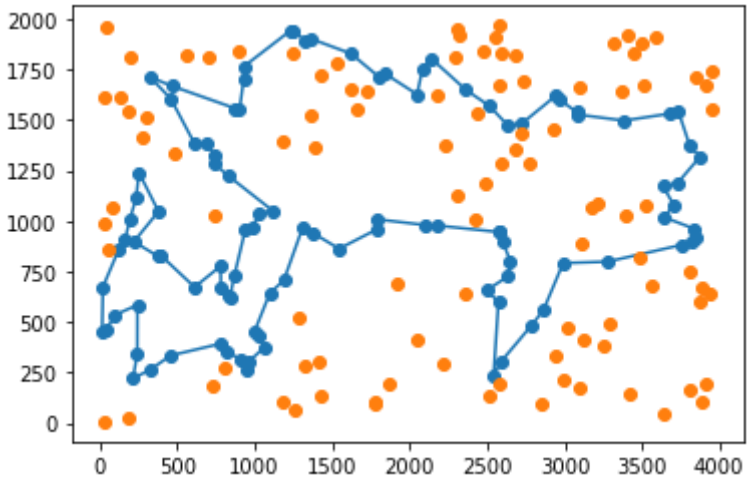
Tabela 1. Wyniki długości ścieżek i czasu działania na wykonywanych algorytmów dla zbioru A

Instancja B	Wersja podstawowa	Ruchy z poprzedniej iteracji	Ruchy kandydackie
Najkrótsza ścieżka	14738	20163	16136
Średnia ścieżka	17053	23645	17716
Najdłuższa ścieżka	18830	27555	20831
Minimalny czas [s]	6.646	0.581	1.969
Średni czas [s]	7.841	0.680	2.220
Maksymalny czas [s]	9.141	0.796	2.512

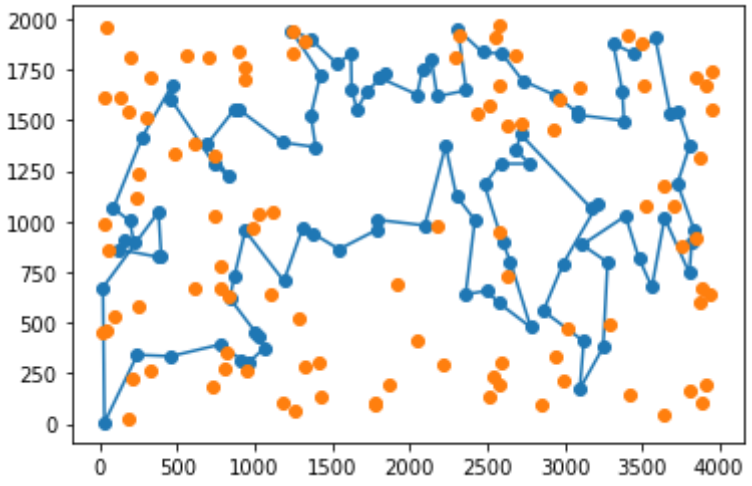
Tabela 2. Wyniki długości ścieżek i czasu działania na wykonywanych algorytmów dla zbioru B

4. Wizualizacje najlepszych rozwiązań dla każdej kombinacji.

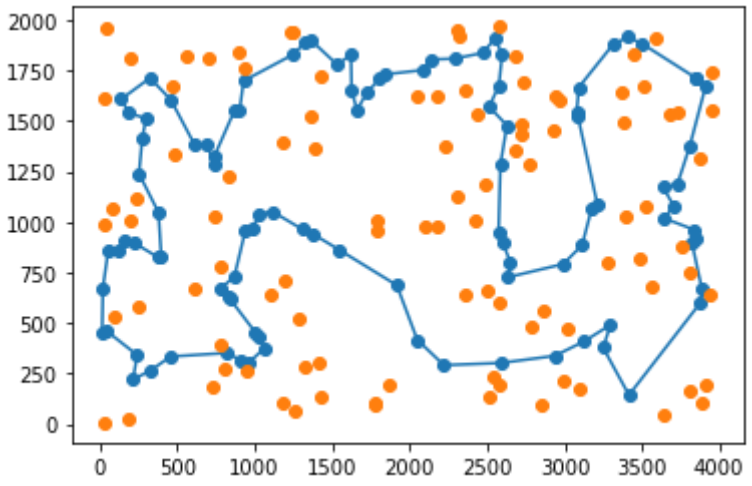
Instancja A:



Rysunek 1. Wizualizacja dla wersji podstawowej

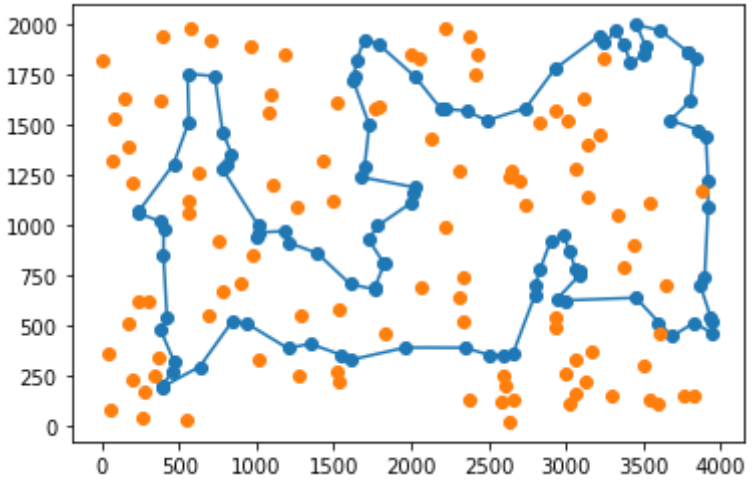


Rysunek 2. Wizualizacja dla ruchów z poprzedniej iteracji

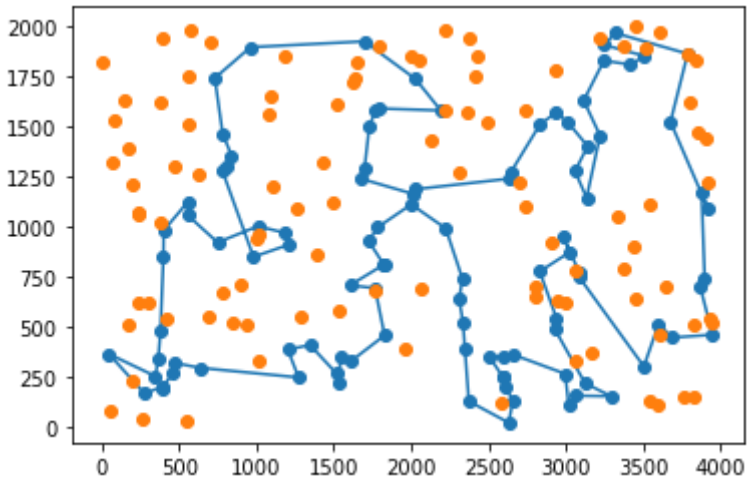


Rysunek 3. Wizualizacja dla ruchu kandydackiego

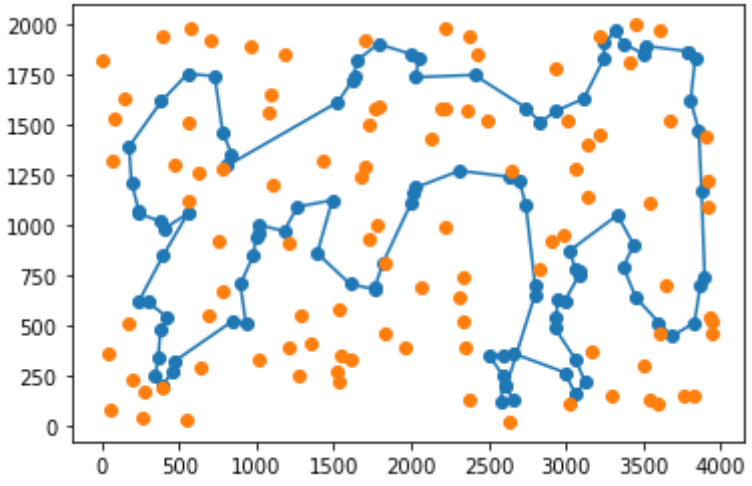
Instancja B:



Rysunek 4. Wizualizacja dla wersji podstawowej



Rysunek 5. Wizualizacja dla ruchów z poprzedniej iteracji



Rysunek 6. Wizualizacja dla ruchu kandydackiego

6. Wnioski.

Algorytm kandydacki względem algorytmu podstawowego zwraca trochę gorsze wyniki długości ścieżek przy znacznej poprawie czasu działania. Algorytm z poprzedniej iteracji jest w porównaniu do algorytmu kandydackiego jeszcze szybszy, lecz przy równoczesnym znacznym wydłużeniu średniej długości ścieżek względem dwóch poprzednich algorytmów. Podsumowując, algorytm kandydacki sprawdzi się jeśli zależy nam na dobrych wynikach długości trasy, natomiast algorytm z poprzedniej iteracji sprawdzi się kiedy zależy użytkownikowi bardziej na prędkości działania niż na dokładności zwracanego rozwiązania.

7. Kod programu (np. w postaci linku).

https://github.com/Kurkum/AEM