MATEMATYCZNE METODY WSPOMAGANIA DECYZJI

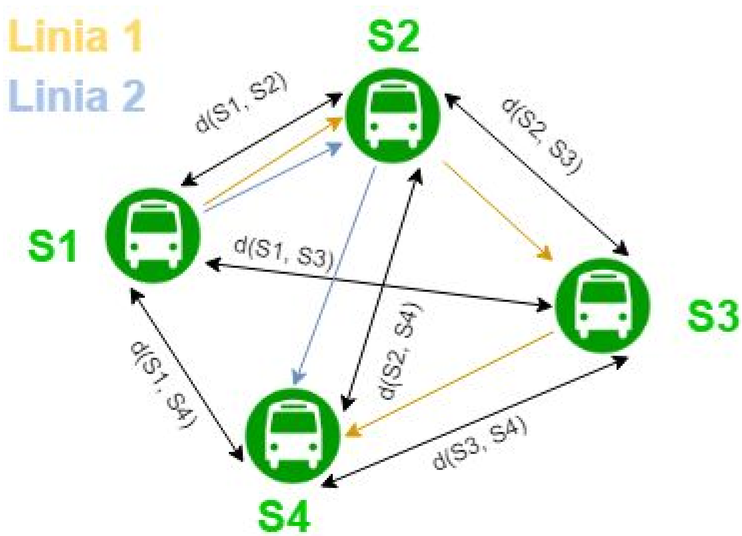
ZAGADNIENIE OPTYMALIZACJI LINII AUTOBUSOWYCH

Sylwester Dawida, Bartłomiej Fryz

Spis treści:

Opis problemu

Przedmiotem pracy jest zagadnienie optymalnego doboru ***l* linii autobusowych**, o n ilości przystanków które będą w stanie obsłużyć miasto. Miasto to jest opisane poprzez listę przystanków, macierz D odległości między nimi oraz macierz P . **Macierz *P* zawiera** informacje o ilości osób chcących się przedostać między dowolnymi dwoma przystankami.



rysunek poglądowy

Naszym celem jest stworzenie listy linii autobusowych, które muszą być w stanie zaspokoić potrzeby jak największej liczby mieszkańców oraz która jest optymalna pod względem kosztów obsługi (najmniejsza wartość funkcji kary) przy zadanych ograniczeniach.

Model matematyczny

Dane wejściowe

lista n przystanków

$$S_i \text{ dla } i = 1...n$$

symetryczna macierz odległości między przystankami:

$$\mathbf{D} = \begin{pmatrix} 0 & - & \dots \\ d_{21} & 0 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

macierz P ilości osób chcących przejechać z przystanku Si do Sj

$$\mathbf{P} = \begin{pmatrix} 0 & p_{12} & \dots \\ p_{21} & 0 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

ilość linii autobusowych: l

Format rozwiązania

Lista linii autobusowych L w formie:

$$L = \{L_1...L_l\} = \{\{S_a...S_b\}...\{S_c...S_d\}\}$$

Funkcja celu

$$\sum_i \sum_j w(i,j) \cdot P_{ij}$$

gdzie P-macierz ilości osób chcących przejechać z przystanku Si do Sj

G(a,b,X) funkcja zwracająca najkrótszą trasę w rozwiązaniu X (lista linii autobusowych) pomiędzy przystankiem a i b lub współczynnik kary jeśli połączenie nie istnieje.

Warunki ograniczające

1. Pojemność autobusów jest nieograniczona.
2. Niemożliwe są przesiadki.
3. Maksymalna ilość linii autobusowych jest ustalona odgórnie.
4. Maksymalna długość linii jest ustalona z góry.
5. Linie mogą się pokrywać, ale bierzemy pod uwagę najkrótszą trasę.
6. Nie wszyscy muszą być przewiezieni, ale istnieje kara za brak połączenia.

Algorytm

Do rozwiązania naszego problemu użyliśmy algorytmu o nazwie “tabu search”. Jego działanie opiszemy na przykładzie pseudokodu przedstawionego poniżej:

wybierz lub wylosuj punkt startowy $x_0 \in X$
$x_{opt} \leftarrow x_0$
tabu_list $\leftarrow \varnothing$
repeat
    znajdź $x \in N''(x_0)$, dla którego $m_{val}(x_0, x)$ jest największa
    $x_0 \leftarrow x$
    if $f(x_0) > f(x_{opt})$ then
        $x_{opt} \leftarrow x_0$
    zweryfikuj tabu_list
    $\forall element \in tabu\_list$ do
        $--kadencja_i$
        if $kadencja_i = 0$ then
            usuń element$(atrybut_i, kadencja_i)$ z $tabu\_list$
until warunek zakończenia

Inicjacja:
W początkowej fazie algorytmu wczytywane są dane wejściowe czyli macierz P, D oraz odpowiednie zmienne takie jak ilość iteracji czy długość linii autobusowych Następnie losowana jest wartość początkowa rozwiązania

Sąsiedztwo:
wewnątrz funkcji iterate początkowo jest wyznaczane sąsiedztwo aktualnego rozwiązania służą do tego dwie funkcje, pierwsza zamienia dwa przystanki w jednej z linii autobusowych druga natomiast usuwa jeden przystanek i na jego miejsce wstawia przystanek który nie występuje aktualnie w linii autobusowej.

Funkcja Celu:
Lista wyznaczonych sąsiadów jest przesyłana do funkcji findBest która wyznacza najlepszego sąsiada służy do tego funkcja celu opisana w modelu matematycznym. findBest sprawdza czy ruch do sąsiada jest w TabuList i odpowiednio odrzuca zablokowane rozwiązania, w algorytmie zaimplementowane jest kryterium aspiracji które jest w stanie złamać powyższą regułę jeśli sąsiad do którego ruch jest zakazany polepsza najlepsze globalne rozwiązanie.

Aktualne Rozwiązanie
funkcja findBest zwraca najlepszego sąsiada ten z kolei staje się aktualnym rozwiązaniem algorytmu i jeśli polepsza najlepsze dotychczas znalezione to staje się rozwiązanie globalnie najlepszym.

tabuList:
ostatnimi krokami w iteracji są dodanie do tabuList ruchu który został wykonany w tej iteracji oraz dekrementacja czasu tabu każdego elementu w liście tabu.

Rozwiązanie:
algorytm zwraca listę linii autobusowych oraz wykresy funkcji celu, pojemności tabuList, użycia kryterium aspiracji, stosunku obsłużonych mieszkańców, stosunku użytych przystanków w zależności od iteracji.

Implementacja

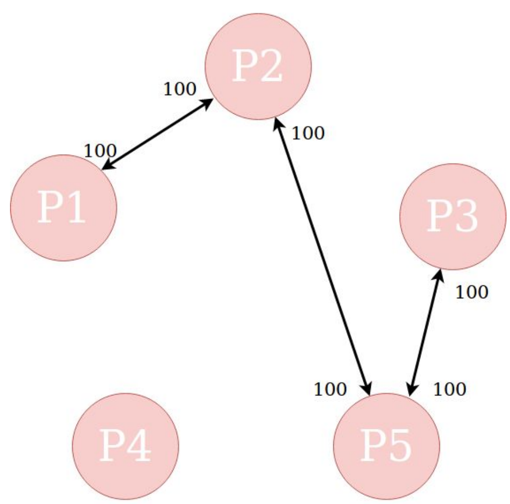
Do implementacji wybraliśmy język programowania Python ponieważ jest on łatwy, przejrzysty oraz wyposażony w wiele przydatnych w analizie bibliotek. Pracowaliśmy wspólnie wykorzystując system kontroli wersji GIT oraz testy jednostkowe.

Testy

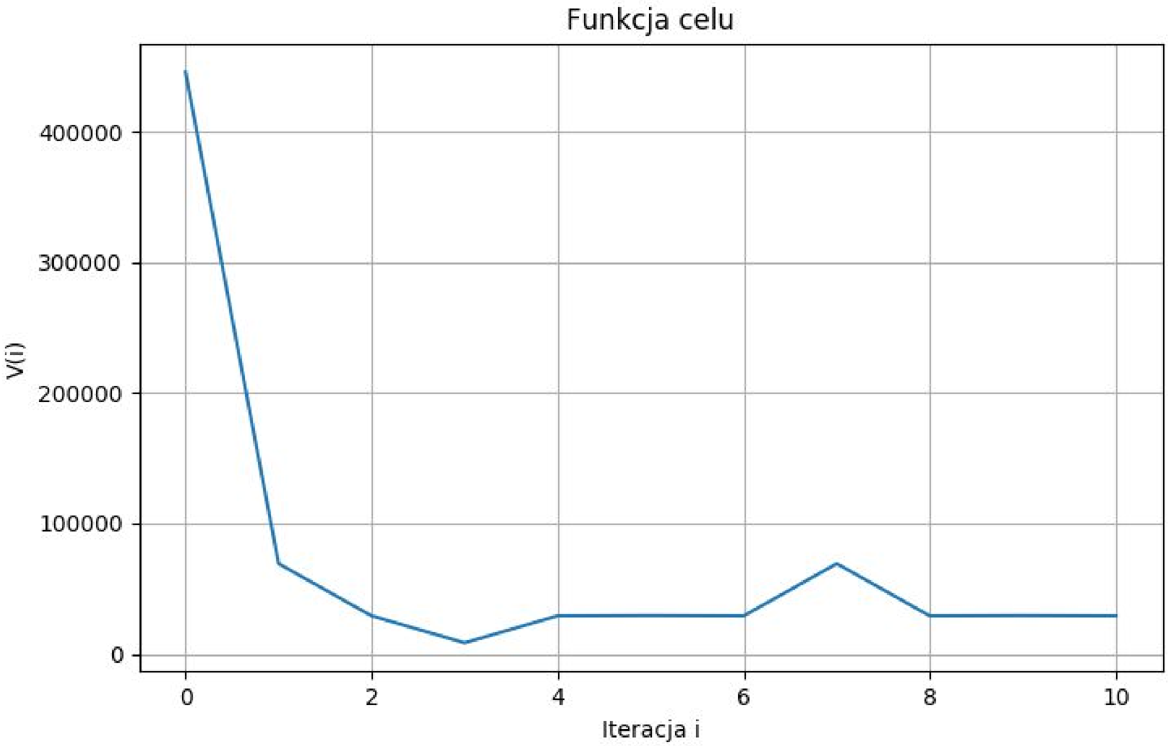
W tym punkcie przedstawiamy szereg przebiegów symulacyjnych obrazujących działanie naszego algorytmu dla przykładowych danych wejściowych.

Przypadek trywialny

Aby można było prześledzić działanie algorytmu na prostym przykładzie, posłużyliśmy się poniższym grafem. Pomiędzy wierzchołkami P1, P2, P5, P3 istnieje optymalna linia - połączenia o liczbie osób 100, długości 1. Wszystkich inne połączenia mają wartość liczby osób 1, a długość 100.



Oczywistym rozwiązaniem zadania jest ciąg P1-P2-P5-P3, który też jest wynikiem działania naszego algorytmu.



Wykres funkcji celu od iteracji

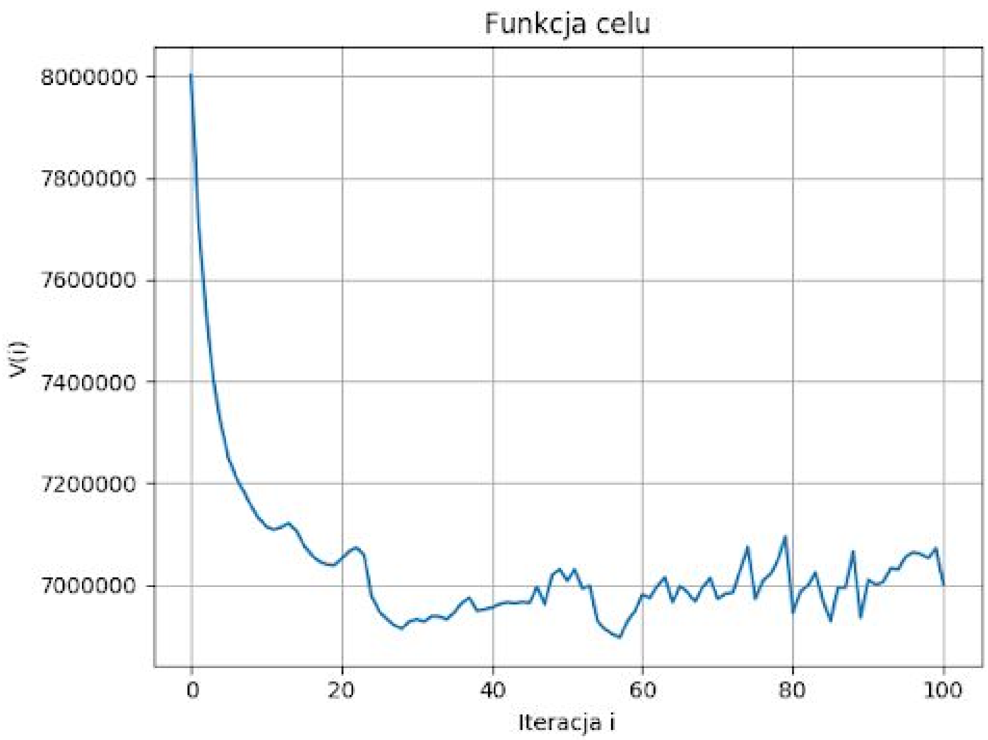
Miasto o rozmiarze 15

Rozwiązywane dla 3 linii o długości 5 przystanków.

Długości tablicy tabu 0

Startowa funkcja celu 8001470

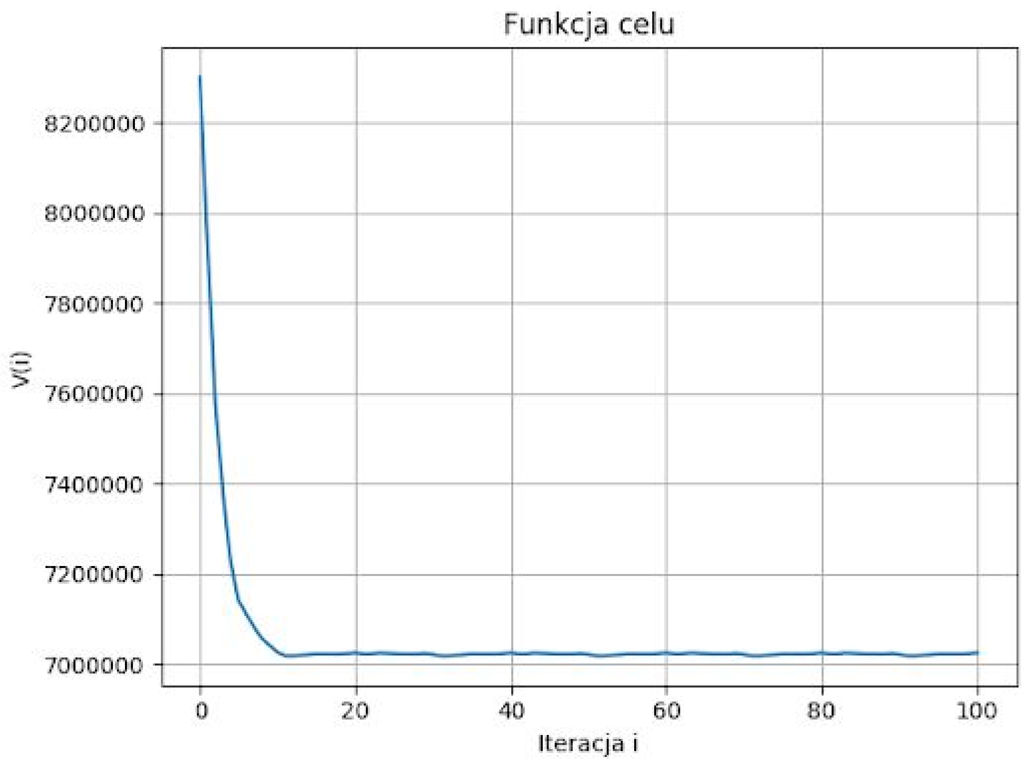
Końcowa funkcja celu 6896633



Długości tablicy tabu 5

Startowa funkcja celu 8300774

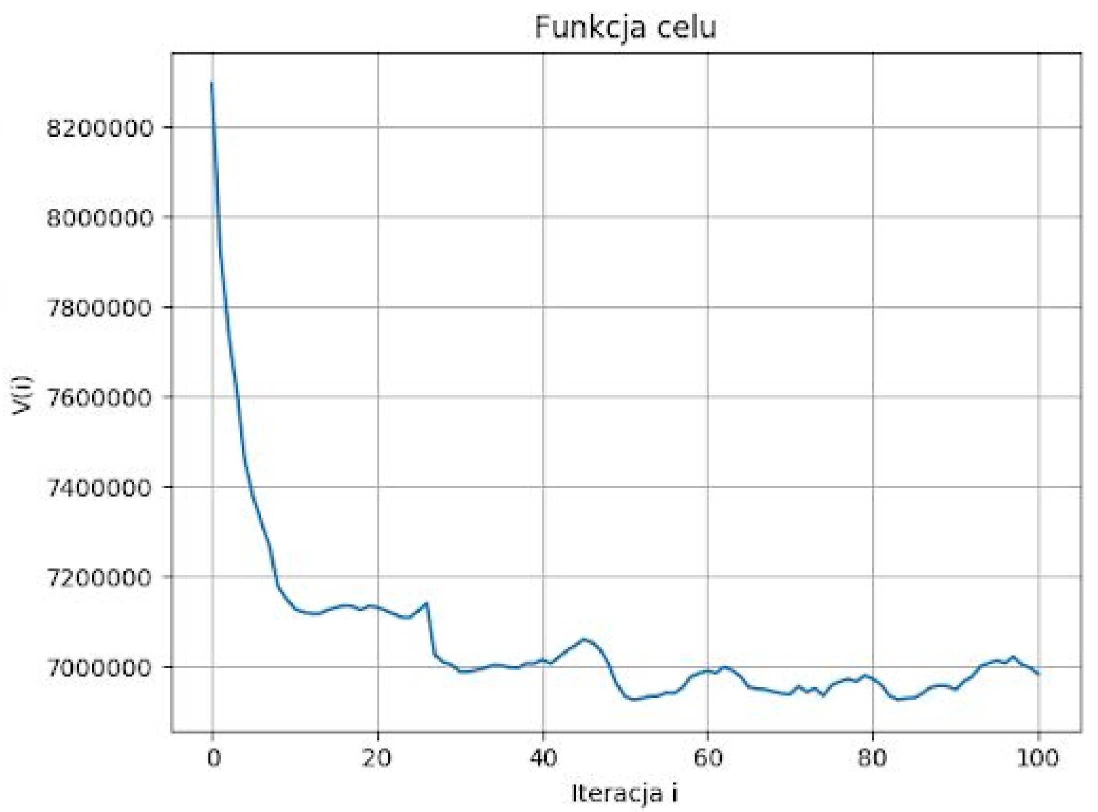
Końcowa funkcja celu 7018206



Długości tablicy tabu 15

Startowa funkcja celu 8295217

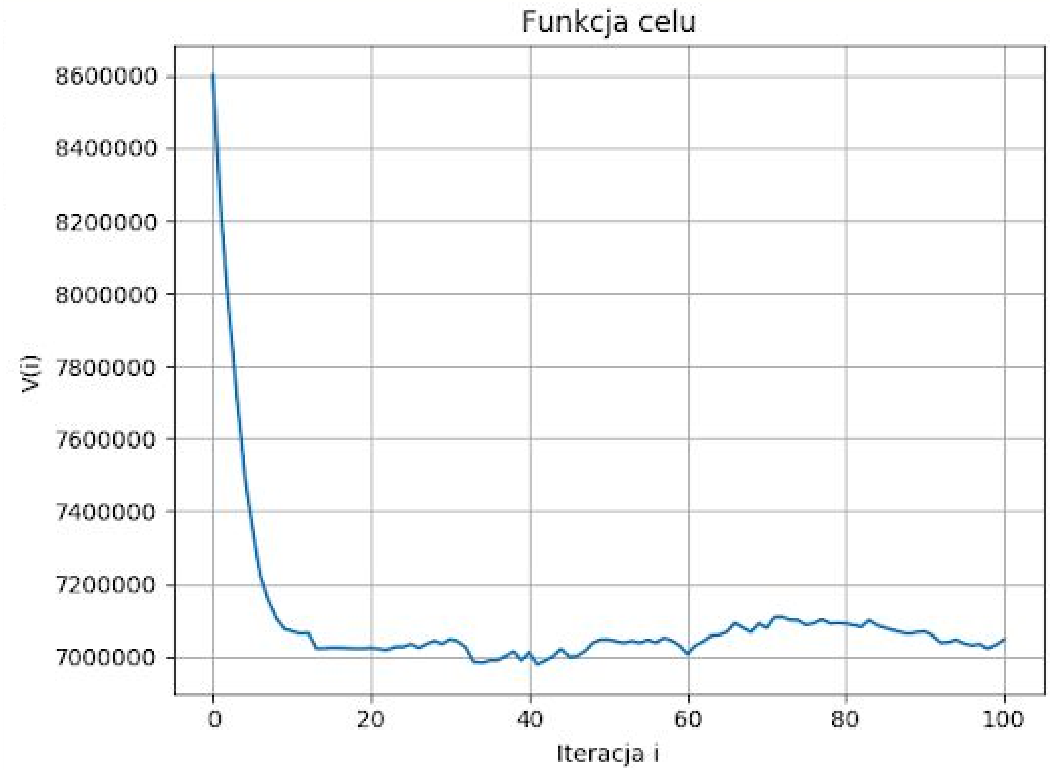
Końcowa funkcja celu 6925999



Długości tablicy tabu 30

Startowa funkcja celu 8599932

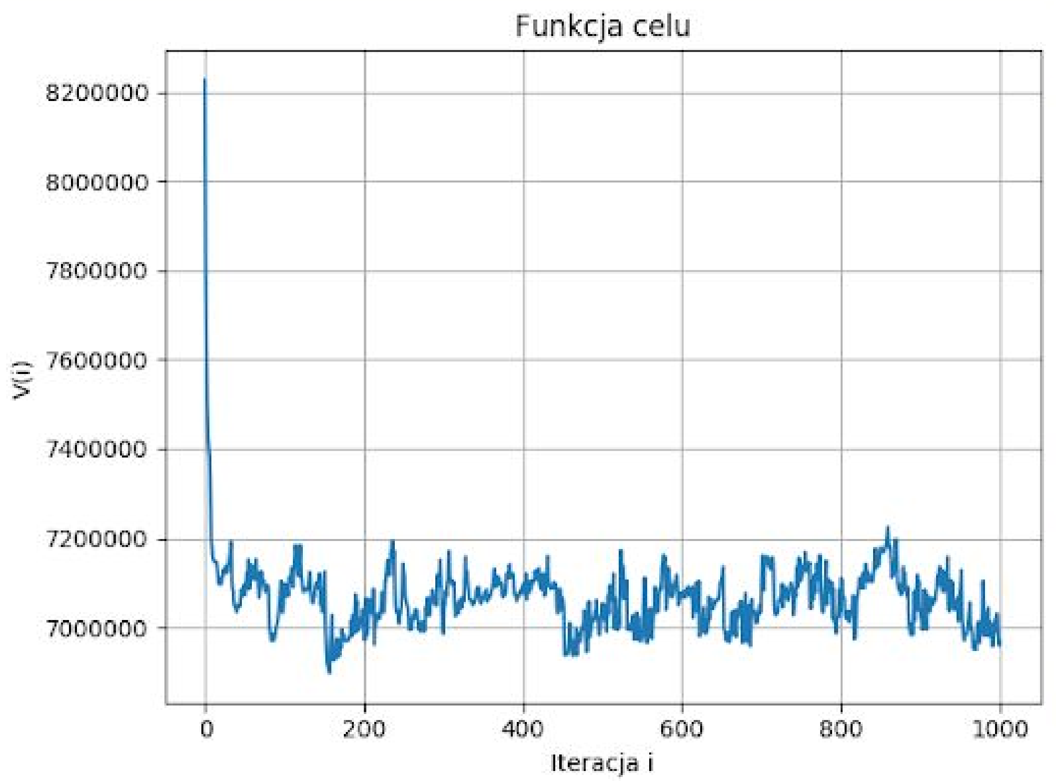
Końcowa funkcja celu 6978799



Długości tablicy tabu 75

Startowa funkcja celu 8226106

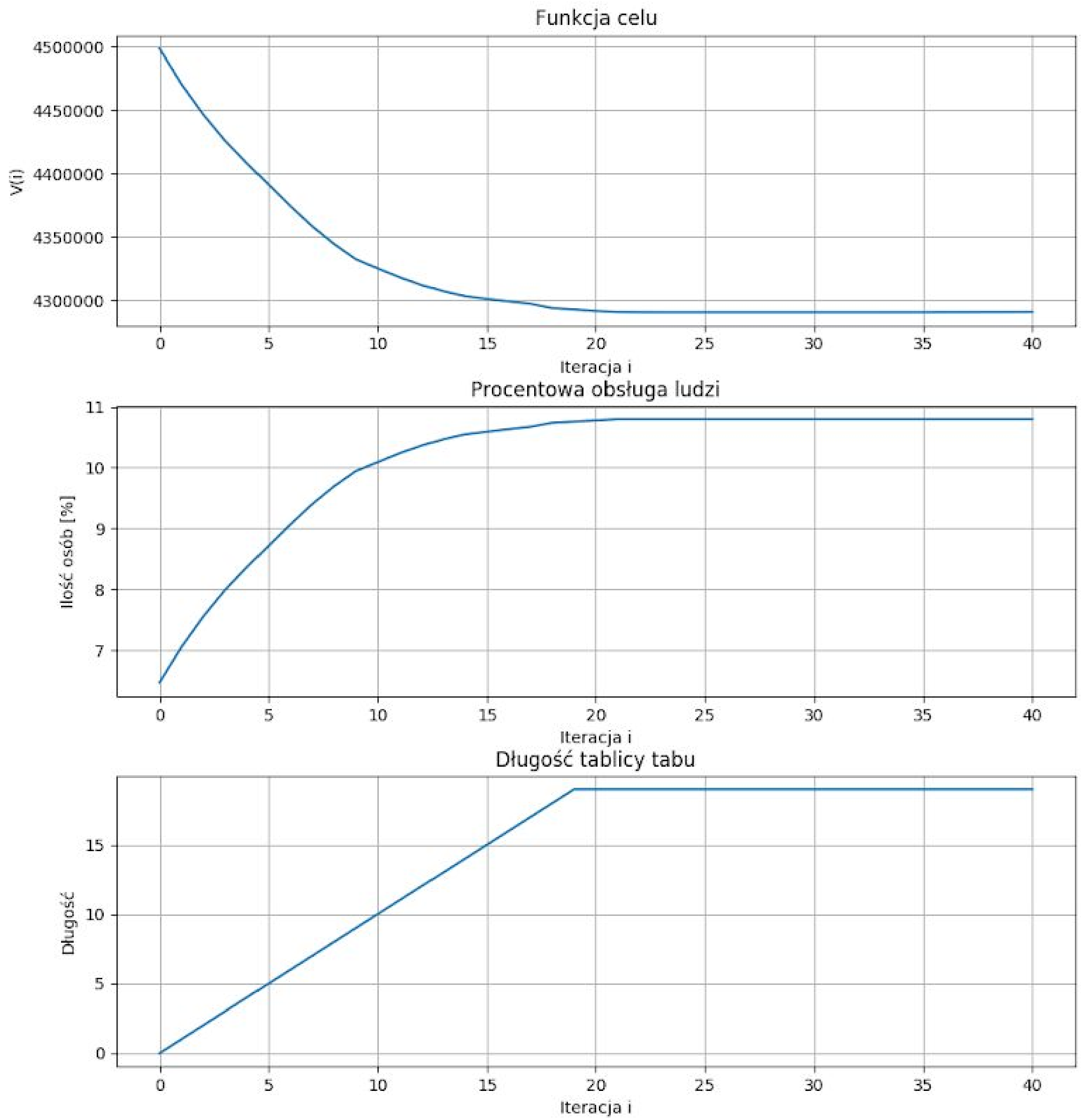
Końcowa funkcja celu 6896633

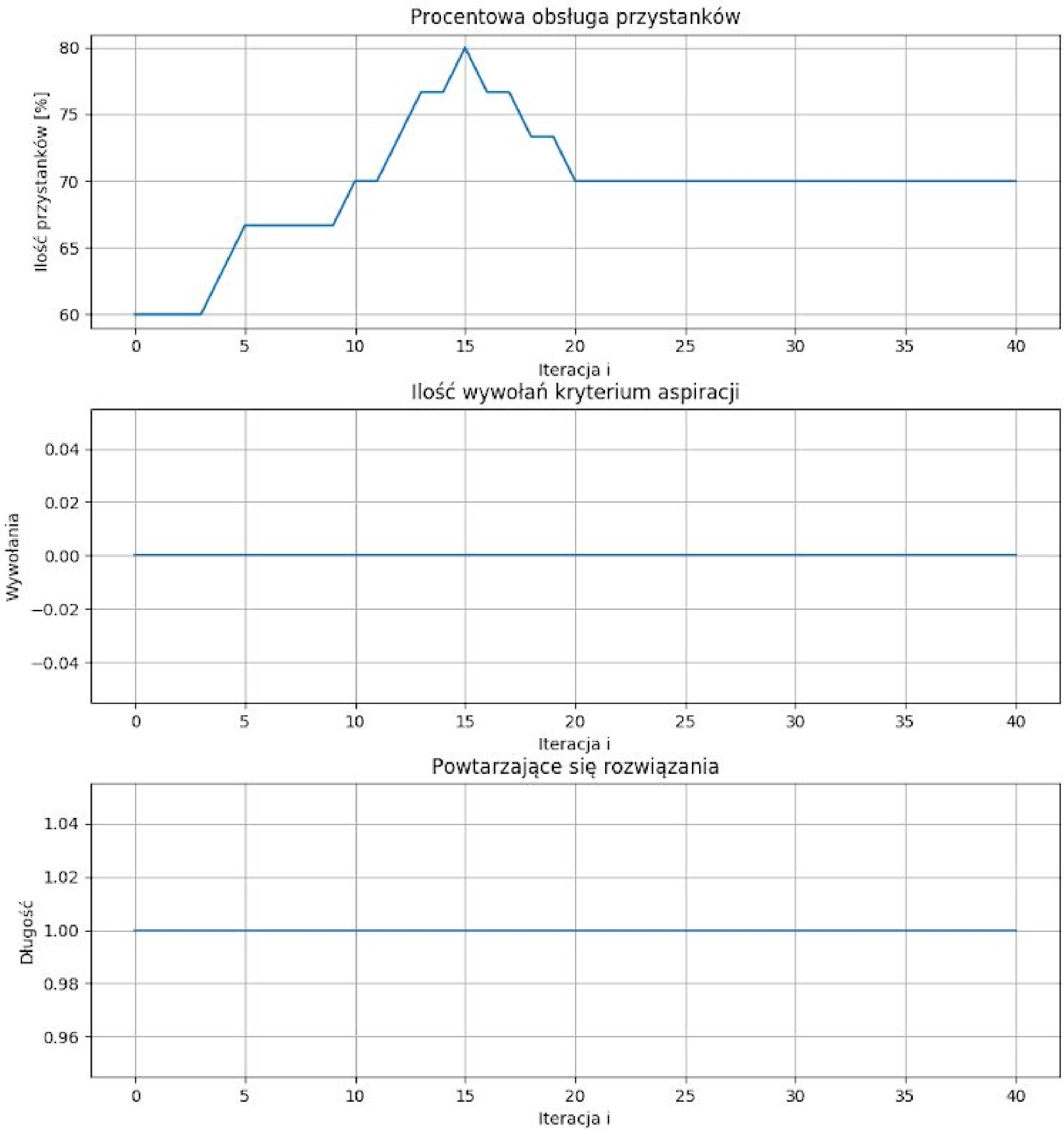


Do analizy wykorzystaliśmy napisaną przez nas aplikację konsolową, która umożliwia wygodną pracę bez potrzeby ciągłego, ręcznego przestawiania parametrów symulacji.

Miasto o rozmiarze 30

Losowe miasto,10 linii po 3 przystanki dla długości tablicy tabu 20.





Startowa funkcja celu 4498480

Końcowa funkcja celu 4290465