Eliza Wielocha
wielochaeliza@gmail.com

Działanie i testy

Algorytm metaheurystyczny ACO

1. Problem:

Dany jest spójny graf G=(V, E) z wagami wi przypisanymi do każdej krawędzi viV. Należy znaleźć ścieżkę łączącą wszystkie wierzchołki (każdy odwiedzony minimum raz) taką, aby zminimalizować jej koszt S. Koszt ten wyliczany jest dynamicznie w trakcie konstrukcji ścieżki w taki sposób, że:

a) stanowi sumę wszystkich wag odwiedzonych do tej pory krawędzi
b) co x odwiedzonych łuków licząc od startu do aktualnej sumy S dodawana jest suma ostatnich x /2 odwiedzonych wag pomnożona przez podwójny stopień wierzchołka, w którym znajduje się algorytm przeszukiwania po przejściu x krawędzi.

Założenia dla instancji problemu: można przyjąć początkowo: |V| minimum: 100, deg(v) = [1, 6], wi = [1, 100], x = minimum 5.

2. Wybrany algorytm

Wybranym algorytmem metaheurystycznym jest **Algorytm mrówkowy (ACO)** zaimplementowany w **Pythonie.**

3. Opis algorytmu

Do opisu algorytmu jak i implementacji niektórych funkcjonalności wykorzystana została wiedza między innymi z poniższych linków:
http://www.cs.put.poznan.pl/mradom/teaching/laboratories/OptKomb/CI_wyklad_ewoluc_4.pdf
http://www.cs.put.poznan.pl/mradom/teaching/laboratories/OptKomb/2010-12-15.pdf

Implementację kodu można podzielić w sumie na 5 części:

- Ustawianie początkowych parametrów
- Metoda odpowiadająca za wygenerowanie instancji
- Metoda odpowiadająca za wyliczenie kosztu dla pojedynczej ścieżki
- Metoda odpowiadająca za mrówkę
- Nasz ‘Main’, w którym odbywa się praca algorytmu

Początkowe parametry:

Część parametrów dla algorytmu takich jak liczba wierzchołków, szansa użycia feromonów, wzrost tej szansy czy granica wygładzania są ustawiane domyślnie w implementacji. Natomiast na samym początku uruchamiania programu użytkownik ma możliwość wyboru wartości kilku parametrów. Miało to na celu sprawniejsze wykonywanie testów ze zamianami poszczególnych wartości. Dla użytkownika pozostawał wybór na przykład czasu wykonywania algorytmu, ilości mrówek czy parametru x potrzebnego do wyliczania kosztu za ścieżkę. Jeśli użytkownik postanowił nic nie wpisać jako wartość danego parametru, program przypisywał wartość domyślną.

Enter the running time of the algorithm (in seconds): 180

Enter the number of ants:

Enter the value of alfa parameter: 1

Enter the value of beta parameter: 1

Enter the x parameter that is needed to calculate the cost for path (more than 5):

Na przykładzie powyżej niektóre wartości zostały wpisane, a niektóre pozostały puste. Tam gdzie użytkownik nie zdecydował się na konkretną wartość, została przypisana ta ustalona w implementacji.

Generowanie instancji:

Tworzymy dwuwymiarową tablicę. Poszczególne pola będą przechowywać wagi krawędzi. Jeżeli wartość danego pola wynosi 0, oznacza to że pomiędzy wybranymi wierzchołkami nie ma połączenia.

Następnie tworzymy połączenia, wypełniając dwie przekątne w tablicy, jedną przesunięta o jedno pole w lewo/dół i drugą przesuniętą o jedno pole w prawo/górę. W ten sposób nasz graf będzie spójny i będzie posiadał ścieżkę łączącą wszystkie wierzchołki.

Dla każdego wierzchołka sprawdzamy ilość prowadzących do niego krawędzi. Jeżeli jest ona mniejsza od minimalnej liczby krawędzi staramy się dodać brakującą ilość. W pętli sprawdzamy, do których wierzchołków możemy poprowadzić nową krawędź. Jeżeli do sprawdzanego wierzchołka prowadzi mniej krawędzi niż maksymalny limit tworzymy połączenie właśnie z nim. W przypadku gdy nie jesteśmy w stanie poprowadzić krawędzi do żadnego wierzchołka zgłaszamy błąd i wychodzimy z programu. W rezultacie powstaje nam macierz w postaci grafu połączeń, w którym jeśli nie ma pomiędzy wierzchołkami krawędzi to w polu wpisane jest 0. Jeśli natomiast jest, to w polu znajduje się waga tej krawędzi (wartość pomiędzy 1 a 100 domyślnie).

Następnie tworzymy jeszcze 2 macierze:

- Macierz feromonów - zostaje ona wypełniona wartościami 0. Następnie w miejscu występowania krawędzi wpisywana jest wartość 1.
- Macierz prawdopodobieństwa - jest ona kopią macierzy feromonów.

Metoda zwraca nam 3 macierze.

Wyliczanie kosztu za ścieżkę:

Wyliczanie kosztu ścieżki zaczynamy od utworzenia dwuwymiarowej tablicy o rozmiarze równym liczbie wierzchołków pomniejszonych o jeden. Wtedy odpowiada ona liczbie zawartych krawędzi w ścieżce.

Następnie w pętli przechodzimy po ścieżce znajdującej się w utworzonym wcześniej grafie odwiedzając przy tym wszystkie wierzchołki grafu, w celu przypisania wag danym krawędziom.

Sprawdzamy warunek czy odwiedzonych zostało minimum x krawędzi. Jeśli tak to przechodzimy do obliczania sumy wag krawędzi stanowiących koszt ścieżki opisany w treści problemu. Do aktualnej wartości wag dodajemy sumę wag ostatnich x/2 odwiedzonych wag pomnożoną przez podwójny stopień wierzchołka, w którym aktualnie znajduje się algorytm.

Mrówka:

Zaczynamy od utworzenia ścieżki będącej dwuwymiarową tablicą o rozmiarze równym ilości wierzchołków w naszym grafie. Dla każdej mrówki wierzchołek początkowy na jej ścieżce wybieramy w sposób losowy. Z pozostałych wierzchołków grafu tworzymy zbiór nie odwiedzonych wierzchołków stanowiących pamięć danej mrówki, co pozwala uniknąć ponownemu odwiedzaniu tych samych wierzchołków.

Następnie tworzymy pętle, która działa dopóki zbiór nie odwiedzonych wierzchołków nie jest zbiorem pustym. Pozwala to na odwiedzenie wszystkich wierzchołków, co jest jednym z warunków przedstawionego problemu.

Kolejny wierzchołek mrówka wybiera na podstawie macierzy prawdopodobieństw. Optymalny wybór stanowi wierzchołek o największym prawdopodobieństwie wyboru ze zbioru nie odwiedzonych wierzchołków. W przypadku jeśli kilka wierzchołków z tego zbioru charakteryzuje się takim samym prawdopodobieństwem wybór dokonywany jest w sposób losowy z zachowaniem równych szans dla każdego z tych wierzchołków. W przypadku jeżeli mrówka nie ma możliwości wyboru wierzchołka ze zbioru wierzchołków nieodwiedzonych, wybiera jakikolwiek sąsiedni wierzchołek o największym prawdopodobieństwie. Jeśli prawdopodobieństwa wyboru dla kilku sąsiednich wierzchołków są takie same to wybór odbywa się w sposób losowy.

Po odwiedzeniu wierzchołka oznaczonego jako nie odwiedzony, zostaje on usuwany z tego zbioru. Jeżeli rozmiar tablicy odpowiadającej ścieżce zostaje wyczerpany to następuje zwiększenie rozmiaru tej tablicy. Dzieje się tak dlatego, że nie zawsze (a raczej rzadko) zdarzy się sytuacja, w której ścieżka będzie miała dokładnie taki rozmiar jaki został wytypowany - to znaczy, że mrówka przejdzie przez jeden wierzchołek dokładnie raz. Zazwyczaj będzie musiała kilka z nich powtórzyć aby dojść do każdego wierzchołka w grafie.

Na koniec zwracamy koszt przejścia przez mrówkę ścieżki w grafie.

Main:

Wywołujemy metodę generowania instancji i przypisujemy nasze 3 macierze do zmiennych. Po utworzeniu tych macierzy, wywołujemy pętlę wykonującą się przez czas, jaki określiliśmy na początku w parametrach.

Najpierw puszczamy mrówki. Każda mrówka ‘produkuje’ jedną ścieżkę, która dodawana jest do listy ścieżek. Otrzymane ścieżki sortujemy według malejącego całkowitego kosztu. To znaczy, że ścieżka z najmniejszym kosztem będzie na samym początku listy.

W kolejnym etapie dochodzi do aktualizacji feromonów poprzez dodanie wartości z przedziału od 0,1 do 1. W ten sposób dodajemy wartość 1 dla najlepszej ścieżki i odpowiednio 0.1 dla najgorszej z otrzymanych ścieżek.

Następnie dochodzi do parowania feromonów i uaktualnienia macierzy prawdopodobieństw na podstawie wartości Alfa i Beta określonych w parametrach początkowych.

Potem uruchamiane jest wygładzanie macierzy feromonowej dla wartości przekraczających parametr granicy wygładzania.

Podczas działania programu zwracane są kolejne mniejsze koszty za ścieżkę pokazując przy okazji w jakim czasie zostały one skalkulowane:

Path with the lowest cost: 4957 at time: 1:15:49
Path with the lowest cost: 4304 at time: 1:15:49
Path with the lowest cost: 4193 at time: 1:15:49
Path with the lowest cost: 4092 at time: 1:15:49
Path with the lowest cost: 3786 at time: 1:15:50
Path with the lowest cost: 3457 at time: 1:15:50
.
.
.

Na samym końcu zwracana jest najlepsza możliwa ścieżka - to jest ścieżka z najmniejszym kosztem. Program również zwraca informacje o parametrach jakie użytkownik ustawił:

Parameters:
Running time of the algorithm: 180 | number of ants: 100 | alfa: 1 | beta: 1 | x parameter: 5
Cost for the best path is 3393.

---

4. Wyniki i strojenie mateheurystyki - **100 instancji**

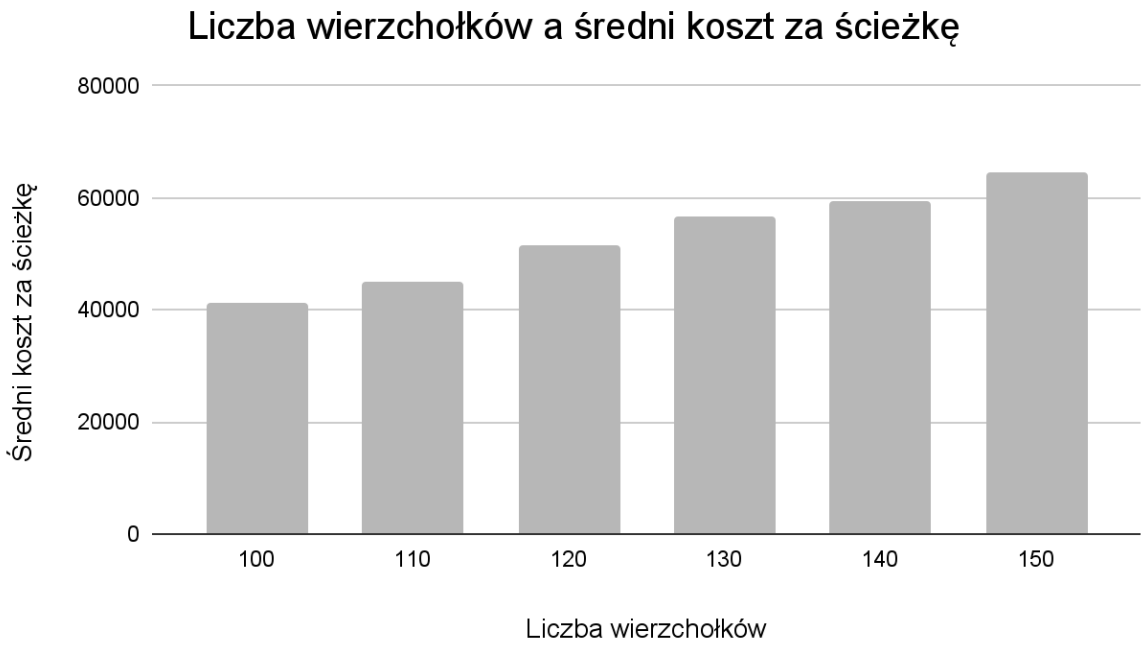
Wierzchołki	100
Krawędzie	od 6 do 30
Wagi	od 1 do 100
Czas algorytmu	300 sekund
Mrówki	100
Stopień parowania	10%
Szansa na użycie feromonów	1%
Wzrost szansy feromonowej	0.1%
Wygładzanie	20
Granica wygładzania	35
Alfa	1
Beta	0
X	5

1) Wpływ liczby wierzchołków na wyniki metaheurystyki

Domyślnie ustawioną wartością liczby wierzchołków na początku było 100. Jako, że algorytm zaimplementowany jest w języku python, a heurystyka kiedyś zatrzymać się musi, to już ta początkowa liczba stanowiła duży problem przy odnajdywaniu coraz to lepszych ścieżek w grafie. Czas jaki ustawiony został dla algorytmu wynosił 5 minut. Przy 100 wierzchołkach program odnajdywał maksymalnie do 10 ścieżek w grafie, których średnie koszty na początku wykazywały duże różnice.

Jednak z upływem następnych kilkudziesięciu sekund spadek ten nie był już taki zauważalny. Wraz ze wzrostem liczby wierzchołków, wzrastał koszt pierwszej znalezionej ścieżki, a co za tym idzie - ostatniej najlepszej ścieżki.

Liczba wierzchołków	Średni koszt znalezionej ścieżki
100	41289
110	44901
120	51436
130	56769
140	59248
150	64442



2) Wpływ liczby mrówek na wyniki metaheurystyki

Zaczynając od 100 mrówek sprawdzany był koszt najlepszej znalezionej ścieżki. Potem zwiększano liczbę mrówek o 30, aby ustalić jak zmienią się wyniki. Przy stopniowym wzroście ilości mrówek, średni koszt ścieżki nieco malał. Nie można tutaj mówić o dużych różnicach między średnimi kosztami, dla danych liczb mrówek. Jednak porównując ze sobą 100 mrówek i 220 mrówek, mamy już znaczniejszą różnicę. Może to oznaczać, że do bardziej widocznych zmian potrzebna byłaby zdecydowanie większa ilość mrówek.

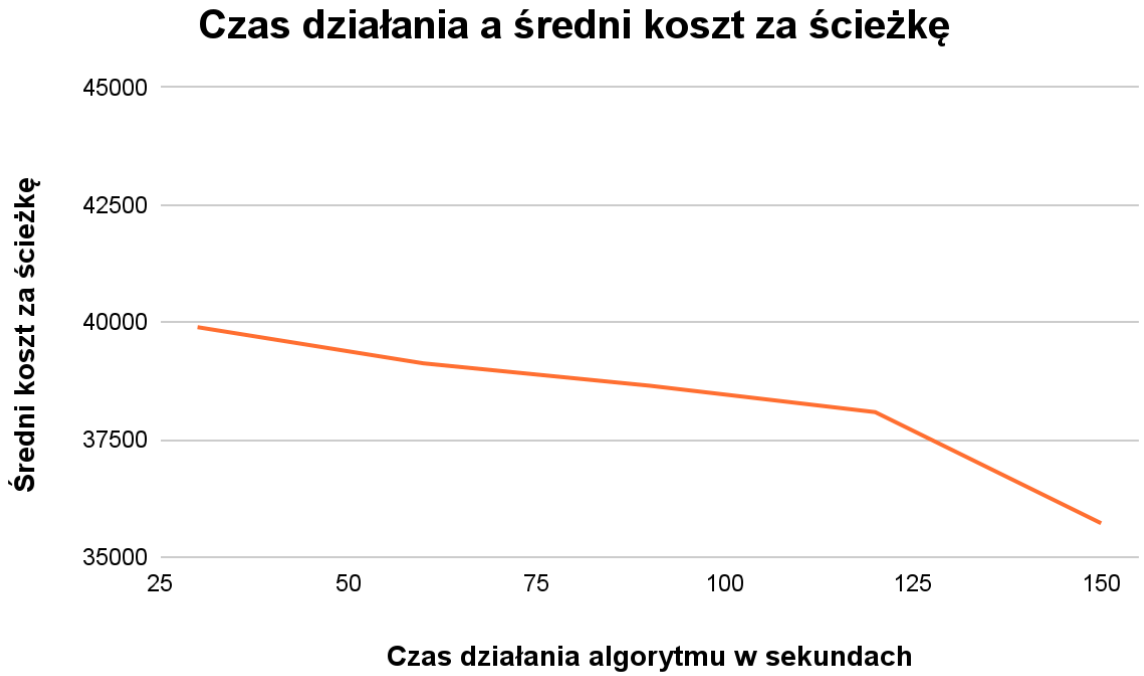
Liczba mrówek	Średni koszt znalezionej ścieżki
100	40089
130	40037
160	38032
190	37611
220	37001

3) Wpływ czasu działania algorytmu na wyniki metaheurystyki

Domyślnie w algorytmie czas ustawiony jest na 5 minut. Dla przetestowania jego działania, zmieniono jego wartość na 30 sekund i podnosiło się o kolejne 30 sekund. Zmieniono również nieco parametr odpowiadający za liczbę wierzchołków - zmniejszono jego wartość o 20. W tym teście występowało tylko 80 wierzchołków. Miało to na celu osiągnięcie bardziej wyrazistych wyników dla mniejszych czasów, jako że domyślnie algorytm działa przez 5 minut na 100 wierzchołkach.

Malejący średni koszt świadczy o tym, że czas ma ogromne znaczenie w użyteczności danej heurystyki. Im więcej czasu da się programowi na znalezienie rozwiązania, tym większe prawdopodobieństwo, że znajdzie lepsze.

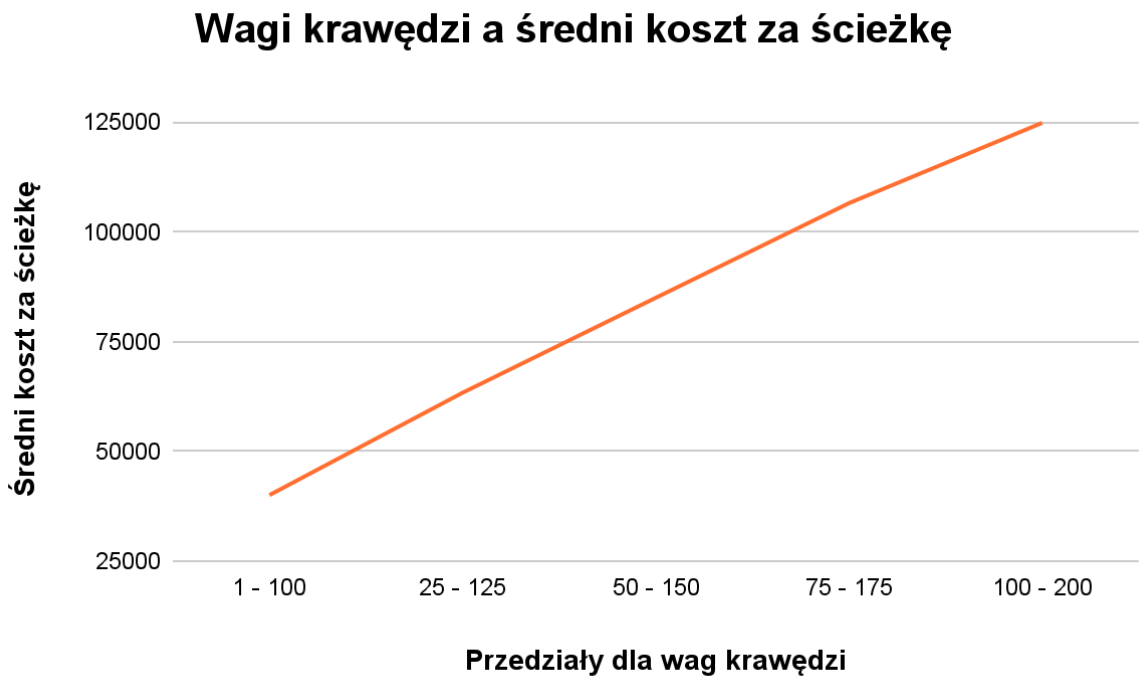
Czas działania algorytmu [s]	Średni koszt znalezionej ścieżki
30	39896
60	39126
90	38650
120	38087
150	35721



4) Wpływ wag krawędzi na wyniki metaheurystyki

Domyślnie ustawionym przedziałem dla wag krawędzi jest od 1 do 100. Przetestowano wpływ zwiększającego się przedziału dla wag na średnie koszty ścieżek. Zgodnie z założeniem wyniki testowania jasno pokazują tendencję wzrostową kosztów. Im wyższy przedział dla wag krawędzie, tym wyższy będzie koszt za ścieżkę.

Waga krawędzi	Średni koszt znalezionej ścieżki
1 - 100	40045
25 - 125	63366
50 - 150	85000
75 - 175	106557
100 - 200	124855

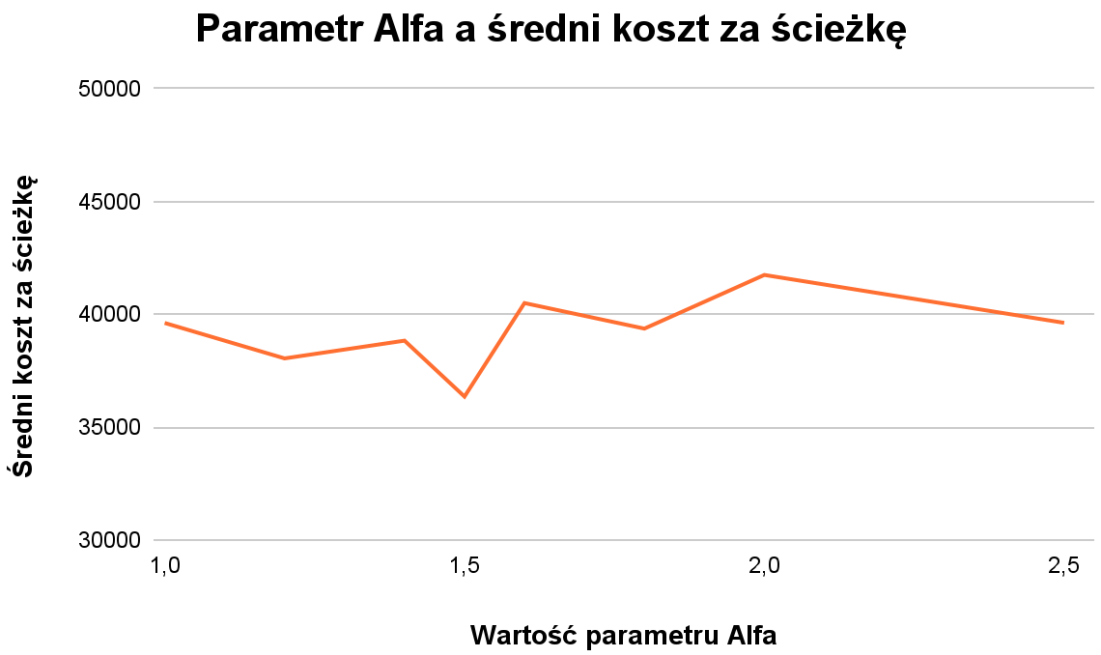


5) Wpływ wartości alfa i beta na wyniki metaheurystyki

W tym przypadku zmieniany był najpierw parametr alfa ze stałą wartością parametru beta, a później na odwrót.

Dla parametru alfa zaczynano od 1 i zwiększano wartość do 0,5. Parametr beta pozostawał taki sam - to jest 0. Jak widać w tabeli i na wykresie poniżej, wyniki wskazują na losowe średnie koszty znalezionych ścieżek. Nie ma tutaj tendencji spadkowej czy wzrostowej w zależności od zwiększania wartości parametru alfa.

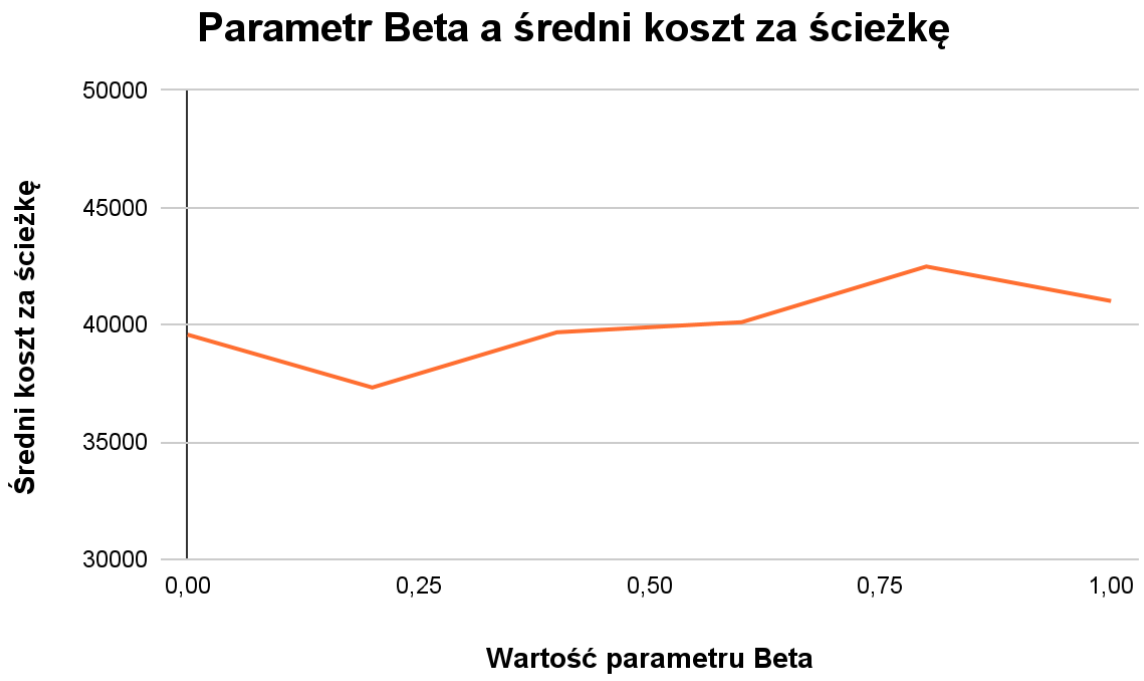
Alfa	Średni koszt znalezionej ścieżki
1	39618
1,2	38047
1,4	38833
1,5	36355
1,6	40499
1,8	39367
2	41745
2,5	39623



Następnie testowano parametr Beta - zwiększono jego wartość zaczynając od 0 a pozostawiając wartość parametru Alfa jako stałą (wartość 1). Tutaj również nie wykazano spadków oraz różnic w wartościach średnich kosztów.

Można wnioskować, że wartości alfa oraz beta nie mają aż tak dużego wpływu na wyniki działania algorytmu, w porównaniu do innych parametrów testowanych w sprawozdaniu.

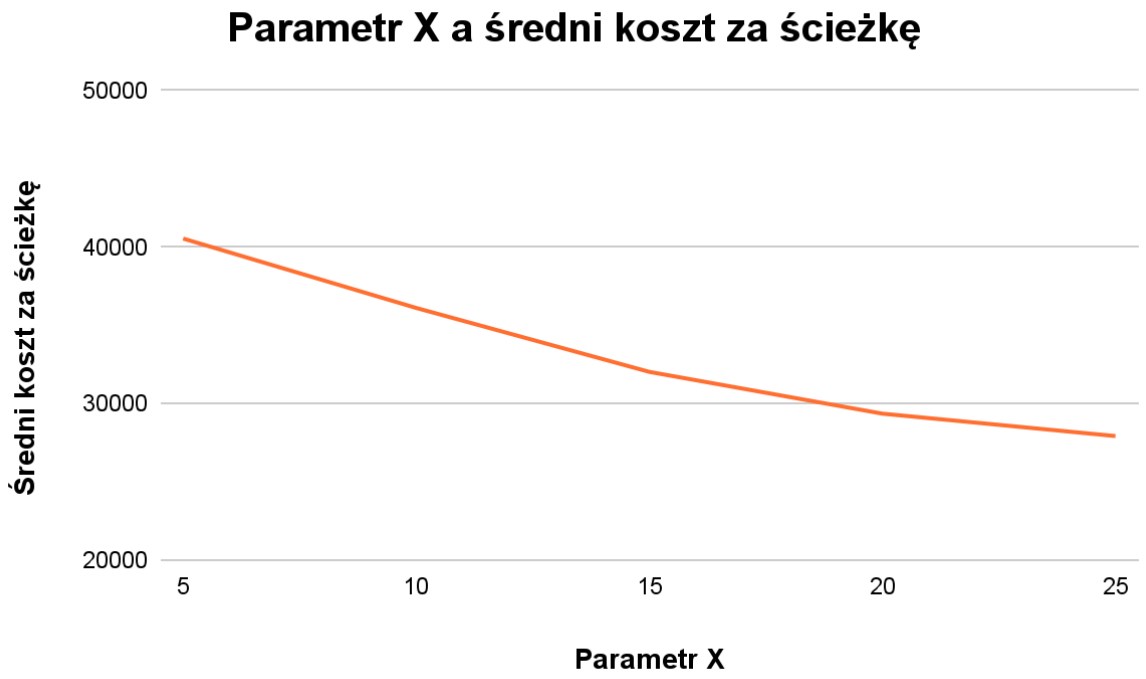
Beta	Średni koszt znalezionej ścieżki
0	39580
0,2	37326
0,4	39681
0,6	40111
0,8	42486
1	41015



6) Wpływ wartości parametru x na wyniki metaheurystyki

Parametr x jest istotny przy obliczaniu kosztu za pojedynczą ścieżkę. Ze wzoru wynika, że zwiększanie jego wartości powoduje zmniejszenie kosztu za daną ścieżkę.

Parametr x	Średni koszt znalezionej ścieżki
5	40540
10	36107
15	32025
20	29355
25	27924



5. Wnioski

Podsumowując cały etap testowania poszczególnych parametrów, można zauważyć że większość z nich badana w sprawozdaniu miała wpływ na wyniki metaheurystyki. Wyjątek stanowiły tutaj parametry alfa i beta, których wartości okazały się nie mieć większego znaczenia w odnajdywaniu lepszych ścieżek w algorytmie.

Wśród parametrów, których zwiększenie wartości przyniosło pozytywne rezultaty znalazły się czas algorytmu i parametr X. Samo założenie heurystyki polega na znajdowaniu jak najlepszych rozwiązań w czasie. Oznacza to, że im więcej czasu ma heurystyka na działanie, tym lepsze rozwiązanie znajdzie. W naszym przypadku oczywiście nie można było tego czasu zwiększyć znacząco, jednak nawet przy przeskokach o 30 sekundach widać ogromną różnicę w działaniu algorytmu. Parametr X natomiast definiuje nam po ilu odwiedzonych łukach zaczynamy wyliczać koszt dla ścieżki. Im wyższy on jest, tym korzystniej dla ścieżki, a także dla późniejszych rozwiązań.

Parametrem niepewnym, który także można zaliczyć pod polepszenie wyników metaheurystyki jest liczba mrówek. Przy testowaniu większej ilości mrówek w algorytmie można było zauważyć spadek kosztów za ścieżki, jednak był nieznaczny. Dopiero po podwyższeniu liczby mrówek do 220 różnica była zauważalna. Może tak być, ponieważ w testowaniu potrzebny był większy przeskok między poszczególnymi wartościami sprawdzanymi. Być może przy różnicy między liczbami mrówek równej 100 można było osiągnąć lepsze rezultaty.

Wśród parametrów, których podwyższanie wartości nie sprzyja polepszeniu działania algorytmu, znalazły się liczba wierzchołków i przedziały dla wag krawędzi. W przypadku liczby wierzchołków, zwiększenie tej wartości prowadzi do zwiększenia postaci całego grafu w rezultacie czego - dłuższych ścieżek z większym kosztem. Natomiast podwyższenie przedziału dla wag krawędzi sprawia, że zwiększa się koszt każdego przejścia mrówki, a tym samym całej jej trasy.

Do ogólnych wniosków można zaliczyć zbyt małe przeskoki między wartościami parametrów. Wyniki, tabele i wykresy mogły być bardziej czytelne gdyby zwiększyć różnice między wartościami badanymi.