Nazwa przedmiotu:	Badania Operacyjne 2
Nazwa projektu:	SA - sieć światłowodowa (dokumentacja)
Dzień zajęć:	Wtorek
Czas zajęć:	16:30 - 18:00
Zespół realizujący projekt:	• Tomasz Bednorz • Michał Spinczyk

Spis treści

1. Podział pracy

Etap	Tomasz Bednorz	Michał Spinczyk
Model zagadnienia	**65%** I. Funkcja celu II. Struktury danych	**35%** I. Zastosowane uproszczenia II. Postać rozwiązania
Opracowanie algorytmu	**40%** I. Rozwiązanie początkowe II. Sąsiedztwa III. Postać rozwiązania	**60%** I. Poprawność rozwiązania II. Schematy wyżarzania III. Całokształt algorytmu
Implementacja aplikacji	700/1300 linii kodu II. Główne metody: create_begining_solution(), update_device_neighbourhood(), update_node_neighbourhood(), generate_edges()	600/1300 linii kodu II. Główne metody: visualization(), check_network_corectness(), calculate_temperature(), run_alghoritm()
Testy	**45%** I. Testy: 5.1, 5.2, 5.6 II. Wykresy: 1, 2 III. Rysunki: 1, 2, 3, 4	**55%** I. Testy: 5.3, 5.4, 5.5 II. Wykresy: 3, 4, 5, 6, 7, 8, 9, 10 III. Rysunki: 5, 6, 7, 8, 9, 10, 11, 12
Dokumentacja	Rozdziały: 2.2, 4, 5.1, 5.2, 5.6, 6.3	Rozdziały: 2.1, 3, 5.3, 5.4, 5.5, 6.1, 6.2

2. Model zagadnienia

2.1 Opis słowny

Zagadnienie rozprowadzenia światłowodu na terenie wiejskim polega na iż mając określoną liczbę domów, utworzyć tak sieć światłowodową aby każdy z domów został podłączony do sieci oraz sieć zawierała odpowiednią liczbę urządzeń. Podczas tworzenia modelu, zostały przyjęte pewne uproszczenia:

- Możliwość poprowadzenia światłowodu z punktu A do punktu B tylko po linii prostej.
- Nie bierzmy pod uwagę czy mamy zgodę oraz możliwość na doprowadzenie światłowodu w danym miejscu (np. przez prywatną działkę albo drogę).
- Doprowadzenie światłowodu, budynki oraz słupy traktowane są jako punkty
- Tylko dwa typy światłowodu: napowietrzny oraz kanalizacyjny. Początkowo miał być jeszcze uniwersalny ale zrezygnowaliśmy z tej opcji.
- Zakładamy że większy światłowód, który jest doprowadzeniem, nie ma ograniczenia co do przewodów światłowodowych.

Informacjami niezbędnymi do rozwiązania problemu są współrzędne geograficzne każdego budynku, słupa oraz doprowadzenia światłowodu, koszty związane z materiałami potrzebnymi do budowy sieci (np. przewody, urządzenia) oraz montażu odpowiednich elementów (urządzenia, pociągnięcie kabli słupami czy kanalizacyjnie). Rozwiązaniem są wartości liczbowe oraz innych struktury danych. Pierwszym elementem rozwiązania jest koszt wybudowania sieci światłowodowej podawana w złotych [zł]. Następnie zwracane są krawędzie (połączenia światłowodowe) oraz użyte wierzchołki (budynki) w postaci słownika. Ostatnim elementem rozwiązania są użyte urządzenia.

2.2 Model matematyczny

Funkcja celu prezentuje się poniższym wzorem:

$$f(v) = \sum_{i=1}^{M} (d_i \cdot p_i) + m \cdot A$$

a) p - koszt i-tej krawędzi w złotówkach

$$p_i = \sum_{i=0}^{n} (c^u + d^w)$$

c - koszt przewodu światłowodowego za metr (u - rodzaj przewodu)
d - koszt montażu światłowodu za metr (w - typ przewodu)
n - ilość przewodów światłowodowych w danej krawędzi

b) d - odległość pomiędzy punktami a i b w terenie w metrach

$$d_i = \sqrt{(x_{a_i} - x_{b_i})^2 + (y_{a_i} - y_{b_i})^2}$$

x, y - współrzędne geograficzne

c) Zmienne pomocnicze:

A - stały koszt montażu światłowodu u klienta - jeden budynek (250 zł)
M - ilość krawędzi w sieci światłowodowej
m - ilość domów w sieci światłowodowej

3. Algorytm

3.1 Krótki opis

W celu rozwiązania zagadnienia został użyty algorytm symulowanego wyżarzania. Podstawową cechą algorytmu jest to możliwość przyjęcia gorszego rozwiązania jako aktualne. Dzieje się tak dlatego iż na początku działania algorytmu temperatura jest wysoka, dzięki czemu algorytm może często zmieniać konfigurację rozwiązania, raz wybierając lepsze a raz gorsze. Wraz ze wzrostem iteracji algorytmu temperatura algorytmu spada i wybierane częściej są lepsze rozwiązania. W końcowych iteracjach temperatura algorytmu jest na tyle niska, że prawdopodobieństwo wyboru gorszego rozwiązania jest bliskie zeru. W tej fazie algorytm działa podobnie do algorytmu iteracyjnego, czyli maksymalnie próbuje ulepszyć rozwiązanie.

Dla naszego problemu zmodyfikowaliśmy nieznacznie algorytm, mianowicie zapamiętujemy najlepsze rozwiązanie, przez co rozwiązanie które jest zwracane jest najlepsze (biorąc pod uwagę cały przebieg algorytmu).

3.2 Schemat implementacyjny algorytmu

```
i: = 0
R := R_startowe        (R – rozwiązanie aktualne)
R_best := R    (R_best – rozwiązanie najlepsze)
T := T_maksymalna
while i<i_max   (i – liczba iteracji)
        j := 0
        while j < liczba_iteracji_w_jednej_temperaturze
                R_tymczasowe := stworzenie_rozwiązania_w_oparciu_o_R    (różne sąsiedstwa)
        if check(R_tymczasowe):                (check – sprawdza poprawność rozwiązania)
                ∆ := F(R_tymczasowe) – F(R)          (F() – funkcja celu)
                if ∆ ≤ 0
                        R := R_tymczasowe
                        if F(R) < F(R_best)
                                R_best := R
                else
                        if  exp(-∆/T) > rand (0, 1)
                                R := R_tymczasowe
        i = i + 1
        T := oblicz_t(i)          (oblicz_t – oblicza temperaturę w oparciu o iteracje)
```

3.3 Opis elementów

I. Pierwszym elementem algorytmu jest stworzenie rozwiązania początkowego. Służy do tego metoda *create_beginning_solution* w klasie *SimulatedAnnealing*. Na początku dla każdego wierzchołka (czyli budynków i słupów) generowane są krawędzie dla każdej ćwiartki. Następnie do każdej krawędzi dodawane są przewody światłowodowe. Do rozwiązania dodawane są też urządzenia. Czynność jest powtarzana dopóki rozwiązanie nie będzie poprawne (każdy dom będzie podłączony do sieci oraz urządzenia będą poprawnie ulokowane).

II. Kolejnym etapem algorytmu jest uaktualnienie sąsiedztwa. Służą do tego trzy różne funkcje:

- *update_node_neighbourhood(NodeType.BUILDING),*
- *update_node_neighbourhood(NodeType.POLE),*
- *update_device_neighbourhood.*

Pierwsza metoda uaktualnia sąsiedztwo budynków, druga słupów natomiast trzecia urządzeń.

III. Trzecim elementem, który istotny jest z punktu poprawnego działania programu, jest sprawdzenie poprawności rozwiązania. Za to odpowiada funkcja *check_network_correctness*. Jest to zmodyfikowany algorytm Bellmana-Ford'a.

IV. Ostatnim elementem jest funkcja obliczającą następną temperaturę. Za obliczanie temperatury odpowiedzialna jest metoda *calculate_temperature*. Jako parametr przyjmuje ona liczbę wykonanych iteracji. W zależności od wybranego schematu chłodzenia, funkcja zwraca nam inną temperaturę.

3.4 Parametry algorytmu

Korzystając z klasy *SA_parameters* można ustawić odpowiednie wartości parametrów. Należą do nich:

- metody wyboru sąsiedztwa – do wyboru są 3 różne aktualizacje rozwiązania: aktualizacja połączeń budynków, aktualizacja połączeń słupów oraz aktualizacja urządzeń w sieci,
- alfa – liniowy współczynnik zmiany temperatury, wartość z zakresu od 0 do 1,
- temperatura początkowa – temperatura startowa, zmniejszana w kolejnych iteracjach,
- liczba iteracji – maksymalna liczba iteracji, w przypadku wykonanej implementacji jest to również liczba iteracji, które algorytm ma wykonać, ze względu na brak innego warunku stopu,
- liczba iteracji w jednej temperaturze – jest to liczba iteracji jaką algorytm ma wykonać z daną temperaturą,
- schemat wyżarzania – do wyboru są 4 różne schematy schładzania: liniowe, kwadratowe, wykładnicze oraz logarytmiczne, opcjonalnie można wybrać brak schładzania, wtedy algorytm wykonuje się dla stałej temperatury.

4. Aplikacja

4.1 Wymagania sprzętowe i oprogramowania

a) Wymagania sprzętowe
Platforma sprzętowa umożliwiająca instalację środowiska do rozwoju oprogramowania w języku Python oraz z dostępem do internetu (np. komputer klasy PC, laptop, SBC (single board computer), itp.)

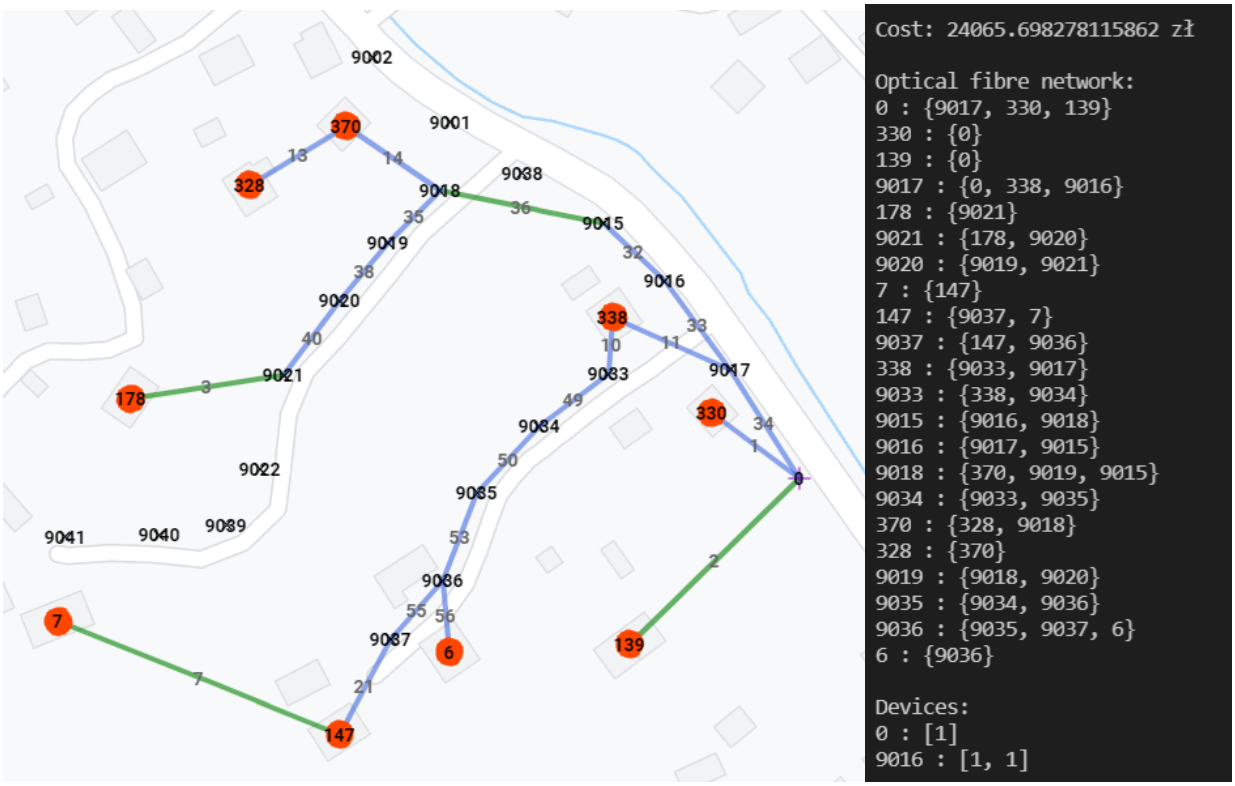
b) Wymagania odnośnie oprogramowania
Środowisko programistyczne umożliwiające rozwój oprogramowania w języku Python wraz z zainstalowanymi bibliotekami:

- Matplotlib
- Numpy
- Gmplot (Google Maps plot)

4.2 Format danych oraz wyników

a) Dane wejściowe
Sieć światłowodowa przechowywana jest za pomocą grafu. Wprowadzenie jego wierzchołków (słupów i budynków osobno) można zrealizować automatycznie za pomocą plików tekstowych, których kolejne linijki są w poniższej postaci:
<długość geograficzna><0x32><szerokość geograficzna><0x32><id>
0x32 - pusta przestrzeń w kodzie ASCII
id - identyfikator wierzchołka
Pozostałe dane takie jak: punkt startowy, sąsiedztwo, temperatura, ilość iteracji, parametr alpha oraz schemat chłodzenia ustawiane są bezpośrednio w skrypcie języka Python.

b) Dane wyjściowe
- Reprezentacja graficzna
Wizualizacja jest naniesiona na mapy Google. Zawiera wierzchołki i krawędzie sieci światłowodowej. Budynki zaznaczone są czerwonymi kropkami, wewnątrz których jest ich numer identyfikacyjny. Słupy zaznaczone są niewielkimi czarnymi krzyżykami na które naniesione są ich numery identyfikacyjne. Zielone krawędzie pomiędzy wierzchołkami oznaczają, że wewnątrz ich są przewody światłowodowe kanalizacyjne, a niebieskie, że napowietrzne.
- Reprezentacja numeryczna
Zawiera koszt sieci światłowodowej, listę sąsiedztwa oraz listę urządzeń.




```
Cost: 24065.698278115862 zł

Optical fibre network:
0 : {9017, 330, 139}
330 : {0}
139 : {0}
9017 : {0, 338, 9016}
178 : {9021}
9021 : {178, 9020}
9020 : {9019, 9021}
7 : {147}
147 : {9037, 7}
9037 : {147, 9036}
338 : {9033, 9017}
9033 : {338, 9034}
9015 : {9016, 9018}
9016 : {9017, 9015}
9018 : {370, 9019, 9015}
9034 : {9033, 9035}
370 : {328, 9018}
328 : {370}
9019 : {9018, 9020}
9035 : {9034, 9036}
9036 : {9035, 9037, 6}
6 : {9036}

Devices:
0 : [1]
9016 : [1, 1]
```

Rys 1. Reprezentacja graficzna / reprezentacja numeryczna

4.3 Funkcjonalność

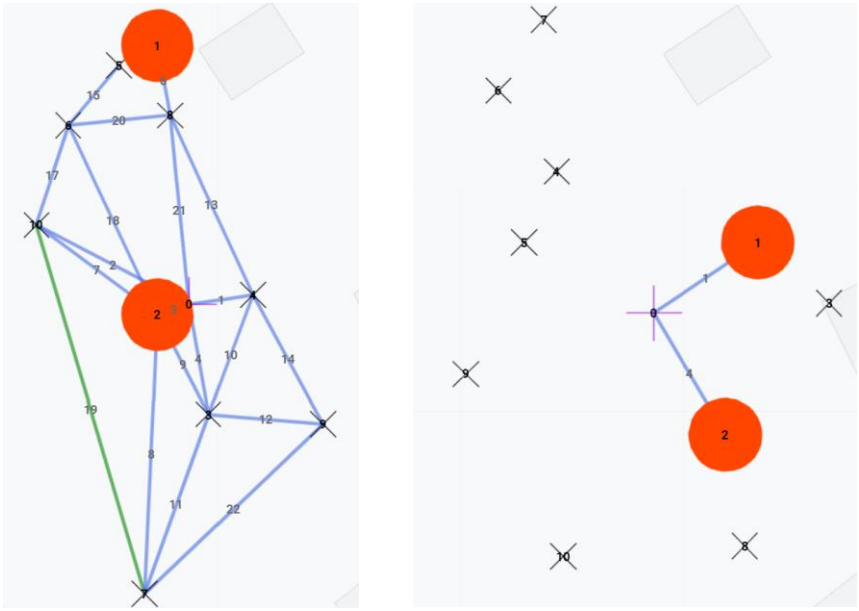
Aplikacja daje możliwość:

- wprowadzenia dowolnych wierzchołków (budynków i słupów) dla grafu, które znajdują się na kuli ziemskiej,
- wyboru sąsiedztwa,
- wyboru temperatury,
- wyboru ilości iteracji i liczba iteracji w jednej temperaturze,
- wyboru parametru alfa,
- wyboru schematu wyżarzania,
- otrzymania wyników zarówno w formie numerycznej jak i graficznej.

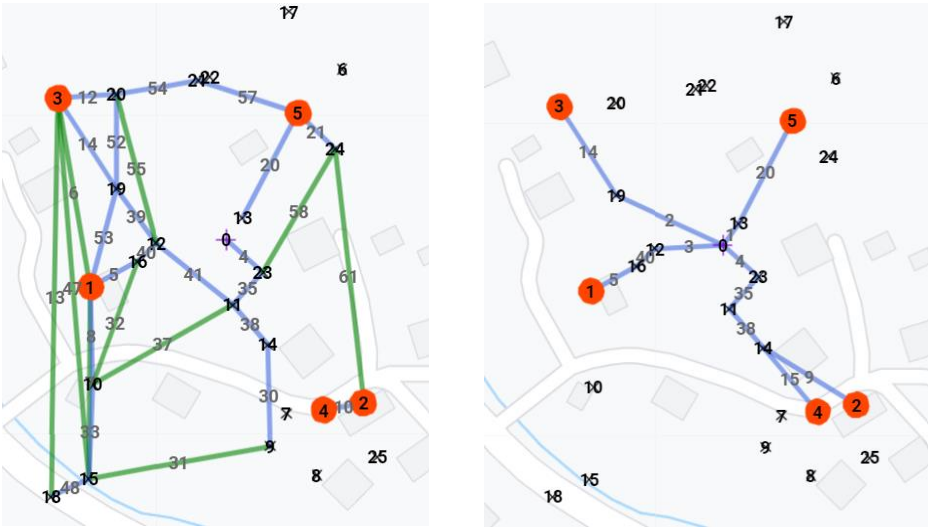
5. Testy

5.1 Przypadki “obojętne” statystycznie

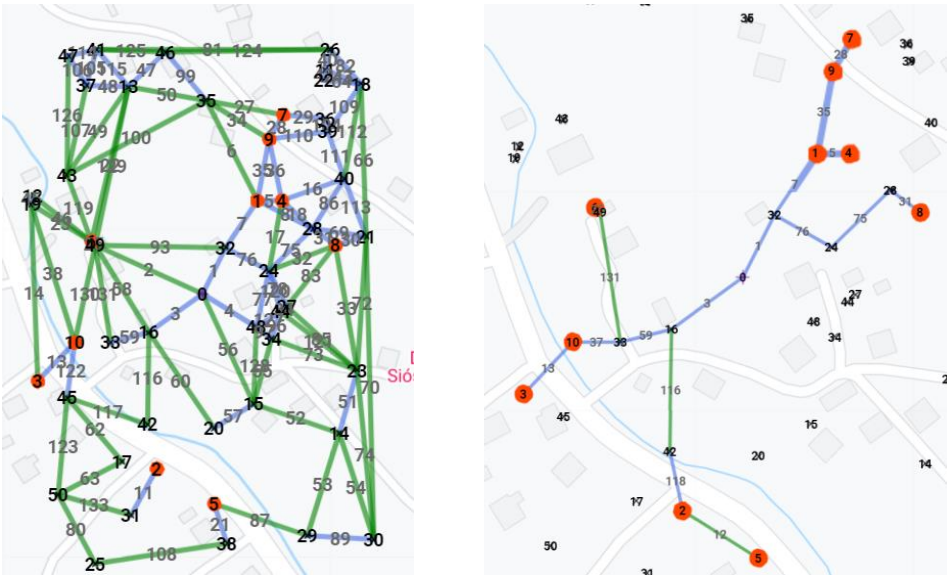
Autor: Tomasz Bednorz
Cel testu: Sprawdzenie poprawności działania algorytmu dla sieci światłowodowych wygenerowanych w sposób losowy.
Opis: Przy pomocy generatora liczb losowych zostaną wygenerowane 3 scenariusze testowe o rozmiarze 10, 25 oraz 50 wierzchołków. Następnie wizualnie zostanie sprawdzone czy otrzymane wyniki są bliskie rozwiązaniu optymalnemu.



Rys 2. Rozwiązanie początkowe / końcowe - scenariusz 1



Rys 3. Rozwiązanie początkowe / końcowe - scenariusz 2



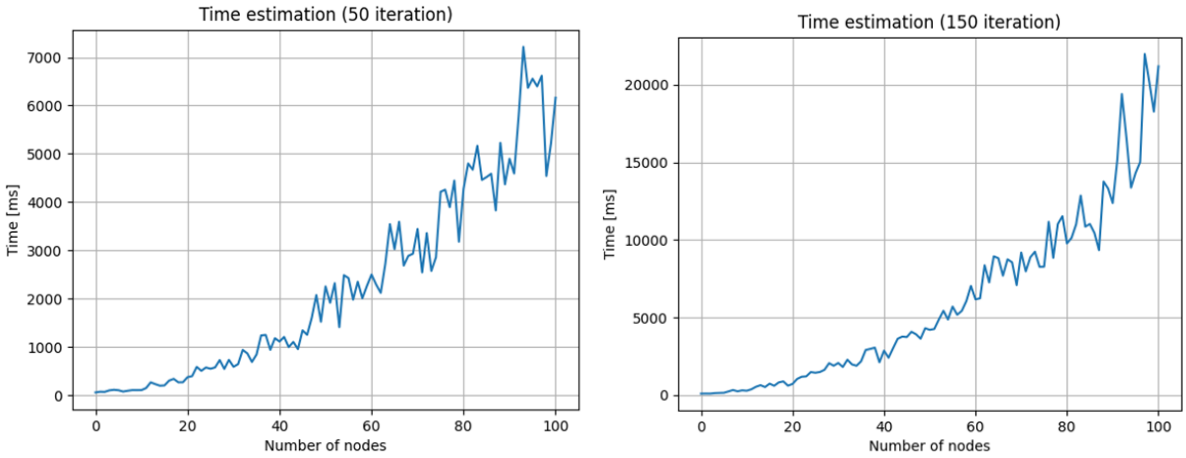
Rys 4. Rozwiązanie początkowe / końcowe - scenariusz 3

Numer scenariusza testowego	Ilość budynków	Ilość słupów	Ilość iteracji	Czas [s]	Początkowa wartość funkcji celu [zł]	Końcowa wartość funkcji celu [zł]
1	2	8	150	0.315	3608.88	1711.66
2	5	20	1000	9.27	34090.10	6255.00
3	10	40	3000	86.75	168505.33	19862.69

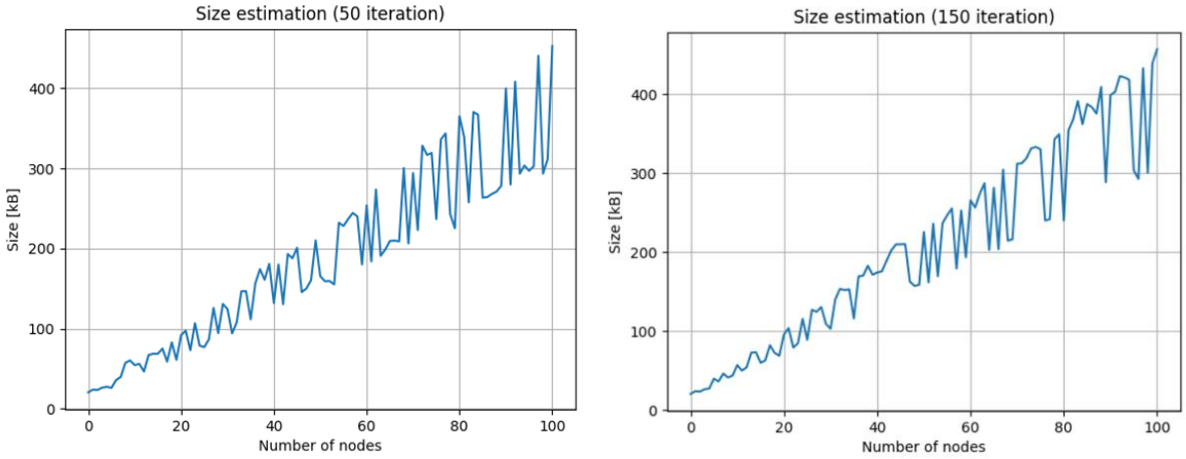
Interpretacja:
Patrząc na rysunki rozwiązań oraz tabelę można zauważyć, że algorytm działa poprawnie. Otrzymane rozwiązania są bliskie rozwiązaniom optymalnym. Wraz z wzrostem ilości wierzchołków wzrasta ilość iteracji potrzebnych do rozwiązania problemu oraz czas jego rozwiązania.

5.2 Czas i rozmiar

Autor: Tomasz Bednorz
Cel testu: Sprawdzenie wpływu zmiany rozmiaru problemu na czas trwania oraz rozmiar wykonywanego algorytmu.
Opis: W pętli generowane będą scenariusze losowe o rozmiarze od 1 do 100 (wierzchołków). Po wygenerowaniu struktury grafu będzie wykonywał się algorytm, a czas jego trwania będzie mierzony. Po wykonaniu algorytmu zostanie sprawdzony rozmiar przechowywanych danych. Wszystko to będzie sprawdzane w dwóch scenariuszach testowych, dla 50 i 150 iteracji trwania algorytmu.



Wykres 1. Czas trwania 50/150 iteracji algorytmu w zależności od rozmiaru sieci światłowodowej



Wykres 2. Rozmiar przechowywanych danych w zależności od rozmiaru sieci światłowodowej dla 50/150 iteracji algorytmu

Interpretacja:
Można zauważyć około 3-krotne wydłużenie działania algorytmu dla 150 iteracji w stosunku do 50 iteracji. Rozmiar sieci światłowodowej zachowywany jest na podobnym poziomie w obu sprawdzanych przypadkach. Na wykresie rozmiaru przechowywanych danych można zauważyć znaczące wahania podczas wzrostu liczby węzłów. Spowodowane jest to najprawdopodobniej rozwiązaniem początkowym, które ma znaczący wpływ na liczbę krawędzi, a co za tym idzie wielkość rozwiązania początkowego.

5.3 Wybór temperatury startowej

Autor: Michał Spinczyk
Cel testu: Sprawdzenie w jaki sposób temperatura startowa wpływa na wybory gorszego rozwiązania.
Opis: Dla jednego zestawu danych (budynki oraz słupy) zostanie wywołany algorytm dla różnych temperatur początkowych. Do tabeli zostaną wpisane wybory gorszych lub lepszych rozwiązań w procentach. Zgodnie z teorią, im niższa będzie temperatura tym mniej powinno być wybieranych gorszych rozwiązań. W ramach testu liczba iteracji zostanie ustawiona na 100 oraz schładzanie na liniowe.

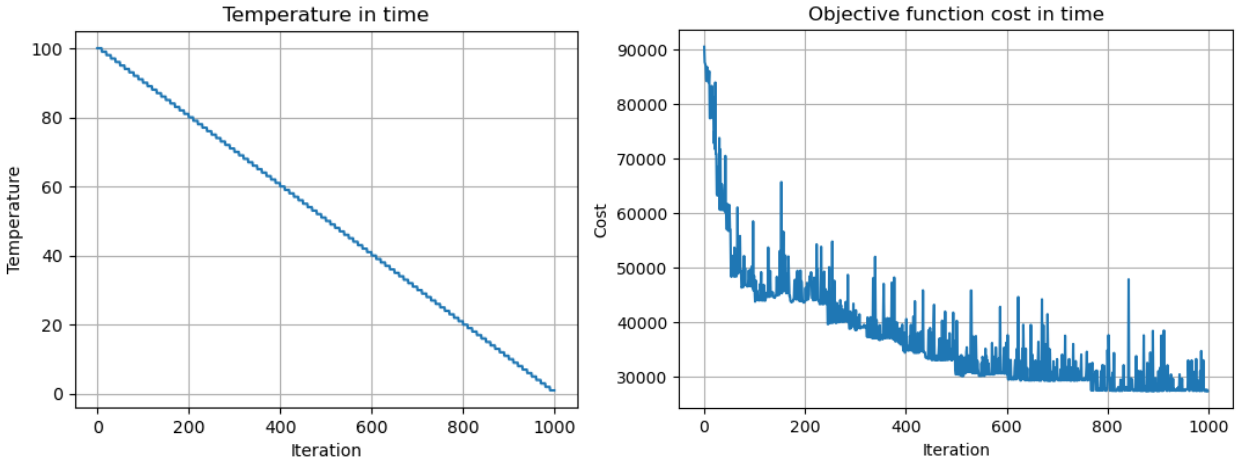
Temperatura startowa [°C]	Gorsze rozwiązanie nie zaakceptowane	Gorsze rozwiązanie (zaakceptowane)	Lepsze rozwiązanie (zaakceptowane)
10	67.0%	0.4%	32.6%
25	63.0%	1.6%	35.4%
50	54.7%	4.3%	41.0%
100	50.2%	9.9%	39.9%
250	42.5%	16.0%	41.5%
500	28.3%	21.6%	50.1%
1000	26.4%	26.1%	47.5%
2500	10.7%	34.7%	54.6%

Interpretacja:
Analizując wyniki wpisane do tabeli, można zauważyć iż wraz ze wzrostem temperatury coraz większa jest wartość wyboru gorszego rozwiązania (zaakceptowanego). Dla temperatury równej 10°C było to 0.4 %, następnie stopniowo ta wartość rosła aż do 2500°C było to 34.7%. Również wraz ze wzrostem temperatury startowej coraz częściej było wybierane lepsze rozwiązanie.

5.4 Wybór schematu wyżarzania

Autor: Michał Spinczyk
Cel testu: Sprawdzenie w jaki sposób schemat wyżarzania wpływa na rozwiązanie końcowe, przebieg funkcji celu oraz wybory rozwiązania.
Opis: Algorytm symulowanego wyżarzania zostanie wywołany dla każdego schematu wyżarzania: liniowego, kwadratowego, wykładniczego oraz logarytmicznego. Dla każdego schematu zostanie załączony wykres temperatury oraz funkcji celu. Liczba iteracji, liczba iteracji w jednej temperaturze, temperatura oraz alfa dla każdego schematu wyżarzania będzie taka sama (odpowiednio – 100, 10, 100, 0.98).

a) schładzanie liniowe

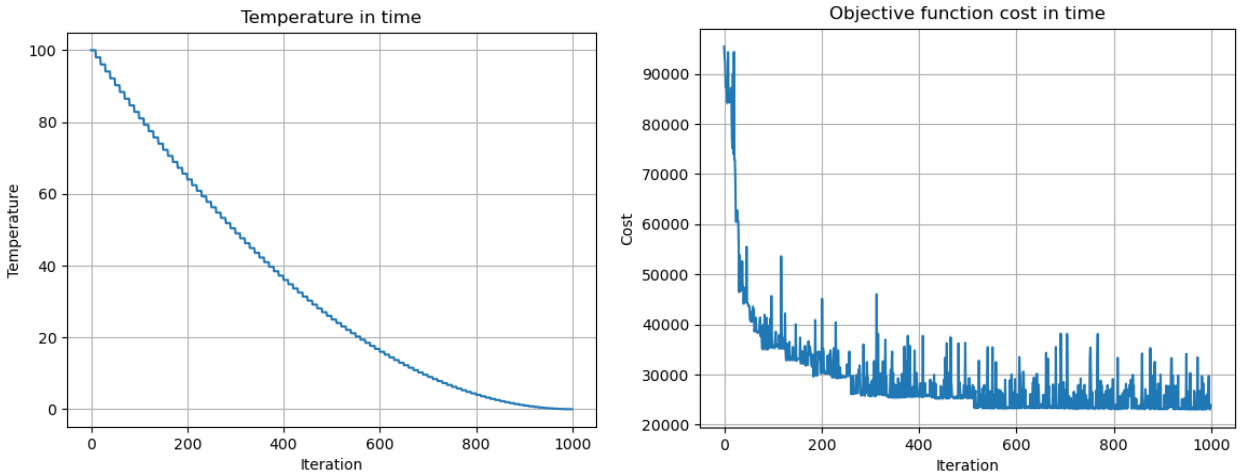


Wykres 3. Zmiana temperatury oraz wartości funkcji celu w czasie (schładzanie liniowe)

```
Objective function cost: 27441.61998102231 zł
Worse cost (not accepted): 467  46.7%
Worse cost (accepted): 117  11.7%
Better cost: 416  41.6%
```

Rys 5. Najlepsza wartość funkcji celu i statystyki algorytmu (schładzanie liniowe)

b) schładzanie kwadratowe

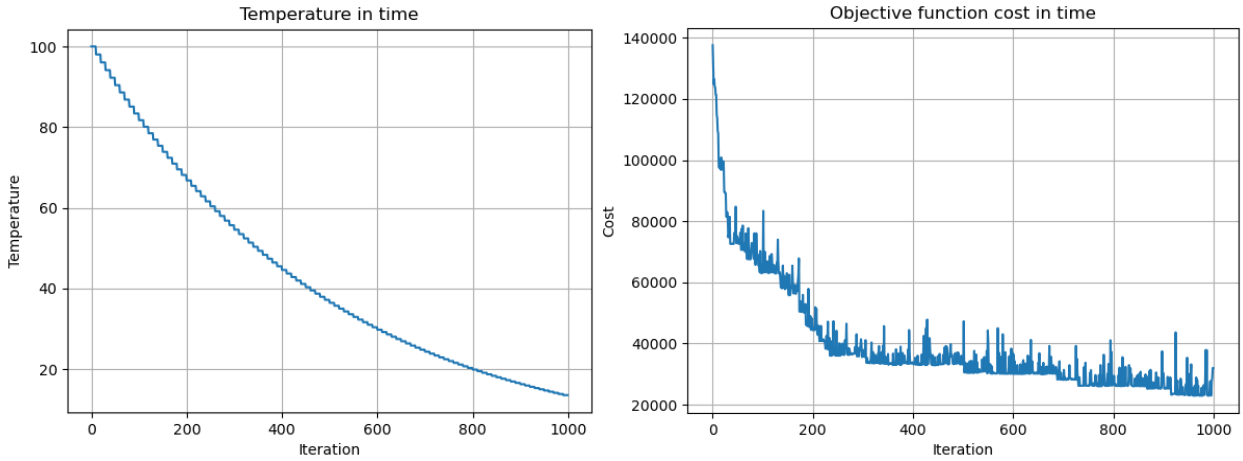


Wykres 4. Zmiana temperatury oraz wartości funkcji celu w czasie (schładzanie kwadratowe)

```
Objective function cost: 23015.203021081386 zł
Worse cost (not accepted): 543  54.3%
Worse cost (accepted): 74  7.4%
Better cost: 383  38.3%
```

Rys 6. Najlepsza wartość funkcji celu i statystyki algorytmu (schładzanie kwadratowe)

c) schładzanie wykładnicze

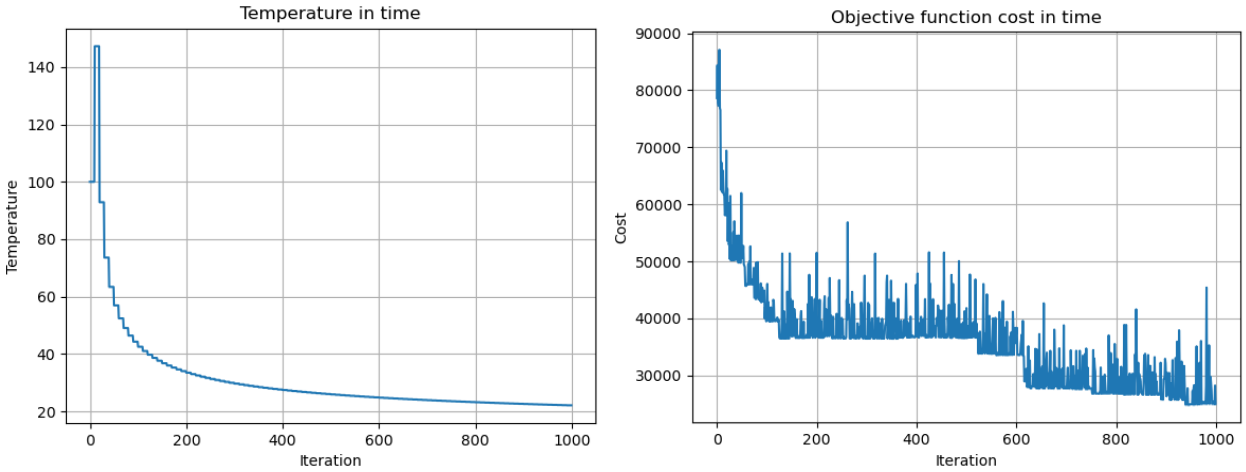


Wykres 5. Zmiana temperatury oraz wartości funkcji celu w czasie (schładzanie wykładnicze)

```
Objective function cost: 22943.723926230916 zł
Worse cost (not accepted): 490  49.0%
Worse cost (accepted): 83  8.3%
Better cost: 427  42.7%
```

Rys 7. Najlepsza wartość funkcji celu i statystyki algorytmu (schładzanie wykładnicze)

d) schładzanie logarytmiczne



Wykres 6. Zmiana temperatury oraz wartości funkcji celu (schładzanie logarytmiczne)

```
Objective function cost: 24851.818172239673 zł
Worse cost (not accepted): 571  57.1%
Worse cost (accepted): 58  5.8%
Better cost: 371  37.1%
```

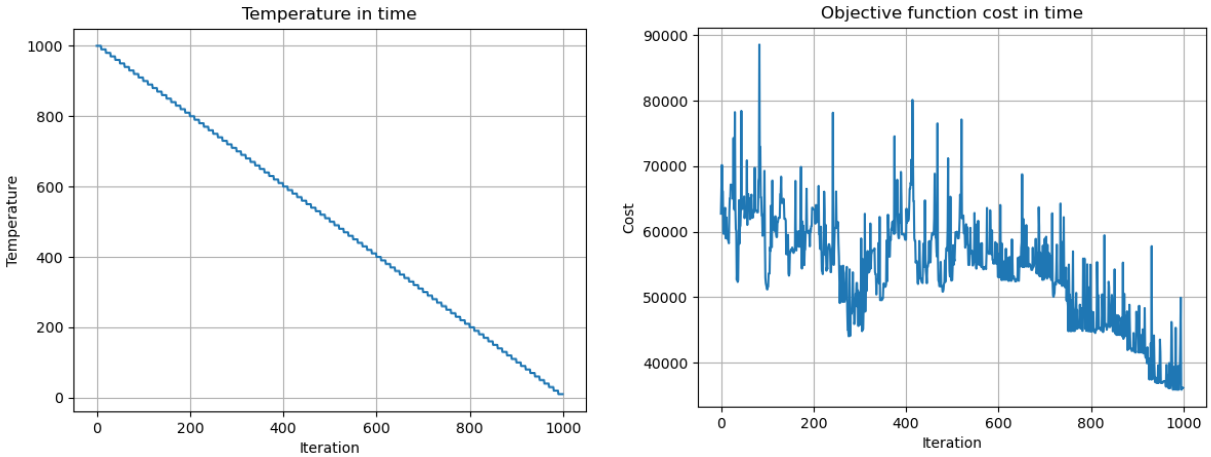
Rys 8. Najlepsza wartość funkcji celu i statystyki algorytmu (schładzanie logarytmiczne)

Interpretacja:
Analizując otrzymane wartości numeryczne oraz dane przedstawione na wykresie można zauważyć iż im funkcja szybciej malej do zera tym mniej jest akceptowanych gorszych rozwiązań. W przypadku schładzania liniowego, które najwolniej maleje do zera było to około 12%, dla kwadratowego oraz wykładniczego około 8%, natomiast w przypadku logarytmicznego (najszybciej maleje do zera) było to już tylko niecałe 6 %. Algorytm najlepiej sobie poradził w przypadku schładzania kwadratowego oraz wykładniczego.

5.5 Wydłużanie serii wraz ze zmniejszaniem temperatury

Autor: Michał Spinczyk
Cel testu: Sprawdzenie w jaki sposób wydłużanie serii ze zmniejszaniem temperatury wpływa na działanie algorytmu.
Opis: W każdym przypadku testowym zostanie zastosowany ten sam rozkład słupów i budynków. Optymalizacji będą podlegać budynki, słupy oraz urządzenia. Afla, liczba iteracji w jednej temperaturze oraz schemat wyżarzania będą jednolite dla każdego przypadku, natomiast zmiennymi będą temperatura początkowa oraz ilość iteracji.

a) liczba iteracji: 100, temperatura: 1000

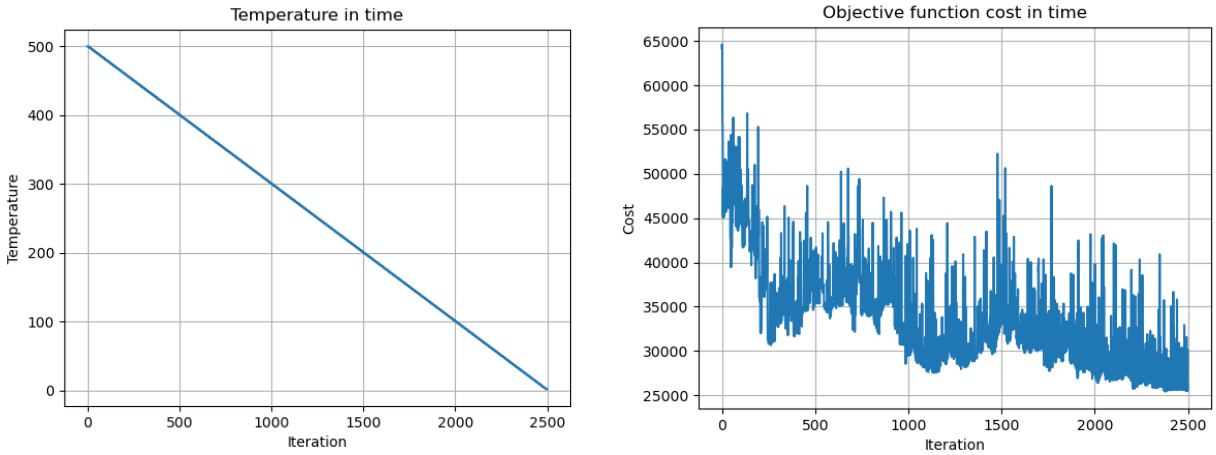


Wykres 7. Zmiana temperatury oraz wartości funkcji celu w czasie (liczba iteracji: 100, temperatura: 1000)

```
Objective function cost: 41571.53091957814 zł
Worse cost (not accepted): 217  21.7%
Worse cost (accepted): 284  28.4%
Better cost: 499  49.9%
```

Rys 9. Najlepsza wartość funkcji celu i statystyki algorytmu (liczba iteracji: 100, temperatura: 1000)

b) liczba iteracji: 250, temperatura: 500

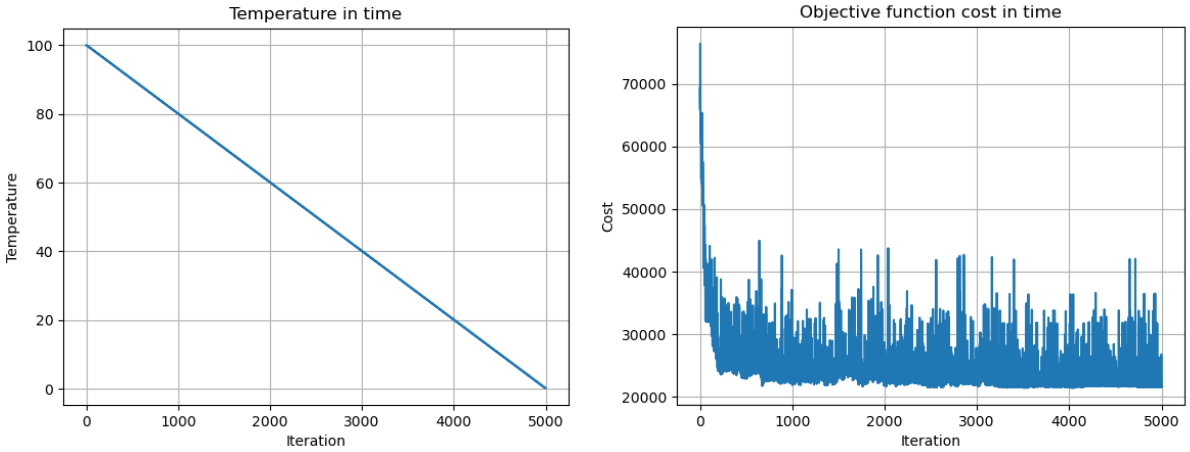


Wykres 8. Zmiana temperatury oraz wartości funkcji celu w (liczba iteracji: 250, temperatura: 500)

```
Objective function cost: 25448.95395325616 zł
Worse cost (not accepted): 805  32.2%
Worse cost (accepted): 535  21.4%
Better cost: 1160  46.4%
```

Rys 10. Najlepsza wartość funkcji celu i statystyki algorytmu (liczba iteracji: 250, temperatura: 500)

c) liczba iteracji: 500, temperatura: 100

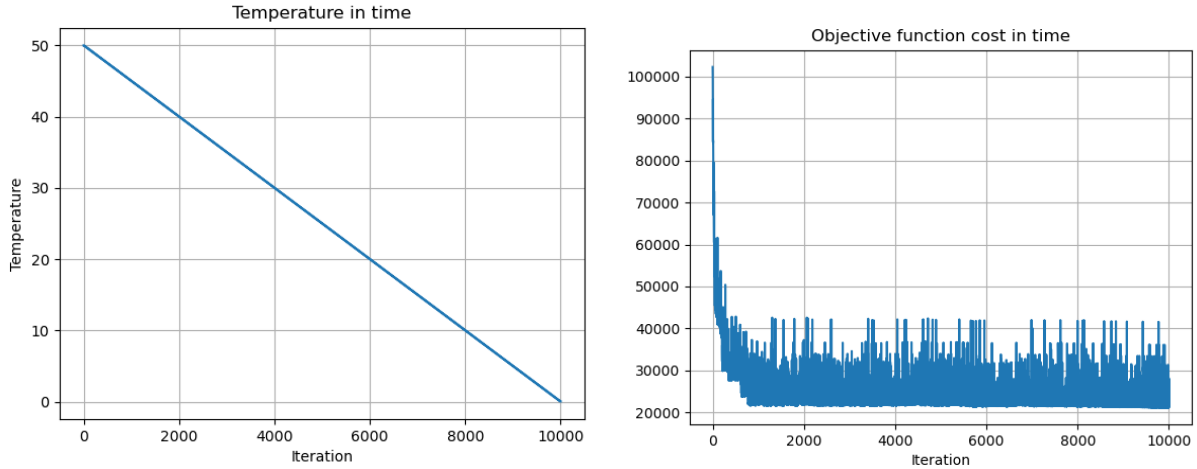


Wykres 9. Zmiana temperatury oraz wartości funkcji celu w czasie (liczba iteracji: 500, temperatura: 100)

```
Objective function cost: 21441.10412636238 zł
Worse cost (not accepted): 2248  45.0%
Worse cost (accepted): 633  12.7%
Better cost: 2119  42.4%
```

Rys 11. Najlepsza wartość funkcji celu i statystyki algorytmu (liczba iteracji: 500, temperatura: 100)

d) liczba iteracji: 1000, temperatura: 50



Wykres 10. Zmiana temperatury oraz wartości funkcji celu w czasie (liczba iteracji: 1000, temperatura: 50)

```
Objective function cost: 21208.8824600808 zł
Worse cost (not accepted): 5662  56.6%
Worse cost (accepted): 599  5.99%
Better cost: 3739  37.4%
```

Rys 12. Najlepsza wartość funkcji celu i statystyki algorytmu (liczba iteracji: 1000, temperatura: 50)

Interpretacja:
Na wykresach widać iż im mniejsza jest temperatura początkowa tym mniej jest wybieranych gorszych rozwiązań. Widać również iż wraz ze wzrostem iteracji oraz spadkiem temperatury maleje wartość funkcji celu. Dla 3 i 4 przypadku ta różnica jest nieznaczna.

5.6 Nakład obliczeniowy

Autor: Tomasz Bednorz
Cel testu: Sprawdzenie złożoności obliczeniowej oraz czasu trwania poszczególnych elementów algorytmu.
Opis: Początkowo zostanie sprawdzona teoretyczna złożoność obliczeniowa na podstawie ilości pętli wewnątrz poszczególnych funkcji odpowiadających za konkretne elementy algorytmu. Następnie zostanie sprawdzony czas trwania ich 1000-krotnego wywołania w pętli dla problemów o dwóch różnych rozmiarach:
- mały rozmiar: 9 budynków, 19 słupów
- duży rozmiar: 43 budynki, 57 słupów
Strukturę sieci (budynki i słupy) dla poszczególnych rozmiarów można znaleźć w folderze Tests pod nazwami: buildings_S.txt, poles_S.txt, buildings_L.txt oraz poles_L.txt.

Algorytm składa się z poniższych elementów:
a) dodania wierzchołków do grafu,
b) stworzenie rozwiązania początkowego,
c) sprawdzenie poprawności rozwiązania,
d) aktualizacja sąsiedztwa - budynki,
e) aktualizacja sąsiedztwa - słupy,
f) aktualizacja sąsiedztwa - urządzenia,
g) wizualizacja rozwiązania,
h) pozostałe mniejsze elementy (generowanie liczb losowych, dodawanie poprzednich iteracji do danych historycznych, obliczanie nowej temperatury, itd.).

Element	Teoretyczna złożoność obliczeniowa	Czas 1000 krotnego wywołania [s] - mały rozmiar	Czas 1000 krotnego wywołania [s] - duży rozmiar
a)	O(n)	0.26	1.05
b)	O(n^3)	10.62	119.88
c)	O(n^3)	3.21	55.54
d)	O(n^2)	2.34	7.17
e)	O(n^2)	2.47	7.40
f)	O(n^2)	2.48	7.80
g)	O(n^2)	23.49	89.07

Interpretacja:
Można zauważyć, że teoretyczna złożoność obliczeniowa ma odzwierciedlenie w czasie 1000 krotnego wywołania danego elementu algorytmu. W zależności od sprawdzanego elementu w dużym problemie, którego liczba wierzchołków jest około trzy i pół razy większa od małego problemu można zauważyć od kilku do kilkunastu krotnego wzrostu czasu trwania danego elementu.

6. Podsumowanie

6.1 Wnioski

Analizując wyniki otrzymane w teście 5.1 można stwierdzić iż zaimplementowany algorytm działa poprawnie. Istotnym zależnością stwierdzoną również w tym teście jest to iż wraz ze wzrostem ilości wierzchołków wzrasta ilość iteracji potrzebnych do rozwiązania problemu oraz czas jego rozwiązania. Testy 5.3, 5.4 oraz 5.5 potwierdzają podstawową własność algorytmu jaką jest możliwość wyboru gorszego rozwiązania. Dane przedstawione w tabelach oraz na wykresach pozwalają stwierdzić, iż im wyższa jest temperatura tym większa jest szansa na przyjęcie gorszego rozwiązania. Na wykresach wartości funkcji celu w czasie można zauważyć iż największe spadki wartości funkcji celu są na początku natomiast z czasem wartości się stabilizują.

6.2 Stwierdzone problemy

Podczas testowania działania programu oraz algorytmu zostały stwierdzone dwie podstawowe wady. Pierwsza z nich związana jest z funkcją tworzącą rozwiązanie początkowe (*create_beginning_solution*). Funkcja ta nie zawsze zwróci rozwiązanie poprawne przez co jest wywoływana dopóki nie zwróci poprawnego rozwiązania. Niepotrzebnie również tworzy niektóre krawędzie. Drugim problemem jest złożoność obliczeniowa związana z funkcja sprawdzająca poprawność rozwiązania - *check_network_correctness*. Jest to zmodyfikowany algorytm Bellmana-Ford'a, przez co za każdym razem gdy zmieni się rozwiązanie, algorytm jest zmuszony do wywołania funkcji o złożoności O(n^3). Im więcej będzie budynków oraz słupów tym algorytm będzie dłużej wykonywał obliczania.

6.3 Kierunki dalszego rozwoju

Badane zagadnienie daje wiele możliwości dalszego rozwoju. Zarówno poprawy istniejącej implementacji jak i rozszerzenie jej o nowe funkcjonalności.
Przykładem poprawy istniejącej implementacji może być poprawa istniejących funkcji w celu zmniejszenia złożoności obliczeniowej. Dałoby to możliwość rozwiązywania o wiele bardziej skomplikowanych problemów w krótszym czasie. Przykładem takich funkcji mogą być funkcja tworząca rozwiązanie początkowe oraz funkcja sprawdzająca poprawność połączeń. Funkcja tworząca początkowe rozwiązanie robi to w sposób losowy co powoduje, że w większości przypadków rozwiązanie początkowe w porównaniu do rozwiązania końcowego zawiera kilku lub kilkunastokrotnie większą ilość stworzonych krawędzi. Dlatego następnie należy wykonać bardzo dużą liczbę iteracji w celu poprawy rozwiązania początkowego.
Przykładem rozszerzenia funkcjonalności mogą być:

- dodatkowe schematy wyżarzania,
- dodatkowe metody optymalizacji (sąsiedztwa),
- możliwość prowadzenia światłowodu nie tylko w linii prostej,
- wprowadzenie większej liczby ograniczeń (np. zakaz przeprowadzania światłowodu naziemnego przez rzeki, a kanalizacyjnego w miejscach zakazanych).