

# MATEMATYCZNE METODY WSPOMAGANIA DECYZJI

## Dostawa firmy MMWD - "Miejskie Matematycznie Wybierane Dostawy"

AUTORZY: ALEKSANDRA BEŃ  
MARCIN BEREŹNICKI  
IGA CHUDZIKIEWICZ

---

### Wprowadzenie do problemu

#### Opis analizowanego zadania

Rozważanym przez nas problemem optymalizacyjnym jest problem z kategorii zagadnień transportowych - firma dostarczająca jedzenie na terenie jednego miasta. W wykreowanym problemie, rozwiązaniem jest znalezienie jak najlepszego przydziału dostaw do dostawców oraz kolejności ich realizacji, mające na celu maksymalizację zysków.

W obrębie sieci dostawców wyróżniono 3 rodzaje pojazdów charakteryzujące się różnymi średnimi prędkościami dostawy, udźwigiem oraz kosztami dostaw:

- rower,
- hulajnoga elektryczna,
- skuter.

Dostarczane produkty będą się różnić wagą oraz punktem odbioru towaru przez dostawcę. Dodatkowym parametrem będzie maksymalny czas, po którym muszą trafić one do osoby zamawiającej. Czas określony jako maksymalny czas dostawy liczony jest od momentu odebrania przez dostawcę jedzenia z punktu odbioru.

Dostawca ma do wyboru realizować zamówienia pojedynczo lub kilka jednocześnie, wykonując kolejne zamówienia w trakcie realizacji poprzedniego, w zależności od tego jaka strategia przyniesie większy zysk. W żadnym momencie jednak jego załadunek nie może przekroczyć maksymalnej wartości udźwigu.

Z każdym dostawcą związane jest ponoszenie kosztów jego pracy w zależności od przydzielonego pojazdu.

Jako firma dostarczająca jedzenie, wybierane są zamówienia z dostępnej bazy zamówień, w sposób pozwalający na maksymalizację zysków.

Punkty odbioru oraz dostaw jedzenia będą reprezentowane przez macierz kosztów, natomiast zarobek uzależniony jest od wagi dostarczanej paczki z jedzeniem (w sposób jednostka monetarna na kilogram).

---

## Przyjęte uproszczenia

- Wszystkie zamówienia mają ten sam priorytet, nie ma z góry przyjętej kolejności dostarczania zamówień
- Nie przewiduje się awarii żadnego ze środków transportu dostawcy - brak ograniczenia ogólnego liczby kursów dostawcy
- Nie bierze się pod uwagę warunków atmosferycznych, zagęszczenia ruchu drogowego w godzinach szczytu - stała prędkość danego rodzaju pojazdu
- Losowo wygenerowana pula zamówień jest znana
- Ograniczony jest czas realizacji puli zamówień, natomiast nie dostarczone zamówienia nie mają konsekwencji poza utraconą możliwością zarobku

Każde z powyższych uproszczeń można wprowadzić do modelu celem rozszerzenia problematyki zagadnienia.

### **Dodatkowe możliwe rozszerzenia problemu optymalizacyjnego:**

- Rejonizacja dostaw - podział punktu odbioru zamówień ze względu na zlecony adres dostawy
- Okna czasowe dostaw
- Wprowadzenie napiwków w zależności od czasu oczekiwania na dostawę
- Zarobek uzależniony od rodzaju produktu, nie proporcjonalny do wagi zamówienia

---

# Model matematyczny

## Struktury danych:

N - liczba zamówień

M - ilość dostawców

A - macierz przedstawiająca mapę miasta

c - koszt jednostki wagowej

$w_i$  - waga i-tego zamówienia, dla  $i = 1, \dots, N$

$x_i$  - adres odbioru i-tego zamówienia, dla  $i = 1, \dots, N$

$y_i$  - adres dostawy i-tego zamówienia, dla  $i = 1, \dots, N$

$r_i$  - długość trasy wyliczana pomiędzy  $x_i$  oraz  $y_i$ , dla  $i = 1, \dots, N$

$a_j$  - koszt dostawcy oraz eksploatacji pojazdu, dla  $j = 1, \dots, M$

## Postać rozwiązania:

$z_j$  - trasa poszczególnego dostawcy, dla  $j = 1, \dots, M$

## Postać funkcji celu:

$$f(z) = \max \left( \sum_{i=1}^N c w_i - \sum_{i=1}^N \sum_{j=1}^M r_i a_j \right)$$

## Warunki ograniczające:

I. Dostawca nie może przewozić większego ciężaru towaru niż wynosi jego udźwig

n - ilość zamówień przetwarzanych przez kierowcę

$u_j$  - udźwig j-tego kierowcy

$$\forall_{j \in M} : \sum_{i=1}^n w_i \leq u_j$$

II. Czas pracy dostawców jest ograniczony

T - maksymalny czas pracy

$p_j$  - prędkość j-tego kierowcy

$$\forall_{j \in M} : \sum_{i=1}^N r_i / p_j \leq T$$

III. Czas dostarczania nie może być dłuższy od maksymalnego czasu dostawy

$t_i$  - maksymalny czas dostawy i-tego zamówienia

$$\forall_{j \in M} \forall_{i \in n} : r_i / p_j \leq t_i$$

---

# Rozwiązanie problemu

## Algorytm rozwiązujący problem

### Opis teoretyczny algorytmu

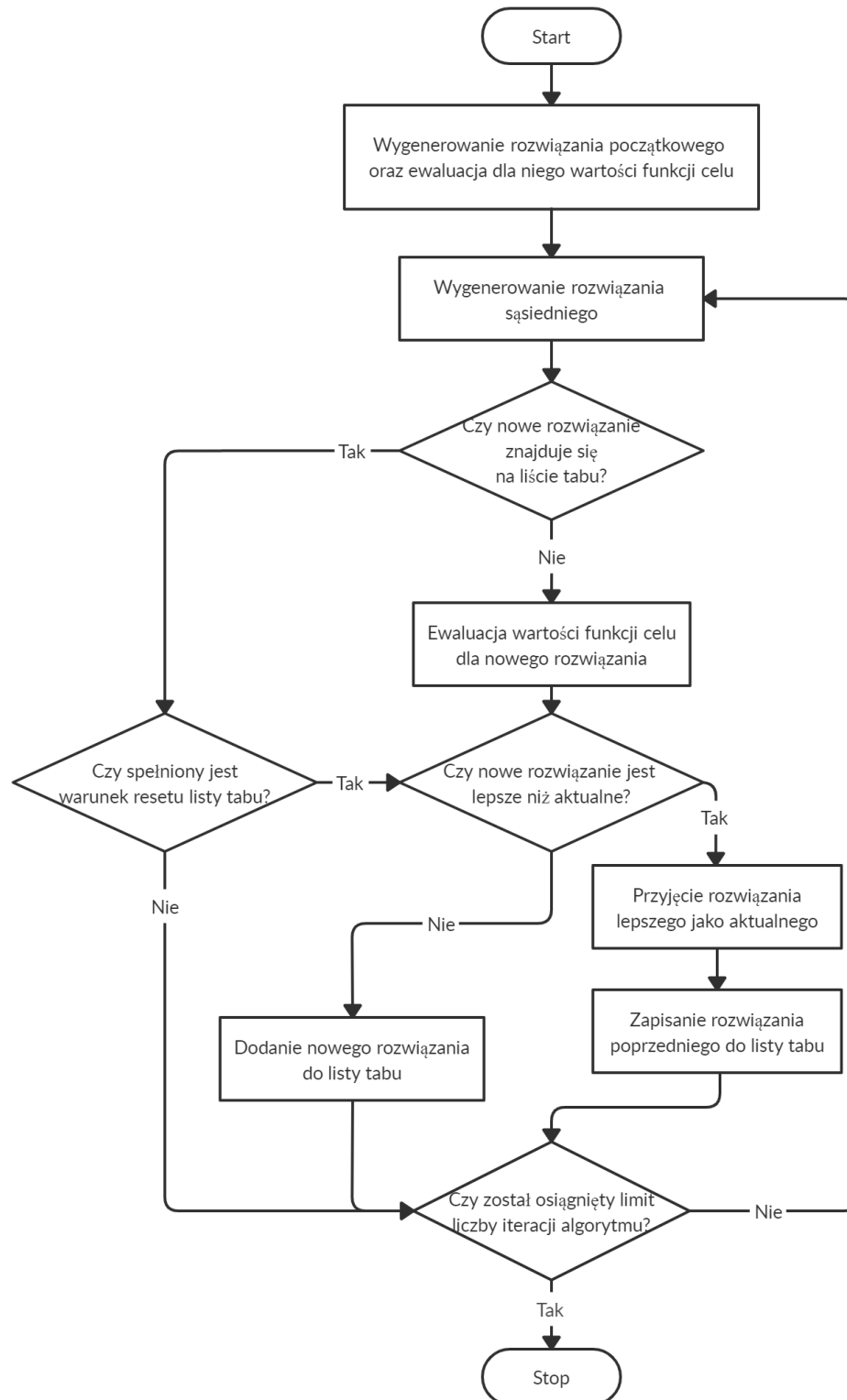
Przeszukiwanie Tabu to powszechnie stosowana metaheurystyka, używana do rozwiązywania problemów optymalizacyjnych. Oparta jest na iteracyjnym przeszukiwaniu przestrzeni rozwiązań, wykorzystując sąsiedztwo pewnych elementów tej przestrzeni oraz zapamiętując przy tym przeszukiwaniu ostatnie ruchy, dopóki nie zostanie spełniony warunek końcowy.

Ruchy zapisywane są w postaci atrybutów przejścia na liście tabu. Obecność danego ruchu na liście tabu jest tymczasowa oraz oznacza, że danego ruchu nie można wykonać przez określoną liczbę iteracji, chyba że ruch spełnia tzw. kryterium aspiracji. Lista tabu ma za zadanie wykluczyć prawdopodobieństwo zapętleń przy przeszukiwaniu i zmusić algorytm do przeszukiwania nowych, niezbadanych rejonów przestrzeni przeszukiwań.

### Dlaczego Algorytm Tabu?

Algorytm Tabu generuje ogólnie dobre rozwiązania problemów optymalizacji, w porównaniu z innymi metaheurystykami, które bazują na rozwiązaniach. Pozytywnym aspektem tego podejścia jest możliwość szybszego wyjścia z regionu przyciągania ekstremum lokalnego, które niekoniecznie musi być globalnie optymalne. Również listę Tabu można wykorzystać do uniknięcia cykli i powrotu do starych rozwiązań. Może być on stosowany zarówno do rozwiązań dyskretnych, jak i ciągłych.

## Uproszczony schemat blokowy



---

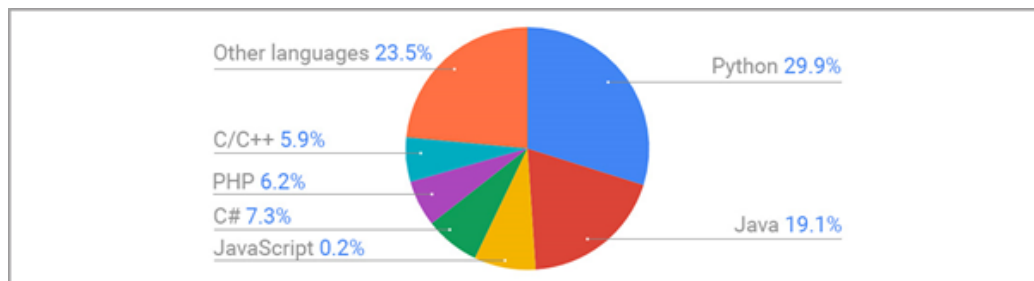
## Dostosowanie algorytmu do problemu

W celu opracowania rozwiązania problemu transportowego, przy użyciu algorytmu Przeszukiwania Tabu, zdecydowano się na następujące podejście do zagadnienia. Za rozwiązanie obrano całkowitą trasę przejazdu kierowcy, do której znajdowano możliwe do wykonania zamówienia. W kolejnych iteracjach trasa zmieniana jest w sposób losowy, z uwagą na maksymalizację ilości oraz wagi przetwarzanych przez kierowców zamówień, których punkty odbioru i dostawy objęte zostały w trasie. W ten sposób, wykonane zamówienia są bezpośrednio zależne od dobranej trasy, a punkty drogi danego kierowcy są jedynie pośrednio zmieniane w oparciu o listę zamówień.

## Wybór języka programowania

W celu realizacji powyższego problemu zdecydowaliśmy się na użycie języka programowania Python. Jest on jednym z najbardziej dostępnych języków programowania, ponieważ ma elegancką i prostą składnię oraz jest uważany za najłatwiejszy do opanowania. Ze względu na łatwość uczenia się i użytkowania, kody Pythona można łatwo pisać i wykonywać znacznie szybciej niż inne języki programowania.

Indeks PYPL za luty 2020:



PYPL obrazuje popularność języków programowania, pokazując jak często ludzie szukają samouczków językowych i kursów do nauki w Google. Podkreśla on więc, że język programowania Python, będąc obecnie najbardziej uczonym, wkrótce zwiększy również swoją popularność.

---

## Implementacja struktur

### Generacja mapy

Funkcja odpowiedzialna za generację mapy przyjmuje cztery elementy: ilość punktów, minimalną oraz maksymalną odległość między punktami a także prawdopodobieństwo, że powstanie droga między dwoma punktami.

Istotnym założeniem jest stworzenie grafu skierowanego oraz spójnego. Oznacza to, że może powstać sytuacja, kiedy jesteśmy w stanie przejść bezpośrednio z wierzchołka A do B, natomiast z wierzchołka B do A już nie. Aby jednak z każdego punktu była możliwość dojścia do dowolnego miejsca (celem możliwości realizacji dowolnego zamówienia i nie utknięcia w jakimś punkcie przez dostawcę), do każdego wierzchołka grafu musi prowadzić i odchodzić co najmniej jedna ścieżka.

Biorąc pod uwagę powyższe kwestie, w pierwszej kolejności generowana jest macierz kwadratowa wypełniona 'nieskończonymi' kosztami przejść. Następnie dla każdego punktu tej macierzy (poza główną przekątną) losowana jest liczba z zakresu  $<0,1$  i jeżeli jest ona mniejsza lub równa założonemu prawdopodobieństwu to krawędź jest tworzona a koszt drogi jest zamieniany na losowy z podanego zakresu odległości między punktami.

### Wybór punktów odbioru dostaw

Funkcja odpowiedzialna za określenie punktów odbioru dostaw przyjmuje dwa parametry: ilość wszystkich punktów na mapie oraz żadaną ilość punktów odbioru. Na tej podstawie z puli wszystkich punktów losowana jest odpowiednia ilość punktów odbioru dostaw, która następnie zapisywana jest w postaci listy.

### Generacja puli zamówień

Funkcja odpowiedzialna za generację zamówień przyjmuje następujące parametry: liczbę zamówień, przedział maksymalnego czasu dostawy, minimalną oraz maksymalną wagę zamówienia, liczbę wszystkich punktów oraz miejsca odbioru dostaw. Pula zamówień przechowywana jest w postaci listy. Zamówienia są natomiast instancjami klasy.

Dla każdego zamówienia losowana jest waga oraz czas dostawy z podanego przedziału. Następnie, na podstawie liczby wszystkich punktów, losowane jest miejsce dostawy oraz odbioru. Istotne jest, aby punkty te były różne, natomiast dopuszcza się aby miejsce dostawy było jednocześnie jednym z innych punktów odbioru dostaw. Dodatkowo, zamówienia te są ponumerowane dla zachowania porządku (zgodnie z kolejnością ich tworzenia).

---

## **Generacja puli kierowców**

Pula kierowców przechowywana jest w postaci listy. Każdemu kierowcy jest przydzielony jeden z trzech rodzajów pojazdów. Zarówno kierowcy jak i pojazdy są instancjami klas. Kolejność kierowców w liście jest posegregowana zgodnie z ich pojazdami: rower, hulajnoga elektryczna, skuter. Każdy pojazd przyjmuje parametry takie jak: prędkość, pojemność oraz koszt przemieszczania, które można dowolnie ustawić. Innymi parametrami związanymi z kierowcami są: wynagrodzenie kierowcy (wypłacane za przejechany dystans) oraz czas pracy.

## **Implementacja algorytmu**

Funkcja `tabu_search` ma za zadanie zarządzać wszystkimi pozostałymi funkcjami zdefiniowanymi w programie w celu przeprowadzenia całkowitego działania algorytmu z punktu widzenia pojedynczego kierowcy, dla wprowadzonych parametrów oraz utworzonych danych. Przyjmuje ona jako argumenty kierowcę, mapę, listę zamówień, zysk jednostki wagowej towaru, liczbę iteracji oraz okres resetu listy tabu. Funkcja ta ogranicza liczbę wykonywanych iteracji zgodnie z przekazaną wartością, a następnie zgodnie z listą kroków przedstawioną na schemacie blokowym wykonuje operacje mające na celu znalezienie rozwiązania o maksymalnej wartości funkcji celu. Po zakończeniu iteracji funkcja aktualizuje atrybuty klasowe dla danego kierowcy oraz zwraca pulę dostępnych zamówień.

## **Wygenerowanie pierwszego rozwiązania**

Jako argumenty funkcja przyjmuje kierowcę oraz wygenerowaną mapę i na ich podstawie generowana jest droga, którą będzie poruszać się kierowca. Lista punktów generowana jest w sposób całkowicie losowy z zachowaniem ograniczeń, takich jak maksymalny czas pracy kierowcy oraz możliwość przemieszczania się pomiędzy poszczególnymi punktami, określona w macierzy kosztów. Poprzez dodanie liczby iteracji wyszukiwania kolejnych punktów trasy w momencie utknięcia przy dużym pozostałym czasie pracy kierowcy, zminimalizowano różnicę pomiędzy maksymalnym a rzeczywistym czasem pracy kierowcy.

## **Wygenerowanie kolejnych rozwiązań**

Rozwiązania kolejne generowane są poprzez modyfikację jednego punktu rozwiązania obecnego. Funkcja przyjmuje jako parametry kierowcę, mapę, rozwiązanie aktualnie najlepsze oraz listę tabu. Działanie funkcji można podzielić na dwa przypadki. Pierwszy z nich to sytuacja kiedy długość aktualnego rozwiązania jest mniejsza bądź równa 3. Wtedy do rozwiązania dołączane są kolejne punkty w sposób losowy, analogicznie do generowania pierwszego rozwiązania i zwracana jest nowa trasa dla kierowcy. W drugim przypadku ma miejsce podstawowe działanie tej funkcji. Opiera się ono na losowym wyborze jednego punktu spośród aktualnego rozwiązania i podmianie go na inne punkty tak aby niezmieniony pozostał punkt przed nim oraz po nim.



---

## Ocena rozwiązania

Funkcja odpowiedzialna za ocenę rozwiązania jako parametry przyjmuje kierowcę, mapę, sprawdzone rozwiązanie, listę zamówień oraz cenę za jednostkę wagi. W pierwszej kolejności obliczana jest odległość jaką przejedzie kierowca, poprzez zsumowanie odległości między kolejnymi punktami rozwiązania.

Następnie na podstawie rozwiązania sprawdzane jest jakie zamówienia jest w stanie wykonać kierowca dla danej trasy. W tym celu dla każdego punktu obliczana jest waga w pojeździe kierowcy w danym momencie trasy. Dla każdej dostawy sprawdzanie możliwości jej realizacji rozpoczyna się od przeszukania rozwiązania w celu znalezienia punktu odbioru. Jeżeli takie istnieją na trasie kierowcy to w kolejnym kroku sprawdzane jest czy istnieje możliwość przyjęcia tego zamówienia. W związku z tym przechodzimy do następnego punktu aż dojdziemy do punktu końcowego. Z każdym kolejnym krokiem sprawdzane jest czy w danym miejscu łączna waga zamówień nie przekracza maksymalnego udźwigu oraz czy nie został przekroczony maksymalny czas dostawy.

W zależności od spełnienia powyższych ograniczeń, zamówienie jest przypisywane kierowcy lub pozostawione do realizacji. Po zakończeniu przeglądu dostępnych dostaw obliczana jest suma ich wag w celu określenia zarobku. Ostatnim krokiem jest obliczenie wartości funkcji celu dla tego rozwiązania oraz zwrócenie jej wraz zamówieniami realizowanymi przez danego kierowcę oraz zaktualizowaną listą pozostałych zamówień.

## Klasy

W celu przechowywania danych w sposób uporządkowany zdefiniowano następujące klasy:

- Order: klasa definiuje pojedyncze zamówienie poprzez przechowywanie jego maksymalnego czasu wykonania od moment odbioru w punkcie początkowym, wagi, punktu odbioru, punktu docelowego oraz numeru porządkowego.
- Vehicle: klasa definiuje rodzaj pojazdu poprzez przyjęcie jego prędkości, maksymalnej pojemności wagowej oraz kosztów związanych z eksploatacją na odcinku jednostkowym drogi.
- Driver: klasa definiuje kierowcę poprzez przydział dla niego rodzaju pojazdu, kosztu pracy za przejechany odcinek jednostkowy drogi, pozostałego oraz całkowitego czasu pracy, jak i przechowuje całkowitą trasę kierowcy, listę realizowanych przez niego zamówień, wartość funkcji celu rozwiązania oraz listy tras i wartości funkcji celu uzyskanych w kolejnych iteracjach.

---

## Testy algorytmu

W celu ewaluacji wyników dla każdego testu obliczana jest wartość odniesienia. Jest to wartość wyliczana poprzez sumowanie całkowitego zysku możliwego do uzyskania przy wykonaniu wszystkich zamówień, pomniejszona o koszty dostawców dla przypadku nieprzerwanej jazdy w czasie całkowitego czasu pracy.

### I Instancja testowa:

Mapa:

	0	1	2	3	4
0	inf	inf	10	14	8
1	18	inf	11	4	5
2	16	14	inf	19	9
3	18	7	3	inf	13
4	8	13	17	inf	inf

Punkty odbioru:

{2, 1, 3}

Zamówienia:

Ilość zamówień: 100

Przedział maksymalnego czasu dostawy: 1-2

Minimalna waga = 1

Maksymalna waga = 5

Cena za jednostkę wagi: 2

Dostawcy i pojazdy:

	Prędkość dostawy	Pojemność pojazdów	Koszt użytkowania pojazdu
<b>Rower</b>	15	20	0
<b>Skuter</b>	30	30	0.05
<b>Hulajnoga elektryczna</b>	20	10	0.005

Opłata dla dostawcy: 0.8

Czas pracy każdego dostawcy: 4

Liczba użytkowanych rowerów: 1

Liczba użytkowanych skuterów: 1

Liczba użytkowanych hulajnóg elektrycznych: 1

---

## Wpływ ilości iteracji na działanie algorytmu

Liczba iteracji = 10

Okres resetu listy Tabu = 2

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 93.8 Trasa: [0, 3, 2, 4, 1, 3, 1] Numer iteracji: 10	Zysk: -8.6 Trasa: [0, 2, 4, 1, 4, 0, 4, 1, 2] Numer iteracji: 4	Zysk: 117.6 Trasa: [0, 3, 2, 1, 0, 2, 3, 2, 4, 1, 4] Numer iteracji: 3	202.8	71	0.046 sekund
Zysk: 73.8 Trasa: [0, 2, 4, 1, 3, 1, 2] Numer iteracji: 2	Zysk: 14.2 Trasa: [0, 3, 1, 3, 2, 1] Numer iteracji: 10	Zysk: 108.6 Trasa: [4, 1, 4, 2, 0, 2, 0, 3, 2, 4, 2] Numer iteracji: 9	196.6	68	0.047 sekund
Zysk: 63.2 Trasa: [4, 1, 4, 0, 3, 2, 4] Numer iteracji: 0	Zysk: 41.1 Trasa: [2, 3, 2, 3, 1, 3, 1] Numer iteracji: 8	Zysk: 134.4 Trasa: [2, 3, 4, 0, 3, 2, 3, 1, 2, 0, 2] Numer iteracji: 10	238.7	78	0.048 sekund
Zysk: 94.4 Trasa: [4, 0, 3, 1, 3, 1, 3, 4] Numer iteracji: 8	Zysk: 39.4 Trasa: [0, 2, 4, 1, 2, 1, 3, 2, 4] Numer iteracji: 10	Zysk: 74.2 Trasa: [4, 1, 3, 2, 3, 0, 4, 0, 2, 1, 3, 1, 2] Numer iteracji: 10	208.1	75	0.046 sekund
Zysk: 105.2 Trasa: [1, 3, 0, 2, 1, 3, 2] Numer iteracji: 2	Zysk: 0.1 Trasa: [0, 3, 4, 1, 2, 0] Numer iteracji: 5	Zysk: 98.0 Trasa: [0, 3, 0, 2, 1, 2, 3, 1, 3, 0] Numer iteracji: 5	203.3	72	0.046 sekund
Zysk: 73.2 Trasa: [4, 2, 1, 2, 4] Numer iteracji: 10	Zysk: 21.0 Trasa: [1, 3, 0, 3, 1, 4, 1, 4, 1] Numer iteracji: 9	Zysk: 123.7 Trasa: [0, 4, 2, 4, 0, 3, 0, 2, 1, 3, 2, 4] Numer iteracji: 10	217.9	75	0.045 sekund
Zysk: 101.2 Trasa: [1, 3, 1, 3, 2, 4, 1, 2] Numer iteracji: 9	Zysk: 17.9 Trasa: [1, 4, 1, 3, 4, 1, 3, 1, 4] Numer iteracji: 5	Zysk: 112.5 Trasa: [4, 0, 2, 3, 2, 0, 3, 4, 1, 3, 2, 1] Numer iteracji: 5	231.6	76	0.048 sekund
Zysk: 68.4 Trasa: [0, 3, 1, 4, 2, 1] Numer iteracji: 6	Zysk: 8.6 Trasa: [0, 3, 0, 2, 3, 2, 0] Numer iteracji: 2	Zysk: 126.2 Trasa: [2, 1, 4, 1, 3, 1, 2, 4, 2, 1, 3, 0] Numer iteracji: 10	203.2	74	0.046 sekund
Zysk: 62.4 Trasa: [4, 2, 4, 0, 3, 1, 3] Numer iteracji: 1	Zysk: 53.3 Trasa: [3, 2, 1, 2, 1, 4, 1, 4] Numer iteracji: 5	Zysk: 109.8 Trasa: [2, 4, 0, 2, 4, 1, 2, 0, 3, 2, 4, 1, 4] Numer iteracji: 0	225.5	77	0.048 sekund
Zysk: 93.2 Trasa: [2, 3, 2, 4, 1, 3, 1, 3] Numer iteracji: 0	Zysk: 26.6 Trasa: [4, 2, 3, 2, 1, 2, 0] Numer iteracji: 10	Zysk: 87.7 Trasa: [2, 4, 1, 0, 2, 4, 0, 2, 3, 0] Numer iteracji: 9	207.5	75	0.046 sekund

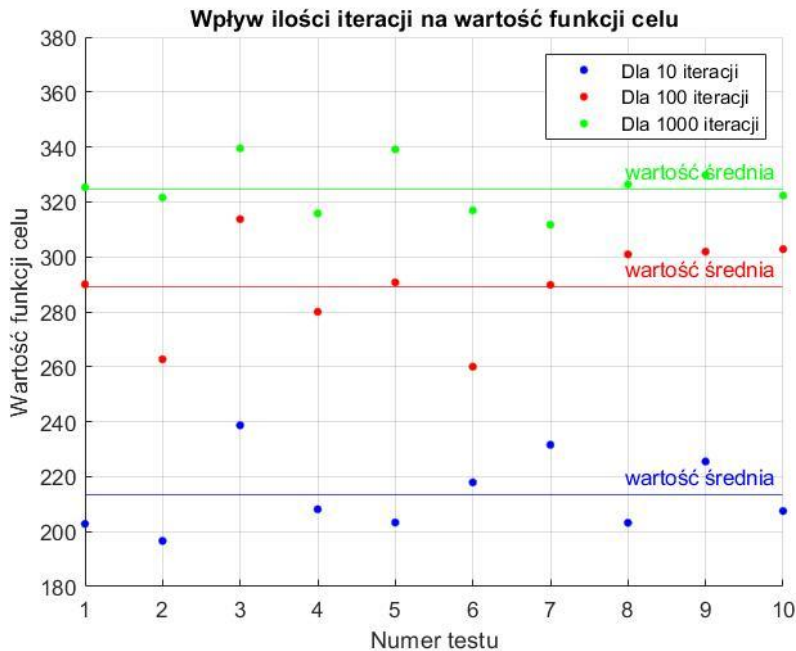
Liczba iteracji = 100  
Okres resetu listy Tabu = 20

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 94.6 Trasa: [4, 1, 3, 2, 4, 1, 3, 1] Numer iteracji: 42	Zysk: 41.7 Trasa: [4, 1, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 7	Zysk: 153.7 Trasa: [2, 1, 4, 1, 3, 2, 4, 0, 3, 2, 4, 1, 3, 0] Numer iteracji: 74	290.0	87	0.426 sekund
Zysk: 118.0 Trasa: [3, 2, 1, 2, 3, 1, 3] Numer iteracji: 12	Zysk: 33.6 Trasa: [1, 4, 1, 3, 0, 3, 2, 3, 2] Numer iteracji: 85	Zysk: 111.0 Trasa: [1, 2, 1, 3, 1, 4, 2, 0, 3, 1, 3, 2, 4] Numer iteracji: 91	262.7	86	0.409 sekund
Zysk: 131.8 Trasa: [2, 4, 1, 3, 2, 1, 3, 2] Numer iteracji: 37	Zysk: 52.2 Trasa: [3, 1, 3, 2, 3, 1, 4] Numer iteracji: 29	Zysk: 129.7 Trasa: [1, 3, 1, 4, 0, 3, 1, 2, 0, 3, 2, 4] Numer iteracji: 80	313.7	84	0.423 sekund
Zysk: 94.6 Trasa: [4, 1, 3, 2, 4, 1, 3, 1] Numer iteracji: 66	Zysk: 53.0 Trasa: [1, 2, 1, 3, 2, 1, 2, 3, 2] Numer iteracji: 91	Zysk: 132.4 Trasa: [1, 3, 2, 4, 0, 3, 1, 0, 2, 0, 3, 4] Numer iteracji: 56	280.0	87	0.41 sekund
Zysk: 133.0 Trasa: [2, 1, 3, 2, 1, 3, 1, 2] Numer iteracji: 91	Zysk: 46.5 Trasa: [1, 3, 2, 3, 2, 3, 1] Numer iteracji: 90	Zysk: 111.2 Trasa: [4, 1, 4, 2, 4, 0, 3, 2, 0, 3, 4] Numer iteracji: 63	290.7	87	0.403 sekund
Zysk: 114.0 Trasa: [4, 1, 2, 1, 3, 2, 1] Numer iteracji: 58	Zysk: 44.8 Trasa: [3, 2, 3, 2, 3, 2, 4, 1, 4] Numer iteracji: 91	Zysk: 101.2 Trasa: [0, 2, 4, 0, 3, 1, 4, 0, 3, 4, 1, 3, 1] Numer iteracji: 92	260.0	86	0.409 sekund
Zysk: 126.0 Trasa: [2, 1, 3, 2, 4, 1, 3, 2] Numer iteracji: 40	Zysk: 40.4 Trasa: [3, 1, 3, 0, 2, 4, 1, 3, 1, 3, 2] Numer iteracji: 33	Zysk: 123.3 Trasa: [4, 1, 3, 4, 0, 2, 3, 1, 4, 2, 4, 0] Numer iteracji: 30	289.8	88	0.409 sekund
Zysk: 118.6 Trasa: [2, 1, 3, 1, 2, 1, 4] Numer iteracji: 70	Zysk: 50.7 Trasa: [3, 1, 3, 1, 3, 1, 3, 2, 1, 3, 1] Numer iteracji: 88	Zysk: 131.6 Trasa: [1, 4, 0, 3, 1, 4, 1, 3, 0, 3, 2, 0] Numer iteracji: 93	300.9	87	0.576 sekund
Zysk: 99.6 Trasa: [4, 2, 1, 3, 1, 3, 2, 4] Numer iteracji: 35	Zysk: 52.9 Trasa: [3, 1, 2, 1, 3, 1, 3, 2, 0] Numer iteracji: 49	Zysk: 149.4 Trasa: [3, 2, 1, 4, 1, 3, 2, 3, 1, 4, 0, 3, 1, 4, 0] Numer iteracji: 97	301.9	89	0.442 sekund
Zysk: 133.2 Trasa: [1, 3, 1, 2, 1, 3, 1, 2] Numer iteracji: 35	Zysk: 39.4 Trasa: [1, 3, 2, 3, 1, 3, 2, 4] Numer iteracji: 86	Zysk: 130.2 Trasa: [4, 1, 3, 2, 0, 3, 1, 4, 1, 3, 0, 2, 4] Numer iteracji: 42	302.8	87	0.449 sekund

Liczba iteracji = 1000  
Okres resetu listy Tabu = 200

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 111.4 Trasa: [2, 1, 2, 3, 1, 4] Numer iteracji: 10	Zysk: 77.8 Trasa: [2, 1, 3, 2, 3, 1, 3, 2, 1, 3, 2] Numer iteracji: 465	Zysk: 136.0 Trasa: [3, 2, 4, 0, 3, 1, 3, 0, 2, 0, 3, 2, 4] Numer iteracji: 858	325.3	94	4.624 sekund
Zysk: 122.2 Trasa: [0, 2, 1, 3, 2, 1, 3, 1] Numer iteracji: 54	Zysk: 52.5 Trasa: [1, 4, 1, 3, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 377	Zysk: 146.9 Trasa: [3, 1, 4, 1, 3, 2, 4, 0, 3, 2, 4, 0, 3, 1, 4, 0] Numer iteracji: 482	321.6	94	4.622 sekund
Zysk: 155.8 Trasa: [1, 3, 2, 1, 3, 1, 3, 2, 1, 3] Numer iteracji: 33	Zysk: 39.4 Trasa: [2, 4, 1, 3, 2, 3, 1, 3, 2, 4] Numer iteracji: 438	Zysk: 144.2 Trasa: [3, 1, 4, 0, 2, 1, 4, 0, 3, 2, 4, 1, 3, 2, 0] Numer iteracji: 316	339.5	97	4.3 sekund
Zysk: 157.6 Trasa: [1, 3, 2, 1, 3, 2, 1, 3, 1, 3, 2] Numer iteracji: 410	Zysk: 22.3 Trasa: [0, 2, 3, 1, 3, 2, 4, 1, 4] Numer iteracji: 110	Zysk: 136.0 Trasa: [1, 3, 1, 4, 0, 3, 2, 4, 0, 2, 4, 1, 3, 0] Numer iteracji: 225	315.8	93	4.294 sekund
Zysk: 174.2 Trasa: [1, 3, 2, 1, 3, 1, 3, 2, 1, 3, 2] Numer iteracji: 248	Zysk: 28.0 Trasa: [3, 1, 4, 1, 4, 2, 1, 3, 2, 4] Numer iteracji: 847	Zysk: 136.9 Trasa: [3, 0, 3, 1, 0, 2, 4, 0, 3, 2, 4, 0] Numer iteracji: 18	339.1	98	4.295 sekund
Zysk: 133.0 Trasa: [3, 1, 2, 1, 3, 1, 2] Numer iteracji: 30	Zysk: 42.8 Trasa: [1, 3, 2, 4, 1, 3, 2, 1, 3, 2, 1, 3] Numer iteracji: 682	Zysk: 141.0 Trasa: [3, 2, 4, 0, 2, 0, 4, 1, 3, 0, 3, 1, 4] Numer iteracji: 386	316.9	93	4.9 sekund
Zysk: 131.8 Trasa: [2, 4, 1, 3, 2, 1, 3, 2] Numer iteracji: 228	Zysk: 37.4 Trasa: [3, 2, 1, 4, 0, 2, 3, 1, 3, 2] Numer iteracji: 241	Zysk: 142.5 Trasa: [2, 4, 1, 4, 0, 3, 1, 4, 2, 1, 3, 0] Numer iteracji: 195	311.7	90	4.789 sekund
Zysk: 154.8 Trasa: [1, 3, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 214	Zysk: 13.0 Trasa: [4, 1, 4, 2, 1, 3, 1, 2, 4] Numer iteracji: 907	Zysk: 158.6 Trasa: [3, 1, 4, 0, 2, 4, 0, 3, 2, 4, 0, 3, 1, 4, 0] Numer iteracji: 293	326.4	96	4.847 sekund
Zysk: 136.8 Trasa: [2, 1, 3, 2, 1, 3, 1, 3, 1] Numer iteracji: 58	Zysk: 52.5 Trasa: [1, 4, 1, 3, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 340	Zysk: 140.4 Trasa: [1, 3, 2, 0, 3, 1, 4, 0, 3, 2, 4, 1, 3, 2, 4, 0] Numer iteracji: 82	329.7	95	4.543 sekund
Zysk: 128.2 Trasa: [0, 3, 1, 3, 2, 1, 3, 1, 3, 2] Numer iteracji: 397	Zysk: 53.0 Trasa: [1, 3, 2, 1, 3, 2, 1, 3, 2, 1, 3, 2, 4] Numer iteracji: 536	Zysk: 141.1 Trasa: [3, 2, 4, 0, 2, 0, 4, 1, 4, 0, 3, 2, 4, 0] Numer iteracji: 691	322.3	95	4.378 sekund

## Wyniki



Wartość odniesienia: 342,0

Podsumowanie wyników dla 10 iteracji:

Wartość średnia funkcji celu: 213,52

Wartość maksymalna funkcji celu: 238,7

Wartość minimalna funkcji celu: 196,6

Podsumowanie wyników dla 100 iteracji:

Wartość średnia funkcji celu: 289,25

Wartość maksymalna funkcji celu: 313,7

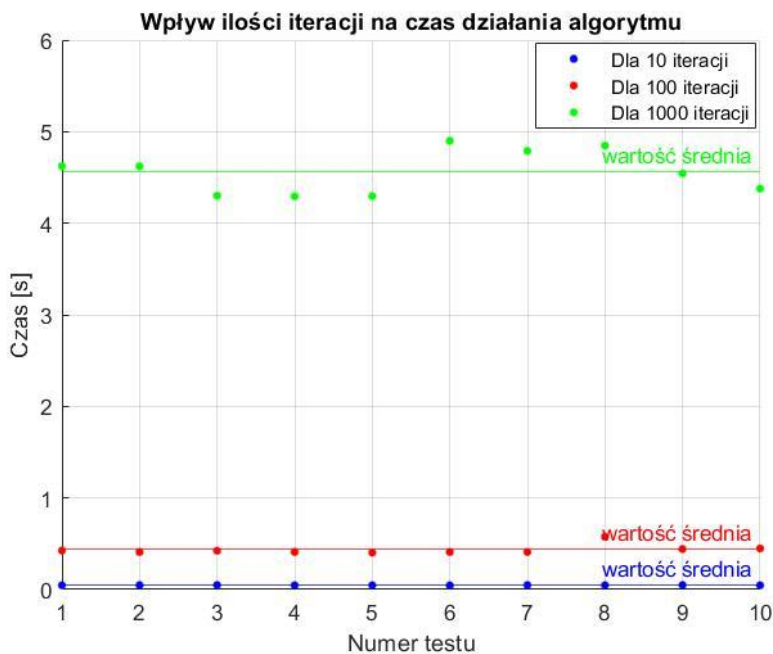
Wartość minimalna funkcji celu: 260,0

Podsumowanie wyników dla 1000 iteracji:

Wartość średnia funkcji celu: 324,83

Wartość maksymalna funkcji celu: 339,5

Wartość minimalna funkcji celu: 311,7



Podsumowanie wyników dla 10 iteracji:

Wartość średnia czasu: 0,047 s

Wartość maksymalna czasu: 0,048 s

Wartość minimalna czasu: 0,045 s

Podsumowanie wyników dla 100 iteracji:

Wartość średnia czasu: 0,436 s

Wartość maksymalna czasu: 0,576 s

Wartość minimalna czasu: 0,403 s

Podsumowanie wyników dla 1000 iteracji:

Wartość średnia czasu: 4,559 s

Wartość maksymalna czasu: 4,9 s

Wartość minimalna czasu: 4,294 s

---

Na podstawie powyższych wykresów można stwierdzić, że wzrost ilości iteracji znacząco wpływa na działanie algorytmu. Wśród badanych przypadków, dla 10 iteracji średnia wartość funkcji celu jest najmniejsza, natomiast dla 1000 iteracji jest największa. Wzrost ilości iteracji zazwyczaj pozwala zbliżyć się do wartości oczekiwanej. Zdarza się jednak, że dla 100 iteracji algorytm uzyska lepszy wynik niż dla 1000, co wynika głównie z losowości w trakcie poszukiwania rozwiązań. Badając czas wykonywania się algorytmu można zauważyć, że jest on niemal proporcjonalny do ilości iteracji. Działanie algorytmu jest więc zgodnie z przewidywaniami. Biorąc pod uwagę uzyskane wyniki, dalsze testy przeprowadzano dla 100 iteracji.

## Wpływ stosunku okresu resetu listy Tabu do ilości iteracji na działanie algorytmu

Liczba iteracji = 100

Okres resetu listy Tabu = 40

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 91.4 Trasa: [0, 3, 1, 4, 0, 3, 1, 3] Numer iteracji: 20	Zysk: 64.9 Trasa: [2, 1, 3, 2, 1, 2, 1, 3, 2] Numer iteracji: 17	Zysk: 125.0 Trasa: [1, 3, 0, 2, 4, 0, 3, 1, 2, 1, 3, 4] Numer iteracji: 78	281.3	86	0.439 sekund
Zysk: 108.8 Trasa: [2, 4, 1, 3, 1, 2, 0] Numer iteracji: 34	Zysk: 26.6 Trasa: [4, 0, 2, 1, 3, 1, 2, 3, 2] Numer iteracji: 25	Zysk: 136.4 Trasa: [0, 3, 4, 1, 3, 2, 0, 2, 4, 1, 3, 2, 4] Numer iteracji: 27	271.9	85	0.401 sekund
Zysk: 146.6 Trasa: [2, 1, 3, 1, 3, 1, 3, 2, 4] Numer iteracji: 62	Zysk: 36.0 Trasa: [2, 3, 1, 2, 3, 2, 1] Numer iteracji: 83	Zysk: 132.8 Trasa: [1, 4, 0, 2, 4, 0, 3, 0, 3, 4] Numer iteracji: 43	315.5	91	0.406 sekund
Zysk: 114.6 Trasa: [0, 3, 1, 3, 2, 1, 3, 1] Numer iteracji: 90	Zysk: 47.0 Trasa: [0, 2, 3, 2, 1, 2, 1, 3, 2] Numer iteracji: 83	Zysk: 121.0 Trasa: [2, 1, 3, 0, 2, 4, 1, 3, 4, 1, 3, 0] Numer iteracji: 25	282.6	88	0.412 sekund
Zysk: 93.8 Trasa: [0, 3, 2, 4, 1, 3, 1] Numer iteracji: 47	Zysk: 53.0 Trasa: [1, 2, 3, 2, 1, 2, 1] Numer iteracji: 16	Zysk: 134.9 Trasa: [1, 3, 0, 3, 1, 0, 2, 0, 3, 4] Numer iteracji: 95	281.7	87	0.479 sekund
Zysk: 158.2 Trasa: [3, 1, 3, 2, 1, 3, 2, 1, 4] Numer iteracji: 66	Zysk: 20.3 Trasa: [4, 1, 3, 2, 3, 2, 3] Numer iteracji: 29	Zysk: 122.3 Trasa: [3, 2, 4, 0, 3, 1, 2, 0, 3, 1, 2, 4] Numer iteracji: 36	300.8	87	0.466 sekund
Zysk: 127.0 Trasa: [2, 1, 3, 2, 1, 3, 1, 3] Numer iteracji: 38	Zysk: 34.4 Trasa: [3, 2, 1, 3, 2, 1, 4, 0, 2, 4] Numer iteracji: 89	Zysk: 122.9 Trasa: [2, 1, 4, 0, 3, 1, 0, 2, 0, 3, 2, 4] Numer iteracji: 19	284.3	84	0.421 sekund
Zysk: 141.6 Trasa: [3, 2, 1, 3, 1, 2, 4] Numer iteracji: 49	Zysk: 38.2 Trasa: [1, 3, 2, 1, 2, 3, 2, 3, 1] Numer iteracji: 89	Zysk: 132.6 Trasa: [2, 1, 4, 1, 2, 4, 0, 3, 0, 3, 4] Numer iteracji: 94	312.4	94	0.434 sekund
Zysk: 125.6 Trasa: [3, 1, 3, 2, 4, 1, 3, 1, 2] Numer iteracji: 28	Zysk: 43.9 Trasa: [1, 3, 2, 3, 2, 1, 3, 2, 1] Numer iteracji: 35	Zysk: 130.6 Trasa: [3, 4, 1, 3, 4, 0, 3, 2, 0, 3, 2, 1, 4] Numer iteracji: 71	300.1	91	0.409 sekund
Zysk: 133.2 Trasa: [1, 3, 1, 2, 1, 3, 1, 2] Numer iteracji: 36	Zysk: 31.0 Trasa: [0, 2, 3, 2, 3, 1, 3, 4] Numer iteracji: 40	Zysk: 128.2 Trasa: [2, 4, 2, 4, 0, 3, 0, 2, 1, 3, 4] Numer iteracji: 20	292.4	88	0.405 sekund

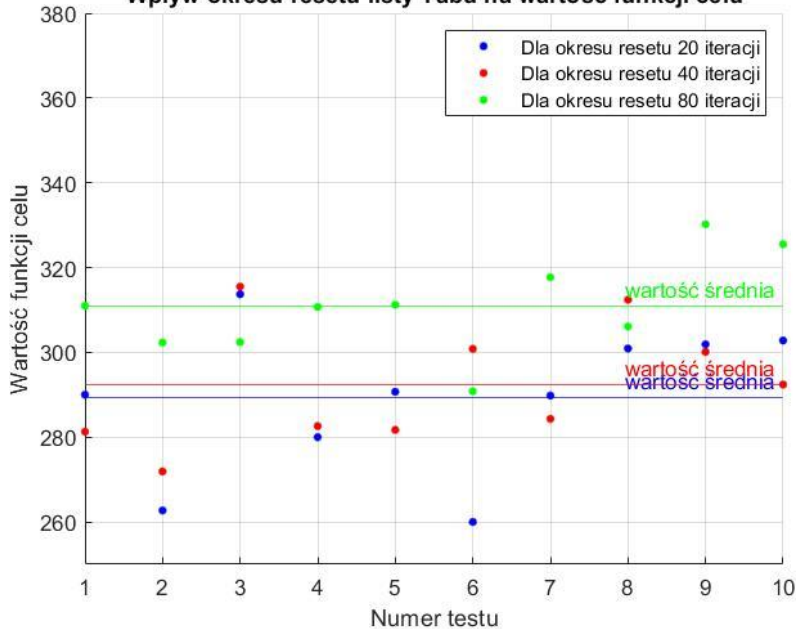


Liczba iteracji = 100  
Okres resetu listy Tabu = 80

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 113.8 Trasa: [2, 4, 1, 3, 1, 3, 1, 3, 2] Numer iteracji: 90	Zysk: 47.0 Trasa: [1, 2, 1, 3, 2, 3, 2, 1, 4] Numer iteracji: 57	Zysk: 150.2 Trasa: [3, 1, 4, 0, 3, 1, 4, 0, 2, 1, 3, 1, 2, 0] Numer iteracji: 66	311.0	90	0.419 sekund
Zysk: 171.0 Trasa: [3, 1, 4, 1, 2, 1, 3, 2] Numer iteracji: 98	Zysk: 28.3 Trasa: [1, 3, 1, 3, 2, 1, 3, 2, 3, 1] Numer iteracji: 91	Zysk: 103.0 Trasa: [0, 3, 4, 0, 3, 2, 4, 0, 2, 1, 3, 0] Numer iteracji: 49	302.3	88	0.647 sekund
Zysk: 144.4 Trasa: [1, 2, 1, 3, 1, 2, 4] Numer iteracji: 44	Zysk: 42.7 Trasa: [3, 1, 3, 2, 1, 3, 2, 3, 2] Numer iteracji: 57	Zysk: 115.2 Trasa: [3, 0, 4, 0, 3, 2, 4, 0, 2, 3, 2, 1, 4] Numer iteracji: 56	302.4	89	0.415 sekund
Zysk: 137.0 Trasa: [3, 2, 3, 1, 3, 2, 1, 4] Numer iteracji: 17	Zysk: 42.3 Trasa: [2, 3, 2, 1, 3, 2, 1, 3, 1] Numer iteracji: 30	Zysk: 131.4 Trasa: [2, 4, 1, 0, 3, 1, 0, 2, 3, 1, 4] Numer iteracji: 54	310.7	90	0.415 sekund
Zysk: 112.4 Trasa: [0, 3, 1, 3, 2, 1, 3, 2] Numer iteracji: 59	Zysk: 44.2 Trasa: [1, 4, 0, 3, 1, 3, 2, 3, 1, 4] Numer iteracji: 54	Zysk: 154.5 Trasa: [3, 2, 4, 0, 2, 4, 0, 3, 2, 4, 1, 3, 2, 0] Numer iteracji: 99	311.2	91	0.537 sekund
Zysk: 148.2 Trasa: [1, 3, 2, 1, 3, 1, 2, 0] Numer iteracji: 21	Zysk: 43.4 Trasa: [1, 3, 2, 3, 1, 2, 3, 1, 4] Numer iteracji: 85	Zysk: 99.2 Trasa: [4, 2, 1, 3, 0, 2, 4, 1, 3, 2, 4, 0] Numer iteracji: 49	290.8	89	0.475 sekund
Zysk: 135.8 Trasa: [2, 4, 1, 3, 2, 1, 3, 1] Numer iteracji: 66	Zysk: 41.9 Trasa: [4, 1, 3, 2, 3, 2, 1, 3, 2] Numer iteracji: 78	Zysk: 140.0 Trasa: [3, 4, 1, 4, 1, 2, 4, 0, 2, 1, 3, 2, 0] Numer iteracji: 36	317.7	92	0.406 sekund
Zysk: 138.4 Trasa: [1, 4, 1, 3, 1, 3, 1, 2, 4] Numer iteracji: 48	Zysk: 39.8 Trasa: [3, 1, 3, 2, 0, 3, 1, 3, 2, 3, 2] Numer iteracji: 50	Zysk: 127.9 Trasa: [0, 2, 0, 2, 1, 3, 0, 2, 3, 1, 4] Numer iteracji: 25	306.1	92	0.416 sekund
Zysk: 127.6 Trasa: [2, 1, 3, 1, 2, 1, 3, 2] Numer iteracji: 20	Zysk: 39.6 Trasa: [1, 3, 1, 3, 0, 2, 3, 1, 3, 1] Numer iteracji: 90	Zysk: 163.0 Trasa: [1, 4, 0, 2, 0, 3, 0, 2, 4, 1, 3, 2, 4] Numer iteracji: 98	330.2	97	0.413 sekund
Zysk: 171.0 Trasa: [3, 1, 4, 1, 2, 1, 3, 2] Numer iteracji: 83	Zysk: 21.6 Trasa: [2, 4, 1, 3, 1, 4, 0, 3, 1, 3, 2] Numer iteracji: 40	Zysk: 132.9 Trasa: [2, 1, 3, 2, 4, 0, 3, 4, 0, 2, 1, 4, 0] Numer iteracji: 100	325.5	94	0.416 sekund

## Wyniki

Wpływ okresu resetu listy Tabu na wartość funkcji celu



Wartość odniesienia: 342,0

\* Podsumowanie wyników dla okresu 20:

Wartość średnia funkcji celu: 289,25

Wartość maksymalna funkcji celu: 313,7

Wartość minimalna funkcji celu: 260,0

Podsumowanie wyników dla okresu 40:

Wartość średnia funkcji celu: 292,3

Wartość maksymalna funkcji celu: 315,5

Wartość minimalna funkcji celu: 271,9

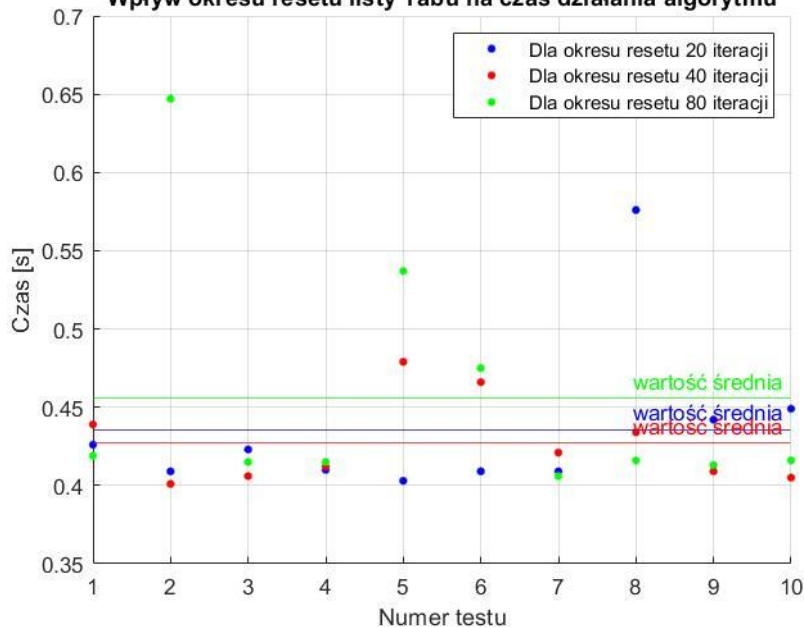
Podsumowanie wyników dla okresu 80:

Wartość średnia funkcji celu: 310,79

Wartość maksymalna funkcji celu: 330,2

Wartość minimalna funkcji celu: 290,8

Wpływ okresu resetu listy Tabu na czas działania algorytmu



\* Podsumowanie wyników dla okresu 20:

Wartość średnia czasu: 0,436 s

Wartość maksymalna czasu: 0,576 s

Wartość minimalna czasu: 0,403 s

Podsumowanie wyników dla okresu 40:

Wartość średnia czasu: 0,427 s

Wartość maksymalna czasu: 0,479 s

Wartość minimalna czasu: 0,401 s

Podsumowanie wyników dla okresu 80:

Wartość średnia czasu: 0,456 s

Wartość maksymalna czasu: 0,647 s

Wartość minimalna czasu: 0,406 s

\* Wyniki dla okresu resetu równego 20 zostały przedstawione we wcześniejszym punkcie.

---

Na podstawie uzyskanych wykresów można stwierdzić, że okres resetu listy Tabu wpływa na działanie algorytmu, jednak w stopniu mniejszym niż ma to miejsce przy zmianie ilości iteracji. Dla okresu resetu 20 i 40 różnica w średniej wartości funkcji celu jest niewielka, natomiast dla 80 już nieco większa. Wzrost okresu resetu powoduje polepszenie średniej wartości funkcji celu, jednak w poszczególnych testach zdarza się, że mniejszy okres daje lepszy wynik. Analizując czas wykonywania algorytmu nie można wyciągnąć jednoznacznych wniosków, bowiem najdłużej wykonywał się algorytm przy okresie resetu 80, a najkrócej przy 40. Im większy okres, tym ciężiej powinno być znaleźć rozwiązanie, które jeszcze nie znajduje się na liście Tabu, w związku z czym uzyskane wyniki dla okresu resetu 20 i 40 są zaskakujące. Różnice te są jednak niewielkie i na uzyskany wynik w znaczącym stopniu mogło mieć wpływ np. aktualne obciążenie komputera. W związku z powyższym, dobór okresu resetu do dalszych testów odbywał się na bieżąco i zależał przede wszystkim od długości rozwiązań.

## Wpływ ograniczeń na działanie algorytmu

### 1) Ograniczenie dotyczące udźwigu kierowcy

W celu przetestowania algorytmu wprowadzono następujące zmiany w pierwszej instancji testowej:

Liczba iteracji = 100

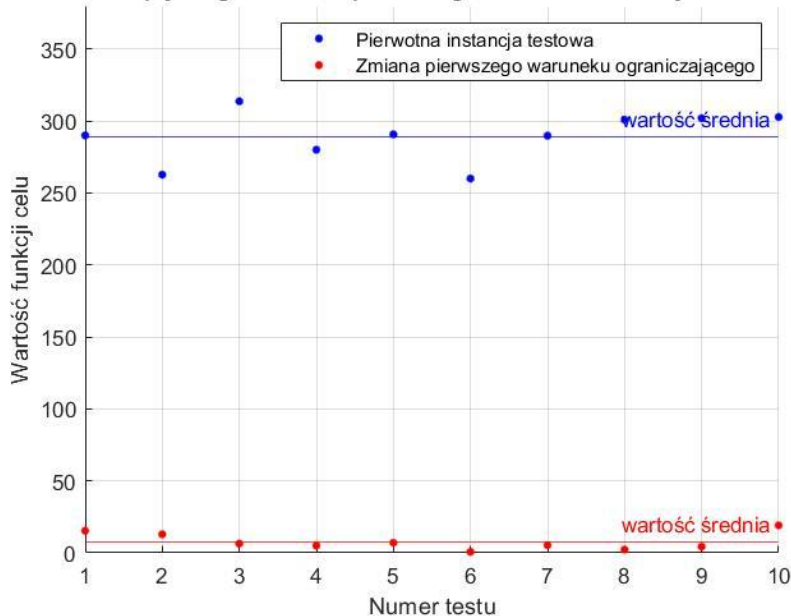
Okres resetu listy Tabu = 20

	Prędkość dostawy	Pojemność pojazdów	Koszt użytkowania pojazdu
<b>Rower</b>	15	20 -> <b>5</b>	0
<b>Skuter</b>	30	30 -> <b>7.5</b>	0.05
<b>Hulajnoga elektryczna</b>	20	10 -> <b>2.5</b>	0.005

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 5.4 Trasa: [0, 3, 1, 3, 1, 3] Numer iteracji: 56	Zysk: -23.4 Trasa: [4, 1, 3, 2, 4, 1, 3, 2] Numer iteracji: 70	Zysk: 33.3 Trasa: [2, 1, 3, 2, 1, 2, 3, 2, 3, 2] Numer iteracji: 52	15.3	31	0.407 sekund
Zysk: 8.8 Trasa: [1, 3, 1, 3, 2, 1, 3, 1] Numer iteracji: 96	Zysk: -25.3 Trasa: [1, 4, 1, 3, 2, 3, 1, 4] Numer iteracji: 37	Zysk: 29.4 Trasa: [2, 1, 3, 2, 3, 2, 3, 1, 2, 1, 3, 1, 3] Numer iteracji: 89	12.9	37	0.442 sekund
Zysk: 10.4 Trasa: [2, 3, 2, 1, 3, 1, 3, 1] Numer iteracji: 73	Zysk: -23.2 Trasa: [3, 2, 4, 1, 3, 1, 3, 1] Numer iteracji: 42	Zysk: 19.2 Trasa: [2, 1, 2, 3, 1, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 72	6.4	36	0.451 sekund
Zysk: 23.0 Trasa: [1, 3, 2, 1, 3, 1, 3, 1, 3] Numer iteracji: 98	Zysk: -29.3 Trasa: [2, 4, 1, 3, 2, 4, 1, 4] Numer iteracji: 38	Zysk: 11.2 Trasa: [0, 3, 1, 4, 1, 3, 1, 2, 1, 3, 2, 3, 1, 3, 1] Numer iteracji: 53	5.0	39	0.538 sekund
Zysk: 8.8 Trasa: [1, 3, 1, 3, 2, 1, 3, 1] Numer iteracji: 59	Zysk: -23.4 Trasa: [1, 4, 0, 3, 1, 3] Numer iteracji: 89	Zysk: 21.5 Trasa: [4, 1, 3, 2, 3, 2, 4, 1, 3, 2, 1, 3, 2, 1, 3, 2] Numer iteracji: 64	7.0	33	0.481 sekund
Zysk: 4.0 Trasa: [1, 2, 1, 2, 1, 3, 2] Numer iteracji: 0	Zysk: -24.9 Trasa: [2, 4, 1, 3, 1, 3, 4] Numer iteracji: 26	Zysk: 21.5 Trasa: [2, 4, 1, 3, 2, 1, 3, 2, 1, 4, 1, 3, 2] Numer iteracji: 62	0.7	33	0.416 sekund
Zysk: 8.4 Trasa: [2, 4, 1, 3, 1, 3, 1] Numer iteracji: 24	Zysk: -26.6 Trasa: [0, 3, 1, 3, 1, 3, 2, 4] Numer iteracji: 96	Zysk: 23.5 Trasa: [1, 2, 1, 2, 3, 2, 1, 2, 3, 1] Numer iteracji: 55	5.3	34	0.414 sekund
Zysk: 21.4 Trasa: [1, 3, 1, 3, 1, 3] Numer iteracji: 85	Zysk: -19.2 Trasa: [2, 4, 0, 3, 2] Numer iteracji: 14	Zysk: 0.1 Trasa: [3, 1, 3, 4, 0, 2, 1, 3, 2, 3] Numer iteracji: 47	2.3	23	0.416 sekund
Zysk: 0.6 Trasa: [1, 3, 1, 3, 2, 1, 2] Numer iteracji: 65	Zysk: -29.9 Trasa: [4, 1, 3, 1, 3, 2, 3, 1, 3] Numer iteracji: 21	Zysk: 33.6 Trasa: [2, 3, 2, 3, 2, 1, 3, 1, 3, 1, 2, 1, 3, 2] Numer iteracji: 84	4.3	37	0.434 sekund
Zysk: 1.0 Trasa: [2, 3, 1, 3, 2, 1, 3, 2] Numer iteracji: 14	Zysk: -23.6 Trasa: [3, 2, 1, 3, 1, 2] Numer iteracji: 21	Zysk: 41.6 Trasa: [2, 1, 3, 1, 3, 1, 2, 1, 3, 1, 3, 1, 3] Numer iteracji: 95	19.1	34	0.56 sekund

## Wyniki

Wpływ ograniczenia pierwszego na wartość funkcji celu



Wartość odniesienia: 342,0

\* Podsumowanie wyników pierwszej instancji testowej:

Wartość średnia funkcji celu: 289,25

Wartość maksymalna funkcji celu: 313,7

Wartość minimalna funkcji celu: 260,0

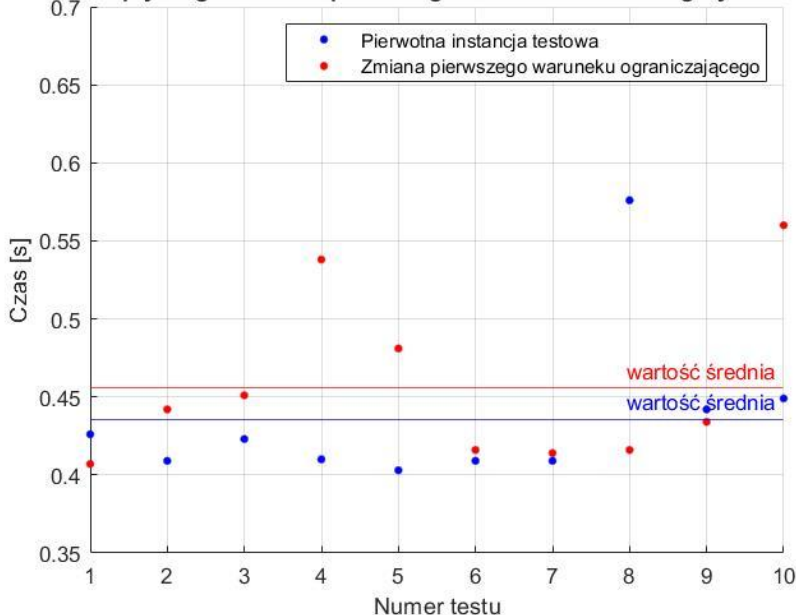
Podsumowanie wyników dla testu ograniczenia:

Wartość średnia funkcji celu: 7,83

Wartość maksymalna funkcji celu: 19,1

Wartość minimalna funkcji celu: 0,7

Wpływ ograniczenia pierwszego na czas działania algorytmu



\* Podsumowanie wyników pierwszej instancji testowej:

Wartość średnia czasu: 0,436 s

Wartość maksymalna czasu: 0,576 s

Wartość minimalna czasu: 0,403 s

Podsumowanie wyników dla testu ograniczenia:

Wartość średnia czasu: 0,456 s

Wartość maksymalna czasu: 0,560 s

Wartość minimalna czasu: 0,407 s

\* Wyniki dla pierwszej instancji testowej zostały przedstawione we wcześniejszym punkcie.

Na podstawie uzyskanych wykresów można stwierdzić, że zmiana wartości udźwigu dostawcy ma wpływ na wartość funkcji celu. Czterokrotne zmniejszenie wartości udźwignięcia

kierowców poszczególnych rodzajów pojazdów powoduje prawie 37-krotne zmniejszenie średniej wartości funkcji celu dla niezmiennych pozostałych wartości i obiektów badanej instancji testowej, a tym samym dla niezmiennych wartości odniesienia. Badanie czasu wykonywania algorytmu pozwala na stwierdzenie, że podobnie jak w przypadku okresu resetu listy Tabu, różnice są nieznaczne i najprawdopodobniej spowodowane aktualnym obciążeniem komputera.

## 2) Ograniczenie dotyczące czasu pracy dostawców

W celu przetestowania algorytmu wprowadzono następujące zmiany w pierwszej instancji testowej:

Liczba iteracji = 100

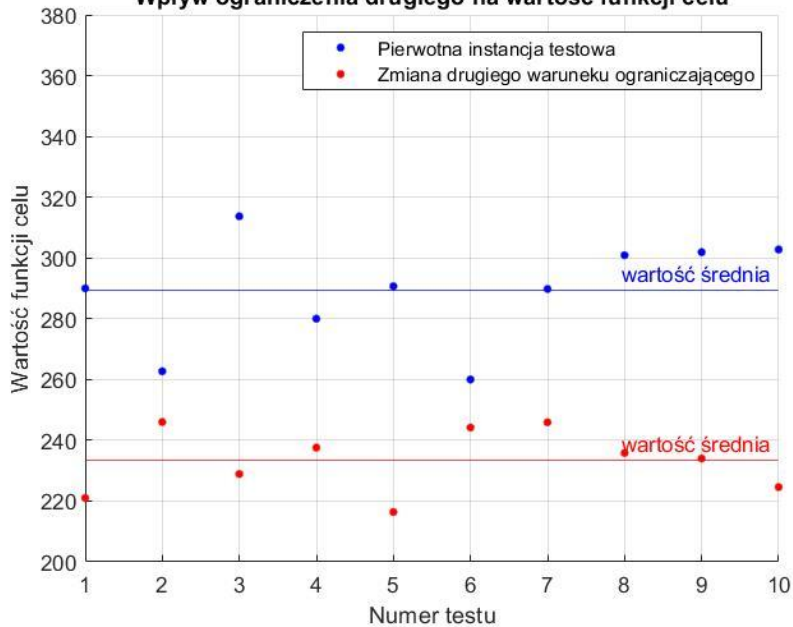
Okres resetu listy Tabu = 20

Czas pracy każdego dostawcy: 4 -> 2

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 83.2 Trasa: [1, 2, 1, 3] Numer iteracji: 36	Zysk: 20.0 Trasa: [0, 3, 2, 1, 3, 2] Numer iteracji: 75	Zysk: 117.8 Trasa: [0, 3, 1, 4, 0, 3, 2, 4] Numer iteracji: 93	221.0	59	0.567 sekund
Zysk: 108.8 Trasa: [1, 3, 2, 1, 3, 2] Numer iteracji: 25	Zysk: 19.0 Trasa: [2, 3, 1, 2] Numer iteracji: 11	Zysk: 118.1 Trasa: [3, 1, 3, 2, 4, 0, 3, 1, 4] Numer iteracji: 76	246.0	65	0.465 sekund
Zysk: 83.2 Trasa: [1, 2, 1, 3] Numer iteracji: 3	Zysk: 23.6 Trasa: [0, 3, 2, 3, 2] Numer iteracji: 69	Zysk: 122.0 Trasa: [2, 4, 0, 3, 1, 3, 4] Numer iteracji: 18	228.9	61	0.458 sekund
Zysk: 83.2 Trasa: [1, 2, 1, 3] Numer iteracji: 9	Zysk: 39.4 Trasa: [1, 3, 2, 1, 3, 1, 4] Numer iteracji: 41	Zysk: 115.0 Trasa: [3, 1, 4, 2, 3, 2, 4] Numer iteracji: 84	237.6	63	0.42 sekund
Zysk: 79.4 Trasa: [3, 1, 3, 4] Numer iteracji: 21	Zysk: 11.8 Trasa: [4, 2, 1, 3, 2] Numer iteracji: 15	Zysk: 125.2 Trasa: [3, 1, 2, 0, 3, 2, 4] Numer iteracji: 64	216.4	59	0.569 sekund
Zysk: 78.4 Trasa: [3, 1, 3, 0] Numer iteracji: 7	Zysk: 27.4 Trasa: [0, 3, 2, 1, 3, 2] Numer iteracji: 77	Zysk: 138.3 Trasa: [3, 1, 2, 3, 1, 4] Numer iteracji: 42	244.2	61	0.438 sekund
Zysk: 78.4 Trasa: [3, 1, 3, 0] Numer iteracji: 0	Zysk: 39.6 Trasa: [3, 2, 3, 1, 3, 2] Numer iteracji: 30	Zysk: 127.9 Trasa: [3, 1, 2, 1, 3, 0] Numer iteracji: 52	245.9	63	0.413 sekund
Zysk: 99.4 Trasa: [3, 2, 1, 3, 1] Numer iteracji: 20	Zysk: 24.0 Trasa: [2, 1, 3, 2, 3] Numer iteracji: 16	Zysk: 112.4 Trasa: [2, 4, 1, 4, 2, 0] Numer iteracji: 75	235.8	62	0.525 sekund
Zysk: 83.2 Trasa: [1, 2, 1, 3] Numer iteracji: 11	Zysk: 39.4 Trasa: [1, 3, 2, 1, 3, 1, 4] Numer iteracji: 89	Zysk: 111.4 Trasa: [2, 4, 1, 3, 2, 1, 3, 4] Numer iteracji: 92	234.0	63	0.417 sekund
Zysk: 83.2 Trasa: [1, 2, 1, 3] Numer iteracji: 12	Zysk: 43.8 Trasa: [3, 2, 3, 1, 4] Numer iteracji: 25	Zysk: 97.6 Trasa: [2, 4, 0, 3, 2, 3, 1] Numer iteracji: 51	224.6	62	0.44 sekund

## Wyniki

Wpływ ograniczenia drugiego na wartość funkcji celu



\* Podsumowanie wyników pierwszej instancji testowej:

Wartość odniesienia: 342,0

Wartość średnia funkcji celu: 289,25

Wartość maksymalna funkcji celu: 313,7

Wartość minimalna funkcji celu: 260,0

Podsumowanie wyników dla testu ograniczenia:

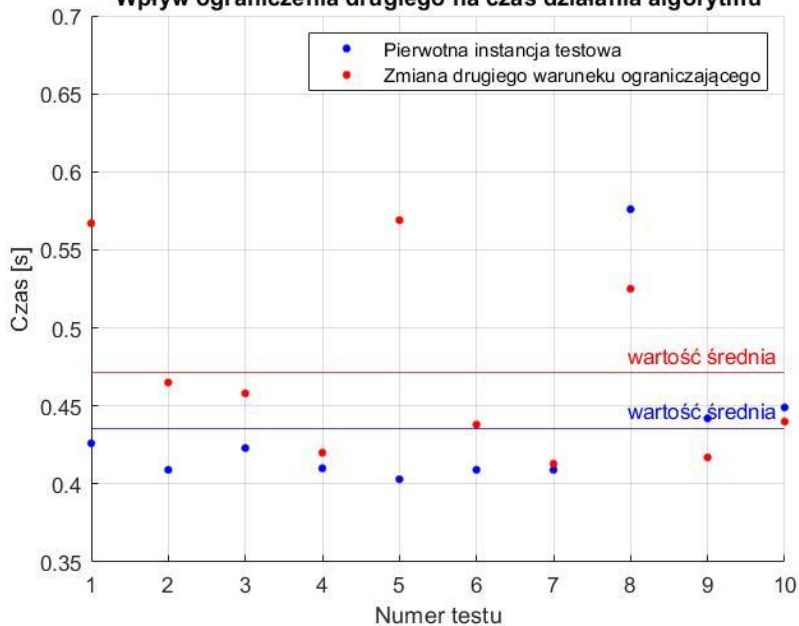
Wartość odniesienia: 449,2

Wartość średnia funkcji celu: 233,44

Wartość maksymalna funkcji celu: 246,0

Wartość minimalna funkcji celu: 216,4

Wpływ ograniczenia drugiego na czas działania algorytmu



\* Podsumowanie wyników pierwszej instancji testowej:

Wartość średnia czasu: 0,436 s

Wartość maksymalna czasu: 0,576 s

Wartość minimalna czasu: 0,403 s

Podsumowanie wyników dla testu ograniczenia:

Wartość średnia czasu: 0,471 s

Wartość maksymalna czasu: 0,569 s

Wartość minimalna czasu: 0,413 s

\* Wyniki dla pierwszej instancji testowej zostały przedstawione we wcześniejszym punkcie.

Na podstawie uzyskanych wykresów można stwierdzić, że zmiana wartości czasu pracy dostawców ma wpływ na wartość funkcji celu. Dwukrotne zmniejszenie czasu pracy

dostawców powoduje 1,24-krotne zmniejszenie średniej wartości funkcji celu dla niezmiennych pozostałych wartości i obiektów badanej instancji testowej. Zmiana ta powoduje jednak także zmianę wartości odniesienia, a w konsekwencji stosunek średniej wartości funkcji celu do wartości odniesienia uległ zmianie z 0,85 na 0,52. Badanie czasu wykonywania algorytmu ponownie pozwoliło na stwierdzenie, że powyższa zmiana powoduje nieznaczne różnice, najprawdopodobniej spowodowane czynnikami zewnętrznymi.

### 3) Ograniczenie dotyczące maksymalnego czasu dostawy

W celu przetestowania algorytmu wprowadzono następujące zmiany w pierwotnej instancji testowej:

Liczba iteracji = 100

Okres resetu listy Tabu = 20

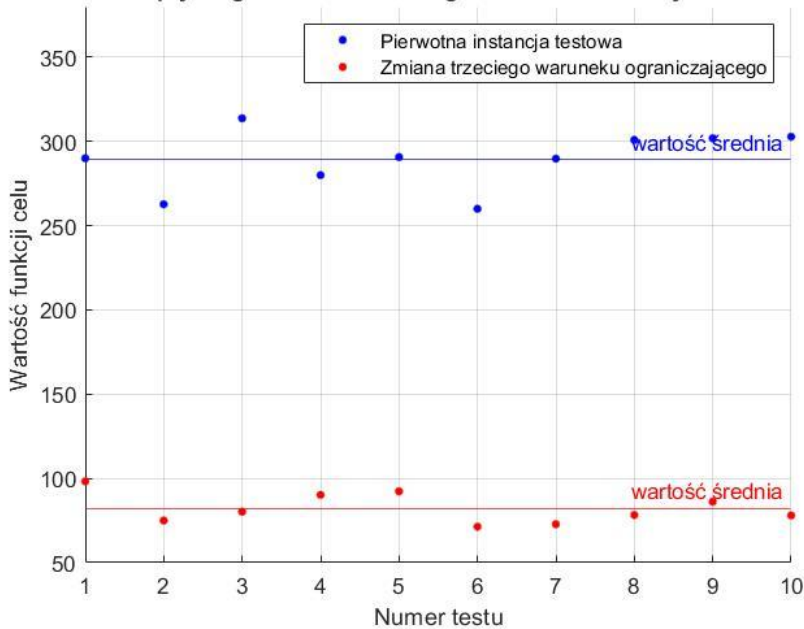
Przedział maksymalnego czasu dostawy: 1-2 -> **0.1 - 0.5**

Kierowca 1	Kierowca 2	Kierowca 3	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
Zysk: 33.2 Trasa: [4, 1, 3, 2, 4] Numer iteracji: 11	Zysk: 19.7 Trasa: [3, 2, 4, 1, 4, 1, 3, 1, 2] Numer iteracji: 23	Zysk: 45.4 Trasa: [1, 3, 1, 2, 1, 3, 1, 4, 0, 4, 1, 3, 2, 4] Numer iteracji: 54	98.2	48	0.468 sekund
Zysk: 35.2 Trasa: [4, 1, 3, 2, 3, 2] Numer iteracji: 58	Zysk: 8.9 Trasa: [4, 0, 3, 2, 4, 1, 3, 1, 4] Numer iteracji: 6	Zysk: 30.9 Trasa: [1, 3, 1, 2, 4, 1, 3, 2, 4, 2, 1, 4, 0, 2] Numer iteracji: 55	75.0	48	0.444 sekund
Zysk: 32.6 Trasa: [3, 2, 4, 1, 3, 2, 4, 1, 4] Numer iteracji: 19	Zysk: 13.1 Trasa: [1, 3, 2, 4, 0, 4, 0, 3, 1, 4] Numer iteracji: 91	Zysk: 34.6 Trasa: [2, 1, 4, 0, 3, 2, 4, 0, 3, 1, 3, 2] Numer iteracji: 77	80.2	48	0.49 sekund
Zysk: 37.0 Trasa: [4, 1, 4, 1, 3, 2] Numer iteracji: 14	Zysk: 20.2 Trasa: [4, 1, 3, 1, 4, 1, 3, 2] Numer iteracji: 58	Zysk: 33.0 Trasa: [4, 0, 2, 4, 1, 3, 2, 4, 1, 4, 0, 3, 1, 4] Numer iteracji: 46	90.2	47	0.565 sekund
Zysk: 29.8 Trasa: [4, 1, 3, 2, 4, 1, 4] Numer iteracji: 73	Zysk: 30.5 Trasa: [0, 3, 1, 4, 1, 3, 1, 3, 2, 4] Numer iteracji: 67	Zysk: 32.1 Trasa: [4, 1, 3, 1, 3, 2, 1, 4, 1, 3, 2, 4] Numer iteracji: 50	92.3	47	0.457 sekund
Zysk: 37.4 Trasa: [3, 1, 3, 2, 1, 4, 1, 3, 2] Numer iteracji: 60	Zysk: 6.3 Trasa: [2, 3, 1, 3, 1, 3, 2, 1, 4] Numer iteracji: 23	Zysk: 27.6 Trasa: [2, 4, 1, 3, 2, 3, 1, 4, 0, 2, 1, 3, 1, 4] Numer iteracji: 85	71.3	48	0.491 sekund
Zysk: 36.0 Trasa: [4, 1, 3, 2, 1, 3, 2] Numer iteracji: 44	Zysk: 9.2 Trasa: [4, 2, 4, 1, 4, 1, 3, 1, 4] Numer iteracji: 74	Zysk: 27.5 Trasa: [3, 1, 3, 2, 1, 4, 1, 0, 3, 1, 3, 2, 4] Numer iteracji: 79	72.8	47	0.454 sekund
Zysk: 46.4 Trasa: [1, 3, 2, 1, 3, 2] Numer iteracji: 21	Zysk: 4.4 Trasa: [4, 0, 3, 1, 3, 2, 4, 1, 4, 2] Numer iteracji: 56	Zysk: 27.5 Trasa: [4, 1, 4, 0, 2, 4, 1, 3, 1, 3, 1, 4, 1] Numer iteracji: 71	78.3	46	0.456 sekund
Zysk: 25.2 Trasa: [2, 1, 3, 2, 1, 3] Numer iteracji: 80	Zysk: 27.3 Trasa: [1, 3, 2, 3, 1, 4, 2, 4] Numer iteracji: 25	Zysk: 33.7 Trasa: [0, 3, 1, 4, 0, 3, 2, 4, 1, 3, 2, 1] Numer iteracji: 71	86.2	47	0.448 sekund
Zysk: 28.2 Trasa: [4, 1, 4, 1, 3, 1, 3, 2] Numer iteracji: 53	Zysk: 23.1 Trasa: [4, 1, 3, 1, 3, 2, 4, 1, 4, 0] Numer iteracji: 90	Zysk: 26.8 Trasa: [4, 1, 3, 2, 4, 1, 4, 0, 3, 2, 1, 3, 1, 3] Numer iteracji: 52	78.0	48	0.563 sekund



## Wyniki

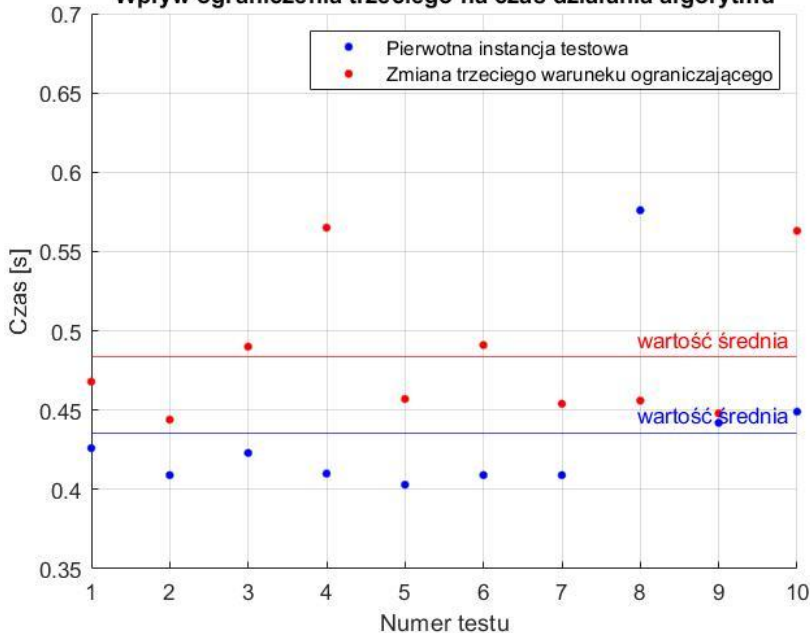
Wpływ ograniczenia trzeciego na wartość funkcji celu



\* Podsumowanie wyników pierwszej instancji testowej:  
Wartość odniesienia: 342,0  
Wartość średnia funkcji celu: 289,25  
Wartość maksymalna funkcji celu: 313,7  
Wartość minimalna funkcji celu: 260,0

Podsumowanie wyników dla testu ograniczenia:  
Wartość odniesienia: 335,0  
Wartość średnia funkcji celu: 82,25  
Wartość maksymalna funkcji celu: 98,2  
Wartość minimalna funkcji celu: 71,3

Wpływ ograniczenia trzeciego na czas działania algorytmu



\* Podsumowanie wyników pierwszej instancji testowej:  
Wartość średnia czasu: 0,436 s  
Wartość maksymalna czasu: 0,576 s  
Wartość minimalna czasu: 0,403 s

Podsumowanie wyników dla testu ograniczenia:  
Wartość średnia czasu: 0,484 s  
Wartość maksymalna czasu: 0,565 s  
Wartość minimalna czasu: 0,444 s

\* Wyniki dla pierwszej instancji testowej zostały przedstawione we wcześniejszym punkcie.

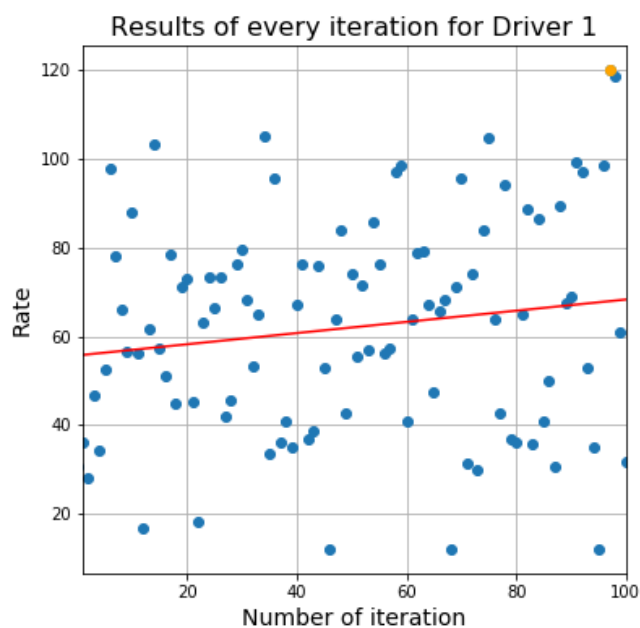
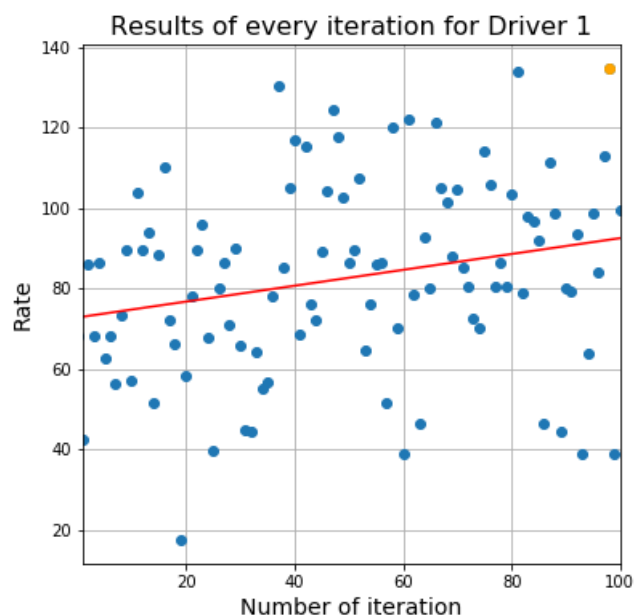
---

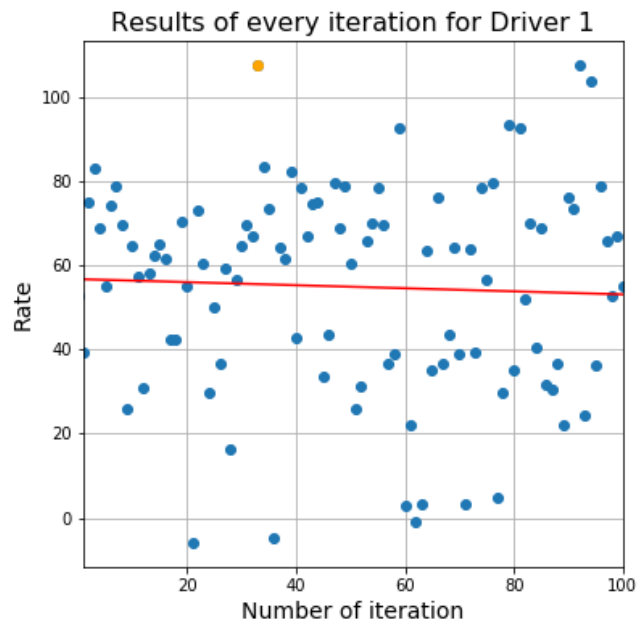
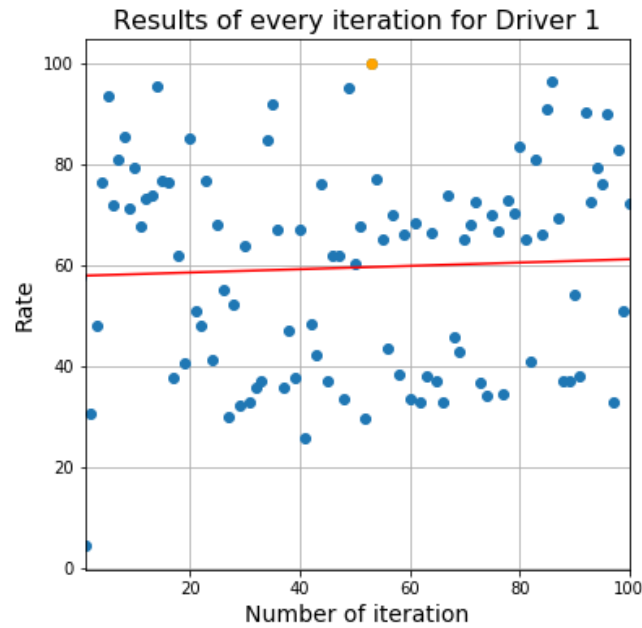
Na podstawie uzyskanych wykresów można stwierdzić, że zmiana maksymalnego czasu dostawy ma wpływ na wartość funkcji celu. Zmniejszenie przedziału maksymalnego czasu dostawy powoduje zmniejszenie średniej wartości funkcji celu dla niezmiennych pozostałych wartości instancji testowej. Zmiana ta wymusiła zmiany w liście zamówień, a w konsekwencji także zmianę wartości odniesienia, co zmniejszyło stosunek średniej wartości funkcji celu do wartości odniesienia z 0,85 na 0,25. Badanie czasu wykonywania algorytmu ponownie pozwoliło na stwierdzenie, że powyższa zmiana powoduje nieznaczne różnice, które najprawdopodobniej są spowodowane czynnikami zewnętrznymi.

---

## Badanie zmian wartości funkcji celu w kolejnych iteracjach

W celu przetestowania algorytmu stworzono instancję testową, o parametrach takich jak w pierwszej instancji testowej, dla jednego dostawcy. Przykładowe przebiegi zmian wartości funkcji celu w kolejnych iteracjach wraz z linią trendu oraz zaznaczonym najlepszym znalezionym rozwiązaniem przedstawiono poniżej:





Na podstawie powyższych wykresów można stwierdzić, że algorytm działa zadowalająco. Na dwóch pierwszych przypadkach widać, że linia trendu dość silnie wzrasta. Zdarzają się jednak sytuacje, kiedy to rośnie tylko nieznacznie, lub nawet maleje w kolejnych iteracjach. Można także zaobserwować, że najlepsze znalezione rozwiązanie niekoniecznie musi znajdować się w jednej z ostatnich iteracji i może być znalezione stosunkowo szybko.

---

## II Instancja testowa

### Wpływ zwiększenia rozmiaru problemu na działanie algorytmu

Liczba iteracji = 100

Okres resetu listy Tabu = 40

#### Mapa:

Liczba punktów na mapie: 100

Liczba punktów odbioru: 20

#### Zamówienia:

Ilość zamówień: 2000

Przedział maksymalnego czasu dostawy: 1-2

Minimalna waga = 1

Maksymalna waga = 5

Cena za jednostkę wagi: 2

#### Dostawcy i pojazdy:

	Prędkość dostawy	Pojemność pojazdów	Koszt użytkowania pojazdu
<b>Rower</b>	15	40	0
<b>Skuter</b>	30	60	0.05
<b>Hulajnoga elektryczna</b>	20	30	0.005

Opłata dla dostawcy: 0.8

Czas pracy każdego dostawcy: 8

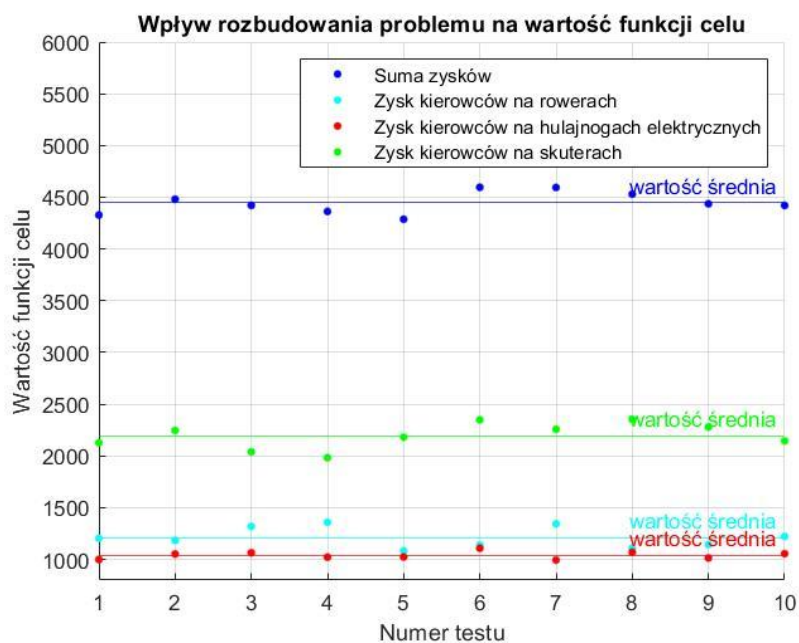
Liczba użytkowanych rowerów: 3

Liczba użytkowanych skuterów: 3

Liczba użytkowanych hulajnóg elektrycznych: 3

### Wyniki

Suma zysków kierowców na rowerach	Suma zysków kierowców na hulajnogach elektrycznych	Suma zysków kierowców na skuterach	Suma zysków	Liczba wykonanych zamówień	Czas wykonywania algorytmu
1202.6	998.6	2126.5	4327.7	930	38.599 sekund
1182.6	1052.0	2245.9	4480.6	961	39.486 sekund
1318.4	1063.8	2038.9	4421.2	954	36.333 sekund
1358.6	1021.0	1982.8	4362.4	948	36.666 sekund
1082.4	1023.2	2181.5	4287.1	934	36.213 sekund
1140.0	1107.2	2348.3	4595.6	972	39.006 sekund
1344.4	993.0	2256.8	4594.2	984	38.297 sekund
1107.8	1068.6	2353.4	4529.8	972	45.399 sekund
1142.8	1013.4	2279.6	4435.8	945	37.689 sekund
1221.6	1054.6	2145.0	4421.2	944	39.213 sekund



Podsumowanie wyników:

Wartość odniesienia: 10666,8

Wartość średniej sumy zysków: 4445,56

Stosunek średniej sumy zysków do wartości odniesienia: 0,42

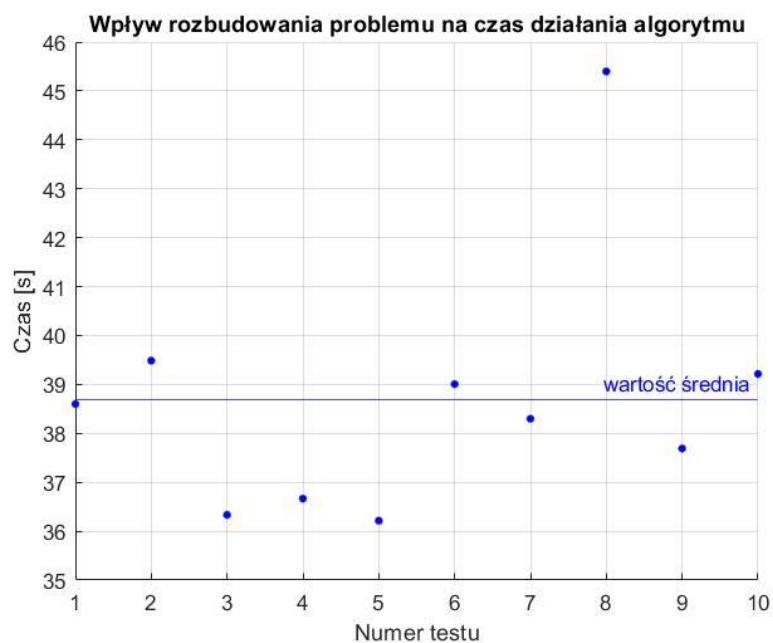
Średnia ilość wykonanych zamówień: 954

Stosunek ilość wykonanych zamówień do łącznej liczby zamówień: 0,48

Wartość sumy zysków kierowców na rowerach: 1210,12

Wartość sumy zysków kierowców na hulajnogach elektrycznych: 1039,54

Wartość sumy zysków kierowców na skuterach: 2195,87



Podsumowanie wyników:

Wartość średnia czasu: 38,69 s

Wartość maksymalna czasu: 45,40 s

Wartość minimalna czasu: 36,21 s

---

Na podstawie uzyskanych wykresów można stwierdzić, że algorytm radzi sobie również z problemami o większym rozmiarze. Dla badanej instancji testowej stosunek średniej sumy zysków do wartości odniesienia wyniósł 0,42, natomiast stosunek ilości wykonanych zamówień do łącznej liczby zamówień 0,48. Można zatem stwierdzić, że dla odpowiednio dobranych danych wejściowych, algorytm działa poprawnie. Należy brać pod uwagę, że średni zysk zamówień które zostały zrealizowane mógł być mniejszy niż zamówień niezrealizowanych, przy stałych kosztach dostaw, co jest głównym czynnikiem powodującym tą niewielką rozbieżność. Czasy wykonywania algorytmu dla tego samego problemu zazwyczaj mają zbliżony czas, natomiast mogą pojawić się odstępstwa np. na skutek chwilowego obciążenia sprzętu na którym algorytm jest wykonywany.

---

## Postać użytkowa algorytmu - GUI

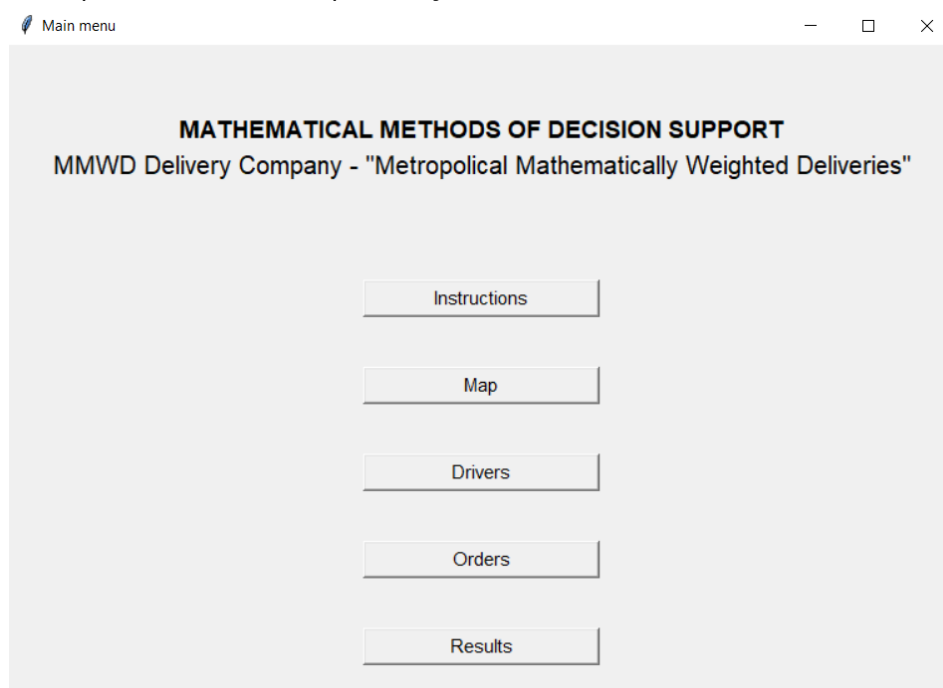
W celu uzyskania postaci użytkowej programu obejmującego powyższe zagadnienie, stworzono aplikację pozwalającą na interaktywną edycję danych wejściowych oraz przedstawienie uzyskanych wyników działania algorytmu.

Aby ją uruchomić należy otworzyć plik **MMWD\_Application.exe**

Aplikacja składa się z kilku menu, każdego odpowiedzialnego za osobną kwestię - wczytywanie partii danych bądź wyświetlanie wyniku działania algorytmu na danym jego etapie.

### Menu Główne - Main Menu

Pierwsze z nich, przedstawione na poniższym zrzucie ekranu:



Służy ono głównie do sprawnego przemieszczania się pomiędzy oknami oraz jako wizytówka aplikacji. Jest w nim umieszczone pięć przycisków przekierowuje kolejno do:

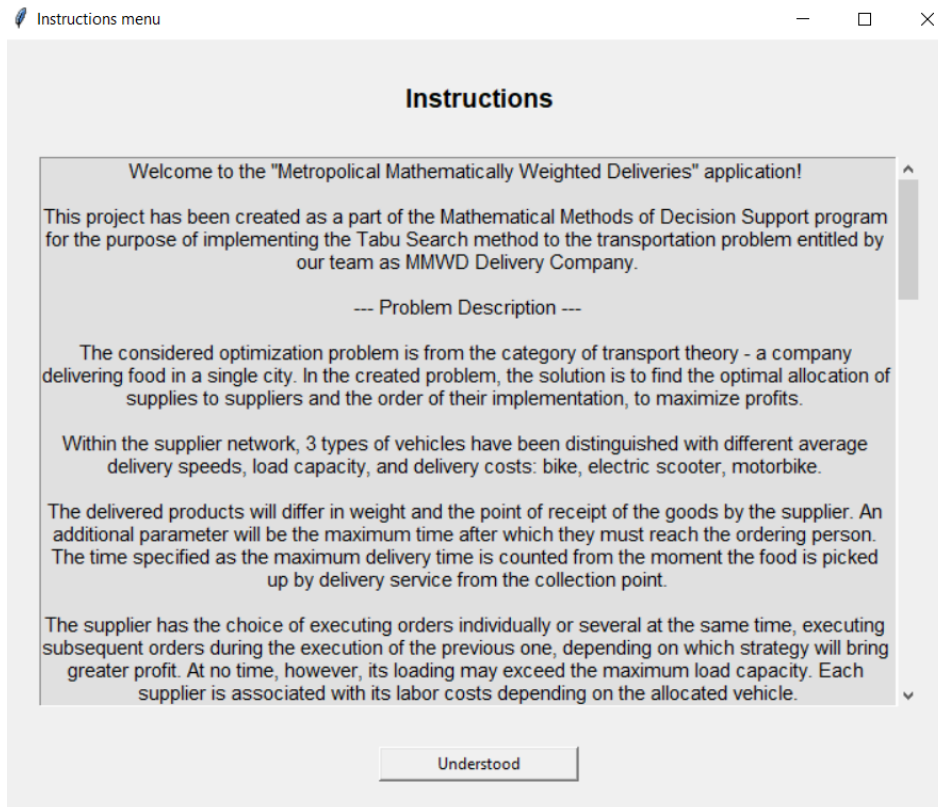
- instrukcji obsługi aplikacji wraz z krótkim opisem problemu,
- generatora mapy,
- modyfikacji listy kierowców,
- tworzenia puli zamówień,
- podglądu wyników działania programu.

Ważną kwestią jest przechodzenie po oknach w kolejności w jakiej zostały one umieszczone w menu, ponieważ bez poszczególnych danych wejściowych dalsze kroki generacji danych nie są możliwe w przypadku obsługiwanego algorytmu.



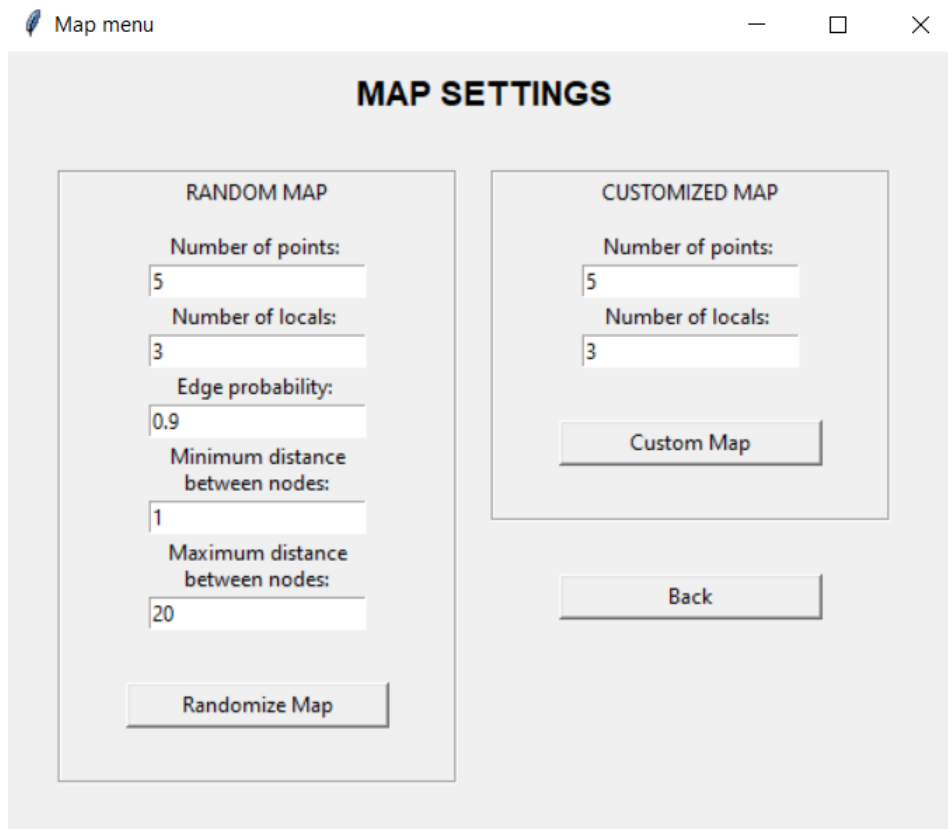
## Menu instrukcji - Instructions Menu

Okno zawiera krótki opis problematyki związanej z realizowanym zagadnieniem oraz szczegółowy opis postępowania w celu prawidłowej obsługi aplikacji. Na końcu dodano zakres datowy pracy nad zagadnieniem oraz autorów. Zamknięcie okna następuje po naciśnięciu przycisku wyrażającego zapoznanie się z instrukcją "Understood".



## Menu mapy - Map Menu

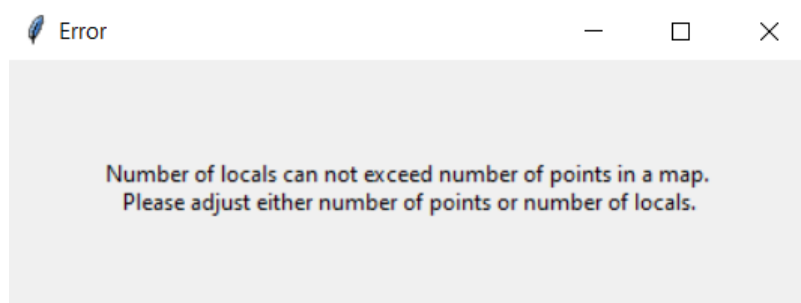
Pojawiające się po uruchomieniu drugiego w kolejności przycisku, umożliwia użytkownikowi dwie opcje - losową generację mapy bądź utworzenie jej samodzielnie. Dane wprowadzane są w przygotowanych do tego polach znajdujących się w zależności od preferencji generacji mapy po prawej lub lewej stronie ekranu.



The screenshot shows a window titled "Map menu" with a standard OS title bar (minimize, maximize, close buttons). The main content area is titled "MAP SETTINGS" and is divided into two panels. The left panel, "RANDOM MAP", contains five input fields: "Number of points:" (value 5), "Number of locals:" (value 3), "Edge probability:" (value 0.9), "Minimum distance between nodes:" (value 1), and "Maximum distance between nodes:" (value 20). Below these fields is a "Randomize Map" button. The right panel, "CUSTOMIZED MAP", contains two input fields: "Number of points:" (value 5) and "Number of locals:" (value 3). Below these fields is a "Custom Map" button. At the bottom of the right panel is a "Back" button.

W przypadku uzupełnienia wszystkich luk następuje przekierowanie do kolejnego menu. Z ustawień mapy można wyjść poprzez użycie załączonego przycisku powrotu. Dla uniemożliwienia błędnego wprowadzenia danych dodano komunikaty ostrzegające o nieprawidłowym zamieszczeniu danych, przedstawione poniżej.

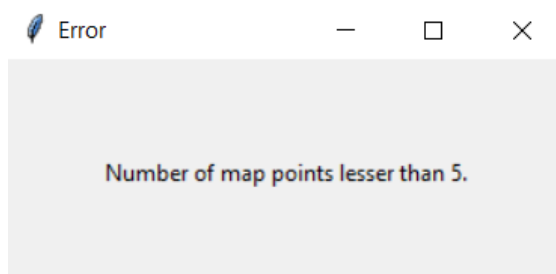
Ostrzeżenie w przypadku podania ilości punktów odbioru większej niż rozmiar mapy:



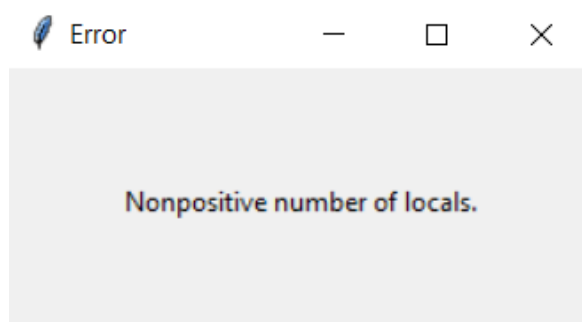
The screenshot shows an "Error" dialog box with a standard OS title bar. The message inside reads: "Number of locals can not exceed number of points in a map. Please adjust either number of points or number of locals."

---

Do prawidłowego działania algorytmu konieczna jest mapa o wielkości minimum pięciu punktów, co opisane zostało również w instrukcji użytkownika. Dlatego też w przypadku, gdy użytkownik poda ich mniejszą ilość również wyświetlane jest stosowne ostrzeżenie:

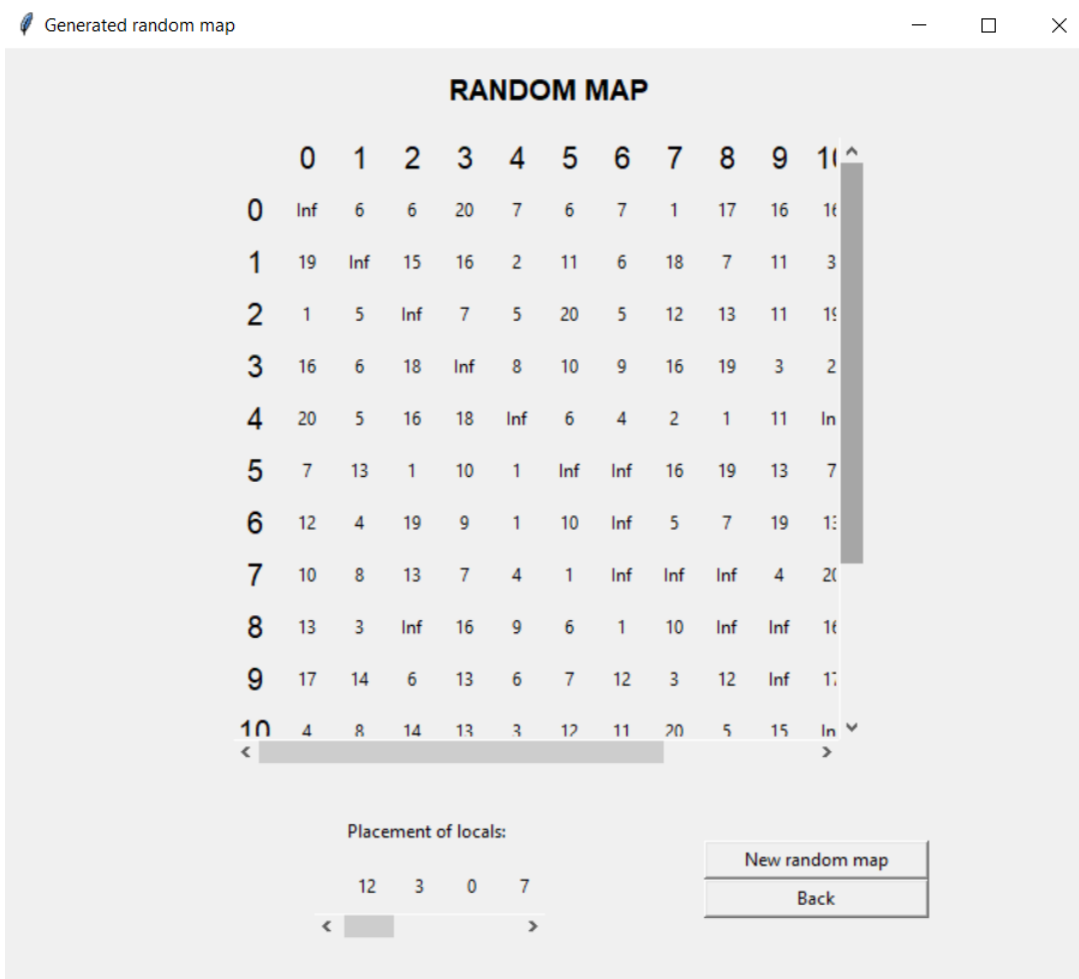


Dodatkowo, zabezpieczono aplikację przed przyjęciem ujemnych wartości parametrów mapy załączając informację o polu, w którym nastąpiła pomyłka:



## Generacja mapy losowej

Po przejściu do menu tworzenia mapy losowej dostajemy gotową mapę, wykreowaną przy pomocy algorytmu o parametrach zdefiniowanych w menu mapy przez użytkownika, oraz listę lokali. Zarówno mapa jak i lista lokali wyposażone zostały w opcję suwaka aby w sytuacji ich większego rozmiaru umożliwić swobodny podgląd całości danych. Menu posiada również opcję odświeżenia wartości równocześnie mapy oraz lokali, generującą nowe losowe dane na podstawie tych samych parametrów wejściowych. Standardowo u dołu ekranu dostępny jest przycisk powrotu.



## Generacja mapy personalizowanej

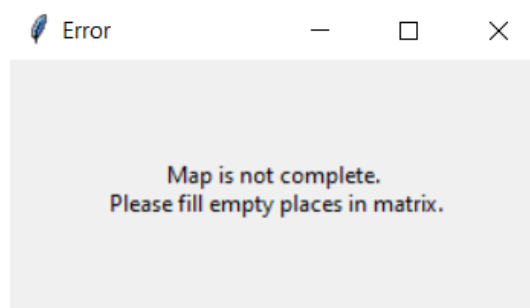
Dla wybranej opcji mapy wpisywanej ręcznie generowane jest okno ze szkieletem mapy tworzonym na podstawie danych wejściowych. W ten sposób po wybraniu wielkości mapy oraz ilości lokali użytkownik ma pełną niezależność jeśli chodzi o wypełnienie macierzy. Przejścia na diagonalu zostały zabronione. Użytkownik również ma możliwość zabronienia przejścia między punktami poprzez wpisanie wartości niedodatniej w pole macierzy. W przypadku pomyłki skorzystać można z opcji wymazania całkowicie wypełnienia zarówno macierzy mapy jak i listy lokali.

The screenshot shows a window titled "CUSTOM MAP" with a 10x10 matrix for defining map transitions. The columns are labeled 0 through 10, and the rows are labeled 0 through 10. The diagonal elements (0,0), (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8), and (9,9) are pre-filled with "Inf". The other cells are empty input boxes. Below the matrix, there is a section labeled "Placement of locals:" with a small grid of four input boxes and a horizontal slider. To the right of this section are three buttons: "Save", "Clear", and "Back".

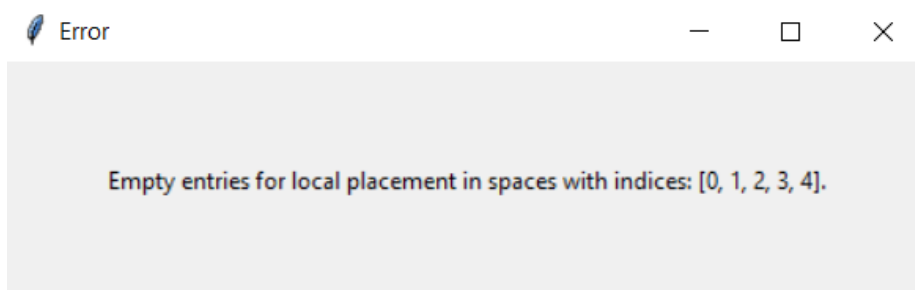
W tym przypadku w celu uniknięcia błędów również dodano ostrzeżenia o nieprawidłowościach we wpisanych danych.

---

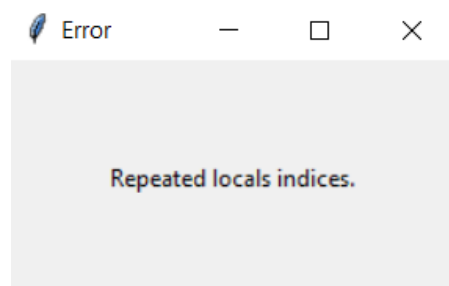
Pierwszym przypadkiem błędu jest nie podanie wszystkich wartości w polu macierzy. Użytkownik jest wtedy proszony o uzupełnienie mapy.



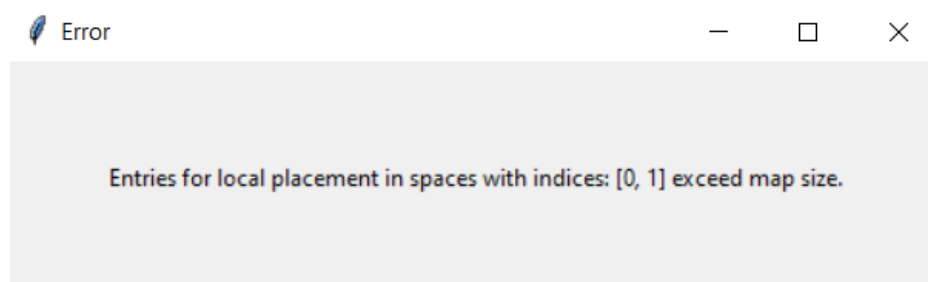
Podobnym błędem jest nie uzupełnienie listy lokali, ostrzeżenie działa w tym przypadku podobnie, dodatkowo jednak podając index pola nieuzupełnionego.



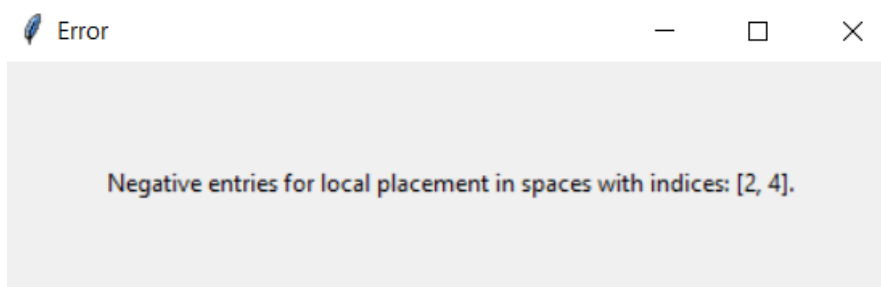
Zabronione jest również powtórzenie indeksów lokali, co zmieniałoby wielkość rozważanego problemu.



Indeksy lokali nie mają prawa przekraczać wielkości mapy - punkty te nie istnieją w przestrzeni mapy.



Podobny komunikat powoduje podanie ujemnej wartości indeksu:

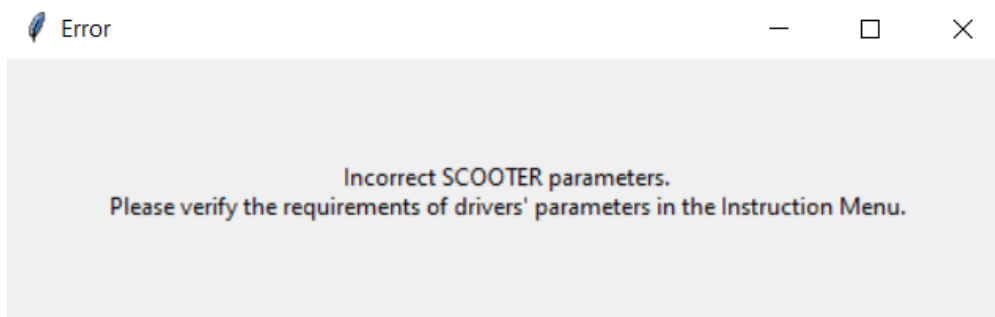


## Menu kierowców - Drivers Menu

Okno służy generacji listy dostawców będących kluczowym elementem działania algorytmu. Podzielone zostało na kilka części, każda odpowiadająca innemu zestawowi danych wejściowych. Kolejno od góry znajdują się bloki zawierające parametry pojazdów - rowera, hulajnogi elektrycznej oraz skutera. Kolejność ich umieszczenia nie jest przypadkowa, lista kierowców tworzona jest z posegregowaniem poszczególnych rodzajów kierowców w zależności od rodzaju pojazdu. Dla każdego pojazdu wybierana jest jego prędkość, udźwig, koszty użytkowania oraz ilość pojazdów danego typu. Dodatkowymi ustawieniami są koszt kierowcy oraz całkowity czas pracy dostarczycieli. Okno zostało wzbogacone prostymi grafikami.

A screenshot of a web application window titled 'Drivers menu'. The window has a title bar with a feather icon and standard window controls. The main content area is titled 'DRIVERS SETTINGS' and contains three sections for vehicle types: BIKE, SCOOTER, and MOTORBIKE. Each section has input fields for Speed, Capacity, Fuel cost, and Quantity, followed by a corresponding icon. The BIKE section has values 15, 20, 0, and 1. The SCOOTER section has values 20, 10, 0.005, and 1. The MOTORBIKE section has values 30, 30, 0.05, and 1. Below these sections is an 'OTHER' section with input fields for Driver cost (0.8) and Work time (4). At the bottom right are 'Save' and 'Back' buttons.

Również te ustawienia zostały stosownie zabezpieczone przed wprowadzeniem niepoprawnych danych. Komunikat podaje nazwę rubryki, w której popełniono błąd oraz przekierowuje do zapoznania się ze szczegółowymi wymaganiami wprowadzanych danych w instrukcji.



## Menu zamówień - Orders Menu

Po przejściu do okna zamówień ukazują nam się miejsca przeznaczone do wpisania parametrów generowanych zgodnie z algorytmem losowym zamówień. Są to między innymi: koszt jednostki wagowej zamówienia, liczba zamówień, zakres maksymalnego czasu dostawy oraz zakres wagowy zamówień. Poniżej, dla przypomnienia, wyświetla się aktualne umieszczenie lokali, będących możliwymi punktami początkowymi zamówień.

Przycisk Save/Refresh służy do generacji losowej listy zamówień lub w przypadku ponownego użycia, zastąpienia starej listy nową, o podanych parametrach. Parametry zamówień w przypadku długiej listy można przesuwac tym samym umożliwiając użytkownikowi podgląd do wszystkich z nich bez konieczności powiększania okna.

Dodatkową opcją jest ręczne wpisanie listy zamówień. Opcję tę umożliwia przycisk znajdujący się u dołu okna. Po jego naciśnięciu użytkownik przeniesiony zostaje do osobnego menu.



Custom orders menu

CUSTOM ORDERS

Unit price:  Number of orders:  Generate blank pattern

Empty canvas will appear here.

Placement of locals:  

0 4 3

Save/Refresh

Clear

Back

W górnej części okna widoczne są dwie luki konieczne do uzupełnienia przez użytkownika przed rozpoczęciem pracy nad listą zamówień. Są to ilość zamówień oraz cena za jednostkę wagową towaru. U dołu ekranu dla przypomnienia jak w poprzednim oknie podano listę lokali. W przypadku podania koniecznych parametrów oraz zatwierdzenia ich przyciskiem po prawej stronie, wygenerowany zostaje szkielet zadanej ilości zamówień, gotowy do uzupełnienia przez użytkownika. Każda z kolumn odpowiada innemu parametrowi kolejnych zamówień, zgodnie z ich podpisami.

Custom orders menu

### CUSTOM ORDERS

Unit price:  Number of orders:  Generate blank pattern

Number	Delivery Time	Weight	Start Point	End Point
0	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
1	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
2	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
3	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
4	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
5	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
6	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
7	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
8	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
9	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
10	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Placement of locals:

0 4 3

Save/Refresh  
Clear  
Back

Przypomnienie indeksów lokali jest w tym przypadku kluczowe, ponieważ podanie punktu startowego dla któregośkolwiek z zamówień wykraczającego poza pulę punktów odbioru, uznane zostaje za błąd. Punkty końcowe zamówień ponownie nie mogą przekraczać zakresu mapy, a czas oraz waga zamówień nie mogą być ujemne. W przypadku niespełnienia któregośkolwiek z założeń wyświetlony zostaje komunikat o błędzie zaznaczający numer zamówienia, w którym popełniono błąd.

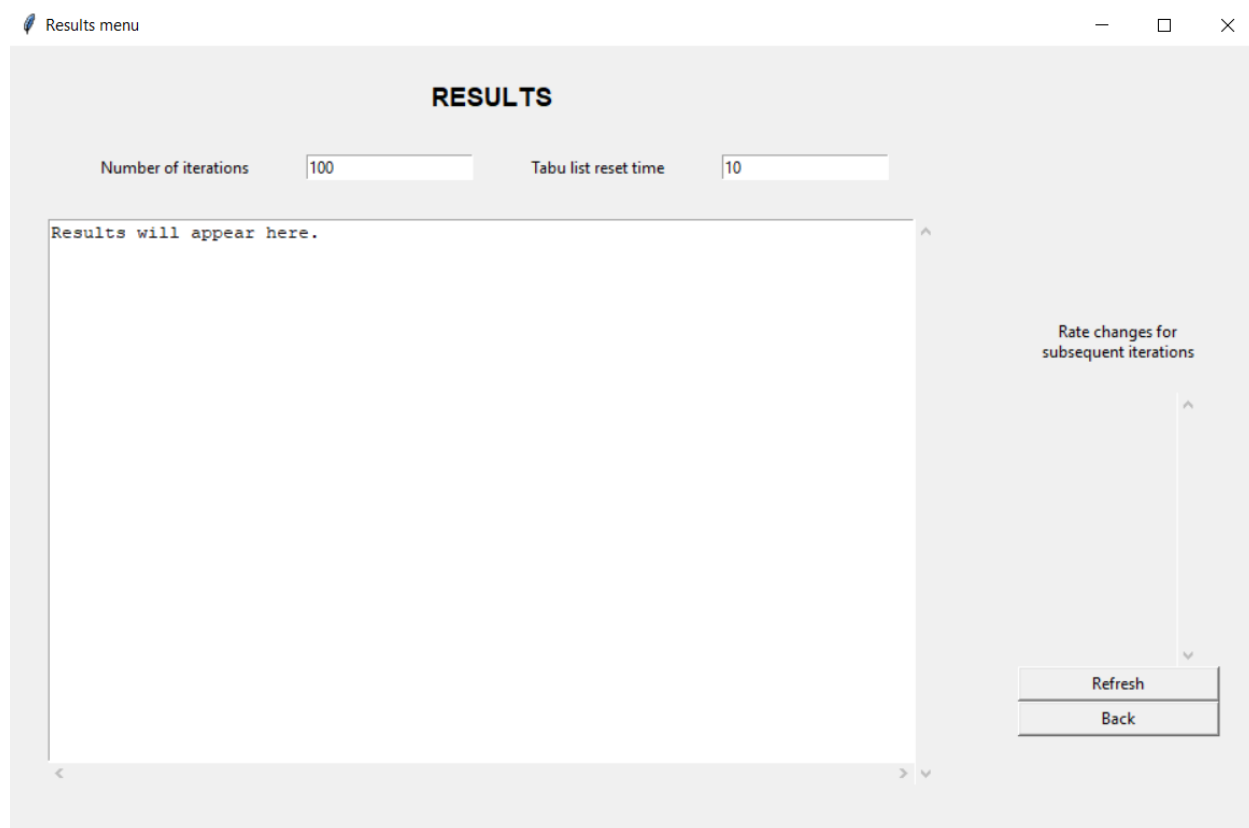
Error

Incorrect order parameters for orders with numbers: [1, 3].

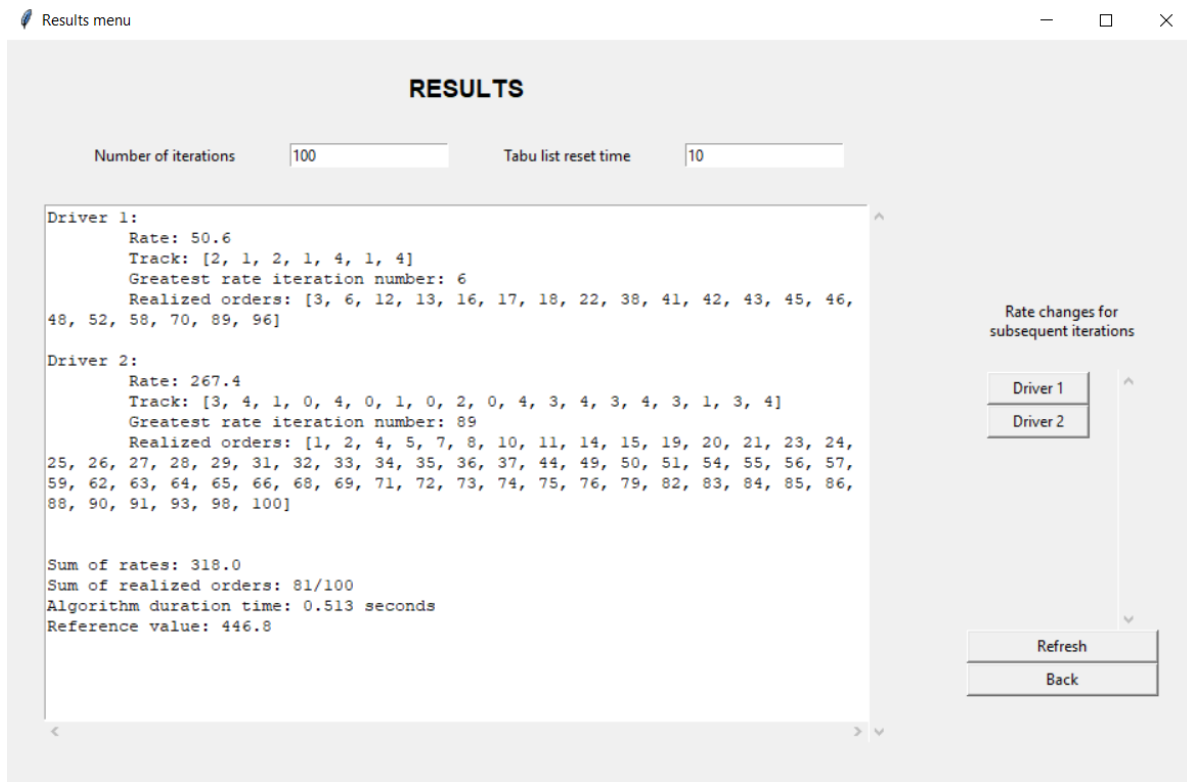
Ważną opcją jest przycisk pozwalający na ponowne sprawdzenie wpisanych danych oraz zapisanie ich. Drugi z przycisków umieszczonych w dolnej części okna pozwala na wymazanie wszystkich uzupełnionych w szkieletcie listy zamówień danych umożliwiając rozpoczęcie pracy od początku.

## Menu wyników - Results Menu

Ostatnie okno, po poprawnym uzupełnieniu parametrów w oknach poprzednich, umożliwia podgląd wyniku działania algorytmu. Najpierw jednak użytkownik zobowiązany jest na uzupełnienie parametrów, takich jak liczby iteracji oraz okresu resetu listy tabu. Po lewej stronie okna widać rubrykę, w obrębie której wyświetlone zostaną wyniki działania programu, natomiast po prawej stronie widnieje pusta jeszcze lista, która pojawi się dopiero po uruchomieniu działalności programu poprzez wciśnięcie przycisku odświeżającego okno.



Wyświetlane są wyniki działania algorytmu dla każdego z dostawców, suma ich wszystkich zysków, stosunek liczby zrealizowanych zamówień do ich całkowitej ilości, czas działania algorytmu podany w sekundach oraz wartość odniesienia dla wprowadzonych danych.



Jak widać na załączonej grafice, uzupełniona zostaje rubryka po prawej stronie okna. Kolejne przyciski podpisane numerami kierowców pozwalają na podgląd wartości zysku kierowcy dla każdej z wykonanych iteracji. Na wykresie zaznaczono linię trendu wartości zysku oraz punkt odpowiadający najwyższej uzyskanej wartości zysku podczas całkowitego działania algorytmu.

Przykładowe wykresy uzyskane tym sposobem zamieszczono w dokumentacji w rozdziale "Badanie zmian wartości funkcji celu w kolejnych iteracjach".

### Wartości domyślne

Dla ułatwienia obsługi programu, w każdym oknie wymagającym wprowadzania danych na wstępie uzupełniono luki parametrami jednej z instancji testowych umożliwiające sprawny test działania programu bez konieczności pilnowania zgodności wprowadzanych danych z narzuconymi ograniczeniami wynikającymi ze specyfiki działania algorytmu.

---

## Podsumowanie działania algorytmu

Przeprowadzone testy pozwalają na stwierdzenie, że algorytm przeszukiwania Tabu pozwala na znalezienie satysfakcjonującego rozwiązania dla badanego problemu transportowego. Ważne jest, aby umiejętnie wybrać wszystkie parametry dotyczące mapy, kierowców i ich pojazdów oraz dostaw, biorąc pod uwagę ich zależności względem siebie. W trakcie testów kierowano się tą zasadą, tworząc instancje testowe z parametrami, które wybrano tak, aby w miarę możliwości mogły odzwierciedlać realny problem.

Na podstawie testów opisanych w dokumentacji oraz tych, które były przeprowadzane na bieżąco implementacji i w celach weryfikacyjnych, zauważono następujące wady działania:

- 1) W trakcie implementacji algorytmu optymalizacja czasu wykonywania nie była brana pod uwagę, co jest widoczne przy rozwiązywaniu problemu o większej skali.
- 2) Kierowcy praktycznie nie są w stanie przepracować całego czasu pracy, wielkość pozostałego czasu jest zależna od punktu końcowego i jego odległości od sąsiadów.
- 3) Obliczana wartość odniesienia nie zawsze jest dobrą drogą do oceny rozwiązania, bowiem trzeba pamiętać o rzeczach takich jak:
  - jeśli pozostały zamówienia niezrealizowane to mogły one mieć skrajnie mały lub duży zysk, przez co np. zrealizowanie połowy zamówień nie gwarantuje zysku równego połowie wartości odniesienia,
  - zysk całkowity może być wyższy niż wartość odniesienia w sytuacji kiedy wszystkie zamówienia zostaną zrealizowane, a kierowcy nie przepracują całego czasu pracy, zmniejszając tym samym koszty dostaw.
- 4) Algorytm może zwrócić ujemną wartość zysku kierowcy. Sytuacja ta jest wbrew logicznemu podejściu, bowiem lepiej byłoby gdyby dany kierowca nie rozpoczął pracy i jego zysk wyniósłby zero. Jednak sytuację tą zaakceptowano m.in. z następujących powodów:
  - modyfikowałoby to wynik algorytmu co mogłoby mieć znaczący wpływ dla bardziej rozbudowanych problemów powodując ukryte niezgodności,
  - zaburzyłoby to porównanie uzyskanego wyniku z wartością odniesienia, bowiem jej wartość byłaby stała pomimo pozbycia się nawet znacznych kosztów.
- 5) Niektóre kroki algorytmu, takie jak wybór pierwszego czy kolejnych rozwiązań są w dużym stopniu oparte na losowości, co może powodować rozrzut w otrzymanych wartościach funkcji celu dla tych samych parametrów przy kolejnych uruchomieniach algorytmu. Jednak na podstawie przeprowadzonych testów dla wybranych danych stwierdzono, że rozrzut ten jest stosunkowo niewielki.
- 6) Aby algorytm na pewno mógł się wykonać poprawnie, minimalna ilość punktów na mapie musi wynosić 5.

---

# Spis treści

<b>Wprowadzenie do problemu</b>	<b>1</b>
Opis analizowanego zadania	1
Przyjęte uproszczenia	2
<b>Model matematyczny</b>	<b>3</b>
<b>Rozwiązanie problemu</b>	<b>4</b>
Algorytm rozwiązujący problem	4
Opis teoretyczny algorytmu	4
Dlaczego Algorytm Tabu?	4
Uproszczony schemat blokowy	5
Dostosowanie algorytmu do problemu	6
Wybór języka programowania	6
Implementacja struktur	7
Generacja mapy	7
Wybór punktów odbioru dostaw	7
Generacja puli zamówień	7
Generacja puli kierowców	8
Implementacja algorytmu	8
Wygenerowanie pierwszego rozwiązania	8
Wygenerowanie kolejnych rozwiązań	8
Ocena rozwiązania	9
Klasy	9
<b>Testy algorytmu</b>	<b>10</b>
Wpływ ilości iteracji na działanie algorytmu	11
Wpływ stosunku okresu resetu listy Tabu do ilości iteracji na działanie algorytmu	16
Wpływ ograniczeń na działanie algorytmu	20
Badanie zmian wartości funkcji celu w kolejnych iteracjach	27
Wpływ zwiększenia rozmiaru problemu na działanie algorytmu	29
<b>Postać użytkowa algorytmu - GUI</b>	<b>32</b>
Menu instrukcji - Instructions Menu	33
Menu mapy - Map Menu	34
Menu kierowców - Drivers Menu	39
Menu zamówień - Orders Menu	40
Menu wyników - Results Menu	43
<b>Podsumowanie działania algorytmu</b>	<b>45</b>
<b>Spis treści</b>	<b>46</b>