ALHE - Raport z projektu

Kaczmarek Kamil, Lewczuk Grzegorz

20 stycznia 2018

Streszczenie

Analiza problemu utworzenia sieci autostrad pomiędzy miastami oraz implementacja algorytmu optymalizującego rozwiązanie

1 Opis zagadnienia

Przygotować algorytm poszukujący optymalnej sieci autostrad tworzącą siatkę połączeń pomiędzy miastami z danego zbioru - położenia miast są nam znane. Rozwiązanie powinno uwzględniać miejsca zjazdów (nie mogą one znajdować się zbyt blisko siebie) oraz pozwalać na przecinanie się autostrad nie tylko w miastach.

2 Zmiany dotyczące założeń

Następiły następujące zmiany w założeniach projektu w celu poprawności działania implementacji problemu:

- Przestrzeń została ograniczona do kwadratu o wymiarach 100x100 – współrzędne x,y z zakresu [0, 100).
- Współrzędne miast i wierzchołki łamanej budującej autostradę są liczbami całkowitymi, natomiast wyliczane punkty zjazdu mogą być punktami o współrzędnych rzeczywistych.
- Losowy sąsiad dla punktów z przestrzeni poszukiwań jest wyznaczany poprzez wylosowanie jednego z wierzchołków autostrady. Następnie losowane są dla niego nowe współrzędne x, y oraz parametr w, który oznacza numer wierzchołka autostrady, z którym jest połączony dany punkt.
- Funkcja kosztu jest obliczana zgodnie założeniami dokumentacji wstępnej. Doprecyzowaniu uległa jedynie kara za niezachowanie spójności autostrady, która została ustalona jako 1e100.

3 Implementacja

Do implementacji projektu użyto języka Python 3.6 wraz z dodatkowymi bibliotekami:

- matplotlib - wyświetlanie wyników w formia graficznej na wykresie.
- numpy - wykonywanie obliczeń i przekształceń na wektorach.
- argparse - przygotowanie argumentów wprowadzanych podczas uruchomienia programu.
- simanneal - zewnętrzna implementacja algorytmu symulowanego wyrzażania, opisanego poniżej.

3.1 Model

Utworzono klasę modelu, zawierającą atrybuty:

- listę miast, z której łatwo także wyznaczć liczbę miast M,
- listę punktów autostrady, z której łatwo wyznaczyć liczbę punktów autostrady K,
- wartość minimalnej odległości pomiędzy zjazdami,
- pomocniczą listę punktów zjazdów z autostrady.

Dodatkowo w modelu zdefiniowane są metody pozwalające na znalezienie sąsiada oraz obliczenia funkcji kosztu używane w algorytmie symulowanego wyrzażania. Oprócz tego są także metody pomocnicze.

3.2 Uruchamianie

Dzięki modułowi argparse, zdefiniowano interfejs użytkownika pozwalający na dodanie w prosty sposób argumentów wywołania programu. Plik główny projektu, to `main.py`. Dodano do niego linijkę `#!/usr/bin/env python`, aby był uruchamialny z poziomu powłoki.

Po wpisaniu `./main.py -h` mozna uzyskać pomoc dotyczącą działania programu:

```
usage: main.py [-h] [-s STEPS] [--show] cities k d

Highway system

positional arguments:
  cities                filename with cities positions, "random" for random
                        cities
  k                     number of highway points
  d                     minimal distance between exits

optional arguments:
  -h, --help            show this help message and exit
  -s STEPS, --steps STEPS
                        steps to perform
  --show                show model through iterations
```

Obowiązkowe jest podanie nazwy pliku zawierającego współrzędne miast, liczby k punktów autostrad oraz wartości d minimalnej odległości pomiędzy zjazdami.
Dodatkowo można podać liczbę kroków algorytmu STEPS oraz ustawić flagę `-show`, dzięki której można śledzić wyniki na bieżąco na wykresie.
Istnieje także możliwość wywołania programu dla losowych miast, poprzez podanie w miejscu nazwy pliku z miastami słowa 'random'.

3.3 Plik konfiguracyjny

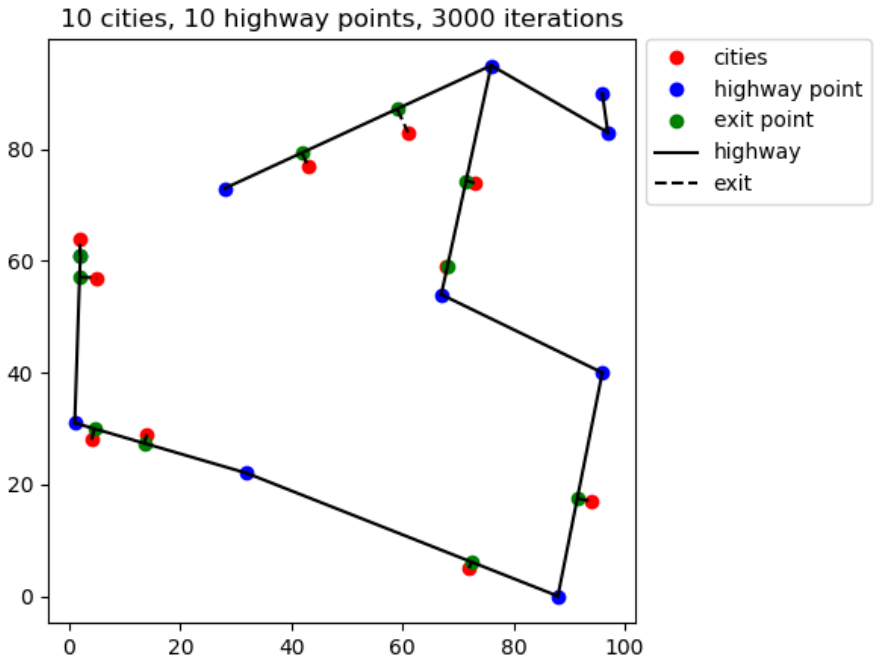
Program zawiera także plik konfiguracyjny określający wartości:

- domylna ilość miast - w przypadku tworzenia losowych współrzędnych.
- maksymalne wartości współrzędnych X oraz Y.
- ustawienia dotyczące algorytmu symulowanego wyrzażania.
- wzór funkcji liniowej oraz wykładniczej kosztu.

3.4 Wizualizacja wyników

Do wizualizacji wyników uzyto biblioteki matplotlib. Przykładowy wykres wygląda następująco. Wszystkie oznaczenia oraz ustawienia opisane są na wykresie.

Dzięki temu możliwe jest w prosty sposób zmiana działania całego programu bez ingerencji w kod.



Rysunek 1: Wizualizacja wyników

4 Metaheurystyka

Do minimalizacji funkcji kosztu autostrady zastosowano metodę optymalizacji symulowanego wyżarzania. Skorzystano z implementacji dostępnej w module simanneal dostępnej pod adresem https://github.com/perrygeo/simanneal. W celu zastosowania tego modułu należało zaimplementować metody move(), której zadaniem jest wybór punktu z sąsiedztwa punktu roboczego oraz metody energy(), która służy do obliczania funkcji celu dla danego punktu. Oprócz tego należało zdefiniować model, który zawierałby wszystkie dane wymagane do wykonywania funkcji move() oraz energy() zgodnie z założeniami projektu. Metoda optymalizacji zawarta w module simanneal jest konfigurowana przez następujące parametry:

- Tmax – temperatura początkowa
- Tmin – temperatura końcowa
- Steps – liczba kroków algorytmu
- Updates – parametr pomocniczy, który określa jak często wypisywać statystyki z dotychczasowego przebiegu algorytmu. Nie ma on wpływu na działanie metody optymalizacji.

Implementacja ta jest klasyczną wersją symulowanego wyżarzania. Przebieg kroku algorytmu jest następujący:

1. Wybierz punkt z y z sąsiedztwa punktu x : y = selRandom(N(x))
2. Jeśli q(y) < q(x), to zapamiętaj punkt y : x = y, gdzie q() – funkcja celu
3. Jeśli q(y) >= q(x) i rand() < p, to również zapamiętaj punkt y: x = y, gdzie p = exp(-|q(y) − q(x)|/T).
4. Algorytm kończy się po wykonaniu zdefiniowanej liczby kroków.

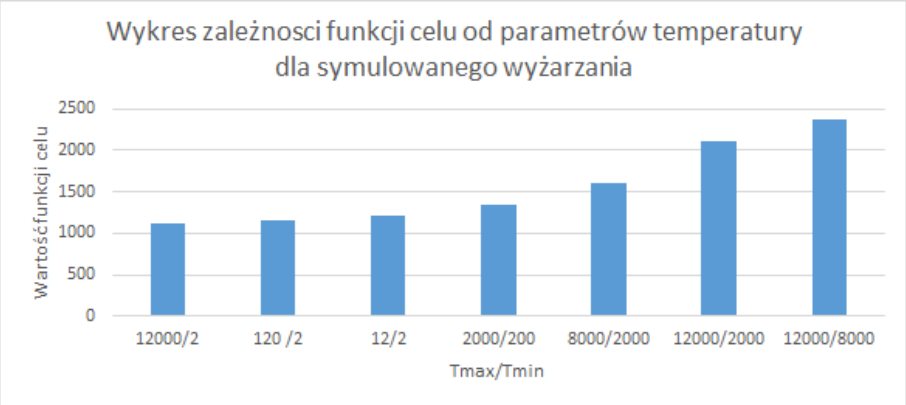
5. Temperatura zmienia się w kolejnych krokach zgodnie z zależnością: T = Tmax * exp(Tfactor * i / steps), gdzie

   - i – numer aktualnego kroku algorytmu
   - Tfactor – współczynnik obliczany w momencie inicjalizacji algorytmu
   - Tfactor = -log(Tmax/Tmin)

   Tak zdefiniowana temperatura zmienia się w czasie kolejnych korków algorytmu od wartości Tmax do Tmin. Jest to funkcja malejąca.

5 Temperatura

W celu dobranie najlepszych parametrów temperatury przeprowadzono próby, których rezultat pokazany jest na poniższym wykresie:(M = 25, STEPS = 1000, k = 10, d = 0)

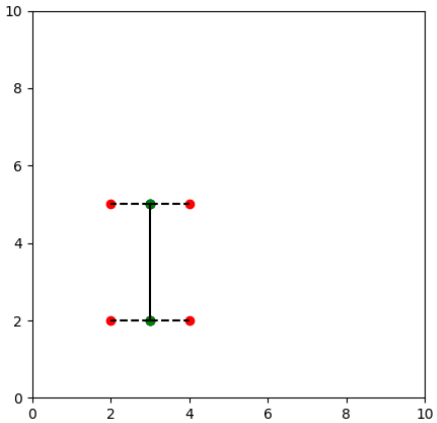


Rysunek 2: Wykres zależności funkcji celu od parametrów temperatury dla symulowanego wyrzażania

Najlepsze wyniki otrzymano dla Tmax =12000 oraz Tmin = 2.

6 Testy

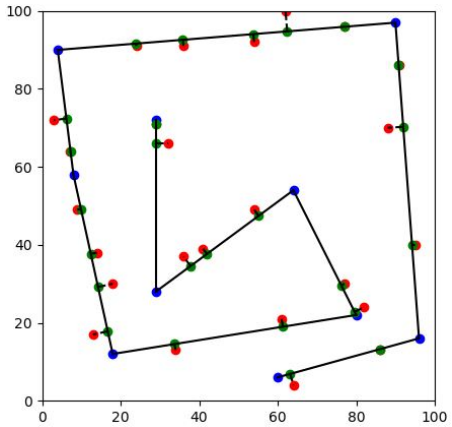
Poprawność działania algorytmu została przetestowana na przykładzie o następujących założeniach. Przestrzeń zmniejszona do kwadratu o wymiarach 10x10, M=4, k=2, d=0.



Rysunek 3: Wynik podstawowego przypadku sprawdzającego poprawność algorytmu

Dla liczby kroków steps = 10000 uzyskiwano rozwiązanie optymalne - minimum globalne funkcji kosztu wynoszące 7 (koszt autostrady = 3, koszt zjazdów = 4 * (2Î - 1) = 4).

Dodatkowo przetestowano przypadek dla M = 25 oraz K = 10 po wykonaniu 100 00 kroków algorytmu. Poniżej efekt.



Rysunek 4: Przykładowy wynik działania algorytmu

Po upewnieniu się, że model został poprawnie zdefiniowany dla metody optymalizacji przystąpiono do testów o większym rozmiarze problemu.

6.1 Test 10 miast

W pierwszym kroku psrawdzono przypadek dla 10 miast. Dodatkowe parametry to:

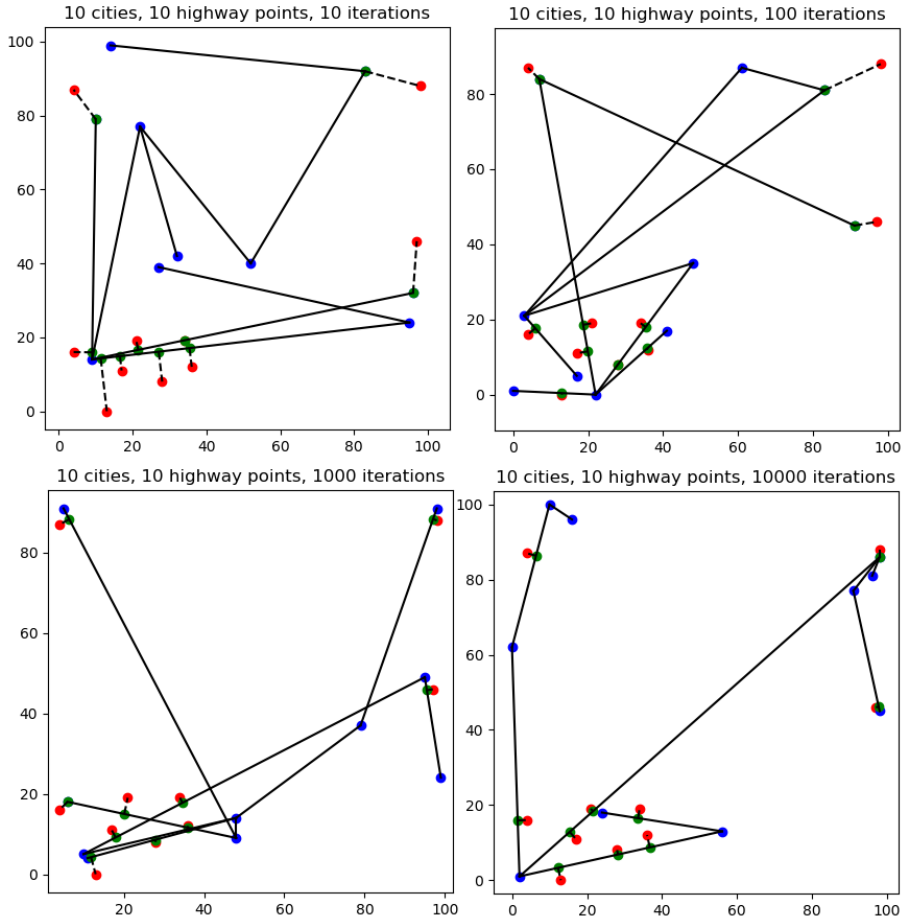
- M = 10
- K = 10
- D = 3
- STEPS - zmienne od 10 do 10 000

Uzyskano następujące wartości funkcji celu w zależności od liczby kroków:

- 10 - 1801449.81
- 100 - 7730.18
- 1000 - 533.60
- 10000 - 421.19

Można zauważyć bardzo szybki spadek wartości funkcji kosztu dla poczętkowych testów, później wartości wolniej się zmniejszają, ponieważ są już bliżej rozwiązania optymalnego. Poniżej grafika reprezentująca wyniki dla każdej z prób.

Rysunek 5: Legenda dla powstałych wizualizacji



Rysunek 6: Wyniki dla 10 miast

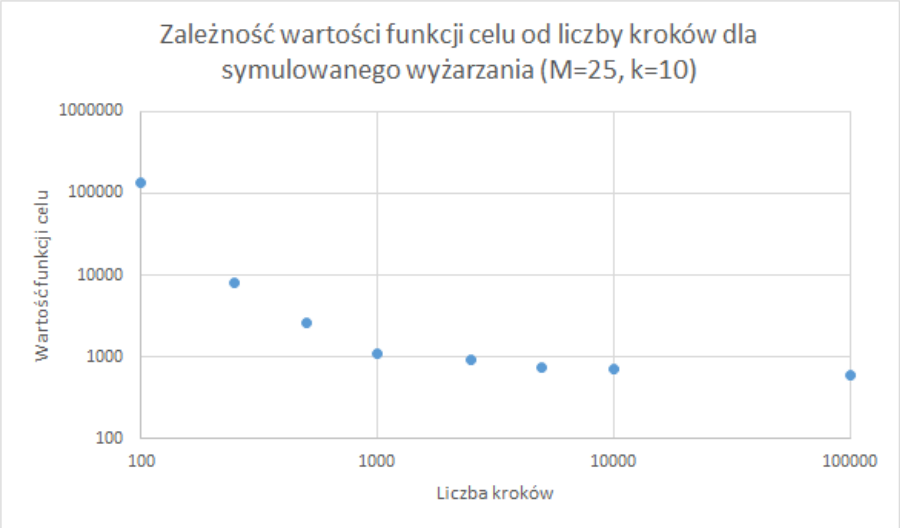
6.2 Test 25 miast

W kolejnym kroku przeprowadzono testy kompleksowe zawierające więcej porównań. Dane początkowe:

- M = 25
- K = 10
- D = 0
- STEPS - zmienne od 10 do 100 000

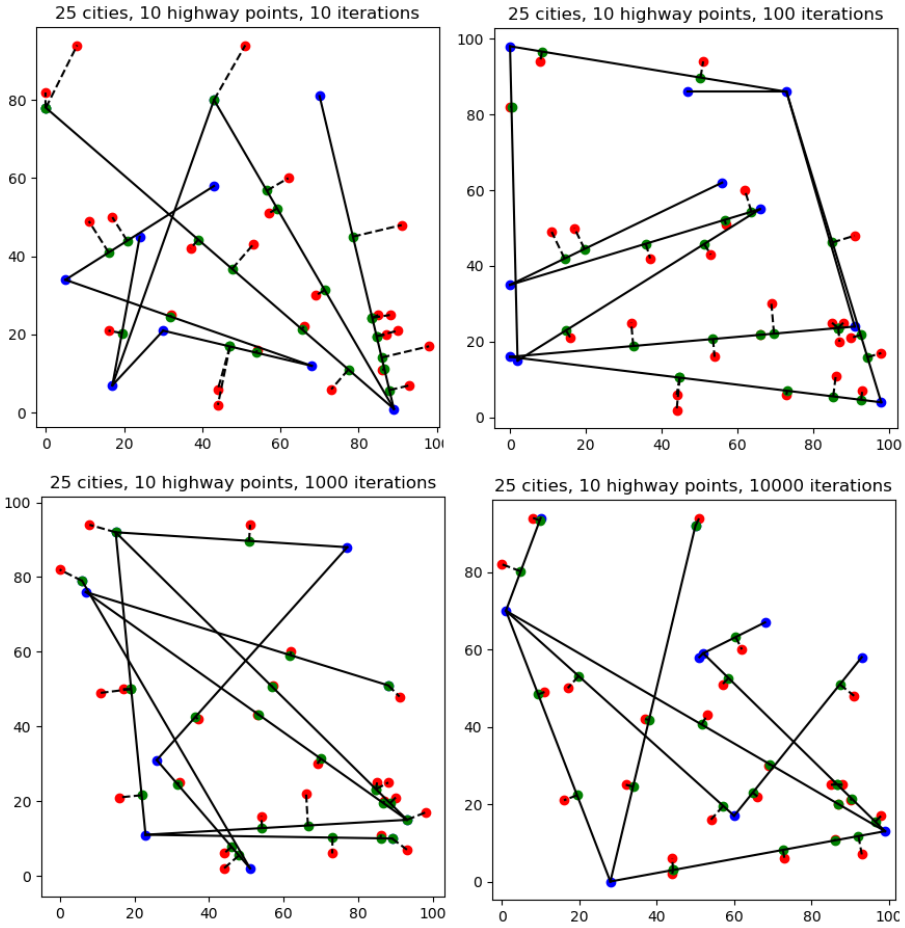
Dla każdej rozpatrywanej liczby kroków przeprowadzono po 10 niezależnych testów, a wynik minimalizacji kosztu autostrady widoczny na wykresie jest średnią z otrzymanych danych. Dla liczby kroków STEPS = 100 000 czas działania algorytmu wynosił średnio 13 minut, natomiast dla steps = 10 000 około 1,5 min.

Na podstawie wykresu przedstawionego na Rysunku 7 można zauważyć, że wartość funkcji celu sukcesywnie maleje wraz z ilością wykonwnych kroków przez algorytm. Początkowe wartości są bardzo wysokie przez co nieakceptowalne. Jednak po wykonaniu ponad 1000 kroków wartości funkcji kosztu wolniej maleją, co oznacza że ciężej znaleźć lepsze rozwiązanie.



Rysunek 7: Wykres zależności funkcji celu od liczby kroków dla 25 miast

Poniżej także wykresy prezentujące wyniki dla liczby kroków od 10 do 10000:



Rysunek 8: Wyniki dla 25 miast

Oraz uzyskane wartości funkcji celu w zależności od liczby kroków:

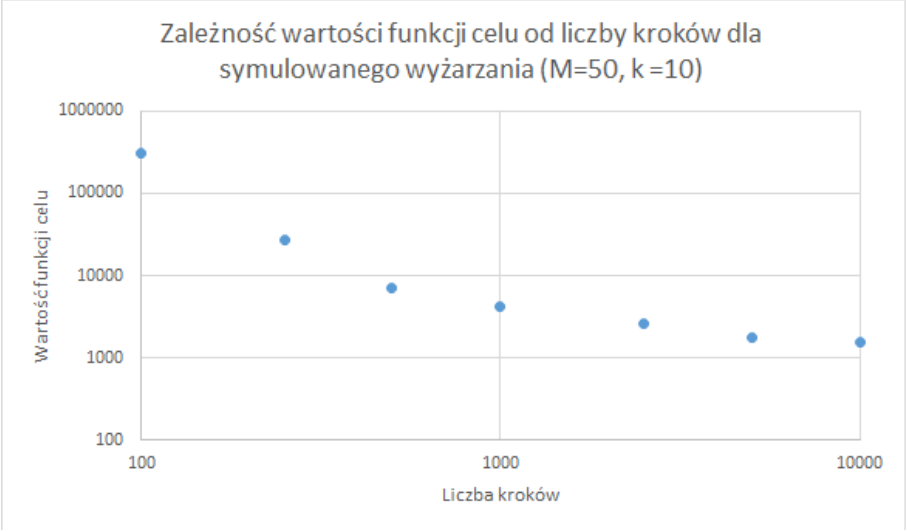
- 10 - 371075.61
- 100 - 2027.27
- 1000 - 1974.19
- 10000 - 794.61

6.3 Test 50 miast

W ostanim przypadku przeprowadzono testy dla 50 miast. Dane początkowe:

- M = 50
- K = 10
- D = 0
- STEPS - zmienne od 10 do 100 000

Podobnie jak w poprzednik przypadku, przeprowadzono wiele prób a wyniki uśredniono. Poniżej wykres zależności wartości funkcji celu od liczby kroków:

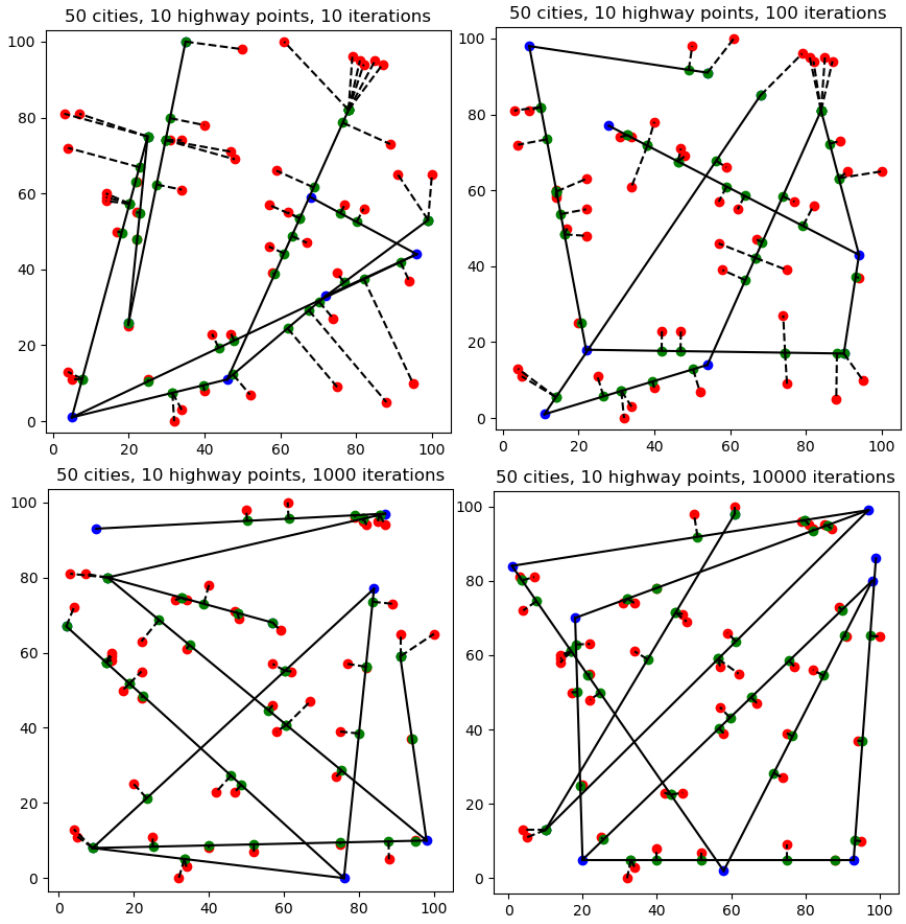


Rysunek 9: Wykres zależności funkcji celu od liczby kroków dla 50 miast

Poniżej wyniki funkcji celu dla przykładowego uruchomienia dla liczby kroków od 10 do 10 000:

- 10 - 4886132988.92
- 100 - 130663.54
- 1000 - 5062.10
- 10000 - 1482.57

Oraz wizualizacja wyników:



Rysunek 10: Wyniki dla 50 miast

6.4 Uruchamianie testów

Wszystkie testy mogą być w prosty sposób odtworzone na podstawie dostarczonych źródeł. W katalogu `tests` znajdują się podkatalogi zawierające współrzędne miast dla każdego testowanego przypadku.

Dodatkowo utworzono skrytp `run_tests.sh`, który wykona jeden przebieg wszystkich opisanych testów, dla ilości kroków 10, 100, 1000, 10 000. Dla każdego wyniku zostanie wyświetlony wynik w postaci wizualnej oraz utworzony plik zawierający uzyskane rozwiązanie, np. `tests/10_cities/100steps`:

```
Cities: [(4, 87), (17, 11), (13, 0), (4, 16), (21, 19), (34, 19),
(36, 12), (98, 88), (28, 8), (97, 46)]
Highway: [[(36, 53) -> (18, 31)], [(4, 25) -> (94, 48)], [(15, 12) -> (3, 74)],
 [(77, 29) -> (15, 12)], [(3, 74) -> (17, 30)], [(98, 91) -> (36, 53)],
 [(94, 48) -> (98, 91)], [(17, 30) -> (36, 53)], [(13, 93) -> (98, 91)],
 [(18, 31) -> (3, 74)]]
Total cost: 7730.180890387163
```

Cities, to współrzędne miast. Natomiast Highway to lista punktów na autostradzie w formacie [punkt -> następny punkt].