



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

MATEMATYCZNE METODY WSPOMAGANIA DECYZJI

ZNAJDYWANIE OPTYMALNEJ TRASY PRZEJAZDU KOMUNIKACJĄ MIEJSKĄ ZA POMOCĄ
ALGORYTMU SYMULOWANEGO WYŻARZANIA

Autorzy: Wojciech Gumuła, Rafał Prusak

1 SPIS TREŚCI

2	Zagadnienie	3
2.1	Motywacja	3
3	Algorytm.....	4
3.1	Opis.....	4
3.2	Parametry	4
3.3	Funkcja oceny rozwiązania	5
3.4	Uproszczenia	5
3.5	Dane wejściowe.....	5
3.6	Konfiguracja algorytmu	6
3.7	Dane wyjściowe.....	6
3.8	Prezentacja przebiegu algorytmu.....	6
3.9	Rozwiązanie początkowe.....	6
3.9.1	Algorytmy BSF i DSF.....	6
3.9.2	Badanie współrzędnych geograficznych.....	7
3.9.3	Rozwiązania losowe.....	7
3.10	Otoczenie rozwiązania.....	7
4	Aplikacja	8
4.1	Struktura danych	8
4.2	Interfejs graficzny	9
5	Testy.....	13
5.1	temperatura początkowa	13
5.2	parametr alfa.....	15
5.3	liczba dozwolonych przesiadek	17
5.4	parametr K.....	19
5.5	kara za przesiadkę	21
5.6	Przypadki złożliwe	22
5.6.1	Długie oczekiwanie na pierwsze połączenie.....	23
5.6.2	Długie oczekiwanie pomiędzy połączeniami	24
5.6.3	Zbyt mała liczba przesiadek.....	25
5.7	Wpływ poszczególnych parametrów na wyniki	25
5.7.1	Temperatura początkowa	25
5.7.2	Temperatura końcowa	26
5.7.3	Liczba iteracji	26
5.7.4	Współczynnik alfa.....	27
5.7.5	Liczba dozwolonych przesiadek.....	27
5.7.6	Współczynnik kary.....	28
5.8	Wnioski	28
5.9	Dopuszczalny rozmiar problemu	29
6	Podsumowanie.....	30

2 ZAGADNIENIE

Celem realizowanego projektu było zapoznanie się z implementacją i działaniem algorytmu przybliżonego (aproksymacyjnego). Algorytmy takie stosuje się do wyznaczania „prawie optymalnego” rozwiązania problemu optymalizacyjnego, dla którego nie są znane szybkie algorytmy wyznaczające dokładne rozwiązanie.

Nasz program realizuje wyznaczanie optymalnej trasy przejazdu komunikacją miejską. Został on przetestowany na realnej instancji problemu, którą jest sieć komunikacyjna Krakowa. Dodatkowo w celach testowania zachowania algorytmu aplikacja została wzbogacona o możliwość generacji testowych instancji testowych (losowa sieć komunikacyjna tworzona poprzez wybranie przystanków z kwadratowej „siatki” oraz statycznego rozkładu jazdy).

2.1 MOTYWACJA

Wyszukiwanie połączeń pomiędzy dwoma punktami jest najbardziej oczywistym zastosowaniem teorii grafów w życiu codziennym. Istnieje wiele programów i stron sieci zajmujących się tym zadaniem i naszym celem nie była rywalizacja z nimi. Badanie problemu komunikacyjnego pozwala nam na intuicyjne podejście do problemu – podróżowanie komunikacją miejską jest rzeczą naturalną dla większości osób. Z tego powodu w trakcie pisania aplikacji wiele zagadnień można było rozwiązać w intuicyjny sposób.

Ponadto, nie chcieliśmy, by badane zagadnienie było abstrakcyjne. Nasza aplikacja pracuje na rzeczywistych danych opisujących komunikację miejską Krakowa. Jest również przenośna – dysponując bazą danych dla dowolnej innej komunikacji, można ją wykorzystać w programie. W przypadku wielu miast dostępne są dane w formacie gtfs, które można by wykorzystać w aplikacji po napisaniu funkcji importującej ten format danych. Zależało nam również, by aplikacja działała poprawnie dla sieci o rozmiarze opisującym duże miasta, co sprawdziliśmy w trakcie testów, badając zachowanie aplikacji dla różnych parametrów.

Zagadnienie, które realizowaliśmy, można na pierwszy rzut oka sprowadzić do problemu wyznaczania najlepszej drogi w grafie. Istnieje wiele algorytmów dedykowanych do tego problemu, takich jak algorytm Dijkstry, Bellmana - Forda, etc. Niestety, wszystkie te algorytmy pracują na grafie o nie zmieniających się wagach krawędzi. W przypadku naszego problemu wagi krawędzi ulegają zmianie wraz z upływającym „wirtualnym” czasem przejazdu.

Powyższa kwestia wymusza konieczność innego podejścia do tego problemu. Do wyznaczenia optymalnej trasy został użyty algorytm symulowanego wyżarzania.

3 ALGORYTM

W trakcie pracy wykorzystaliśmy algorytm symulowanego wyżarzania, dający dobre wyniki w przypadku problemów dyskretnych o dużym rozmiarze.

3.1 OPIS

Krok 1:

Podstaw $T = T_0$

Wygeneruj rozwiązanie początkowe x_a

Podstaw $x^* = x_a, f^* = f(x_a)$

Krok 2: dopóki $T > T_{min}$

Powtarzaj k -krotnie:

znajdź nowe rozwiązanie x_n w otoczeniu punktu x_a

oblicz wartość funkcji dla nowego rozwiązanie oraz różnicę $d = f(x_n) - f^*$

jeśli nowe rozwiązanie jest lepsze ($d < 0$) to znalezione rozwiązanie staje się obecnym rozwiązaniem $x^* = x_n, f^* = f(x_n)$

w przeciwnym przypadku:

wylosuj liczbę q z przedziału $[0,1]$

jeśli zachodzi nierówność $q < e^{\frac{-d}{T}}$ to gorsze rozwiązanie staje się obecnym
 $x^* = x_n, f^* = f(x_n)$

zmniejsz temperaturę $T = \alpha T$

3.2 PARAMETRY

T - obecna temperatura

T_0, T_{min} – temperatura początkowa i końcowa

f – funkcja oceny rozwiązania

x_a – obecne rozwiązanie

x^*, f^* - najlepsze rozwiązanie i jego ocena

k – ilość iteracji algorytmu dla każdej temperatury

α – współczynnik temperatury

3.3 FUNKCJA OCENY ROZWIĄZANIA

Funkcja oceny f miała dwie składowe – funkcję celu, oraz funkcję kary.

Funkcja celu wyznaczana jest na podstawie czasu przejazdu. Jest to po prostu całkowity czas w minutach potrzebny na wykonanie trasy.

Równie istotnym elementem jest **funkcja kary**, umożliwiająca jednocześnie przegląd rozwiązań niedopuszczalnych, jak i zapewnienie, że będą one nieatrakcyjne z punktu widzenia algorytmu. Zgodnie z założeniami programu, funkcja kary wyznaczana jest na podstawie ilości przesiadek – jeśli jest ich więcej, niż założona przy uruchomieniu wartość, do funkcji oceny dodawana jest wyrażenie

$$f_k(x) = (\text{limit przesiadek} - \text{ilość przesiadek dla rozwiązania}) * \omega,$$

Gdzie ω to współczynnik kary – im większy, tym bardziej niechętnie wybierane będą rozwiązania niedopuszczalne.

3.4 UPROSZCZENIA

W trakcie pracy nad algorytmem przyjęliśmy, że komunikacja miejska kursuje zgodnie z rozkładem, a więc są już uwzględnione korki i możliwe opóźnienia, co jak wiadomo, nie zawsze jest prawdą. Ponadto, ze względu na ograniczenia bazy danych, założyliśmy że wszystkie połączenia dostępne są każdego dnia. Czas połączenia mierzony jest z dokładnością do minut, co powinno być wystarczające do większości zastosowań, choć z pewnością znajdą się zagadnienia, w których dokładność powinna być znacznie większa. Dodatkowo, założyliśmy że użytkownik nie porusza się pieszo pomiędzy przystankami, może zmieniać tylko kierunek jazdy przy przystankach podwójnych. Wynika to z faktu, że algorytm musiałby przeszukiwać znacznie większą ilość rozwiązań dopuszczalnych – zamiast kilku połączeń dla każdego przystanku, byłoby ich tyle, ile znajduje się przystanków w sieci. Kosztem optymalizacji rozwiązania należałoby więc zwiększyć czas trwania algorytmu kilkadziesiąt, czy nawet kilkaset razy.

3.5 DANE WEJŚCIOWE

Dane wejściowe stanowi baza danych, na której operować ma algorytm, a także parametry szukanego połączenia – przystanek początkowy, przystanek końcowy, czas rozpoczęcia trasy.

Możliwe jest również generowanie losowej bazy danych o zadanych rozmiarach i zadany współczynnik wypełnienia. Jest to szczególnie przydatne w trakcie testowania pracy algorytmu. Generacja polega na stworzeniu struktury połączeń na planie kwadratu o wymiarach m -wierszy i n -kolumn. W trakcie generowania, na siatkę składającą się z $m * n$ punktów nakładane są najpierw przystanki, a następnie połączenia między nimi. Współczynnik wypełnienia decyduje, jak wiele punktów trafi ostatecznie na siatkę – reprezentuje prawdopodobieństwo od 0 (nigdy) do 1 (zawsze). Po wypełnieniu siatki punktami tworzone są połączenia między nimi – są to proste ciągi przystanków północ-południe, południe-północ, zachód-wschód i wschód-zachód. Łatwo jednak uzasadnić, że reprezentują one dowolną strukturę, która jest tylko przedstawiana na planie kwadratu.

3.6 KONFIGURACJA ALGORYTMU

Przed uruchomieniem algorytmu konieczne jest podanie wszystkich parametrów opisywanych w punkcie 3.2. Dodatkowo, należy pamiętać o podaniu dopuszczalnej ilości przesiadek oraz kary za każdą przesiadkę ponad dopuszczalne.

3.7 DANE WYJŚCIOWE

Zależnie od trybu pracy, dane wyjściowe reprezentowane są w formie tekstowej (Listing 1 Przykładowe wyjście program) lub graficznej (Rysunek 5 Wynik pracy programu). W trakcie wykonywania testów, następuje również zapis parametrów algorytmu (wartość obecnego rozwiązania, wartość współczynnika kary obecnego rozwiązania i wartość najlepszego rozwiązania do tej pory), przydatny w późniejszej analizie.

3.8 PREZENTACJA PRZEBIEGU ALGORYTMU

Nie zdecydowaliśmy się na prezentację przebiegu algorytmu na wykresie czy mapie w trybie rzeczywistym. Wynika to z kilku względów: przede wszystkim, rysowanie wykresu przy jednoczesnej pracy algorytmu zajmuje dużą część czasu pracy procesora, a więc pogorszyłoby wydajność całego programu. Generowanie mapy byłoby jeszcze bardziej czasochłonne – musi być ona pobrana z sieci, co zajęłoby jeszcze więcej czasu. Ponadto, trudno byłoby zapewnić kompatybilność wersji konsolowej, używanej do testów, z wersją z interfejsem graficznym. Z tego powodu zdecydowaliśmy się na monitorowanie pracy algorytmu przy użyciu regularnych komunikatów tekstowych. Są one wysyłane na standardowy strumień wyjścia i informują o obecnej wartości temperatury, a także wartości temperatur początkowej i końcowej.

Dodatkowo uznaliśmy, że optymalnym rozwiązaniem będzie prezentacja wykresów niezależnie od pracy programu. Stworzyliśmy proste narzędzie w języku Python do generowania wykresów na podstawie plików zapisywanych w trakcie testów, które następnie można obejrzeć w dowolnym środowisku graficznym.

3.9 ROZWIĄZANIE POCZĄTKOWE

Przed rozpoczęciem właściwej części konieczne jest wykonanie kroku 1 – wyznaczenia rozwiązania początkowego. Rozwiązaniem może być dowolne połączenie pomiędzy dwoma zadanymi punktami, pod warunkiem, że nie zawiera cykli. W trakcie szukania pierwszego rozwiązania nie braliśmy pod uwagę ilości przesiadek. Można podejść do tego tematu na wiele sposobów, w trakcie pracy nad algorytmem przetestowaliśmy trzy z nich.

3.9.1 ALGORYTMY BSF I DSF

Algorytmy przeszukiwania w szerz i w głąb należą do najprostszych rozwiązań problemu szukania rozwiązania w drzewie. Algorytm przeszukiwania w szerz daje świetne wyniki, jeśli ilość kroków do otrzymania dowolnego rozwiązania nie jest duża. Każdy kolejny krok w głąb drzewa zwiększa jednak wykładniczo nakład pracy, więc konieczne jest przerwanie działania algorytmu w przypadku, gdy nie

znalazł on rozwiązania wystarczająco szybko. Algorytm przeszukiwania w głąb dawał lepsze wyniki w przypadku połączeń pomiędzy bardziej oddalonymi od siebie punktami, ale również nie był idealny.

3.9.2 BADANIE WSPÓŁRZĘDNYCH GEOGRAFICZNYCH

Stworzona przez nas baza danych posiadała przypisane do każdego przystanku współrzędne geograficzne, umożliwiające ustawienie go na mapie. Dzięki temu mogliśmy wyznaczać rozwiązania podążając zgodnie z optymalnym kierunkiem – z obecnego przystanku przesunąć się na taki, który najbardziej zbliża nas do rozwiązania, bądź też takiego, który jest najmniej oddalony od prostej łączącej przystanek obecny z docelowym.

Metoda ta cechowała się zazwyczaj dużą zbieżnością i dobrą optymalizacją. Z punktu widzenia algorytmu wykorzystywanego później była „zbyt dobra” – dobre przybliżenie rozwiązania optymalnego sprawiało, że algorytm symulowanego wyżarzania badał tylko skrawek otoczenia – gdyby istniało połączenie dwóch zadanych przystanków, które jest szybsze, ale zajmuje większą odległość (np. szybkie połączenie omijające centrum miasta), prawdopodobnie nie zostałoby nigdy zbadane.

3.9.3 ROZWIĄZANIA LOSOWE

Ostatnią z badanych metod było losowe podążanie po drzewie. Przy wyborze następnego przystanku na liście nie uwzględniamy żadnego wskaźnika jakości, podążamy zawsze do losowego, z którym jest połączenie, a który nie został jeszcze odwiedzony. Nawet jeśli rozwiązanie będzie liczyć kilkaset połączeń, zostanie ono wstępnie wygładzone po kilkuset iteracjach algorytmu SA. Z punktu widzenia całego procesu jest to liczba dopuszczalna.

Możliwą modyfikacją tej metody jest połączenie jej z metodą geograficzną – przypisać wagę wylosowania danego połączenia powiązaną z oceną rozwiązań – im rozwiązanie lepsze, tym większe szanse, że je wybierzemy, analogicznie do algorytmu SA.

3.10 OTOCZENIE ROZWIĄZANIA

Rozwiązanie w otoczeniu definiowane jest jako takie, które w pewnym fragmencie różni się od obecnego. Dzięki temu zapewniamy, że przeszukana zostanie jak największa ilość rozwiązań dopuszczalnych i niedopuszczalnych, a jednocześnie nie „oddalimy” się od rozwiązania obecnie uważanego za najlepsze. Skupienie na rozwiązaniu najlepszym sprawia, że możemy pominąć oddalone rozwiązania, które nie należą do otoczenia, a mogą okazać się lepsze od obecnych. Z tego powodu przydatna jest pewna losowość, która sprawi, że rozwiązanie zupełnie inne od obecnego zostanie uznane za optymalne. Można to uzasadnić tym, że ilość połączeń dla każdego przystanku jest tylko niewielkim ułamkiem możliwych połączeń, a więc nie można sprawdzić wszystkich możliwości badając wyłącznie otoczenie obecnego rozwiązania.

Na nasz program składają się dwa główne „moduły”: aplikacja napisana w języku C++ (z wykorzystaniem mechanizmów standardu języka C++11) oraz interfejsu graficznego zrealizowanego z wykorzystaniem biblioteki Qt5.

Uruchomienie aplikacji możliwe jest zarówno w formie graficznej, jak i konsolowej. Ta druga opcja użyteczna jest zwłaszcza w celu wywołania szeregu testów, bez konieczności ręcznego podawania parametrów w trybie graficznym, a w sposób całkowicie zautomatyzowany. Program w trakcie działania utworzy szereg plików, dla każdego testu zapisując parametry pracy algorytmu.

W trybie graficznym możliwa jest praca w trybie interaktywnym – użytkownik podaje wybrane parametry, a aplikacja prezentuje wyniki w formie graficznej.

4.1 STRUKTURA DANYCH

Sieć komunikacyjna reprezentowana jest przez kilka różnych obiektów. Głównym pojemnikiem jest obiekt klasy *Network*, przechowujący wewnątrz obiekty typu *Node* i *Edge*, reprezentujące graf składający się z przystanków i połączeń między nimi. Obiekt *Route* reprezentuje ciąg połączeń pomiędzy kilkoma przystankami – czyli trasę, po której może poruszać się użytkownik aplikacji. Konieczne było również użycie pomocniczych obiektów do reprezentacji czasu, linii komunikacyjnych, czy typów pojazdów.

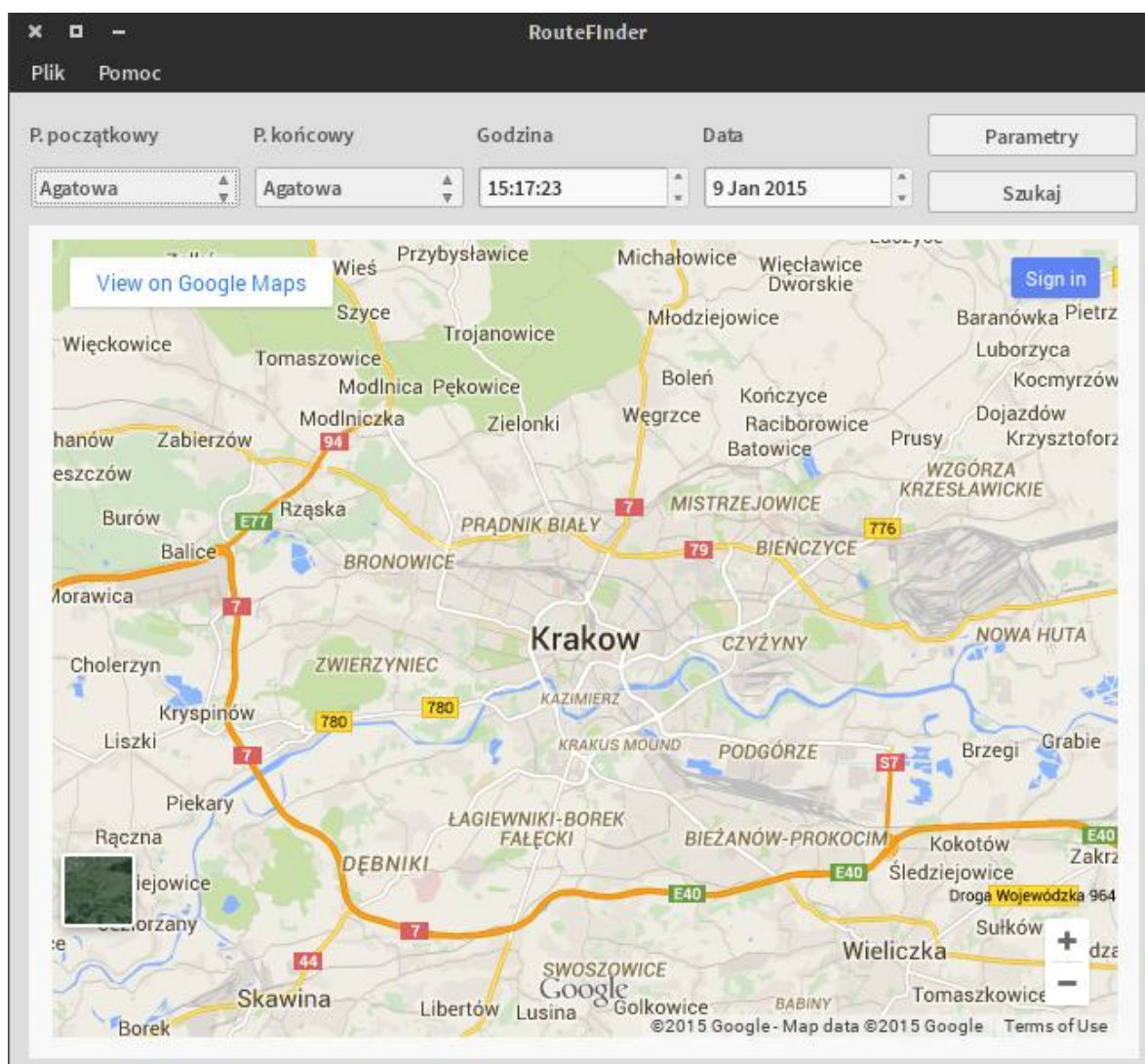
Algorytmy są klasami dziedziczącymi po klasie abstrakcyjnej *Solver*. Dzięki temu łatwo jest zaimplementować kolejne algorytmy – muszą one jedynie posiadać metodę *solve*, zwracającą rozwiązanie w formie obiektu *Route*.

Stworzona została również klasa *Tester*, służąca do przeprowadzania testów. Po stworzeniu obiektu i wywołaniu odpowiedniej metody, na zadanej sieci komunikacyjnej przeprowadzany jest szereg zaprogramowanych wcześniej testów.

Powtarzalność testów jest możliwa dzięki opcji generowania losowej sieci komunikacyjnej o zadanych wymiarach. Dzięki wykorzystaniu mechanizmu *ziarna* inicjującego wszystkie obiekty generowane losowo, praca programu jest powtarzalna.

Dokładniejsza dokumentacja techniczna znajduje się w katalogu *doxygen/doc*.

4.2 INTERFEJS GRAFICZNY



Rysunek 1 Główne okno programu

Znaczenie poszczególnych elementów:

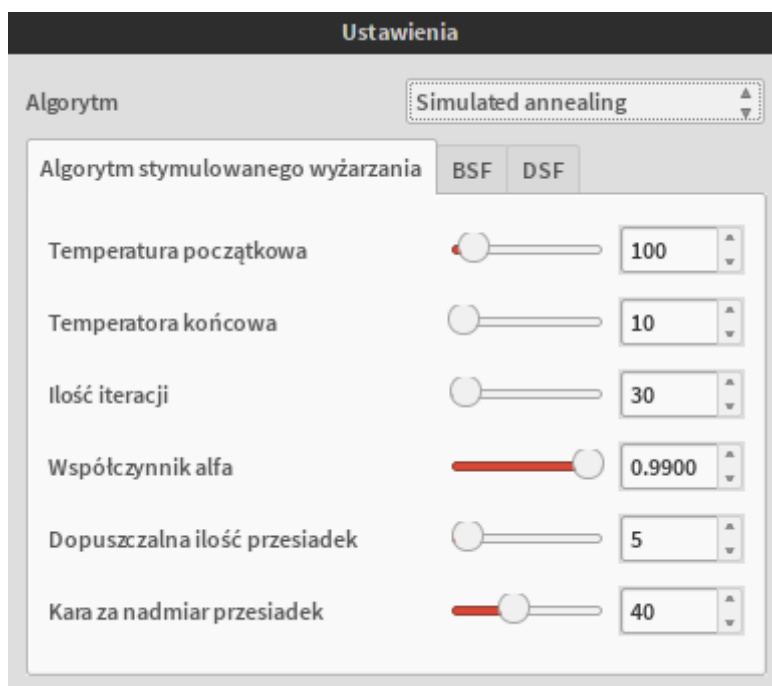
P. początkowy i **P. końcowy** – listy wszystkich przystanków znajdujących się w załadowanej sieci.

Godzina – wskazuje czas początkowy dla algorytmu – w czasie pracy algorytmu ściśle przestrzegane są ograniczenia czasowe, więc należy zwrócić uwagę, czy zadane parametry są poprawne.

Data – jak wyżej, wskazuje datę przejazdu. Należy pamiętać, że baza danych musi posiadać informacje o datach pracy poszczególnych linii w sieci komunikacyjnej, by to pole miało znaczenie w trakcie pracy algorytmu – ze względu na ograniczenia dostępności informacji, w naszym przypadku pozostaje nieaktywne.

Szukaj – przycisk, po wywołaniu którego rozpoczyna się weryfikacja wprowadzonych parametrów i praca algorytmu lub ostrzeżenie o niepoprawności danych.

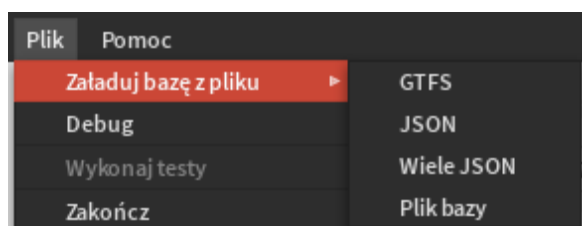
Parametry – przycisk wyświetla okno, w którym możliwe jest wybranie algorytmu pracy i ustawienie wszystkich potrzebnych parametrów.



Rysunek 2 Okno parametrów pracy

Na liście u góry możliwy jest wybór dowolnego algorytmu z zaimplementowanych, po wybraniu możliwa jest zmiana wszystkich parametrów pracy danego algorytmu – dla algorytmu symulowanego wyżarzania zostały one opisane wcześniej.

Po uruchomienie programu konieczne jest załadowanie struktury danych reprezentującą sieć komunikacyjną.



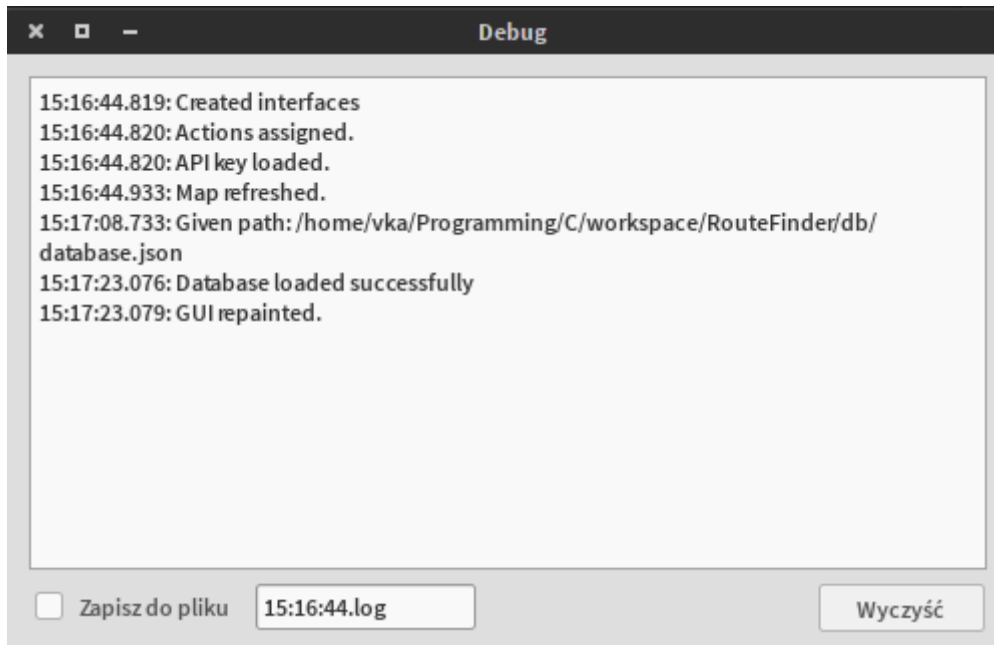
Rysunek 3 Menu Plik

Dostępne jest kilka opcji ładowania struktury danych. Trzy pierwsze są stosunkowo nieprzydatne w trakcie pracy w trybie graficznym – po ich użyciu wymagana jest długotrwała konwersja danych do typu użytecznego dla algorytmu. W trakcie normalnej pracy optymalnym rozwiązaniem jest użycie opcji **Plik bazy**, ładującej gotową do pracy strukturę danych. W trakcie pracy z bazą Krakowa należy załadować plik *db/database.json*. Ładowanie trwa kilka sekund.

Wykonaj testy – uruchamia szereg testów na załadowanej obecnie bazie danych. Opcja jest nieaktywna w trybie graficznym ze względu na czas trwania testowania – aplikacja pozostaje nieaktywna

przez tak długi czas, że system operacyjny może zakończyć proces, uznając że program się zawiesił. Problem ten nie występuje w trybie konsolowym, dodatkowo jest to szybszy sposób testowania.

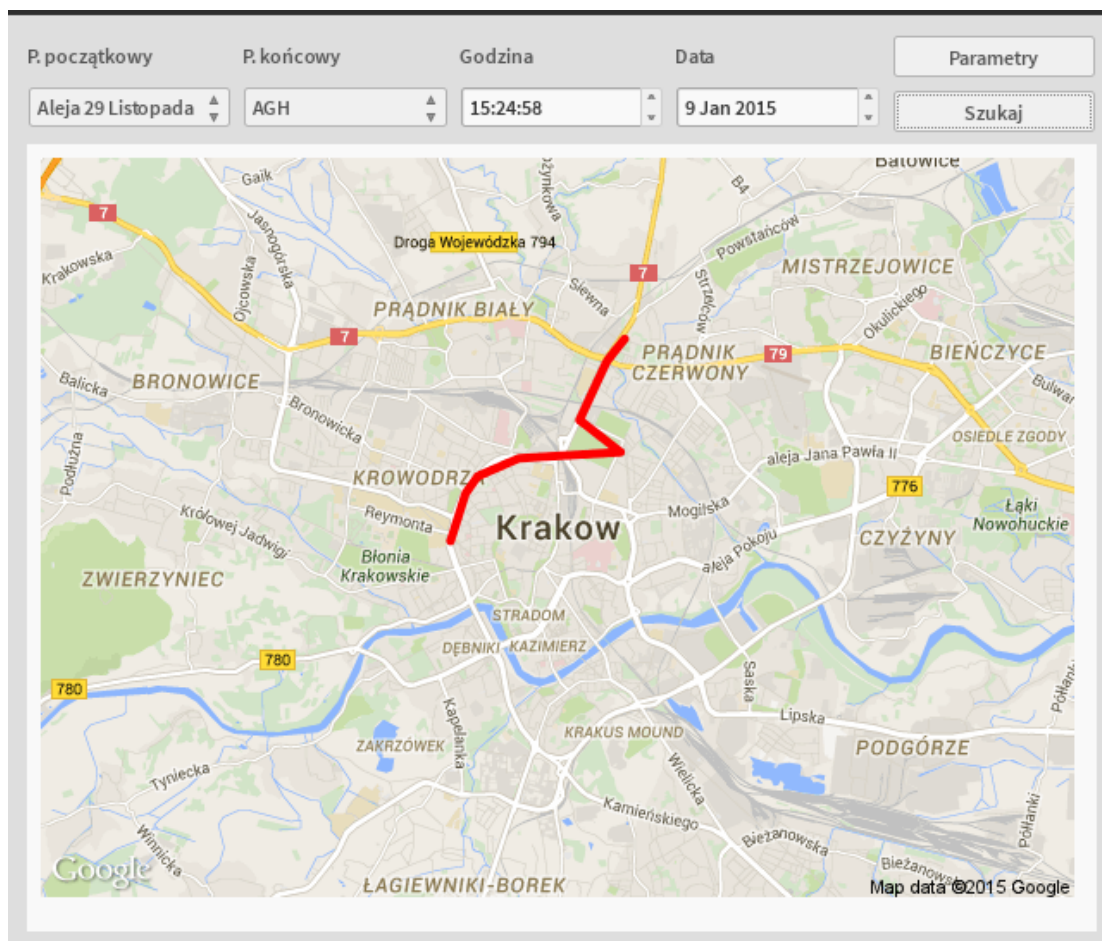
Debug – uruchamia okno informacyjne, pozwalające śledzić przebieg pracy programu a także zapis logu do pliku.



Rysunek 4 Okno Debug

Wynik pracy algorytmu prezentowany w formie graficznej, na mapie w głównym oknie programu.

Reprezentacja graficzna ma za zadanie przedstawić przebieg wynikowego połączenia w ogólnej formie, tak by intuicyjnie można było ocenić, czy wynik jest zadowalający. W tym celu zdecydowaliśmy się na przedstawienie na mapie, co było możliwe dzięki współrzędnym geograficznym każdego przystanku, które znajdowały się w bazie danych.



Rysunek 5 Wynik pracy programu

Towarzyszy temu również wypisanie wyniku w formie tekstowej – ograniczyliśmy się do takiej formy prezentacji wyników, by niepotrzebnie nie powiększać nakładu pracy potrzebnego na dopracowanie interfejsu – uznaliśmy, że kluczowym zagadnieniem jest dopracowanie algorytmu.

Route:

[5]->	[723]-	[987623]	Aleja 29 Listopada	Opolska Estakada
[723]->	[1146]-	[1566159]	Opolska Estakada	Uniwersytet Rolniczy
[1146]->	[80]-	[110426]	Uniwersytet Rolniczy	Biskupa Prandoty
[80]->	[191]-	[260986]	Biskupa Prandoty	Cmentarz Rakowicki Zachód
[191]->	[818]-	[1117579]	Cmentarz Rakowicki Zachód	Politechnika
[818]->	[694]-	[948822]	Politechnika	Nowy Kleparz
[694]->	[323]-	[441912]	Nowy Kleparz	Grottgera
[323]->	[793]-	[1083561]	Grottgera	Plac Inwalidów
[793]->	[2]-	[3525]	Plac Inwalidów	AGH

Total time: 20

From: 15:22 to: 15:42

Listing 1 Przykładowe wyjście program

Prezentuje on ciąg odwiedzonych przystanków, identyfikatory punktów oraz połączeń, a także informacje na temat czasu połączenia.

5 TESTY

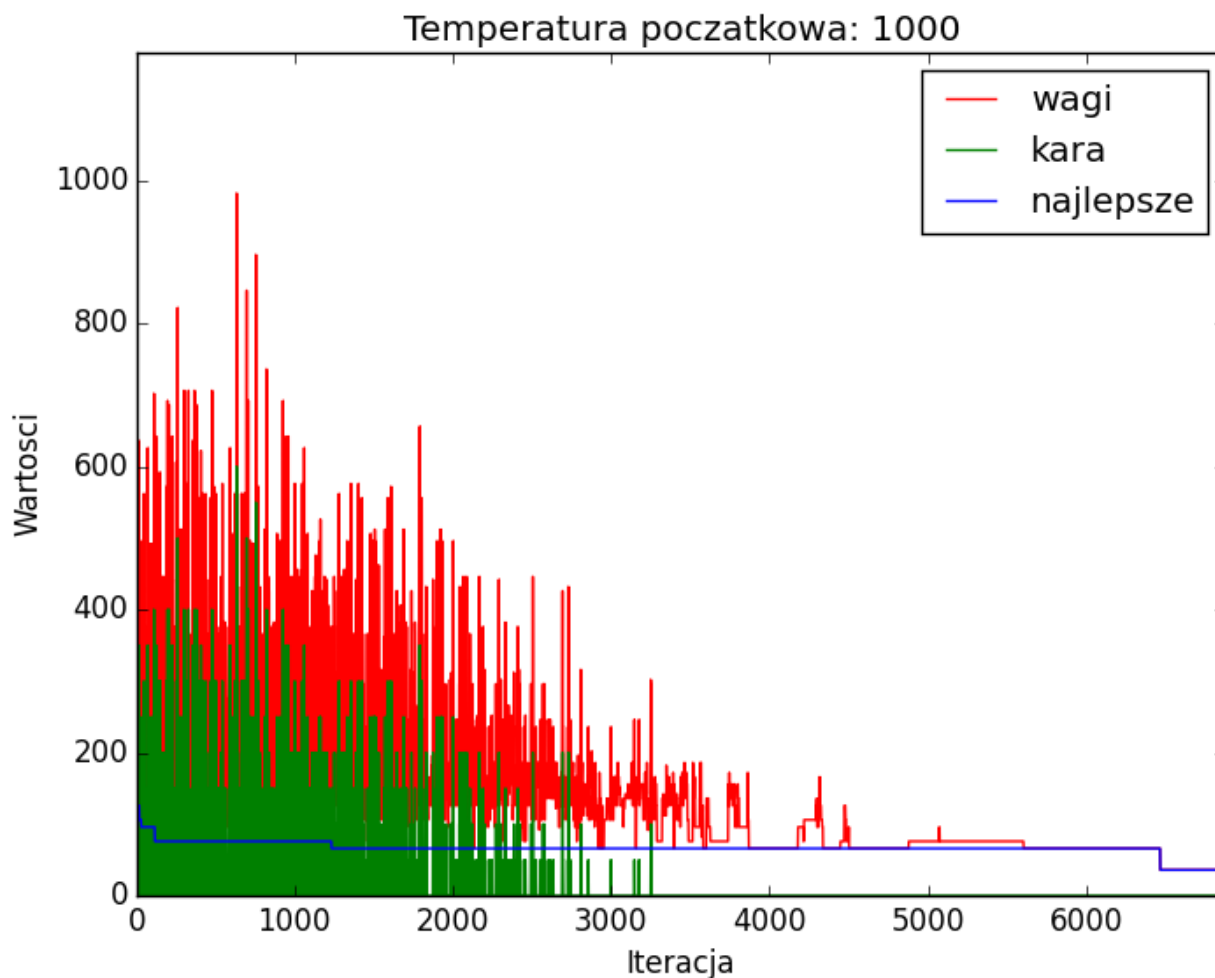
Testy zachowania algorytmu zostały przeprowadzone na losowo wygenerowanej siatce punktów rozmiaru $n \times n$ z której pewien procent punktów został wskazany jako przystanek. Rozkład jazdy został ustalony statycznie na zadzie co przejazdów góra - dół, lewo-prawo co określony czas.

Testom zostały poddane następujące parametry:

5.1 TEMPERATURA POCZĄTKOWA

Temperatura jest jednym z najważniejszych parametrów algorytmu. Wpływa on na to jak „chętnie” algorytm akceptuje gorsze rozwiązania. W początkowej fazie program jest w stanie często zaakceptować gorsze rozwiązanie, po to aby spróbować je poprawić. Działanie takie może dać lepszy rezultat końcowy niż akceptacja oraz poprawa tylko lepszych rozwiązań. Oczywistym jest także wpływ wartości temperatury początkowej na ilość iteracji i długość wykonywania się programu.

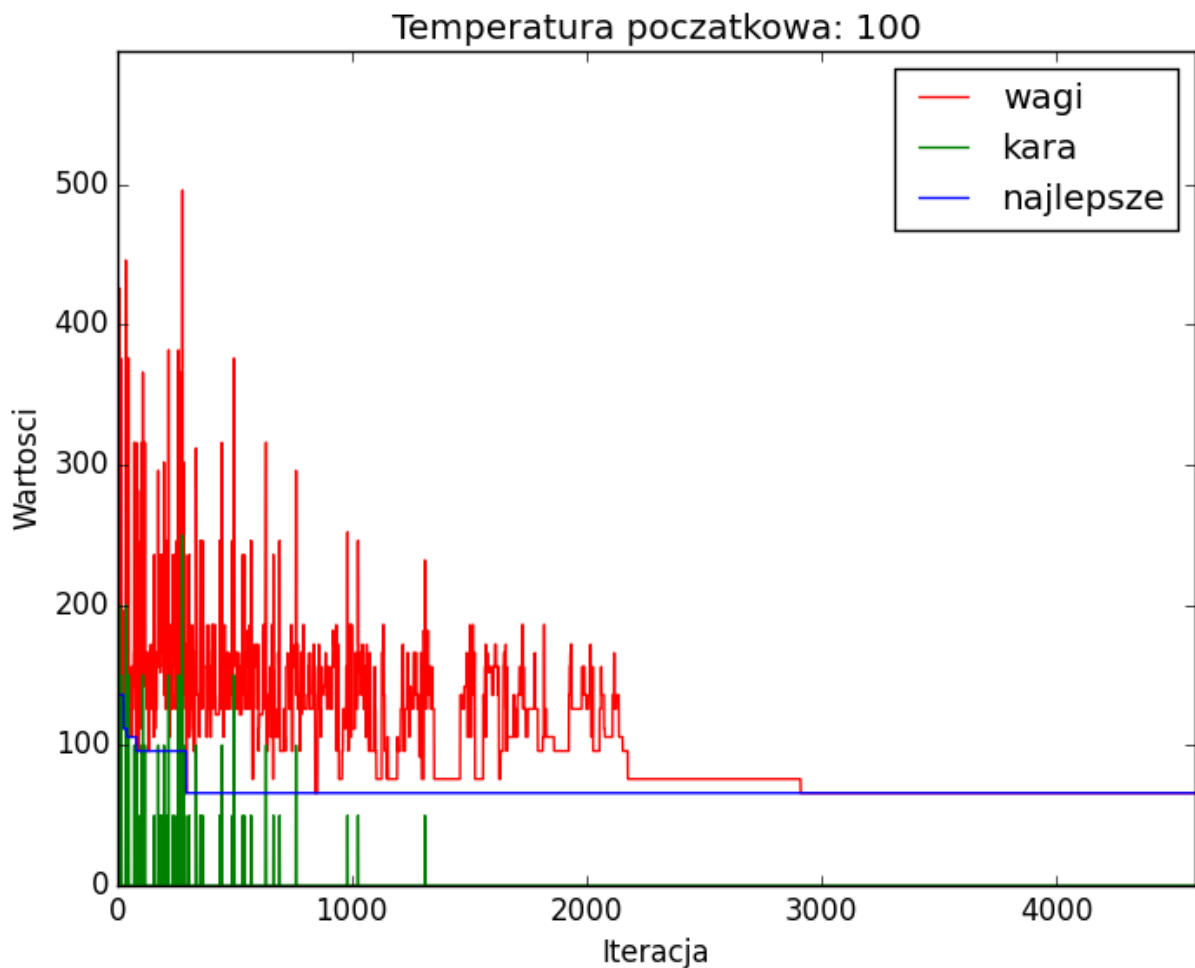
Przykładowy przebieg operacji znajdowania rozwiązania dla temperatury startowej $T=1000$:



Rysunek 6 Zapis pracy algorytmu dla sieci o 400 przystankach.

Na powyższym obrazku widać że algorytm początkowo obraca się w sferze rozwiązań niedopuszczalnych, o bardzo niekorzystnej wartości. Wraz ze spadkiem temperatury spada także wartość funkcji celu aż zostanie znalezione rozwiązanie minimalne, a rozwiązania „nieoptymalne” nie są już tak chętnie akceptowane.

Poniżej znajduje się przebieg znajdowania rozwiązania z inną wartością początkową $T=100$:

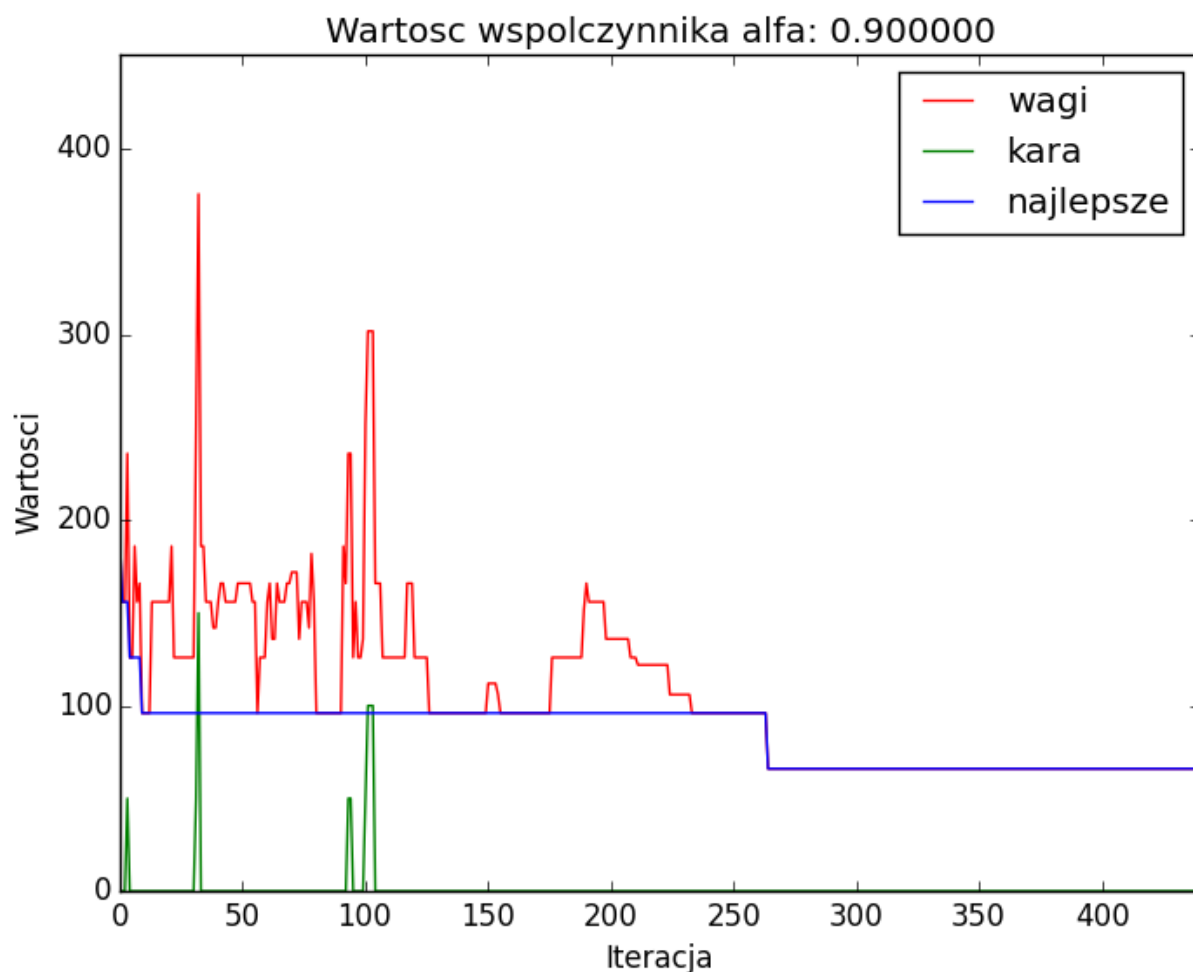


Rysunek 7 Zapis pracy algorytmu dla sieci o 400 przystankach.

W tym przypadku algorytm działał szybciej oraz mniej czasu przebywał w obszarze niedopuszczalnych rozwiązań. Minusem zastosowania mniejszej temperatury było jednak znalezienie nieco gorszego rozwiązania.

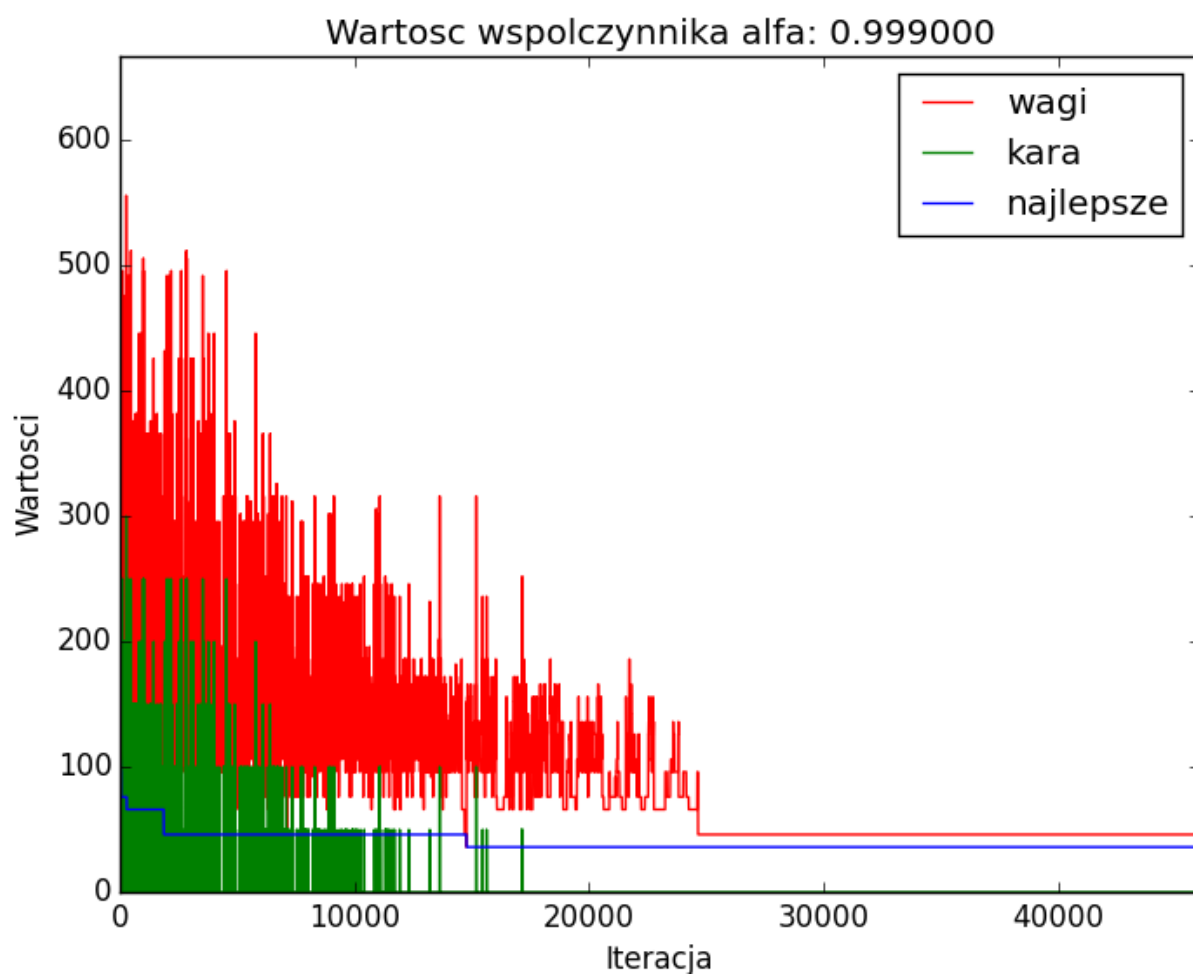
5.2 PARAMETR ALFA

Parametr alfa określa szybkość spadku temperatury po wykonaniu zadanej ilości iteracji, co ma wpływ na przetwarzanie przez algorytm rozwiązań o mniej korzystnych wartościach. Zmniejszenie tego parametru przyspiesza ochładzanie temperatury, skraca czas działania algorytmu co może powodować pogorszenie ostatecznego rozwiązania.



Rysunek 8 Zapis pracy algorytmu dla sieci o 400 przystankach.

Analizując dwa przykładowe przebiegi (drugi znajduje się na następnej stronie), można zauważyć znaczenie parametru alfa dla pracy algorytmu. Zwiększenie jego wartości wpływa wykładniczo na długość pracy algorytmu, a więc również na ilość badanych rozwiązań. Należy więc dobrać go tak, by czas wykonywania programu nie był zbyt długi, a jednocześnie przegląd rozwiązań – zadowalający. W naszym przypadku, wartość optymalna zbliżona była do wartości 0.99-0.999.

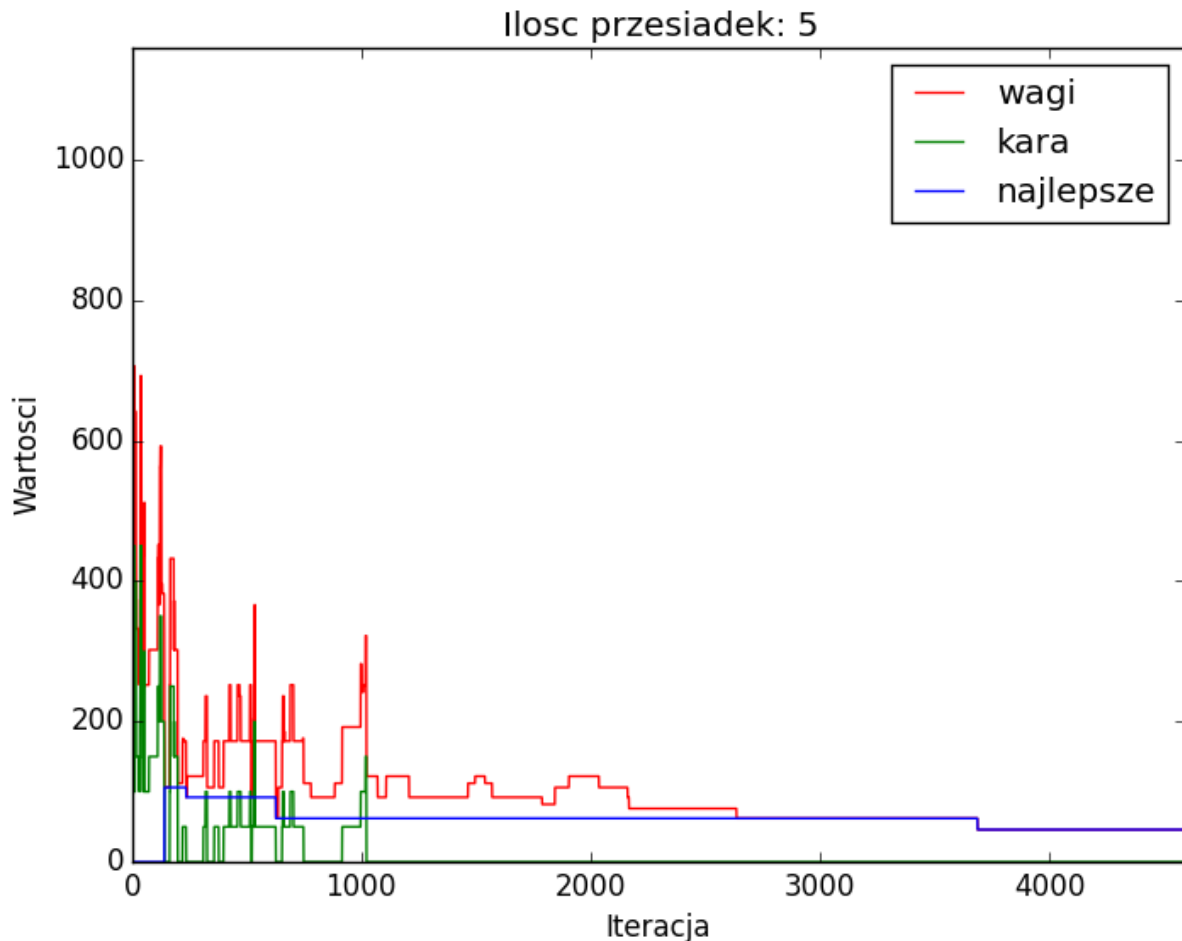


Rysunek 9 Zapis pracy algorytmu dla sieci o 400 przystankach.

Wartość współczynnika alfa wpływa na ilość iteracji względem temperatury – ma to istotne znaczenie, ponieważ przy wyższych temperaturach prawdopodobieństwo wybrania mniej korzystnego rozwiązania jest większe. Zmniejszając współczynnik alfa zmniejszamy więc czas pracy, gdy algorytm jest chętny na przyjmowanie nieoptymalnych rozwiązań.

5.3 LICZBA DOZWOLONYCH PRZESIADK

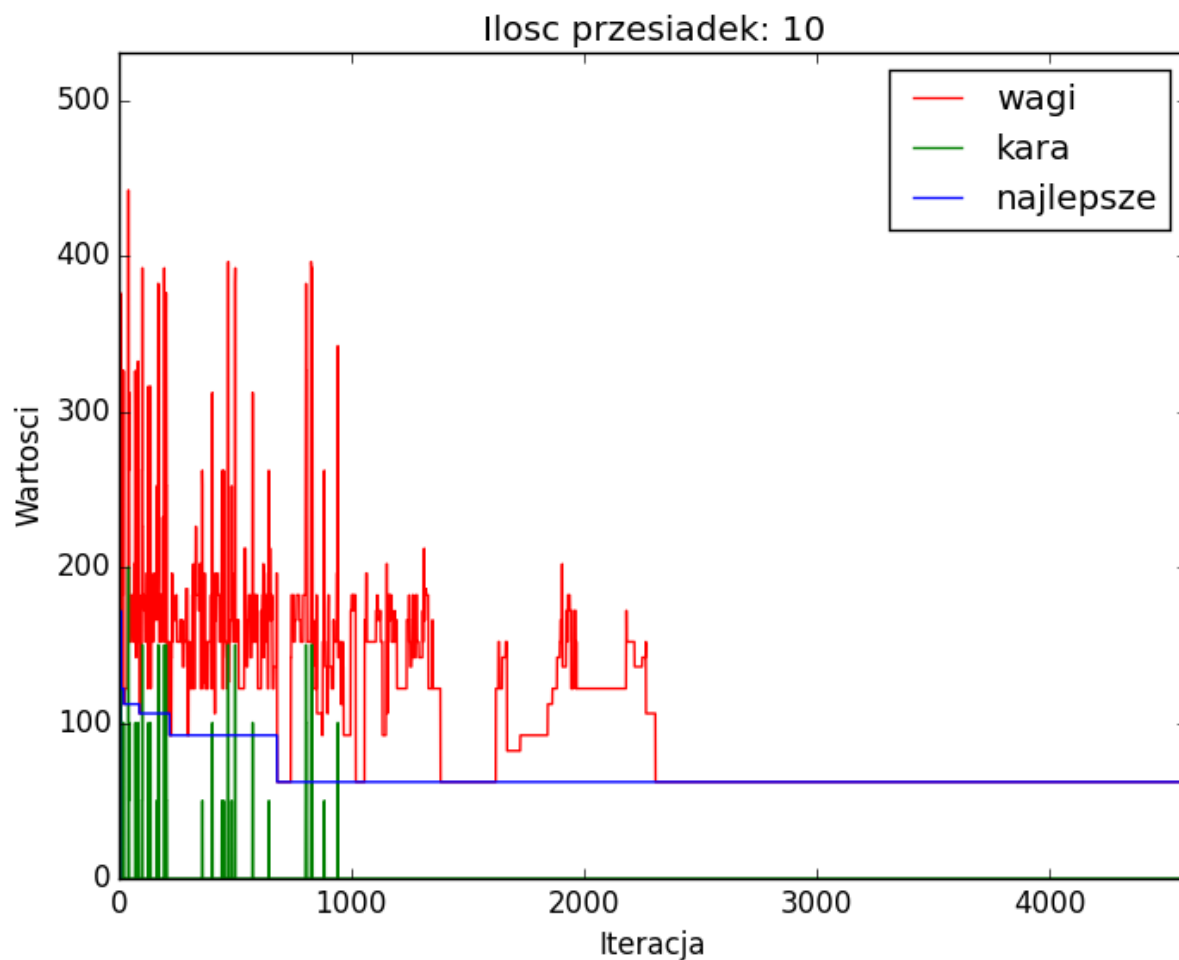
Liczba dozwolonych przesiadek jest parametrem wpływającym na to czy rozwiązanie jest dopuszczalne co w wyniku wpływa na ocenę rozwiązania.



Rysunek 10 Zapis pracy algorytmu dla sieci o 2500 przystankach.

W powyższym przypadku algorytm przeszukuje rozwiązania niedopuszczalne o bardzo dużej wartości funkcji kary. Sprawia to że wraz ze spadkiem temperatury algorytm opuszcza ten obszar znajdując rozwiązanie optymalne.

Warto zwrócić uwagę na przebieg wartości przypisywanej najlepszemu rozwiązaniu. Przez dłuższy czas rozwiązanie najlepsze nie istnieje, co potwierdza również niezerowa wartość funkcji kary na tym przedziale. Pozostaje to więc w zgodzie z założeniami przy wyszukiwaniu pierwszego rozwiązania – nawet jeśli rozwiązanie będzie niedopuszczalne, to zostanie szybko poprawione do formy dopuszczalnej.



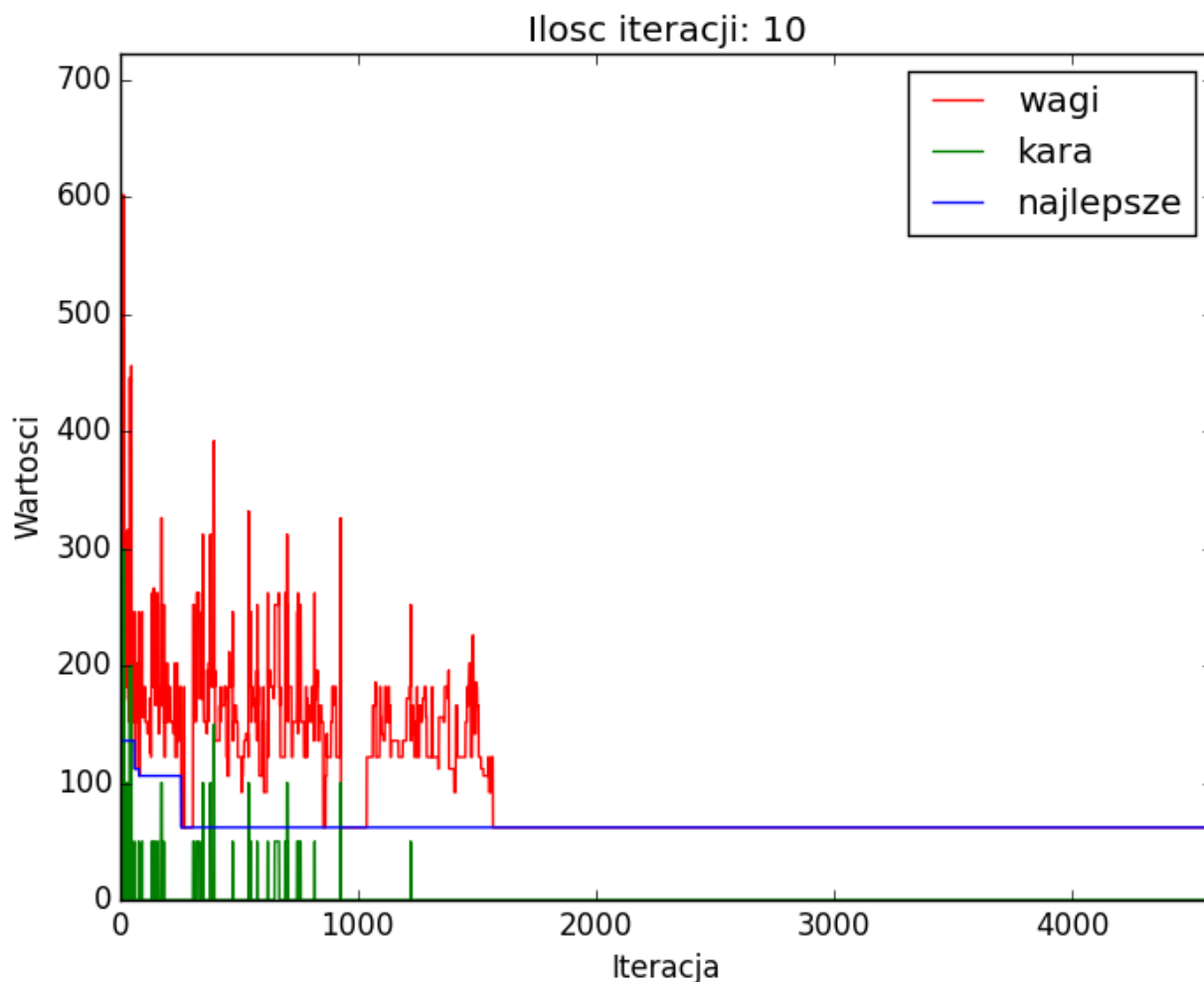
Rysunek 11 Zapis pracy algorytmu dla sieci o 2500 przystankach.

Przy zwiększonej dopuszczalnej liczbie przesiadek wagi znajdujących rozwiązań w początkowej fazie algorytmu są mniejsze, przez co algorytm zmierza wolniej do rozwiązania optymalnego.

W tym przypadku pierwsze rozwiązanie spełnia warunki ilości dopuszczalnych przesiadek, więc następuje szybsza optymalizacja w początkowej fazie.

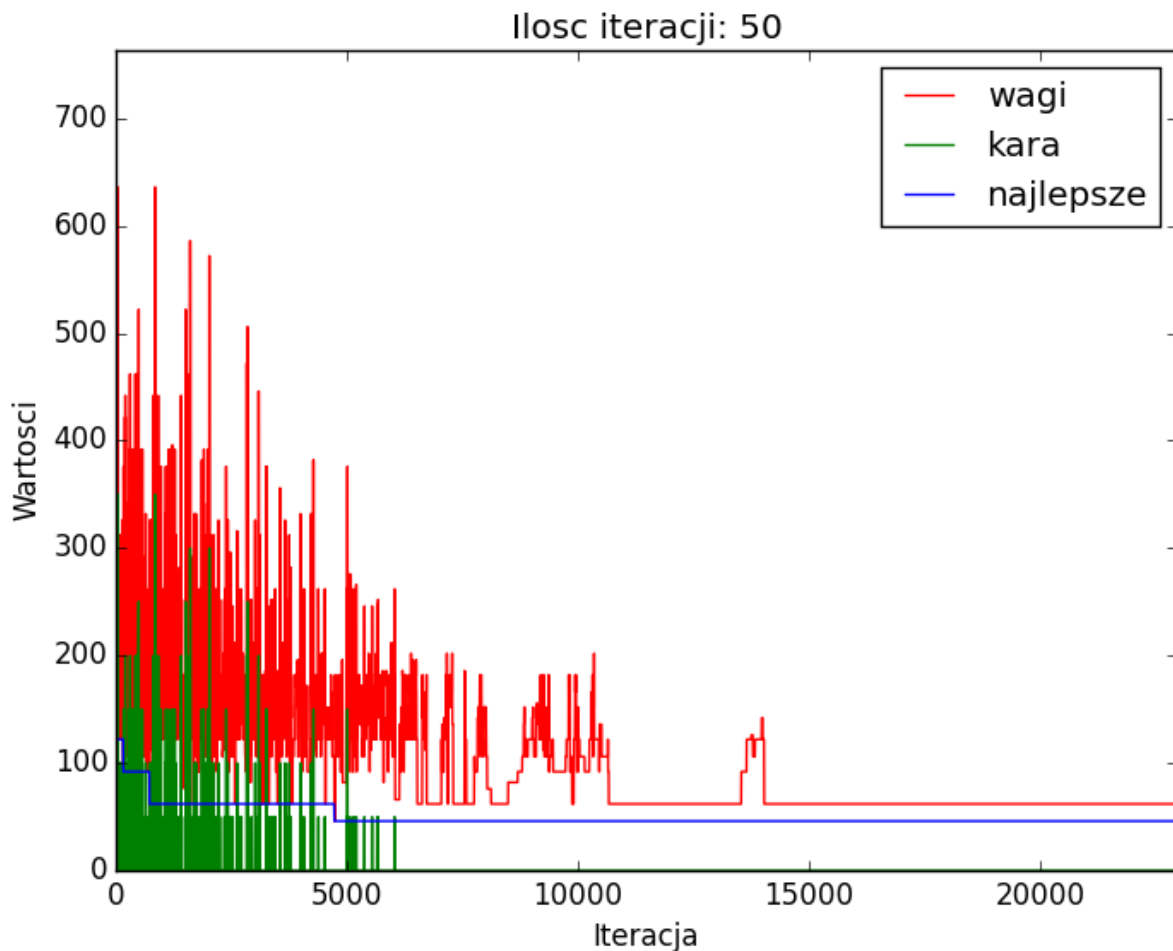
5.4 PARAMETR K

Kolejne dwa wykresy przedstawiają wpływ parametru K na działanie programu. Zwiększenie tej liczby powoduje wzrost ilości iteracji i czasu obliczeń. Z drugiej strony wynikiem takiego działania jest bogatsze przeszukiwanie otoczenia co może prowadzić do znalezienia lepszego rozwiązania.



Rysunek 12 Zapis pracy algorytmu dla sieci o 2500 przystankach.

Zmiana parametru K wpływa linowo na czas trwania programu. Na wykresie powyżej można zauważyć, że wykonanie trwało około 4500 iteracji, w tym tylko około 1500 przeszukiwało „aktywnie” otoczenie – wynika to z doboru parametrów, w tym przypadku prawdopodobieństwo przyjęcia mniej optymalnych rozwiązań było zbyt małe, więc algorytm utknął na jednym rozwiązaniu.



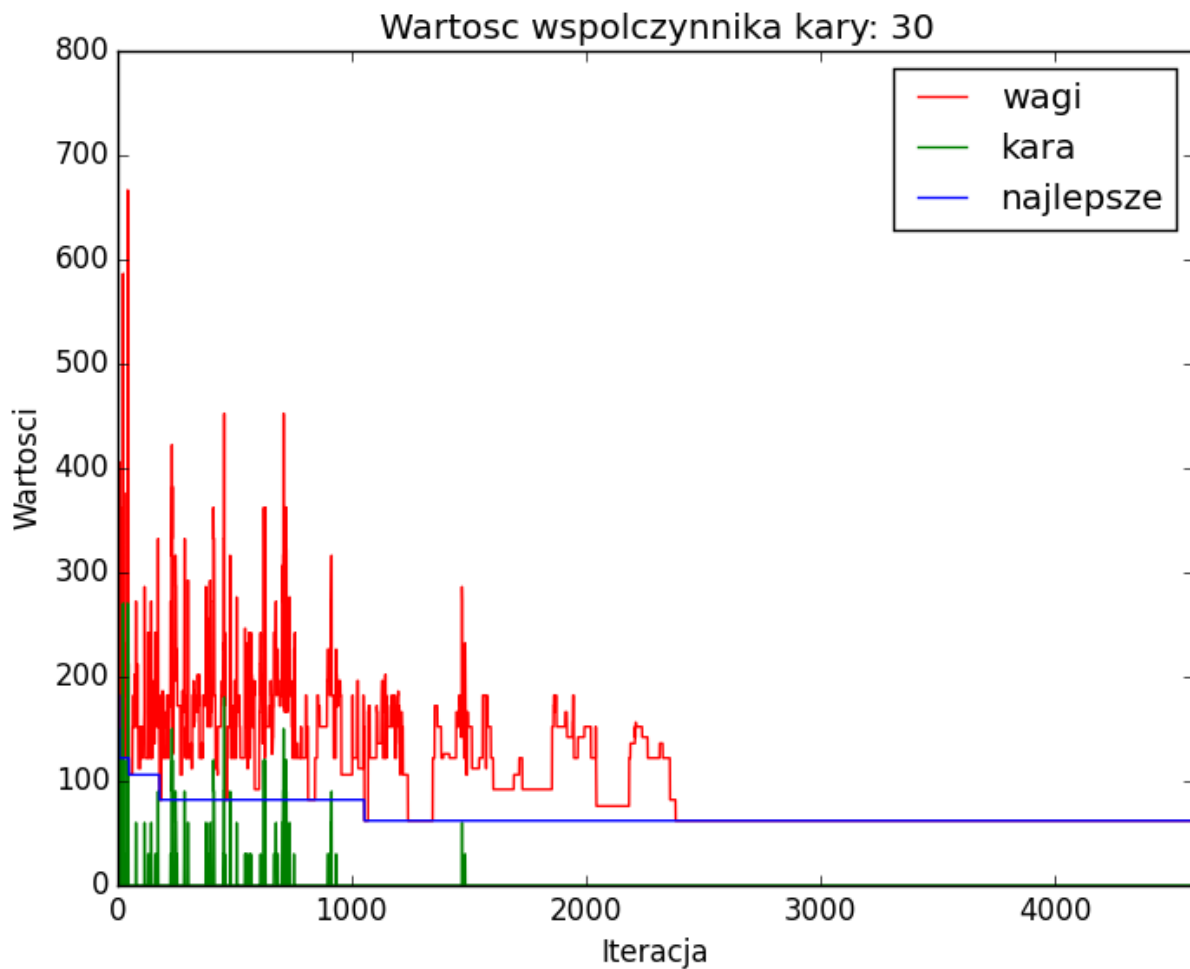
Rysunek 13 Zapis pracy algorytmu dla sieci o 900 przystankach.

W przeciwieństwie do współczynnika alfa, parametr K nie wpływa na rozkład prawdopodobieństw – oczywistym jest jednak, że zmniejszając współczynnik możemy sprawić, że otoczenie rozwiązań nie będzie dokładnie przeszukiwane, co wpłynie na wartość rozwiązania.

W tym przypadku algorytm wykonywał się około 22000 iteracji, w tym prawie 15000 tysięcy pozwalało na aktywne przeszukiwanie rozwiązań. Warto zwrócić uwagę, że wszystkie rozwiązania od 6000 iteracji były dopuszczalne – kara za akceptację rozwiązania niedopuszczalnego stała się w tym momencie zbyt duża w porównaniu do wartości najlepszego rozwiązania dopuszczalnego.

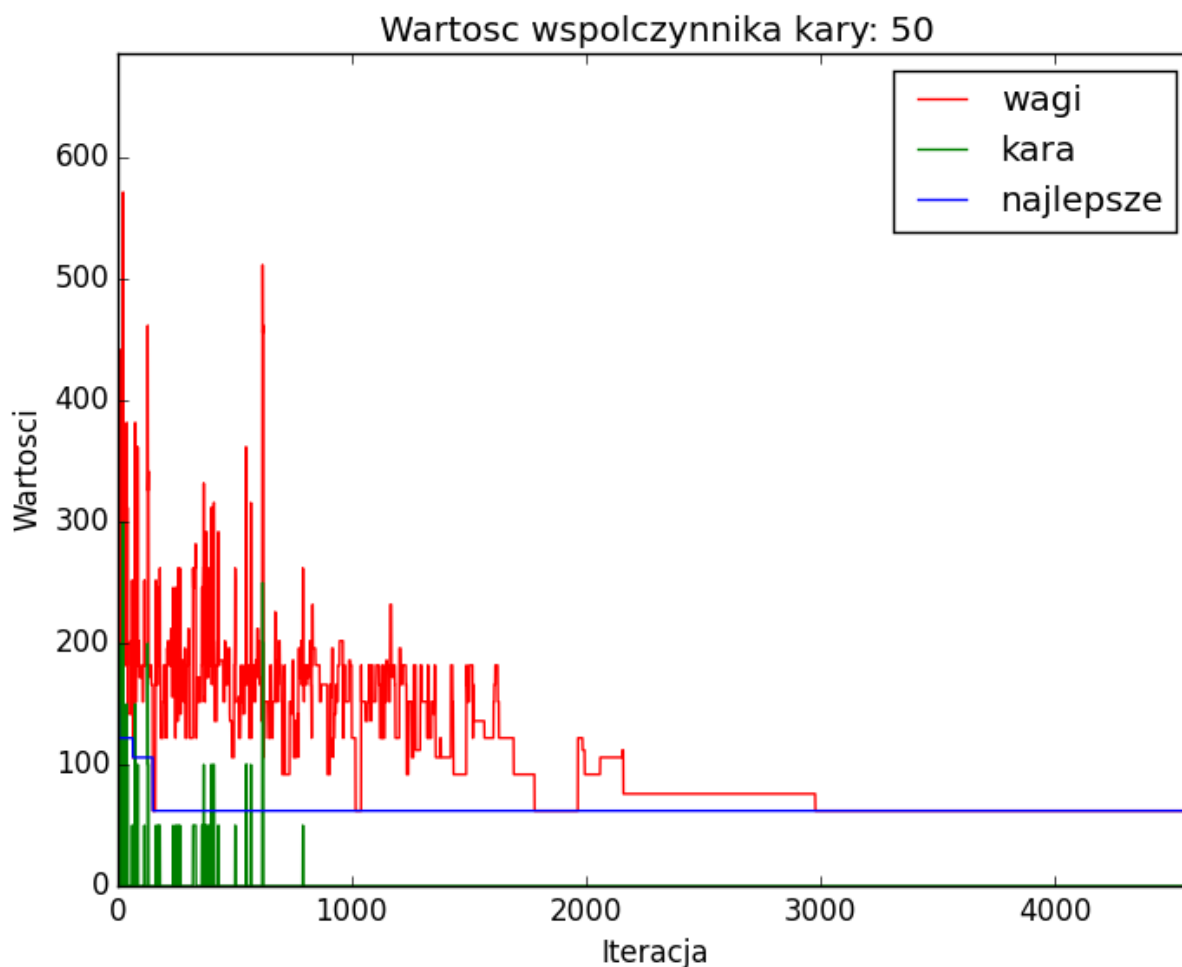
5.5 KARA ZA PRZESIADKĘ

Kara za przesiadkę ma wpływ na wartość funkcji kary, będącej składnikiem funkcji oceny rozwiązania.



Rysunek 14 Zapis pracy algorytmu dla sieci o 2500 przystankach.

Niska wartość współczynnika kary sprawia, że rozwiązania niedopuszczalne są chętniej wybierane. Widać to w początkowej fazie pracy algorytmu, gdzie wartości funkcji kary są duże, a mimo to rozwiązanie jest akceptowalne.



Rysunek 15 Zapis pracy algorytmu dla sieci o 2500 przystankach.

Wzrost wartości współczynnika kary pociąga za sobą wzrost wartości rozwiązań niedopuszczalnych. Zjawisko to powoduje że algorytm rzadziej przyjmuje do poprawy takie rozwiązania, skupiając się na przeszukiwaniu i optymalizacji rozwiązań dopuszczalnych. Istotny jest więc taki dobór współczynnika kary, by rozwiązania nieoptymalne były analizowane w stopniu umożliwiającym dobrą optymalizację rozwiązania, ale by algorytm bardziej skupiał się na rozwiązaniach dopuszczalnych.

Można również rozważyć skuteczność zmiennego w czasie współczynnika funkcji kary – małego, lub nawet zerowego na początku i rosnącego wraz z każdą iteracją. Pozwoliłoby to na swobodną analizę wszystkich rozwiązań i stopniowe pozbywanie się niedopuszczalnych wraz z kolejnymi iteracjami.

5.6 PRZYPADKI ZŁOŚLIWE

W trakcie pracy algorytmu może dojść do sytuacji, które spełniają wszystkie założenia pracy, jednak wpływają z różnych powodów negatywnie na możliwość znalezienia optymalnego rozwiązania. Do takich złośliwych sytuacji, które udało nam się znaleźć w trakcie pracy, należą związane z długim oczekiwaniem – jest to szczególnie powiązanie z dwoma rozkładami jazdy – nocnym i dziennym, co powoduje duże opóźnienia, jeśli dany przystanek posiada połączenia związane tylko z jednym z nich. Innym oczywistym przypadkiem byłaby próba szukania połączenia bez przesiadek, gdy żadne bezpośrednie połączenie między dwoma przystankami nie istnieje.

5.6.1 DŁUGIE OCZEKIWANIE NA PIERWSZE POŁĄCZENIE

Pierwszy kurs jest dostępny dopiero za 70 minut, więc sztucznie zawyża wartość funkcji celu.

przystanek początkowy Balice I

przystanek końcowy: Bojki

czas rozpoczęcia: 10:00

Tend: 100

k: 10

alfa: 0.95

allowed: 5

punish: 25

wait 71 minutes

go from Balice I on time 11:11 to Balice II on time 11:12 with trip 186

go from Balice II on time 11:12 to Budzyń Zalew na Piaskach on time 11:16 with trip 186

go from Budzyń Zalew na Piaskach on time 11:16 to Kryspinów on time 11:18 with trip 186

go from Kryspinów on time 11:18 to Granica Miasta on time 11:19 with trip 186

go from Granica Miasta on time 11:19 to Bielany on time 11:20 with trip 186

go from Bielany on time 11:20 to Bielany Szkoła on time 11:21 with trip 186

go from Bielany Szkoła on time 11:21 to Bielany Klasztor on time 11:22 with trip 186

go from Bielany Klasztor on time 11:22 to Wodociągi on time 11:23 with trip 186

go from Wodociągi on time 11:23 to Bielańskie Skały on time 11:25 with trip 186

go from Bielańskie Skały on time 11:25 to Na Krępaku on time 11:26 with trip 186

go from Na Krępaku on time 11:26 to Zaskale on time 11:27 with trip 186

go from Zaskale on time 11:27 to Przegorzały on time 11:28 with trip 186

go from Przegorzały on time 11:28 to Glinnik on time 11:29 with trip 186

go from Glinnik on time 11:29 to Benedyktowicza on time 11:30 with trip 186

go from Benedyktowicza on time 11:30 to Księcia Józefa on time 11:31 with trip 186

go from Księcia Józefa on time 11:31 to Malczewskiego on time 11:32 with trip 186

wait 49 minutes

go from Malczewskiego on time 12:21 to Zielińskiego on time 12:23 with trip 51

go from Zielińskiego on time 12:23 to Kapelanka on time 12:25 with trip 51

go from Kapelanka on time 12:25 to Szwedzka on time 12:26 with trip 51

go from Szwedzka on time 12:26 to Centrum Kongresowe on time 12:28 with trip 51

wait 3 minutes

go from Centrum Kongresowe on time 12:31 to Ludwinów on time 12:32 with trip 152

go from Ludwinów on time 12:32 to Rondo Matecznego on time 12:34 with trip 152

wait 8 minutes

go from Rondo Matecznego on time 12:42 to Puskarska on time 12:45 with trip 140

go from Puskarska on time 12:45 to Zajezdnia Wola Duchacka on time 12:47 with trip 140

go from Zajezdnia Wola Duchacka on time 12:47 to Wola Duchacka on time 12:49 with trip 140

go from Wola Duchacka on time 12:49 to Karpińskiego on time 12:50 with trip 140

go from Karpińskiego on time 12:50 to Nowosądecka on time 12:52 with trip 140

go from Nowosądecka on time 12:52 to Witosa on time 12:54 with trip 180

go from Witosa on time 12:54 to Kurdwanów on time 12:56 with trip 180

go from Kurdwanów on time 12:56 to Bujaka on time 12:57 with trip 180

wait 19 minutes

go from Bujaka on time 13:16 to Wysłouchów on time 13:18 with trip 98

go from Wysłouchów on time 13:18 to Bojki on time 13:19 with trip 98

Listing 2 Przebieg pracy algorytmu

5.6.2 DŁUGIE OCZEKIWANIE POMIĘDZY POŁĄCZENIAMI

przystanek początkowy AWF
przystanek końcowy: Buków
czas rozpoczęcia: 10:15
Tend: 100
k: 10
alfa: 0.95
allowed: 5
punish: 25

```
trip start time: 10:15
go from AWF on time 10:15 to Stella-Sawickiego on time 10:17 with trip 7
wait 10 minutes
go from Stella-Sawickiego on time 10:27 to Nowohucka on time 10:28 with trip 157
go from Nowohucka on time 10:28 to M1 Nowohucka on time 10:30 with trip 157
go from M1 Nowohucka on time 10:30 to Elektrociepłownia Kraków on time 10:32 with
trip 157
go from Elektrociepłownia Kraków on time 10:32 to Koszykarska on time 10:33 with
trip 157
go from Koszykarska on time 10:33 to Stoczniovców on time 10:35 with trip 157
go from Stoczniovców on time 10:35 to Kuklińskiego on time 10:37 with trip 157
go from Kuklińskiego on time 10:37 to Powstańców Wielkopolskich on time 10:40 with
trip 157
go from Powstańców Wielkopolskich on time 10:40 to Tischnera on time 10:44 with
trip 157
go from Tischnera on time 10:44 to Łagiewniki on time 10:46 with trip 157
wait 879 minutes
go from Łagiewniki on time 01:25 to Borek Fałęcki on time 01:28 with trip 357
go from Borek Fałęcki on time 01:28 to Góra Borkowska on time 01:30 with trip 357
go from Góra Borkowska on time 01:30 to Judyma on time 01:32 with trip 357
go from Judyma on time 01:32 to Kąpielowa on time 01:33 with trip 357
wait 137 minutes
go from Kąpielowa on time 03:50 to Opatkowice Wiadukt on time 03:51 with trip 229
go from Opatkowice Wiadukt on time 03:51 to Opatkowice on time 03:53 with trip 229
go from Opatkowice on time 03:53 to Ważewskiego on time 03:54 with trip 229
wait 27 minutes
go from Ważewskiego on time 04:21 to Libertów on time 04:23 with trip 283
go from Libertów on time 04:23 to Gaj Szkoła on time 04:25 with trip 283
go from Gaj Szkoła on time 04:25 to Gaj Zadziele on time 04:26 with trip 283
go from Gaj Zadziele on time 04:26 to Mogilany Rynek on time 04:28 with trip 283
wait 4 minutes
go from Mogilany Rynek on time 04:32 to Mogilany Cmentarz on time 04:34 with trip
245
go from Mogilany Cmentarz on time 04:34 to Kopce on time 04:36 with trip 245
go from Kopce on time 04:36 to Chorowice on time 04:38 with trip 245
go from Chorowice on time 04:38 to Kulerzów Wąwóz on time 04:40 with trip 245
go from Kulerzów Wąwóz on time 04:40 to Kulerzów on time 04:41 with trip 245
go from Kulerzów on time 04:41 to Buków Szkoła on time 04:43 with trip 245
go from Buków Szkoła on time 04:43 to Buków Klin on time 04:44 with trip 245
wait 1106 minutes
go from Buków Klin on time 23:10 to Buków on time 00:00 with trip 245
```

Listing 3 Przebieg działania algorytmu

W pewnych przypadkach może zdarzyć się, że pasażer „utknie” na przystanku, gdzie będzie musiał czekać wiele godzin na połączenie, kiedy przystanek ten obsługuje wyłącznie połączenia dzienne /

nocne. W prezentowanym przypadku algorytm znalazł połączenie, które wymaga niemal całego dnia oczekiwania na następny ruch. Wynika to z niefortunnego doboru parametrów – wystarczyło nieznacznie zmienić temperaturę początkową i znalezione rozwiązanie było znacznie lepsze.

5.6.3 ZBYT MAŁA LICZBA PRZESIADK

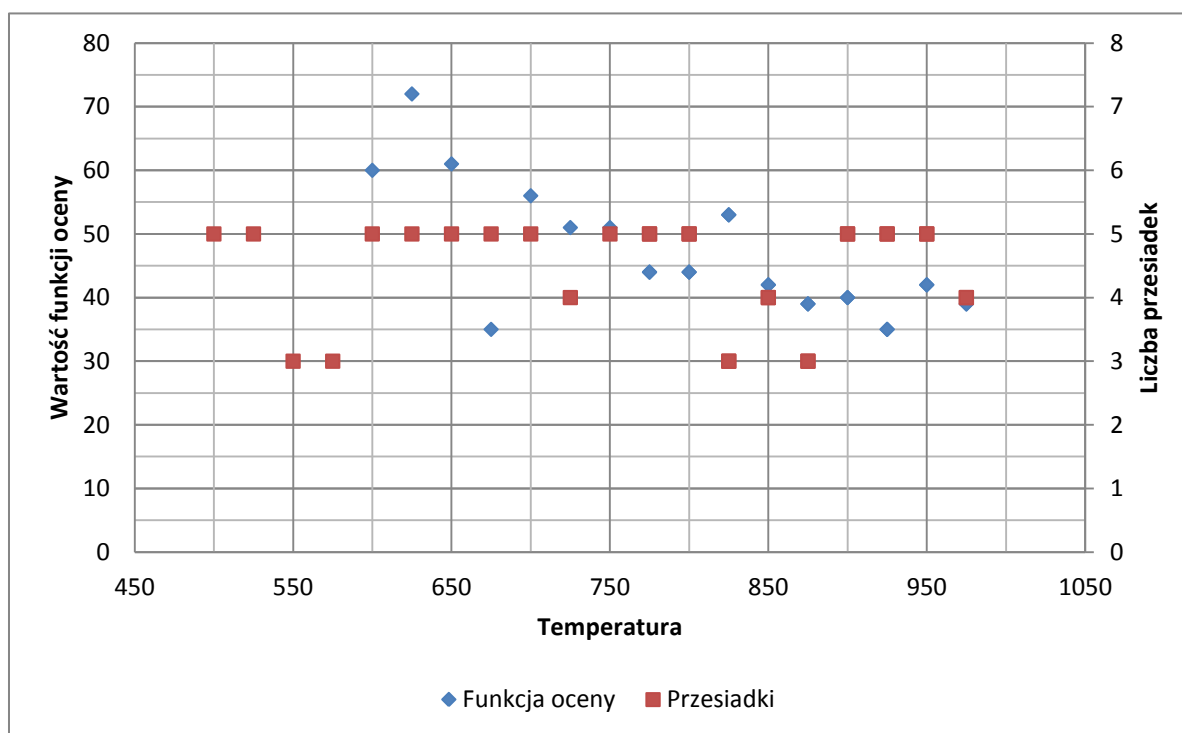
Innym złośliwym zagadnieniem może być zbyt mała liczba dopuszczalnych przesiadek. Jeśli wszystkie znalezione rozwiązania nie będą dopuszczalne, algorytm zwróci błąd i brak rozwiązania. Nie mają wtedy znaczenia żadne inne parametry, który być może wydłużą niepotrzebnie czas symulacji, a nie mają wpływu na końcowy wynik.

5.7 WPŁYW POSZCZEGÓLNYCH PARAMETRÓW NA WYNIKI

Poniżej przedstawiamy wykresy pokazujące wpływ poszczególnych parametrów na wynik pracy algorytmu.

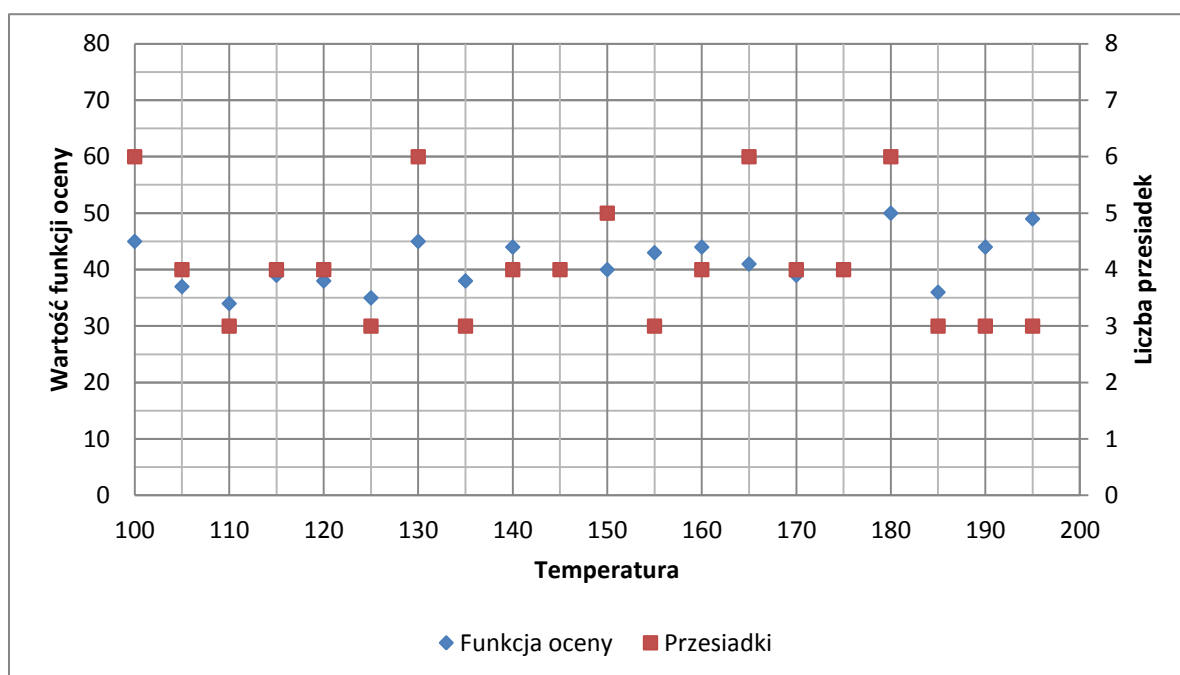
Domyślne parametry to: trasa z M1 Al. Pokoju do AGH, czas rozpoczęcia to 9:30. K: 10, alfa: 0.95, przesiadki: 10, współczynnik kary: 25.

5.7.1 TEMPERATURA POCZĄTKOWA



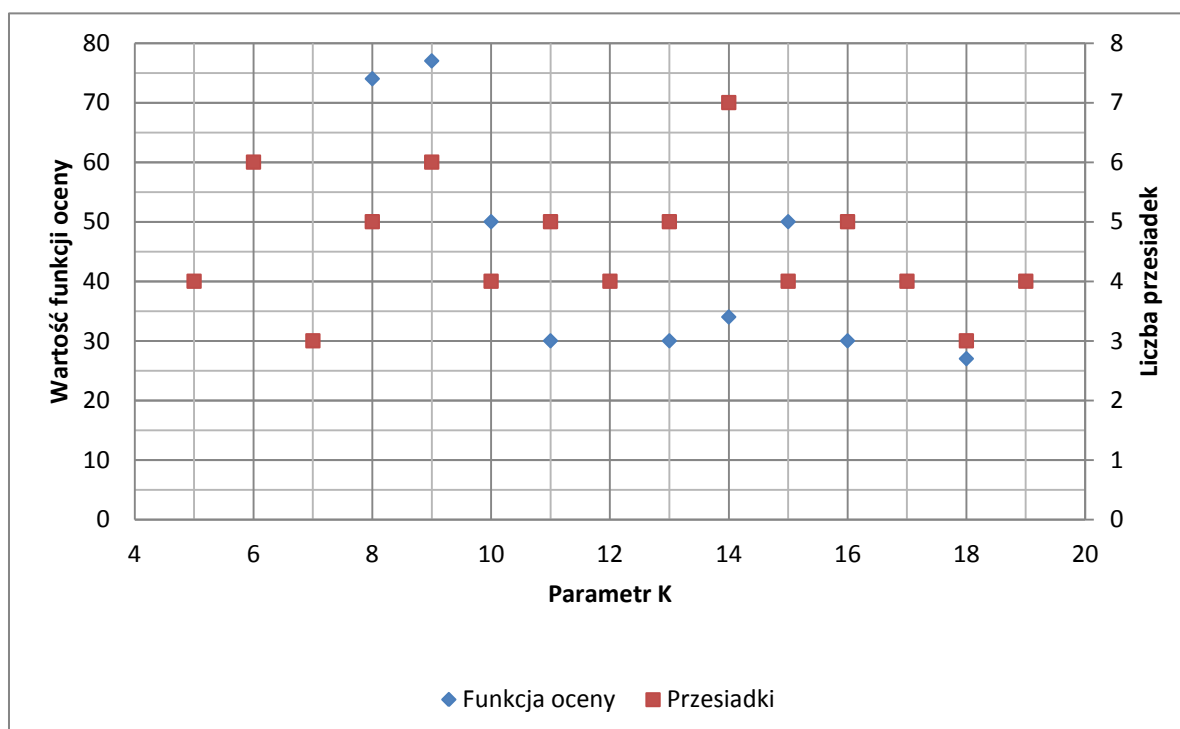
Rysunek 16 Wpływ temperatury początkowej na rozwiązanie.

5.7.2 TEMPERATURA KOŃCOWA



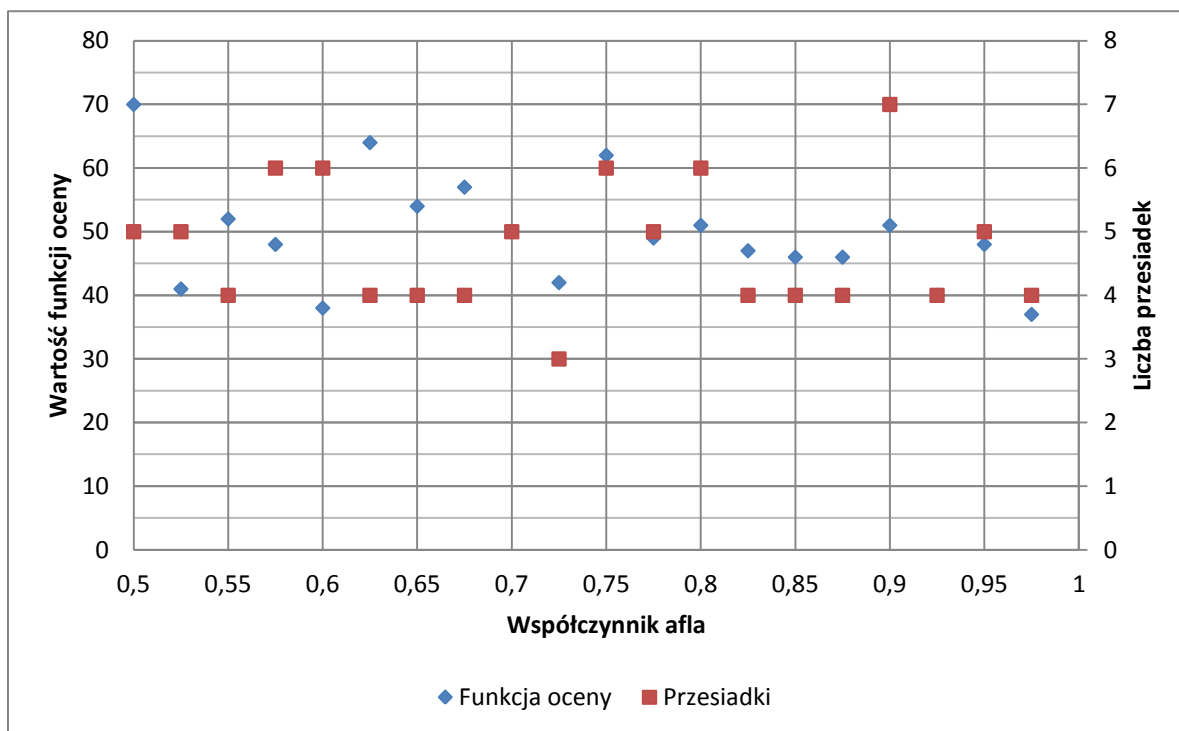
Rysunek 17 Wpływ temperatury początkowej na rozwiązania.

5.7.3 LICZBA ITERACJI



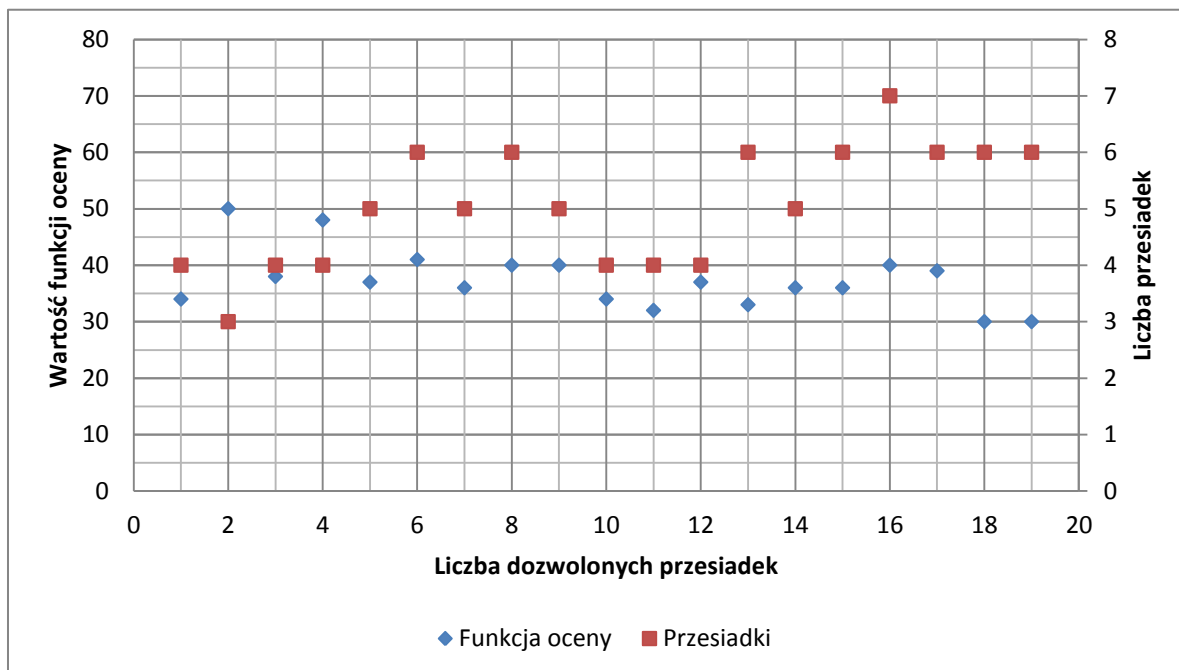
Rysunek 18 Wpływ współczynnika K na rozwiązania.

5.7.4 WSPÓŁCZYNNIK ALFA



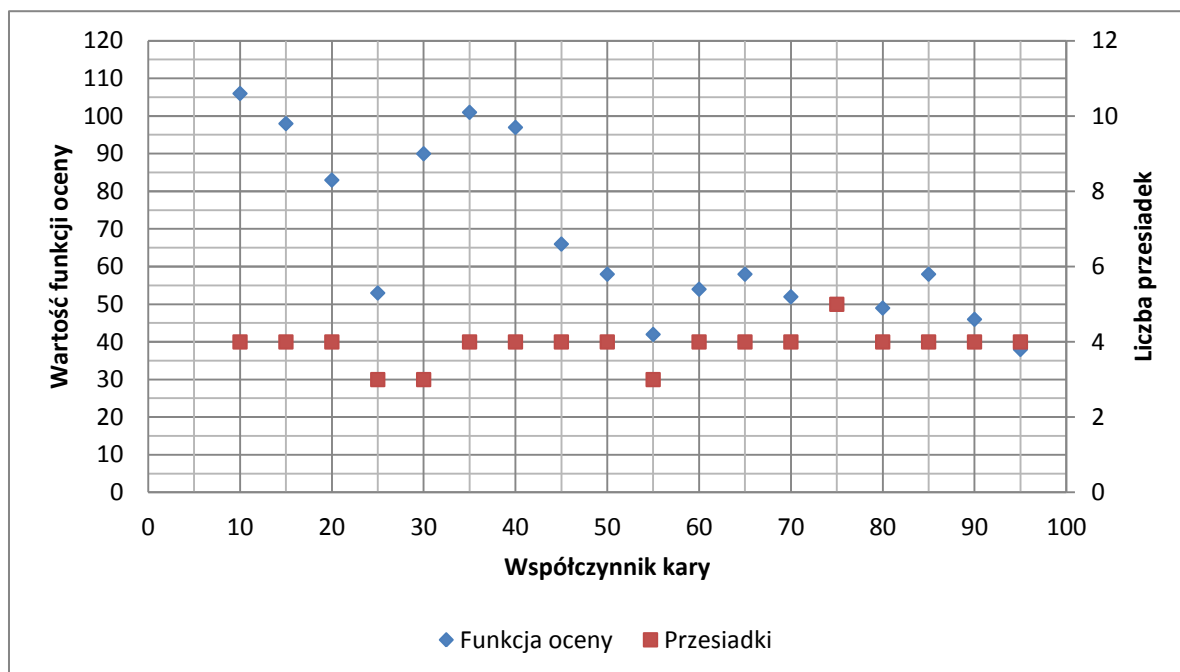
Rysunek 19 Wpływ współczynnika alfa na rozwiązania.

5.7.5 LICZBA DOZWOLONYCH PRZESIADEK



Rysunek 20 Wpływ liczby dozwolonych przesiadek na rozwiązania.

5.7.6 WSPÓŁCZYNNIK KARY



Rysunek 21 Wpływ współczynnika kary na rozwiązania.

5.8 WNIOSKI

Wszystkie z badanych parametrów mają wpływ na pracę algorytmu. Celem testowania jest obserwacja ich i dobranie takich wartości, które dla zadanego problemu dają najlepsze wyniki. Należy przy tym wyważyć wartość optymalizowanego zadania i czas wykonywania algorytmu. Jest to z pewnością zadanie trudne, lecz nasze ograniczone testy pozwoliły nam uzyskać wyniki, które mają pozytywny wpływ na pracę algorytmu.

Można przyjąć, że zbliżanie temperatury końcowej do 0 i zwiększanie wszystkich pozostałych przypadków wpływa pozytywnie na znalezione rozwiązania. Wiąże się to jednak z wydłużeniem pracy algorytmu do czasu minut, a w krytycznych sytuacjach nawet godzin. Dodatkowo, zwiększanie wartości współczynnika kary bez dokładniejszej analizy może doprowadzić do sytuacji, że większość rozwiązań nie pozostaje zbadana, ponieważ od razu są odrzucane jako nieoptymalne.

Zdecydowaliśmy, że temperatury początkowa i końcowa będą równe 100 i 1 stopni (temperatura końcowa powinna być zbliżona do 0, by rozkład prawdopodobieństw był optymalny, jednak ze względów technicznych, zakładając wartość 0 stopni algorytm nie kończyłby się nigdy lub pracował bardzo długo).

Współczynnik K oceniliśmy na około 50-100. Można uznać, że jest to wartość mała, ale specyfika problemu pozwala na taką wartość - otoczenie rozwiązania jest skończone, każdy przystanek posiada zazwyczaj kilka połączeń z innymi przystankami, nie trzeba więc analizować ogromnej ilości możliwych rozwiązań.

Współczynnik alfa pozwala na dobrą optymalizację już od wartości 0.99, więc taką przyjęliśmy za domyślną. Nawet niewielka zmiana tego współczynnika wpływa znacznie na czas trwania programu.

Ostatnim parametrem jest ilość dozwolonych przesiadek i w dużej mierze zależy on od preferencji użytkownika. Należy jednak pamiętać, że często nie ma bezpośredniego połączenia pomiędzy dwoma dowolnymi przystankami, dlatego rozsądnie jest dopuścić opcję 2-3 przesiadek w trakcie transportu.

5.9 DOPUSZCZALNY ROZMIAR PROBLEMU

Przedmiotem testów była również wydajność algorytmu w zależności od rozmiaru przeszukiwanej sieci. Ilość możliwych połączeń pomiędzy punktami rośnie szybciej niż liniowo wraz z liniowym wzrostem ilości przystanków. Stworzony przez nas algorytm dobrze radził sobie z sieciami o rozmiarze do kilku tysięcy przystanków. Jednak wraz z dalszym wzrostem wymiarów, badanie parametrów stawało się uciążliwe i czasochłonne. Z tego powodu oszacowaliśmy maksymalne wartości, dla których algorytm pracuje wydajnie na około 5000-10000 przystanków. Jest to liczba niespotykana zazwyczaj w rzeczywistości, jednak wciąż możliwa – w przypadku poszukiwania połączeń na obszarze kraju z dokładnością nie tylko do miejscowości, ale ulicy, z pewnością przekroczona zostałaby wartość 10000.

Ze względu na specyfikę problemu, dużo czasu procesu poświęćano było na obliczanie wartości funkcji oceny – konieczne było znalezienie w rozkładzie jazdy takiej wartości, która minimalizowała czas oczekiwania. Pomimo wykorzystania szybkich algorytmów przeszukiwania, okazało się, że zajmuje to około połowę czasu wykonania. Można skrócić ten czas kosztem zwiększenia danych przechowywanych w pamięci. Jest to problem optymalizacji złożoności czasowej i pamięciowej algorytmu i przekraczał zagadnienia badane w trakcie projektu.

Celem naszego projektu było realizacja algorytmu znajdującego optymalny czas oraz drogę przejazdu komunikacją miejską. W tym celu stworzyliśmy aplikację w języku c++ z wykorzystaniem biblioteki qt5.

Założeniem było zapoznanie się ze sposobem działania i implementacji algorytmów przybliżonych. W naszym przypadku implementowaliśmy algorytm symulowanego wyżarzania, gdyż tradycyjne algorytmy przeszukiwania grafu nie potrafią „obsługiwać” grafów o dynamicznych wagach.

W ramach testowania programu zapoznaliśmy się również z wpływem poszczególnych parametrów algorytmu takich jak temperatura, iteracje na spadek temperatury, współczynnik spadku temperatury.

Ważnymi elementami algorytmu jest sposób wyznaczania wartości rozwiązania, sprawdzenia jego dopuszczalności oraz wyznaczania otoczenia. Wartość rozwiązania była wyznaczana w kilku krokach: najpierw sprawdzenie możliwości dojazdu linią autobusową/tramwajową, następnie wybór tych linii po czym wyliczenie wartości na podstawie rozkładu jazdy. Rozwiązanie w otoczeniu dla danego rozwiązania było kreowane poprzez próbę zmiany fragmentu istniejącego rozwiązania na inny, tj poprzez próbę jechania inną trasą czy innymi połączeniami. Dopuszczalność rozwiązania została sprawdzana poprzez zliczanie przesiadek podczas wirtualnego przejazdu. Jeżeli przekraczała ona dozwoloną liczbę, za przesiadki ponad limit została nakładana kara. Próbowaliśmy także innego podejścia do funkcji kary: gdy rozwiązanie było niedopuszczalne kara została nakładana na wszystkie przesiadki. Działanie takie miało na celu dyskryminowanie niedopuszczalnych rozwiązań, ale niestety miało negatywny wpływ na przebieg algorytmu, szczególnie dla dużej wartości kary za pojedynczą przesiadkę.

W naszym programie przyjęliśmy pewne uproszczenia. Pierwszym z nich było to że nie bierzemy pod uwagę spóźnień, korków, preferencji podróżnych co w realnym przypadku ma duży wpływ na podróż. Drugim uproszczeniem było także przyjęcie, że podróżny zawsze zdąża na przesiadkę, niezależnie ile czasu ma pomiędzy nimi. Trzecim uproszczeniem jest brak możliwości komunikacji pieszej.