Akademia Górniczo - Hutnicza

Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej

Matematyczne metody wspomagania decyzji

Projektowanie tras biegowych w mieście zrealizowane z wykorzystanie algorytmu Tabu Search

Projekt zrealizowali:
Żaneta Błaszczuk
Jakub Porębski

Prowadzący:
Dr inż. Piotr Kadłuczka

12 stycznia 2015

Spis treści

1 Opis problemu

Zadnie projektowania trasy biegowej to zadanie poszukiwania trasy do biegu w mieście. Trasa taka powinna łączyć ze sobą 2 punkty na mapie i jednocześnie uwzględniać preferencje biegacza odnośnie tej trasy. W problemie przyjęliśmy, że kryteriami oceny trasy jest jej długość, procentowa ilość odcinków betonowych na trasie oraz różnica wysokości podczas biegu.

2 Parametry modelu

Mapa miasta jest reprezentowana jako graf, na którym wierzchołki to skrzyżowania dróg, a krawędzie są odcinkami trasy. Celem algorytmu jest znalezienie minimum globalnego funkcji celu wyrażonej ogólnym wzorem:

2.1 Funkcja celu

$$f(d,h,b) = w_1 \left| \overline{d} - \sum_{i=1}^{n} d_i \right| + w_2 \sum_{i=1}^{n} \left| \overline{h} - |h_i| \right| + w_3 \left| \overline{b} - \frac{\sum_{i=1}^{n} b_i}{n} \right| \tag{1}$$

$d_i > 0$ – długość i-tej krawędzi
h_i – różnica wysokości na i-tej krawędzi
b_i – rodzaj nawierzchni na i-tej krawędzi (0 – nawierzchnia dobra, 1 – nawierzchnia betonowa)
$\overline{d}$ – zadana wartość drogi
$\overline{h}$ – zadana wartość różnicy wysokości na każdej krawędzi
$\overline{b}$ – zadana ilość dróg o nawierzchni betonowej w %
w_1, w_2, w_3 – wagi dobierane przez użytkownika

2.2 Uproszczenia i uogólnienia

W trakcie implementacji algorytmu przyjęto założenie, że badany graf jest grafem planarnym, który dobrze opisuje układ dróg w mieście. Algorytm przybliża długości dróg do linii prostych, nie uwzględniając zakrętów na trasie. Równocześnie przyjęto, że atrakcyjność danej trasy zależy wyłącznie od różnicy wysokości na trasie oraz od ilości dróg asfaltowych na trasie.

2.3 Parametry i zależności

Głównymi parametrami trasy jest mapa krawędzi, po których może przebiegać trasa. Zawiera ona informacje o odległościach pomiędzy sąsiednimi wierzchołkami, wysokości w każdym węźle grafu oraz o rodzaju nawierzchni.

Do parametrów algorytmu zaliczamy również zadaną przez użytkownika odległość, preferencje własne dotyczące trasy oraz wagi każdego z parametrów.

2.4 Postać rozwiązania

Rozwiązaniem naszego problemu jest uporządkowana lista krawędzi grafu, która osiąga minimalną wartość funkcji celu.

2.5 Ograniczenia

Podczas implementacji algorytmu przyjęliśmy kilka niezbędnych ograniczeń

a) Biegacz nie przebiega dwukrotnie po tej samej drodze.

b) Maksymalna długość trasy biegu wynosi 1000000 jednostek (metrów). Jest to ograniczenie obliczeniowe algorytmu.

c) Maksymalna ilość iteracji algorytmu nie może przekroczyć 1000.

d)

3 Algorytm Tabu Search

Algorytm Tabuu Search, nazywany również algorytmem z zabronieniami jest algorytmem poszukiwania rozwiązania przybliżonego, który pozwala na ominięcie minimów lokalnych poprzez zastosowanie Listy Tabu (LT).

3.1 Schemat działania

3.2 Lista Tabu

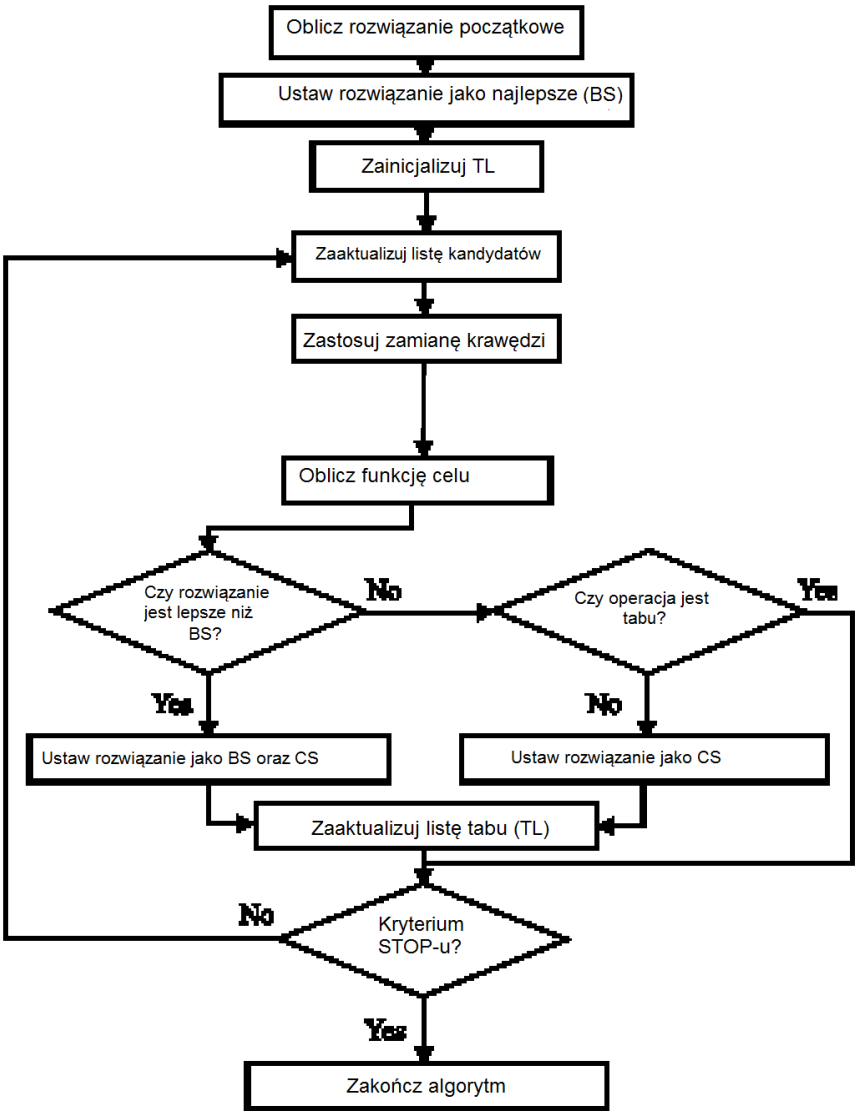
Lista zabronionych ruchów - długoterminowa (uwalniana tylko przez kryterium aspiracji) i któtkoterminowa (uwalniana zarówno przez kryterium aspiracji jak i przez czas).

3.3 Kryterium aspiracji

Pozwala przy spełnieniu pewnych warunków na odrzucenie zabronienia i wykonanie ruchu.

3.4 Warunku stopu

1. KZ1:
2. KZ2:
3. KZ3:
4. KZ4: Algorytm nie znajduje w sąsiedztwie żadnych rozwiązań polepszających funkcję celu, które nie zostały już wcześniej sprawdzone.
5. KZ5:



Rysunek 1: Schemat blokowy algorytmu Tabu Search

4 Implementacja algorytmu

Algorytm został zaprogramowany w środowisku QtCreator w języku C++. Środowisko to pozwala na szybkie obliczenia (wymagane przy dużym rozmiarze problemu) oraz na łatwą wizualizacje wygenerowanych wyników.

4.1 Postać danych

Dane w trakcie wykonywania algorytmu są przechowywane w kilku macierzach przyległości. Program w celu rozpoczęcia pracy potrzebuje 7 plików wejściowych z kolejnymi strukturami danych:

1. Współrzędne wierzchołków
    - rozszerzenie: .xy
    - Pierwsza linia: h w – wysokość i szerokość grafu

- Kolejne linie: wartości współrzędnych X Y każdego punktu

2. Krawędzie
   - rozszerzenie: .kr
   - Pierwsza linia: n – liczba krawędzi
   - Kolejne linie: X1, Y1, X2, Y2
3. Macierz przyległości
   Macierz kwadratowa przeskalowanych odległości między wierzchołkami
   - rozszerzenie: .txt
   - Pierwsza linia: h*w – liczba wierzchołków
   - Kolejne linie: wartości kolejnych komórek, 0 oznacza nieskończoność
4. Lista wysokości
   - rozszerzenie: .wys
   - Pierwsza linia: min max - minimalna i maksymalna wysokość
   - Kolejne linie: wysokości kolejnych wierzchołków
5. Macierz wysokości
   - rozszerzenie: .mwys
   - Pierwsza linia: h*w – liczba wierzchołków
   - Kolejne linie: jak w macierzy przyległości. Zawiera różnicę między dwoma łączonymi wierzchołkami
6. Lista betonowości
   - rozszerzenie: .bet
   - Pierwsza linia: h*w – liczba krawędzi
   - Kolejne linie: wartości, 0 oznacza brak betonu, 1 to beton (bardzo nie atrakcyjne)
7. Macierz wysokości
   - rozszerzenie: .mbet
   - Pierwsza linia: h*w – liczba wierzchołków
   - Kolejne linie: jak w macierzy przyległości. Symetryczna, nieujemna

4.2 Struktury danych

4.2.1 Klasa graph

Klasa graph opisuje w pełni wszystkie połączenia i wierzchołki w badanym grafie.

4.2.2 Klasa trasa

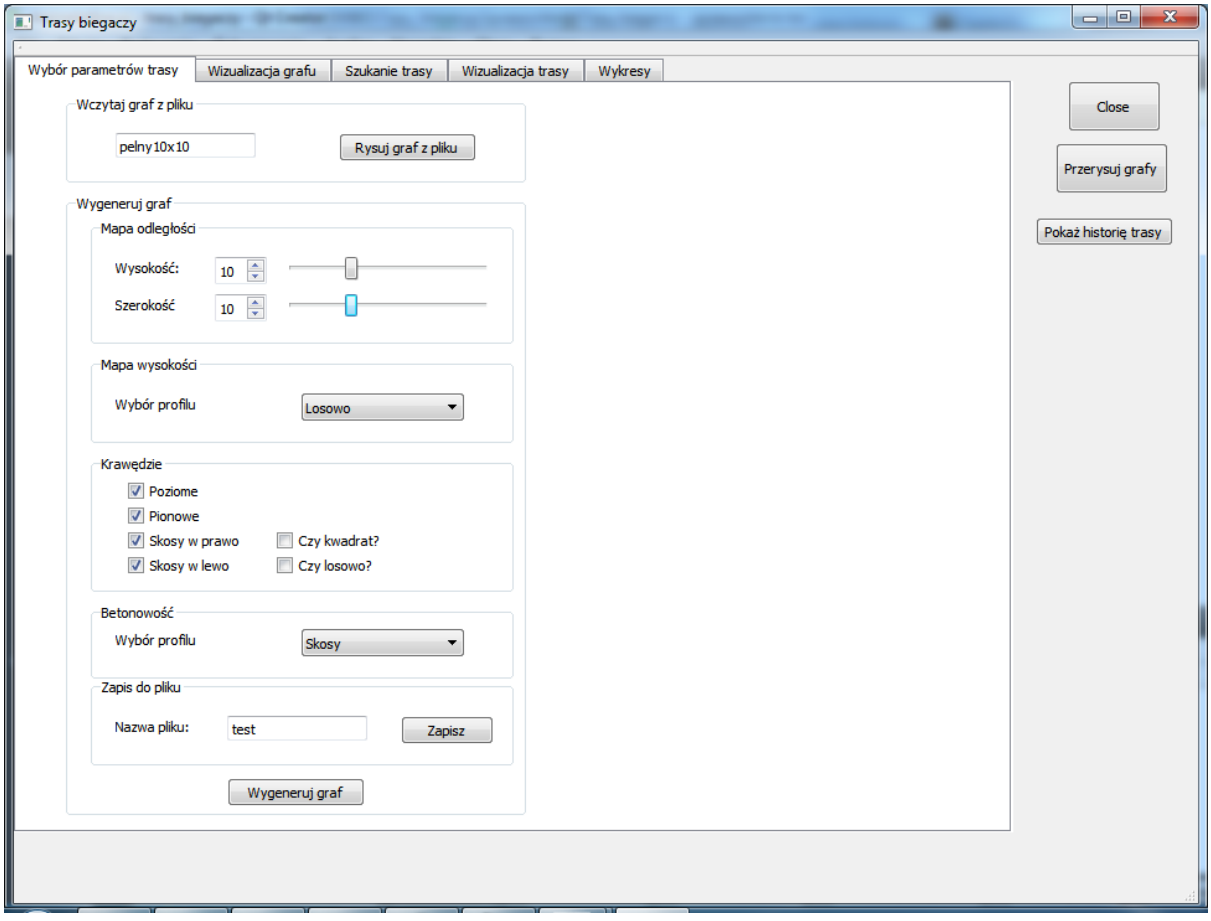
Klasa trasa wraz z zaimplementowanymi metodami realizuje w całości algorytm Taboo Search.

4.3 Interface użytkownika

Podczas pracy programu użytkownik ma pełną kontrolę nad wyborem parametrów algorytmu.

4.3.1 Generowanie grafu

Program pozwala użytkownikowi na wygenerowanie dowolnej mapy i automatyczny zapis jej do plików, jak i również wczytanie mapy z plików. Użytkownik nie musi samodzielnie wprowadzać danych dotyczących każdego wierzchołka, lecz może skorzystać z gotowych funkcji generujących mapy o zadanych właściwościach – na przykład miasto w dolinie, czy prostopadła siatka dróg betonowych.

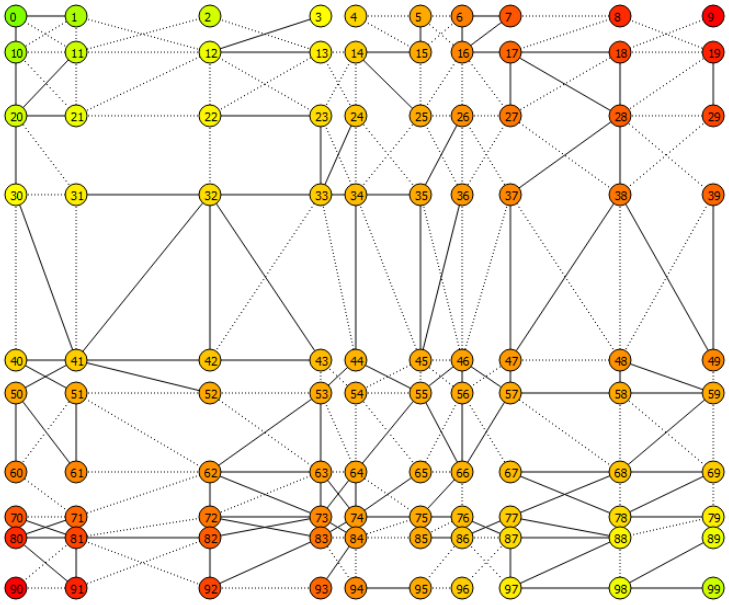


Rysunek 2: Zakładka pozwalająca na generowanie mapy

4.3.2 Wizualizacja grafu

Wygenerowany lub wczytany graf zostaje automatycznie wizualizowany w następnej zakładce. Drogi pokryte betonem zaznaczono linią przerywaną, a ścieżki bardziej atrak-

cyjne dla biegacza linią ciągłą. Kolory wierzchołków oznaczają względną wysokość danego wierzchołka. Wierzchołki zielone są niżej, a wierzchołki czerwone wyżej na mapie.



Rysunek 3: Wizualizacja przykładowego grafu

4.3.3 Szukanie trasy

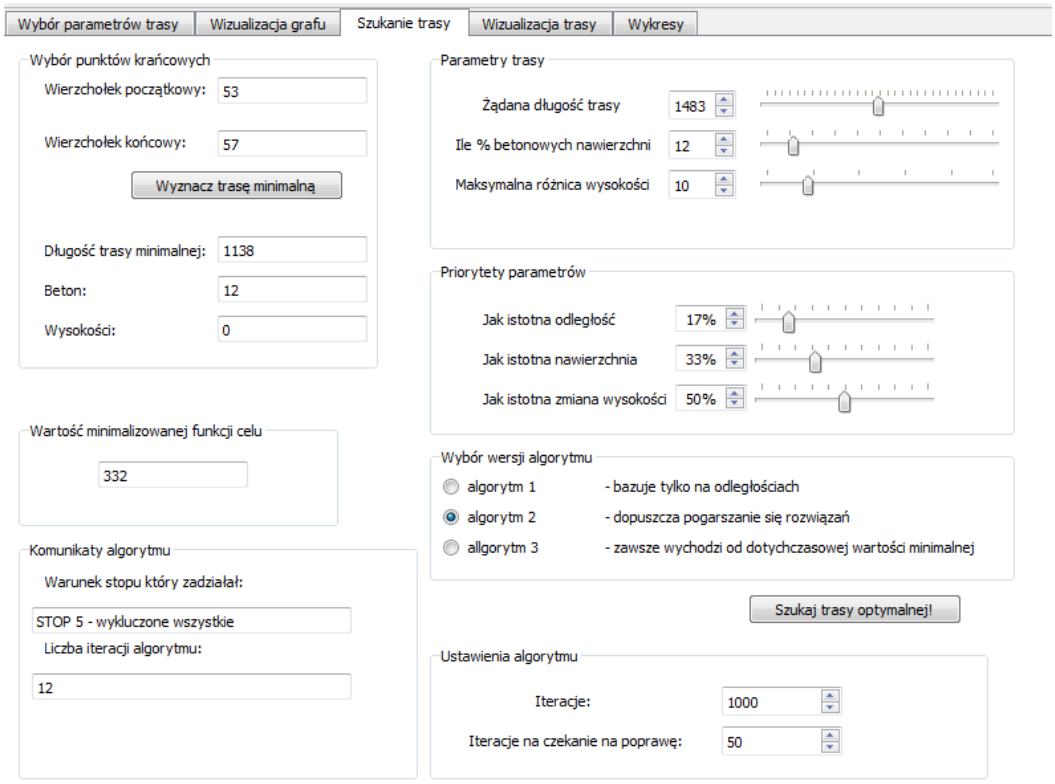
Kolejna zakładka pozwala na wybór opcji poszukiwanej trasy. Możemy tu wybrać jaka ma być długość trasy, ile na trasie ma być krawędzi betonowych oraz jak duże pochyłości są jeszcze akceptowalne. Dodatkowo możliwy jest wybór różnych wersji algorytmu. Następnie po kliknięciu na przycisk „Wygeneruj najlepszą trasę!" program generuje trasę najlepiej odpowiadającą wybranym parametrom.

4.3.4 Wizualizacja trasy

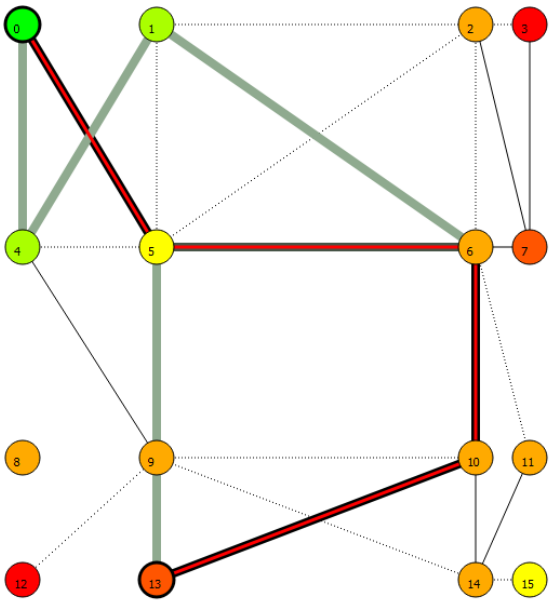
W zakładce „Wizualizacja trasy" możemy zobaczyć na czerwono zaznaczoną trasę optymalną obliczona przez algorytm. Po kliknięciu na przycisk „ Pokaż historię trasy" program zwizualizuje w odcieniach szarości krawędzie, po których algorytm „chodził". Im ciemniejsza krawędź tym częściej algorytm znajdował się na niej. Pozwala to na łatwe podejrzenie działania algorytmu.

4.3.5 Dane dodatkowe dotyczące algorytmu

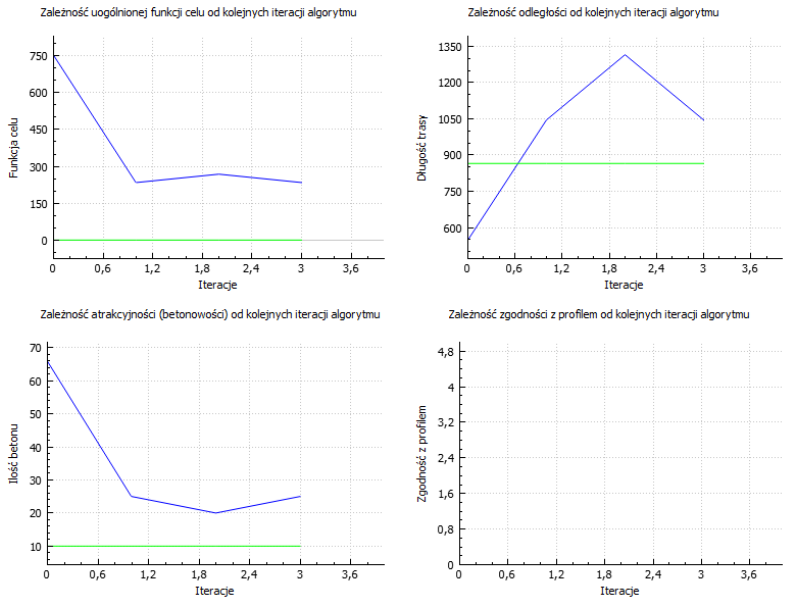
Ostatnia zakładka programu to „Wykresy". Po wykonaniu algorytmu program generuje wykresy zależności każdej ze składowych funkcji celu oraz samej funkcji celu od numeru iteracji.



Rysunek 4: Wybór parametrów trasy



Rysunek 5: Zakładka „Wizualizacja trasy” wraz z wyświetlonym przykładowym rozwiązaniem i historią przebiegu.

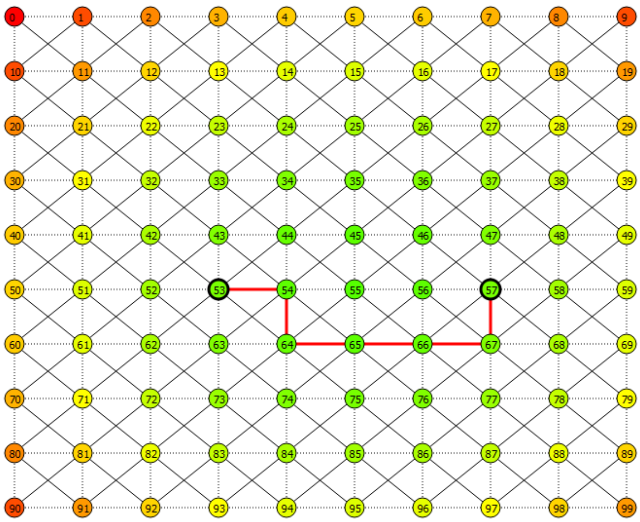


Rysunek 6: Przykładowe wykresy wygenerowane po realizacji algorytmu

5 Testowanie algorytmu

5.1 Test 1 – Badanie znajdywania rozwiązania algorytmu dla małych problemów obliczeniowych

Test 1 polega na badaniu zachowania algorytmu podczas wyszukiwania drogi o długości 392 między odcinkami 53 i 57 na grafie planarnym o rozmiarze 10x10 ze wszystkimi możliwymi połączeniami. Droga minimalna między wierzchołkami istnieje i jej przykładowa realizacja jest przedstawiona na rysunku 7. W celu utrudnienia zadania optymalizacji w



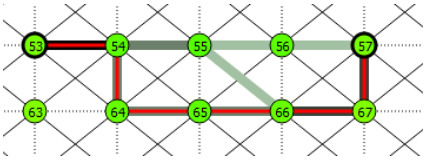
Rysunek 7: Przykładowa realizacja trasy optymalnej od długości 392

teście wykluczono wpływ betonu na wyszukiwanie, a różnica wysokości powinna być jak najmniejsza.

5.1.1 Wersja 1

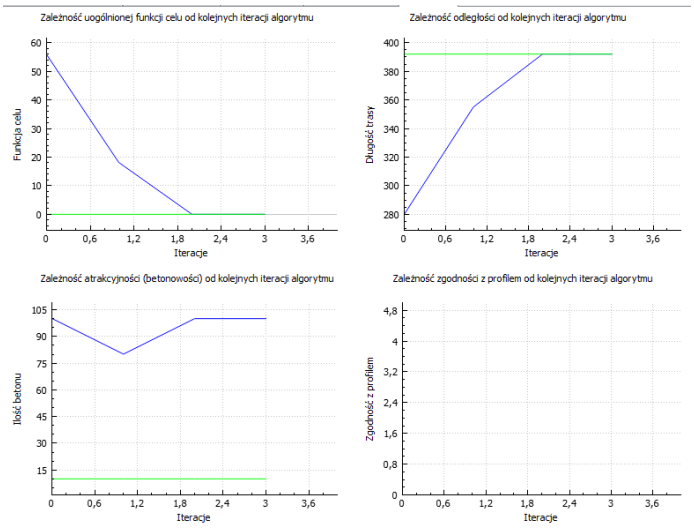
5.1.2 Wersja 2

Algorytm w wersji 2 posiada element losowości, lecz jednym z jego rozwiązań jest rozwiązanie przedstawione na rysunku 8. Algorytm osiągnął wynik po 7 iteracjach i zatrzymał



Rysunek 8: Wynik algorytmu w wersji 2 wraz z zaznaczonym przebiegiem algorytmu

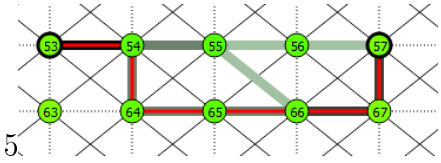
się poprzez nie znalezienie w sąsiedztwie lepszych rozwiązań. Wykresy obrazujące rozwiązanie algorytmu przedstawione są na rysunku 9.



Rysunek 9: Wynik algorytmu w wersji 2 wraz z zaznaczonym przebiegiem algorytmu

5.1.3 Wersja 3

Algorytm w wersji 3 posiada element losowości, lecz jednym z jego rozwiązań jest rozwiązanie przedstawione na rysunku 10. Algorytm osiągnął wynik po 7 iteracjach i zatrzymał



Rysunek 10: Wynik algorytmu w wersji 2 wraz z zaznaczonym przebiegiem algorytmu

się poprzez nie znalezienie w sąsiedztwie lepszych rozwiązań. Wykresy obrazujące rozwiązanie algorytmu przedstawione są na rysunku 11.

Rysunek 11: Wynik algorytmu w wersji 2 wraz z zaznaczonym przebiegiem algorytmu

5.2 Test 2 – badanie działania algorytmu dla nieistniejącego rozwiązania optymalnego

5.3 Test 3 – badanie działania algorytmu dla dużej instancji problemu

6 Analiza efektywności

7 Podsumowanie