<table>
<tr><td rowspan="3"> 
Politechnika
Wrocławska</td><td colspan="2">Projektowanie efektywnych algorytmów – projekt nr 2</td></tr>
<tr><td>Autor:

Łukasz Gawron,
nr indeksu 264475</td><td>Termin oddania projektu:

11.12.2023</td></tr>
<tr><td>Termin zajęć:

Poniedziałek godz. 15:15</td><td>Prowadzący:

Dr. Inż. Jarosław Mierzwa</td></tr>
</table>

Spis treści

1. Wstęp

Celem projektu była implementacja oraz zbadanie efektywności algorytmów heurystycznych (Symulowanego wyżarzania oraz Przeszukiwania tabu) dla problemu Komiwojażera – zagadnieniu z teorii grafów, które polega na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Badany wariant problemu to ATSP (asymetryczny problem komiwojażera) – co oznacza, że waga ścieżki z wierzchołka A do wierzchołka B jest różna od tej z wierzchołka B do wierzchołka A. Główną trudnością problemu jest duża liczba danych do analizy - problem Komiwojażera jest problemem klasy NP (o wielomianowym czasie rozwiązywania).

Algorytmy heurystyczne stanowią implementację metod znajdowania rozwiązań problemów, takich jak problem Komiwojażera, jednak metoda ta nie ma gwarancji znalezienia rozwiązania optymalnego. Pozwalają one jednak na znalezienie rozwiązań przybliżonych (w szczególnych przypadkach także tych dokładnych), dla problemów, gdzie znane algorytmy dokładne nie pozwalają na wystarczająco szybkie znalezienie rozwiązań dokładnych.

2. Zaimplementowane algorytmy

2.1 Symulowane wyżarzanie (ang. Simulated annealing)

2.1.1 Opis algorytmu

Algorytm SA opiera się na procesie wyżarzania stosowanym w metalurgii. Proces poszukiwania rozpoczyna się od stanu o wysokiej energii (gdy prawdopodobieństwo akceptacji gorszego rozwiązania jest wysokie – stanowi ono mechanizm ucieczki z minimum lokalnego) i stopniowo obniża temperaturę, do momentu, gdzie prawdopodobieństwo akceptacji gorszych rozwiązań będzie niskie (zakładamy stabilizację i staramy się zbliżyć naj najbardziej minimum globalnemu).

Kryterium akceptacji SA wywodzi się z mechaniki statystycznej i opiera się na rozkładzie prawdopodobieństwa Boltzmanna.

2.1.2 Opis implementacji

W algorytmie zaimplementowana została definicja sąsiedztwa polegająca na zamianie dwóch wierzchołków w ścieżce (losowo wybranych).

1. Początkowe rozwiązanie generowanie jest przy pomocy algorytmu zachłannego z wierzchołkiem początkowym dającym najlepszy możliwy wynik algorytmu zachłannego. (Badane są rozwiązania startując kolejno od wszystkich wierzchołków i wybierane jest najlepsze wraz ze ścieżką).
2. Algorytm przechowuje jednocześnie 2 rozwiązania: najlepsze ogółem *bestPath* oraz aktualne rozwiązanie *currentPath* wraz z ich kosztami. Przy inicjalizacji algorytmu do obu zmiennych przypisane zostaje rozwiązanie algorytmu zachłannego.
3. Obliczana jest temperatura początkowa za pomocą metody *calculateInitialTemperature.* Temperatura obliczana jest na podstawie wzoru:
$$T_0 = -\frac{\overline{\Delta E}}{\ln(X_0)}$$
- gdzie $\overline{\Delta E}$ jest przybliżonym średnim kosztem wykonania ruchu (zamiany wierzchołków) w grafie. Koszt ten generowany jest podczas wykonywania losowych ruchów na losowo wygenerowanej ścieżce grafu – wyznaczany jest średni koszt. X_0 to natomiast nasze pożądane prawdopodobieństwo akceptacji (podczas początkowej fazy działania algorytmu). Za wartość X_0 przyjęto 0.85.

4. Za ilość iteracji bez schładzania (epokę) przyjęta zostaje wartość: $rozmiar_{macierzy} \cdot 10$
5. Za współczynnik schładzania α zostaje zadana przez użytkownika wartość z zakresu $0 \leq \alpha \leq 1$ (typ zmiennoprzecinkowy double)
6. Po inicjalizacji wszystkich wcześniej wymienionych parametrów przechodzimy do właściwej części algorytmu – pętli głównej (while) wykonującej się, dopóki czas działania algorytmu nie przekroczy zadanego przez nas kryterium stopu w jednostce sekund.
7. W pętli głównej *while* wykonywane są iteracje poszczególnych epok w pętli for – *for(i=0; i<epoka).*
8. Wewnątrz pętli for (dla zadanej długości epoki) wykonywane są następujące operacje:
    1. Wyznaczane jest nowe rozwiązanie (według definicji sąsiedztwa) wraz z jego kosztem poprzez zamianę dwóch losowych wierzchołków w obecnej ścieżce.
    2. Jeśli koszt nowego rozwiązania jest lepszy (mniejszy) od aktualnego rozwiązania, do obecnej ścieżki przypisana zostaje nowa ścieżka. Dodatkowo w celu zapamiętania najlepszego napotkanego rozwiązania, jeśli koszt nowej ścieżki jest mniejszy do najlepszej ścieżki, do najlepszej ścieżki zostaje przypisana nowa ścieżka.
    3. W wypadku, gdy nowe rozwiązanie jest gorsze od aktualnego rozwiązania, wyliczana jest wartość z funkcji akceptacji według wzoru:
    $P(x_o, x) = \min\left(1, \exp\left(-\frac{f(x)-f(x_0)}{T_i}\right)\right)$, gdzie x_0 to obecne rozwiązanie, x to nowe rozwiązanie, $f(x)$ to koszt obecnego rozwiązania, $f(x_0)$ to koszt nowego rozwiązania, T_i to temperatura w obecnej iteracji.
    4. Następnie losowana jest wartość (liczba zmiennoprzecinkowa double) prawdopodobieństwa z zakresu $0 \leq p \leq 1$.
    5. Jeśli wylosowana wartość prawdopodobieństwa p jest mniejsza od wartości funkcji akceptacji, to do obecnego rozwiązania przypisane zostaje nowe rozwiązanie (nowa ścieżka) – oznacza to akceptację gorszego rozwiązania w celu potencjalnej ucieczki z minimum lokalnego.

9. Po zakończeniu epoki, w której wykonywane są wyżej wymienione operacje, temperatura zostaje schładzana według wzoru $T_{next} = T_{current} \cdot \alpha$, gdzie $T_{current}$ to obecna temperatura, a współczynnik α to współczynnik schładzania.
10. Algorytm ten jest powtarzany aż do osiągniecia czasowego kryterium stopu. Mechanizm epok został wprowadzony w celu „schodkowego" schładzania, gdyż jest ono bardziej efektywne dla wyżej wymienionej funkcji schładzania.

Struktury danych wykorzystane przy implementacji tego algorytmu:

- Tablica dwuwymiarowa (macierz sąsiedztwa grafu wczytywanego z pliku) o rozmiarze $V \cdot V$ gdzie V to liczba wierzchołków grafu.

- Klasa std::vector (tablica dynamiczna) służąca przechowaniu ścieżki – gdzie jednocześnie przechowywane były 3 takie struktury (dla ścieżek: najlepszej, obecnej, nowo wygenerowanej), o rozmiarze $V + 1$, gdzie V to liczba wierzchołków grafu.

2.2 Przeszukiwanie Tabu (ang. Tabu Search)

2.2.1 Opis algorytmu

W każdej iteracji TS oceniamy próbkę sąsiadów bieżącego rozwiązania. Na każdym etapie można zaakceptować ruchy pogarszające się, jeśli nie jest dostępny ruch poprawiający (np. gdy wyszukiwanie utknęło na minimum lokalnym). Dodatkowo wprowadzane są zakazy (stąd określenie tabu), które mają zniechęcić poszukujących do powrotu do wcześniej odwiedzonych rozwiązań. W zależności od implementacji, słabość algorytmu może stanowić fakt, iż dokonujemy w nim częściowego (lub całkowitego) przeglądu sąsiedztwa, co zwiększa złożoność czasową i może doprowadzić do tego, iż w zadanym czasie algorytm TS przeszuka mniejszy obszar rozwiązań niż algorytm SA (Simulated annealing).

2.2.2 Opis implementacji

W wykonanej implementacji przeszukiwania tabu zrealizowano jedną definicję sąsiedztwa, w której zbiór rozwiązań sąsiednich stanowią ścieżki z zamienionymi ze sobą 2 wybranymi wierzchołkami. Podczas przeglądu sąsiedztwa sprawdzane są wszystkie możliwe zamiany wierzchołków, a następnie zamiany (numery wierzchołków) wraz z kosztem takiego rozwiązania trafiają do listy sąsiadów.

Z listy sąsiadów zostają następnie odfiltrowane ruchy (zamiany dwóch wierzchołków), które znajdują się w liście tabu.

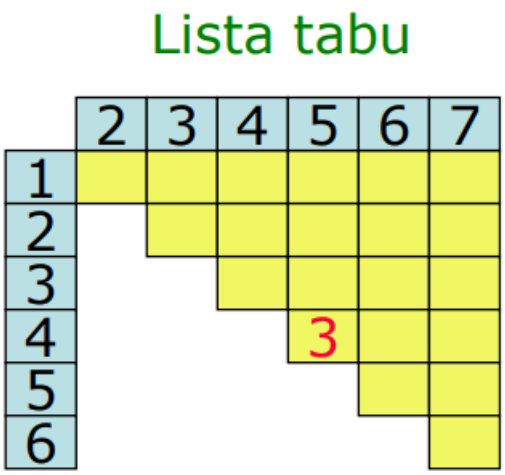
Z odfiltrowanej listy tabu zostaje następnie wybranych N najlepszych kandydatów na rozwiązania – implementacja tzw. „Elitarnej listy sąsiadów" (master list). Dodatkowo wprowadzone zostało pierwsze kryterium aspiracji. Polega ono na usunięciu ograniczenia, jeśli dany ruch daje rozwiązanie lepsze od najlepszego znalezionego do tej pory. W przypadku, gdy wszystkie ruchy są tabu – również bierzemy pod uwagę tylko N najlepszych kandydatów rozwiązań.

1. Podobnie jak w przypadku algorytmu symulowanego wyżarzania, początkowe rozwiązanie generowanie jest przy pomocy algorytmu zachłannego.
2. Algorytm przechowuje jednocześnie trzy rozwiązania w tablicy dynamicznej typu std::vector: aktualnie najlepsze rozwiązanie, pierwsze pojawienie się najlepszego rozwiązania oraz aktualne rozwiązanie.
3. Podczas inicjalizacji przypisywane są także następujące parametry algorytmu:
    1. Kadencja – ilość iteracji, jaka po wykonaniu danego ruchu, będzie on zapisany w liście tabu jako niedozwolony. Kadencja wyznaczana jest według wzoru $K = 2 \cdot \sqrt{V}$, gdzie V to liczba wierzchołków grafu.
    2. Kryterium stopu (czas graniczny trwania algorytmu podany w sekundach)
    3. Ilość iteracji, po której, w przypadku, gdy nie zostanie znalezione nowe lepsze rozwiązanie, wywołana zostanie metoda implementująca strategię dywersyfikacji, tzw. *CriticalEvent.* (W pętli algorytmu zliczane są iteracje bez znalezienia nowego lepszego rozwiązania. W przypadku, gdy lepsze rozwiązanie zostanie znalezione, licznik zostaje wyzerowany). Liczba ta wyznaczona została według następującego wzoru: $D = 2 \cdot \sqrt{V} \cdot 3$, czyli trzykrotność kadencji tabu.
    4. Długość listy kandydatów (jako że zaimplementowana została strategia „elitarnej listy kandydatów" – master list) – jest to długość listy N kandydatów, zaciągana podczas każdego pobrania informacji o sąsiedztwie. Długość wyznaczana jest według wzoru: $N = \begin{cases} V \cdot 0{,}05 \; jeśli \; V > 200 \\ 10 \; dla \; pozostałych \end{cases}$, gdzie V to liczba wierzchołków grafu.
4. Po inicjalizacji wcześniej wymienionych parametrów, przechodzimy do głównej pętli algorytmu (pętla while wykonująca się, dopóki nie zostanie osiągnięte czasowe kryterium stopu), w której wykonywane są następujące operacje:
    1. W pierwszym kroku pobierane jest następne rozwiązanie. Najlepsi kandydaci znajdują się w liście posortowanej niemalejąco według kosztów ścieżek. Następne rozwiązanie jest pobierane ze szczytu listy. Rozwiązanie to jest następnie z niej usuwane. Wyróżnia się następujące przypadki szczególne:
        - W przypadku, gdy lista kandydatów jest pusta, wykonujemy na nowo przegląd sąsiedztwa. Podczas pobieranie listy najlepszych N kandydatów, zapamiętujemy koszt najgorszego rozwiązania tej listy (koszt ostatniego elementu).
        - W przypadku, gdy koszt najlepszego rozwiązania z listy jest większy od zapamiętanego kosztu najgorszego rozwiązania z oryginalnej listy kandydatów (podczas jej pobierania), także wykonujemy ponownego przeglądu sąsiedztwa.
    2. Najlepszy kandydat zostaje przypisany jako aktualne rozwiązanie.
    3. Jeśli koszt nowego rozwiązania jest mniejszy bądź równy kosztowi aktualnie najlepszego rozwiązania, to do aktualnie najlepszej ścieżki przypisana zostaje nowo wygenerowana ścieżka. Dodatkowo w celach testowych, aby śledzić pierwsze pojawienie się jakiegoś rozwiązania (pojawienie się nowego lepszego kosztu), jeśli pojawi się nowe lepsze rozwiązanie globalnie, aktualizowany jest najlepsze globalne rozwiązanie.
    4. Aktualizowana jest lista tabu (tablica tabu). Lista tabu zapamiętuje liczbę iteracji dla danego ruchu. Dla ruchu wykonanego w danej iteracji, ustawiana jest liczba iteracji tabu równa kadencji (wyznaczanej na początku działania algorytmu). Dla pozostałych ruchów (które jeszcze znajdują się w liście tabu) zmniejszana jest liczba iteracji tabu.

5. Jeśli nowo wygenerowane rozwiązanie nie dało lepszego rozwiązania, zwiększany jest licznik uruchamiania strategii dywersyfikacji. W przeciwnym wypadku licznik ten zostaje wyzerowany.
6. **STRATEGIA DYWERSYFIKACJI:** W przypadku osiągnięcia liczby iteracji bez znalezienia lepszego rozwiązania na poziomie wcześniej wymienionego parametru $D = 2 \cdot \sqrt{V} \cdot 3$, realizowana jest strategia dywersyfikacji.
   Strategia dywersyfikacji została zaimplementowana, aby zawierała elementy losowości. Dla 3/5 wywołań metody strategii dywersyfikacji, generowane jest nowe rozwiązanie losując 2 wierzchołki grafu, tworząc odcinek w ścieżce. Następnie wierzchołki w ścieżce są losowo zamieniane. Odcinek nie może przekroczyć połowy całkowitej ścieżki.
   W pozostałych 2/3 przypadków nowe rozwiązanie generowane jest za pomocą odwracania wybranych fragmentów ścieżki. Zwracana jest wtedy ścieżka z najmniejszym kosztem.
   Częstość obu akcji została wyznaczona w sposób eksperymentalny jako dająca najlepsze rezultaty.

Struktury danych wykorzystane przy implementacji tego algorytmu:

- Lista tabu – została zaimplementowana jako dwuwymiarowa tablica, tak jak na rysunku poniżej.



Rysunek 1. Implementacja listy tabu za pomocą tablicy. Źródło: http://www.pi.zarz.agh.edu.pl/intObl/notes/IntObl_w2.pdf

Dla przyjętej definicji sąsiedztwa, w której sąsiedzi definiowani są poprzez zamianę dwóch wierzchołków, implementacja listy tabu jako tablica dwuwymiarowa jest najwygodniejsza i najszybsza (dzięki bezpośredniemu dostępowi do elementów pod zadanym indeksem). Wadą takiego rozwiązania jest fakt, iż połowa tablicy dwuwymiarowej pozostaje pusta (wynika z oczywistego faktu, że w przypadku problemu Komiwojażera zamiana dwóch wierzchołków, np. 1 oraz 2 stanowi ruch identyczny co zamiana wierzchołków 2 z 1).

Każdy poszczególny ruch (zamiana dwóch wierzchołków) posiada swój indeks w tablicy, pod danym indeksem (ruch, zamiana 5 i 7 posiada odpowiadający indeks w tablicy [5][7]). Jeśli pod zadanym indeksem wpisana jest liczba większa od 0, stanowi ona kadencję (liczbę iteracji), przez jaką dany ruch pozostaje tabu.

- Klasa std::vector (tablica dynamiczna) służąca przechowaniu ścieżki – gdzie jednocześnie przechowywane były 4 takie struktury (dla ścieżek: obecnie najlepszej, o obecnym najlepszym koszcie (pierwsze pojawienie się rozwiązania o danym koszcie), obecnej ścieżki oraz nowo wygenerowanej), o rozmiarze V+1, gdzie V to liczba wierzchołków grafu.

- Lista kandydatów typu std::list<pair<int, int>, int> - przechowująca ruch (zamianę dwóch wierzchołków w postaci pary liczb integer) oraz koszt ścieżki po wykonaniu takiego ruchu. Maksymalny rozmiar takiej lity to $(V \cdot V)/2$, gdzie V to liczba wierzchołków. Jednakże rozmiar listy w kolejnych iteracjach jest mniejszy z powodu odrzucania ruchów tabu.

- Lista elitarnych kandydatów std::list<pair<int, int>, int> zawierająca N najlepszych kandydatów z wcześniej wymienionej listy kandydatów oraz dodatkowo kandydatów spełniających kryterium aspiracji (akceptujemy ruch tabu jeśli daje rozwiązanie lepsze niż najlepsze dotychczas).

3. Opis najważniejszych klas programu

3.1 AppController

Klasa ta, jak sama nazwa mówi, łączy infrastrukturę programu. Odpowiada za wywołanie widoku konsolowego (menu), po czym wywołuje wybrane przez użytkownika akcje tj. wywołanie algorytmów, wczytanie grafu z pliku itd. W polu klasy znajdują się wskaźniki do klas zawierających algorytm symulowanego wyżarzania (SimulatedAnnealing), algorytm przeszukiwania tabu (TabuSearch), klasy obudowującej macierz grafu (ATSPMatrix). Klasa wywołuje również statyczne metody klas pomocniczych – m.in. do odczytywania oraz zapisywania danych w plikach oraz pomiaru czasu.

3.2 DataFileUtility

Klasa posiada statyczne metody służące wczytaniu danych z plików, m.in. do wczytania pliku zawierającego ścieżkę. Znajdują się w niej także metody do zapisu danych do pliku: m.in. wynikowej ścieżki po uruchomieniu algorytmu oraz plików zawierających dane zbierane podczas testów algorytmów (wyniki, temperatura, znaczniku czasu znalezienia lepszego rozwiązania, poszczególne rozwiązania).

3.3 ATSPMatrix

Klasa opakowująca alokowaną dynamicznie tablicę dwuwymiarową stanowiąca macierz sąsiedztwa wczytywanych przez program grafów (z plików). Klasa ta udostępnia klasom zawierającym logikę algorytmów wskaźnik do macierzy, rozmiar oraz posiada kilka metod pomocniczych, takich jak na przykład metodę alokującą dynamicznie tablicę o rozmiarze odpowiadającym wczytywanemu grafowi oraz przypisuje zadane w nim wagi do owej tablicy.

3.4 Timer

Klasa pomocnicza służąca do pomiaru czasu. Sposób pomiaru czasu opisany został w kolejnej sekcji omawiającej plan eksperymentu. Wykorzystuje ona *QueryPerformanceCounter* z biblioteki windows.h.

3.5 RandomDataGenerator

Klasa pomocnicza, która zamiera metody służące wygenerowaniu danych pseudolosowych. Służą one m.in. do wygenerowania prawdopodobieństwa do funkcji akceptacji z zakresu $\langle 0,1 \rangle$ – typu zmiennoprzecinkowego double. Posiada również metodę generującą liczbę typu integer dla zadanego zakresu, w celu wygenerowania np. wierzchołków do zamiany w przypadku algorytmu symulowanego wyżarzania. Wykorzystuje ona generator liczb pseudolosowych z biblioteki random.

3.6 GreedyAlgorithm

Klasa ta, jak sama nazwa mówi, zawiera metodę implementującą algorytm zachłanny dla problemu komiwojażera. Posiada również metodę wywołującą algorytm zachłanny dla każdego z możliwych wierzchołków i zwracającą rozwiązanie z najlepszym wierzchołkiem początkowym.

3.7 SimulatedAnnealing

Klasa zamierająca logikę odpowiedzialną za algorytm symulowanego wyżarzania wraz z funkcjami wykonującymi poszczególne operacje algorytmu. Szczegóły implementacyjne zostały przedstawione we wcześniejszych punktach.

Najważniejsze metody:

- mainFun – służy inicjalizacji parametrów
- solveTSP – zawiera w sobie główną pętlę algorytmu
- calculateCost – funkcja oblicza koszt ścieżki
- perturbPath – funkcja odpowiedzialna za wygenerowanie nowego rozwiązania (poprzez zamianę dwóch losowych wierzchołków)
- calculateInitialTemperature – funkcja służąca obliczeniu początkowej temperatury na podstawie średniego kosztu wyznaczonego poprzez losowe ruchy w grafie
- generateRandomSolution – funkcja pomocnicza generująca losowe rozwiązania (cykle Hamiltona) wykorzystywane w funkcji obliczającej temperaturę początkową.

3.8 TabuSearch

Klasa zamierająca logikę odpowiedzialną za algorytm przeszukiwania tabu wraz z funkcjami wykonującymi poszczególne operacje algorytmu. Szczegóły implementacyjne zostały przedstawione we wcześniejszych punktach.

Najważniejsze metody:

- mainFun - służy inicjalizacji parametrów
- solveTSP – zawiera w sobie główną pętlę algorytmu
- getNextMovesFromNeighbours – funkcja generująca sąsiednie rozwiązania podczas przeglądu sąsiedztwa (wszystkie możliwe zamiany 2 wierzchołków). Kandydaci zwracani są w postaci listy zawierające numery wierzchołków do potencjalnej zamiany oraz koszt zmodyfikowanej ścieżki.
- getSwappedPathCost – funkcja pomocnicza, obliczająca koszt ścieżki po modyfikacji zdefiniowanej jako zamiana dwóch wierzchołków.
- filterTabuSwaps – funkcja służąca odfiltrowaniu ruchów tabu z listy sąsiadów
- updateTabuList – funkcja służąca aktualizacji listy tabu po przejściu iteracji w pętli głównej
- generateDiversificationCandidate – funkcja zwracająca nowe rozwiązanie jeśli uruchomiona została strategia dywersyfikacji
- calculatePathCost – funkcja służąca obliczeniu kosztu ścieżki (cyklu Hamiltona)
- getNextMovesFromNeighboursMasterList – funkcja zwracająca „Listę elitarnych kandydatów” (master list) zawierającą N najlepszych ruchów z listy sąsiadów. Uwzględnione zostało także kryterium aspiracji – jeśli ruch tabu generuje rozwiązanie lepsze niż najlepsze dotychczas, zostaje ono dodane do listy
- updateMasterListCosts – funkcja służąca aktualizacji kosztów listy elitarnych kandydatów po przejściu iteracji
- swapVectorElements – funkcja pomocnicza, odpowiadająca za zamianę dwóch zadanych wierzchołków w ścieżce (cyklu Hamiltona).

4. Plan eksperymentu

4.1 Założenia

1. W celu zbadania efektywności zaimplementowanych algorytmów, należało zbadać ich dokładność dla podanych 3 plików opisujących niezależny problem ATSP (Asymetryczny problem komiwojażera):

   - ftv55.atsp – rozmiar mały
   - ftv170.atsp – rozmiar średni
   - rbg358.atsp – rozmiar duży

2. Czas działania algorytmów ustawiony był na sztywno:

   - 2 minuty dla rozmiaru małego,
   - 4- średniego,
   - 8-dużego

3. Algorytm symulowanego wyżarzania zbadano dla 3 różnych współczynników schładzania, dla każdego z grafów.

4. Algorytm przeszukiwania tabu zawierał tylko jedną implementację sąsiedztwa, więc wykonano tylko jedną iterację testów dla danego grafu.

5. Dla obu algorytmów przyjęto te same czasowe kryteria przerwania.

6. Dla każdego pliku algorytm uruchomiony został 10 razy. Wyniki umieszczone zostały w dalszej części sprawozdania.

4.2 Sposób generowania liczb pseudolosowych oraz pomiaru czasu

Generator liczb losowych wykorzystywał maszynę losującą: 32-bit Mersenne twister engine z biblioteki random, czyli *std::mt19937 gen.*

Listing 1. Metoda służąca do wygenerowania liczby pseudolosowej z podanego zakresu.

```
int RandomDataGenerator::generateVertexInRange(int from, int to) {
    std::random_device rdev;
    std::mt19937 gen(rdev());
    std::uniform_int_distribution<> dist(from, to);
    return dist(gen);
}
```

W celu uzyskania dużej dokładności pomiarowej czasu (o rozdzielczości mikrosekundowej), wykorzystana została klasa *QueryPerformanceCounter().*

Listing 2. Funkcja służąca odczytaniu licznika procesora.

```
long long int Timer::read_QPC() {
    LARGE_INTEGER count;
    QueryPerformanceCounter(&count);
    return ((long long int) count.QuadPart);
}
```

W ten sposób zmierzony został stan licznika przed rozpoczęciem testowanej operacji oraz po jej wykonaniu. Następnie, Końcowy rezultat uzyskuje się przez podzielenie odmierzonej liczby impulsów licznika przez częstotliwość, którą otrzymuje się za pomocą wywołania *QueryPerformanceFrequency().*

Listing 3. Funkcja wyznaczająca czas między dwoma zmierzonymi punktami pomiaru licznika procesora.

```
double Timer::getMicroSecondsElapsed(long long int start, long long int
end) {
    long long int elapsed, frequency;
    elapsed = end - start;
    QueryPerformanceFrequency((LARGE_INTEGER *) &frequency);
    return ((1000000.0 * elapsed) / frequency);
}
```

4.3 Ogólny plan przeprowadzania eksperymentu

- Oba algorytmy zostały przetestowane dla 3 zadanych instancji problemu zawartych w plikach ftv55.atsp, ftv170.atsp oraz rbg358.atsp. Instancje problemu stanowią kolejno problem mały (przyjęto czasowe kryterium stopu 2 minuty), średni (przyjęto czasowe kryterium stopu 4 minuty) oraz duży (przyjęto czasowe kryterium stopu 6 minut). Algorytmy dla zadanej instancji zostały uruchomione po 10 razy w celu otrzymania uśrednionych wyników.
- Poniżej przedstawione wyniki pomiarów to wartości uśrednione na podstawie 10 przebiegów danego algorytmu dla podanego pliku .atsp.
- Dla obu algorytmów zapisywany był czas znalezienia najlepszego rozwiązania w danym przebiegu. Wartości pomiarów czasu przedstawione zostały w jednostce Milisekund (ms).
- Błąd względny wyznaczony został na podstawie najlepszych znalezionych rozwiązań oraz najlepszego znanego rozwiązania dla danego grafu:

  $\frac{|f_{zn}-f_{opt}|}{f_{zn}}$, gdzie f_{zn} – najlepsza wartość obliczona przez nasz algorytm, f_{opt} – wartość optymalna (najlepsze znane rozwiązanie).

5. Wyniki eksperymentów

5.1 Symulowane wyżarzanie

Algorytm symulowanego wyżarzania dla każdego z grafów przetestowano dla 3 różnych schematów schładzania (współczynników alfa). Sposób dobierania współczynników był następujący:

1. Na początku wybierano współczynnik optymalny, według wyrażenia exp(-1/Tk), gdzie Tk stanowiła temperatura końcowa algorytmu. Wyrażenie to wynika z funkcji akceptacji algorytmu symulowanego wyżarzania i jego wynik stanowi prawdopodobieństwo akceptacji gorszego rozwiązania podczas końcowej epoki działania algorytmu. Prawdopodobieństwo to powinno wtedy oscylować w okolicach kilku procent. W ramach tego eksperymentu, optymalny współczynnik wyznaczano, aby wyrażenie wynosiło pomiędzy: $0{,}01 \leq \exp(-1/\mathrm{Tk}) \leq 0{,}10$.
2. Współczynnik mniejszy od optymalnego (dla którego chłodzenie odbywało się zbyt szybko) dobrano tak, aby wartość wyrażenia była możliwie jak najniższa, lecz aby algorytm nadal dawał rozwiązania lepsze od rozwiązania algorytmu zachłannego. Zbyt szybkie chłodzenie powoduje, że algorytm działa w sposób prawidłowy jedynie przez ułamek czasu działania.
3. Współczynnik większy (dla którego chłodzenie odbywało się zbyt wolno) dobierano tak, aby wartość wyrażenia oscylowała pomiędzy: $0{,}5 \leq \exp(-1/\mathrm{Tk}) \leq 0{,}99$. W takim przypadku algorytm podczas końcowej epoki działania nadal akceptuje zbyt wielką ilość rozwiązań gorszych od aktualnego, co w końcowej fazie nie powinno mieć miejsca.

Należy zaznaczyć, że pomimo faktu iż 2 pozostałe od optymalnego współczynniki powodują gorsze działanie algorytmu, zdarzyło się podczas testów że otrzymywano dla nich pojedyncze wyniki lepsze niż w przypadku współczynnika optymalnego. Wynika to z faktu, że algorytm jest niedeterministyczny i istnieje szansa, że algorytm natrafi na rozwiązanie znacznie lepsze już na samym początku działania algorytmu.

5.1.1 Graf ftv55

Czas działania algorytmu dla tego problemu ustawiono na 2 minuty.

Dla tego problemu przyjęte zostały następujące współczynniki:

- $\alpha = 0.9998$ (chłodzenie zbyt szybkie)
- $\alpha = 0.99988$ (chłodzenie optymalne)
- $\alpha = 0.99993$ (chłodzenie zbyt wolne)

Poniższe tabele przedstawiają rezultaty otrzymane podczas testów. Sporządzony został wykres błędu względnego w funkcji czasu dla 3 schematów schładzania.

Rezultaty dla zbyt szybkiego schematu schładzania:

	Rozmiar grafu: 56					
	Współczynnik chłodzenia: **0.9998**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	37115.80	37.12	1948	**1748**	0.00231	5.35E-189
2	45771.80	45.77	1948	**1762**	0.00285	7.48E-153
3	37866.50	37.87	1948	**1609**	0.00229	2.97E-190
4	37480.70	37.48	1948	**1659**	0.00253	3.59E-172
5	39904.60	39.90	1948	**1691**	0.00228	1.85E-191
6	44323.00	44.32	1948	**1725**	0.00237	4.60E-184
7	37126.10	37.13	1948	**1791**	0.00220	4.16E-198
8	36965.70	36.97	1948	**1776**	0.00220	7.18E-198
9	40915.10	40.92	1948	**1693**	0.00225	1.83E-193
10	35589.90	35.59	1948	**1761**	0.00561	4.09E-78

Najlepszy znany koszt: 1608		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1748	8.71%
2	1762	9.58%
3	1609	0.06%
4	1659	3.17%
5	1691	5.16%
6	1725	7.28%
7	1791	11.38%
8	1776	10.45%
9	1693	5.29%
10	1761	9.51%

Rezultaty dla optymalnego schematu schładzania:

	Rozmiar grafu: 56					
	Współczynnik chłodzenia: **0.99988**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	64281.20	64.28	1948	**1745**	0.333	0.0494
2	57181.80	57.18	1948	**1808**	0.308	0.0387
3	63201.20	63.20	1948	**1708**	0.339	0.0525
4	74362.40	74.36	1948	**1691**	0.362	0.0632
5	65044.40	65.04	1948	**1754**	0.329	0.0477
6	60685.20	60.69	1948	**1716**	0.323	0.0450
7	72224.30	72.22	1948	**1677**	0.318	0.0432
8	65857.60	65.86	1948	**1694**	0.305	0.0375
9	57439.00	57.44	1948	**1681**	0.305	0.0376
10	60266.00	60.27	1948	**1778**	0.311	0.0402

Najlepszy znany koszt: 1608		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1745	8.52%
2	1808	12.44%
3	1708	6.22%
4	1691	5.16%
5	1754	9.08%
6	1716	6.72%
7	1677	4.29%
8	1694	5.35%
9	1681	4.54%
10	1778	10.57%

Rezultaty dla zbyt wolnego schematu schładzania:

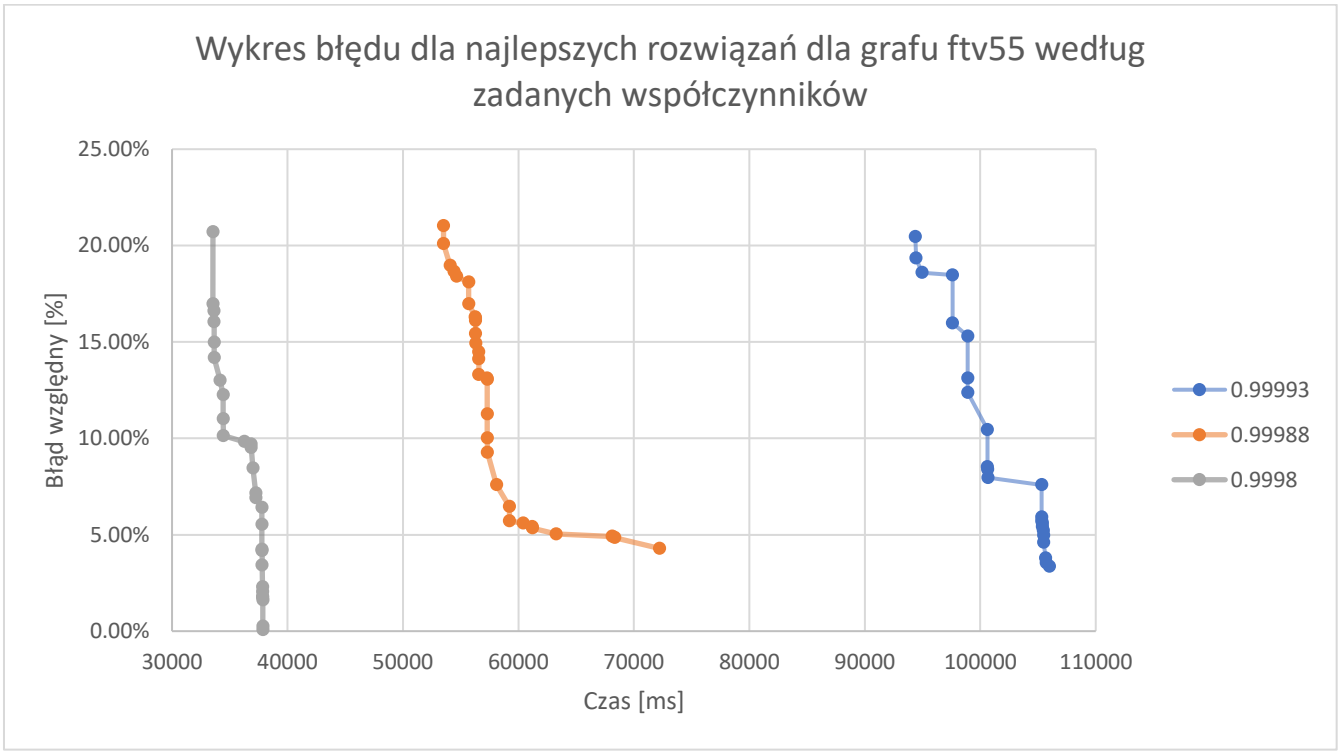
	Rozmiar grafu: 56					
	Współczynnik chłodzenia: **0.99993**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	106234.00	106.23	1948	**1696**	7.739	0.879
2	108845.20	108.85	1948	**1710**	8.255	0.886
3	109128.00	109.13	1948	**1700**	8.590	0.890
4	108160.00	108.16	1948	**1726**	7.941	0.882
5	117235.00	117.24	1948	**1688**	8.880	0.893
6	105988.00	105.99	1948	**1662**	7.695	0.878
7	107451.00	107.45	1948	**1748**	7.922	0.881
8	112200.04	112.20	1948	**1705**	10.032	0.905
9	118013.00	118.01	1948	**1673**	10.206	0.907
10	110843.00	110.84	1948	**1669**	9.678	0.902

Najlepszy znany koszt: 1608		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1696	5.47%
2	1710	6.34%
3	1700	5.72%
4	1726	7.34%
5	1688	4.98%
6	1662	3.36%
7	1748	8.71%
8	1705	6.03%
9	1673	4.04%
10	1669	3.79%

Wykres błędu względnego w funkcji czasu dla wszystkich schematów schładzania:

Wykres przedstawia przebiegi funkcji błędu w dziedzinie czasu dla najlepszych rozwiązań dla danego współczynnika.

- $\alpha = 0.9998$, najlepsze rozwiązanie: 1609, błąd względny: 0.06%
- $\alpha = 0.99988$, najlepsze rozwiązanie: 1677, błąd względny: 4.29%
- $\alpha = 0.99993$, najlepsze rozwiązanie: 1662, błąd względny: 3.36%



5.1.2 Graf ftv170

Czas działania algorytmu dla tego problemu ustawiono na 4 minuty.

Dla tego problemu przyjęte zostały następujące współczynniki:

- $\alpha = 0.99975$ (chłodzenie zbyt szybkie)
- $\alpha = 0.9998$ (chłodzenie optymalne)
- $\alpha = 0.99985$ (chłodzenie zbyt wolne)

Rezultaty zbyt szybkiego schematu schładzania:

	Rozmiar grafu: 171					
	Współczynnik chłodzenia: **0.99975**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	141427.00	141.427	3582	**3482**	0.0555	1.50E-08
2	136857.00	136.857	3582	**3427**	0.0656	2.42E-07
3	4.35	0.004	3582	**3582**	0.0654	2.31E-07
4	137995.00	137.995	3582	**3478**	0.0589	4.21E-08
5	140496.00	140.496	3582	**3511**	0.0927	2.07E-05
6	126183.00	126.183	3582	**3403**	0.0594	4.92E-08
7	139183.00	139.183	3582	**3418**	0.0665	2.93E-07
8	123232.00	123.232	3582	**3372**	0.0645	1.86E-07
9	119636.00	119.636	3582	**3543**	0.0772	2.39E-06
10	4.50	0.004	3582	**3582**	0.0725	1.01E-06

Najlepszy znany koszt: 2755		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	3482	26.39%
2	3427	24.39%
3	3582	30.02%
4	3478	26.24%
5	3511	27.44%
6	3403	23.52%
7	3418	24.07%
8	3372	22.40%
9	3543	28.60%
10	3582	30.02%

Rezultaty dla optymalnego schematu schładzania:

	Rozmiar grafu: 171					
	Współczynnik chłodzenia: **0.9998**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	149536.00	149.536	3582	**3471**	0.266	0.0232
2	176326.00	176.326	3582	**3517**	0.266	0.0234
3	154533.00	154.533	3582	**3398**	0.298	0.0347
4	164767.00	164.767	3582	**3527**	0.256	0.0202
5	4.64	0.005	3582	**3582**	0.278	0.0275
6	149485.00	149.485	3582	**3338**	0.237	0.0147
7	174601.00	174.601	3582	**3335**	0.544	0.1590
8	174703.00	174.703	3582	**3317**	0.365	0.0648
9	166536.00	166.536	3582	**3312**	0.394	0.0792
10	4.22	0.00422	3582	**3582**	0.411	0.0876

Najlepszy znany koszt: 2755		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	3471	25.99%
2	3517	27.66%
3	3398	23.34%
4	3527	28.02%
5	3582	30.02%
6	3338	21.16%
7	3335	21.05%
8	3317	20.40%
9	3312	20.22%
10	3582	30.02%

Rezultaty dla zbyt wolnego schematu schładzania:

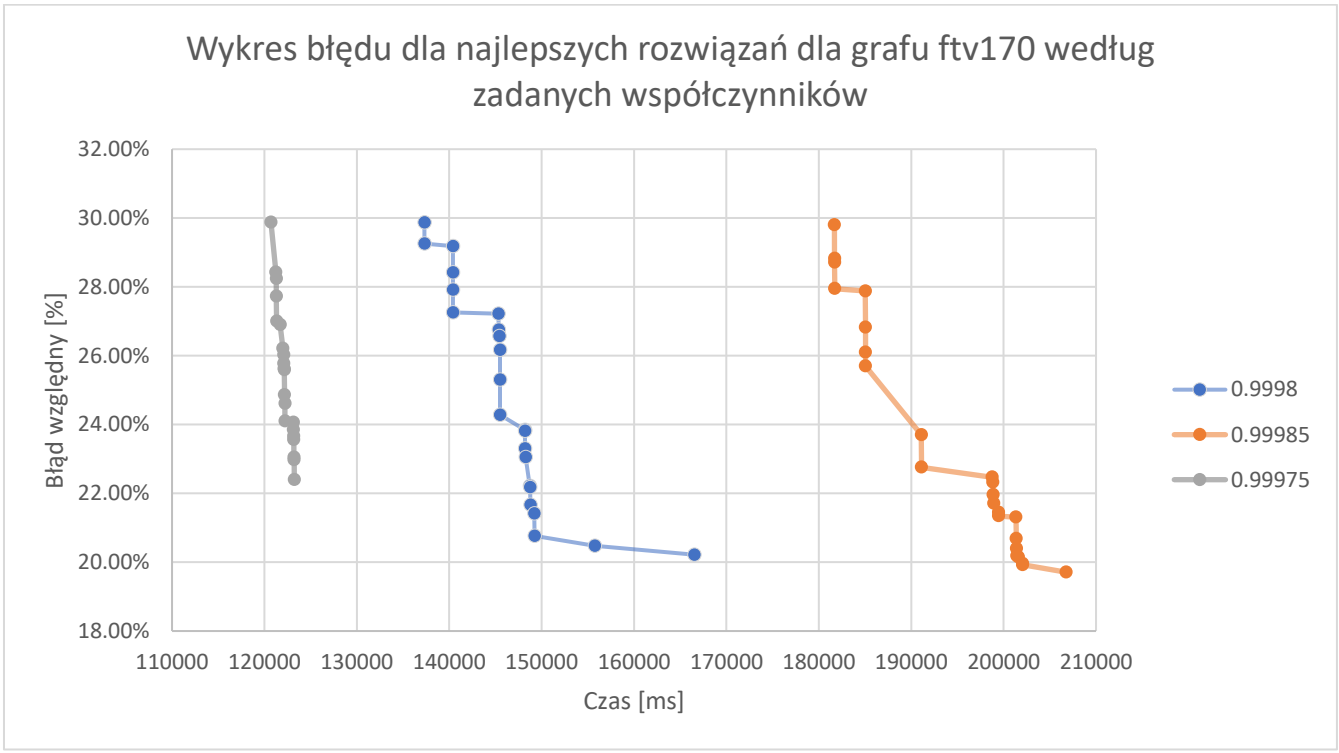
	Rozmiar grafu: 171					
	Współczynnik chłodzenia: **0.99985**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	4.68	0.00468	3582	**3582**	2.497	0.670
2	4.36	0.00436	3582	**3582**	2.320	0.650
3	4.53	0.00453	3582	**3582**	2.355	0.654
4	225301.00	225.301	3582	**3535**	2.243	0.640
5	206751.00	206.751	3582	**3298**	2.260	0.642
6	240953.00	240.953	3582	**3475**	2.357	0.654
7	212664.00	212.664	3582	**3550**	2.262	0.643
8	218177.00	218.177	3582	**3308**	2.250	0.641
9	221139.00	221.139	3582	**3467**	2.511	0.672
10	225278.00	225.278	3582	**3450**	2.596	0.680

Najlepszy znany koszt: 2755		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	3582	30.02%
2	3582	30.02%
3	3582	30.02%
4	3535	28.31%
5	3298	19.71%
6	3475	26.13%
7	3550	28.86%
8	3308	20.07%
9	3467	25.84%
10	3450	25.23%

Wykres błędu względnego w funkcji czasu dla wszystkich schematów schładzania:

Wykres przedstawia przebiegi funkcji błędu w dziedzinie czasu dla najlepszych rozwiązań dla danego współczynnika.

- $\alpha = 0.99975$, najlepsze rozwiązanie: 3372, błąd względny: 22.40%
- $\alpha = 0.9998$, najlepsze rozwiązanie: 3312, błąd względny: 20.22%
- $\alpha = 0.99985$, najlepsze rozwiązanie: 3298, błąd względny: 19.71%



5.1.3 Graf rbg358

Czas działania algorytmu dla tego problemu ustawiono na 6 minut.

Dla tego problemu przyjęte zostały następujące współczynniki:

- $\alpha = 0.99975$ (chłodzenie zbyt szybkie)
- $\alpha = 0.9998$ (chłodzenie optymalne)
- $\alpha = 0.99985$ (chłodzenie zbyt wolne)

Rezultaty dla zbyt szybkiego schematu schładzania:

	Rozmiar grafu: 358					
	Współczynnik chłodzenia: **0.99975**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	340296.00	340.30	1747	**1209**	0.0862	9.10E-06
2	318452.00	318.45	1747	**1191**	0.0853	8.15E-06
3	303047.00	303.05	1747	**1197**	0.0811	4.40E-06
4	301077.00	301.08	1747	**1220**	0.0797	3.54E-06
5	326589.00	326.59	1747	**1232**	0.0838	6.59E-06
6	326549.00	326.55	1747	**1220**	0.0823	5.32E-06
7	293273.00	293.27	1747	**1219**	0.0805	4.03E-06
8	313765.00	313.77	1747	**1207**	0.0820	5.08E-06
9	351173.00	351.17	1747	**1207**	0.0817	4.87E-06
10	324359.00	324.36	1747	**1219**	0.0770	2.31E-06

Najlepszy znany koszt: 1163		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1209	3.96%
2	1191	2.41%
3	1197	2.92%
4	1220	4.90%
5	1232	5.93%
6	1220	4.90%
7	1219	4.82%
8	1207	3.78%
9	1207	3.78%
10	1219	4.82%

Rezultaty dla optymalnego schematu schładzania:

	Rozmiar grafu: 358					
	Współczynnik chłodzenia: **0.9998**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	351898.00	351.90	1747	**1227**	0.314	0.0413
2	357506.00	357.51	1747	**1220**	0.316	0.0421
3	359858.00	359.86	1747	**1209**	0.315	0.0420
4	356246.00	356.25	1747	**1198**	0.309	0.0392
5	358473.00	358.47	1747	**1214**	0.316	0.0422
6	359707.00	359.71	1747	**1210**	0.310	0.0396
7	359091.00	359.09	1747	**1197**	0.307	0.0387
8	356230.00	356.23	1747	**1204**	0.314	0.0414
9	359771.00	359.77	1747	**1224**	0.312	0.0406
10	354144.00	354.14	1747	**1215**	0.308	0.0390

Najlepszy znany koszt: 1163		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1227	5.50%
2	1220	4.90%
3	1209	3.96%
4	1198	3.01%
5	1214	4.39%
6	1210	4.04%
7	1197	2.92%
8	1204	3.53%
9	1224	5.25%
10	1215	4.47%

Rezultaty dla zbyt wolnego schematu schładzania:

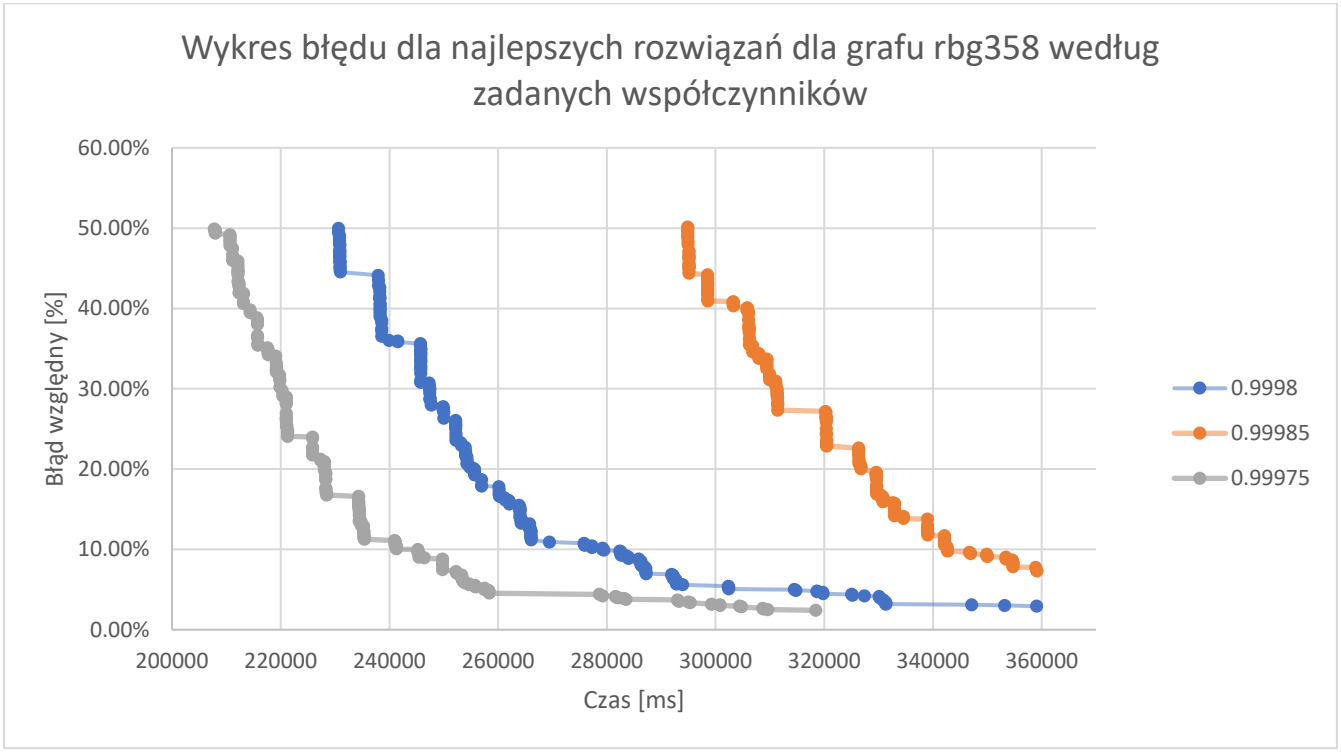
	Rozmiar grafu: 358					
	Współczynnik chłodzenia: **0.99985**					
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm	Temperatura końcowa	exp(-1/Tk)
1	358233.00	358.23	1747	**1289**	1.201	0.435
2	359133.00	359.13	1747	**1251**	1.187	0.431
3	359855.00	359.86	1747	**1277**	1.192	0.432
4	354109.00	354.11	1747	**1248**	1.217	0.440
5	359688.00	359.69	1747	**1246**	1.194	0.433
6	359850.00	359.85	1747	**1292**	1.201	0.435
7	353168.00	353.17	1747	**1251**	1.212	0.438
8	359904.00	359.90	1747	**1264**	1.207	0.437
9	357377.00	357.38	1747	**1267**	1.233	0.444
10	351335.00	351.34	1747	**1263**	1.216	0.439

Najlepszy znany koszt: 1163		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1289	10.83%
2	1251	7.57%
3	1277	9.80%
4	1248	7.31%
5	1246	7.14%
6	1292	11.09%
7	1251	7.57%
8	1264	8.68%
9	1267	8.94%
10	1263	8.60%

Wykres błędu względnego w funkcji czasu dla wszystkich schematów schładzania:

Wykres przedstawia przebiegi funkcji błędu w dziedzinie czasu dla najlepszych rozwiązań dla danego współczynnika.

- $\alpha = 0.99975$, najlepsze rozwiązanie: 1191, błąd względny: 2.41%
- $\alpha = 0.9998$, najlepsze rozwiązanie 1197, błąd względny: 2.92%
- $\alpha = 0.99985$, najlepsze rozwiązanie 1246, błąd względny: 7.14%



5.2 Przeszukiwanie tabu

Jako że zaimplementowana została tylko jedna definicja sąsiedztwa w przypadku algorytmu przeszukiwania tabu, wykonana została jedynie jedna seria po 10 testów dla danego problemu.

5.2.1 Graf ftv55

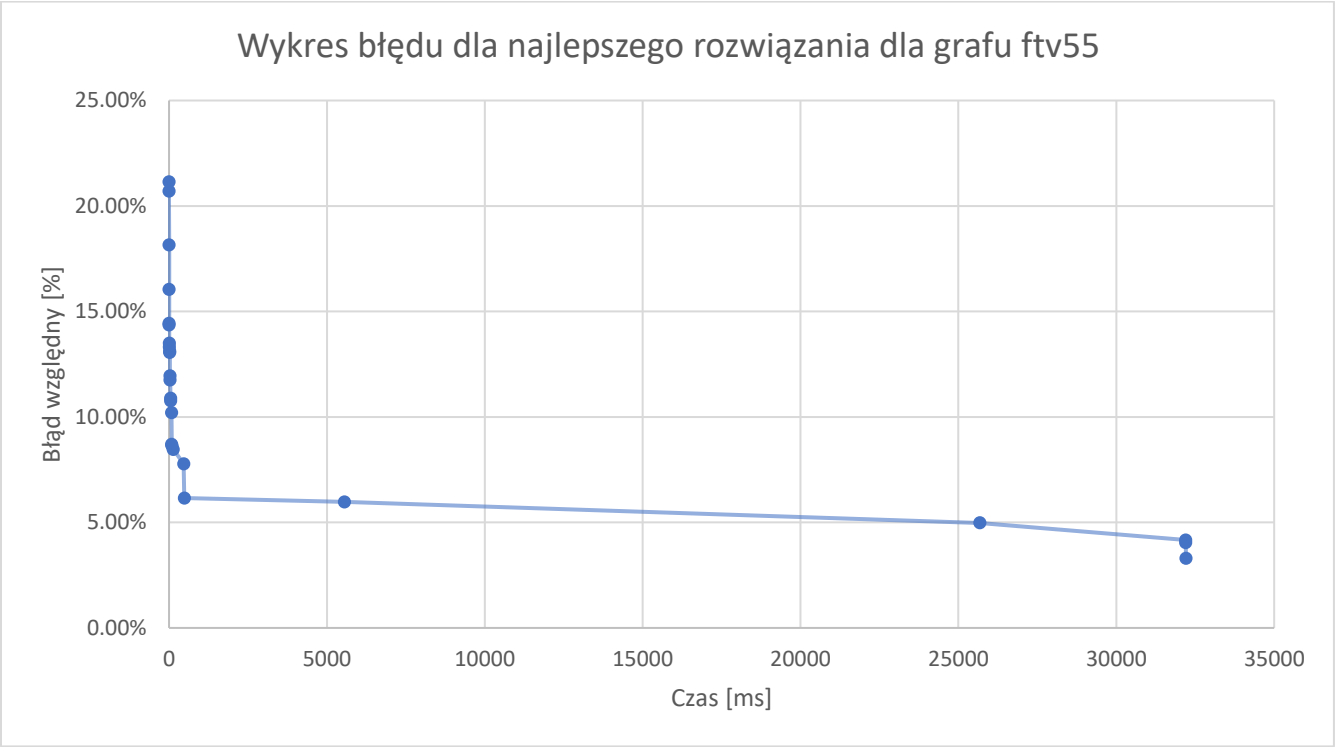
Czas działania algorytmu dla tego problemu ustawiono na 2 minuty.

	Rozmiar grafu: 56			
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm
1	84704.9	84.7049	1948	**1661**
2	27578.4	27.5784	1948	**1704**
3	115605	115.605	1948	**1694**
4	6454.52	6.45452	1948	**1704**
5	32203	32.203	1948	**1661**
6	57251.7	57.2517	1948	**1704**
7	7645.9	7.6459	1948	**1704**
8	89129.4	89.1294	1948	**1688**
9	24682.6	24.6826	1948	**1704**
10	10038.5	10.0385	1948	**1704**

Najlepszy znany koszt: 1608		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1661	3.30%
2	1704	5.97%
3	1694	5.35%
4	1704	5.97%
5	1661	3.30%
6	1704	5.97%
7	1704	5.97%
8	1688	4.98%
9	1704	5.97%
10	1704	5.97%

Wykres błędu względnego w funkcji czasu:

- Najlepsze znalezione rozwiązanie: 1661, błąd względny: 3.30%



5.2.2 Graf ftv170

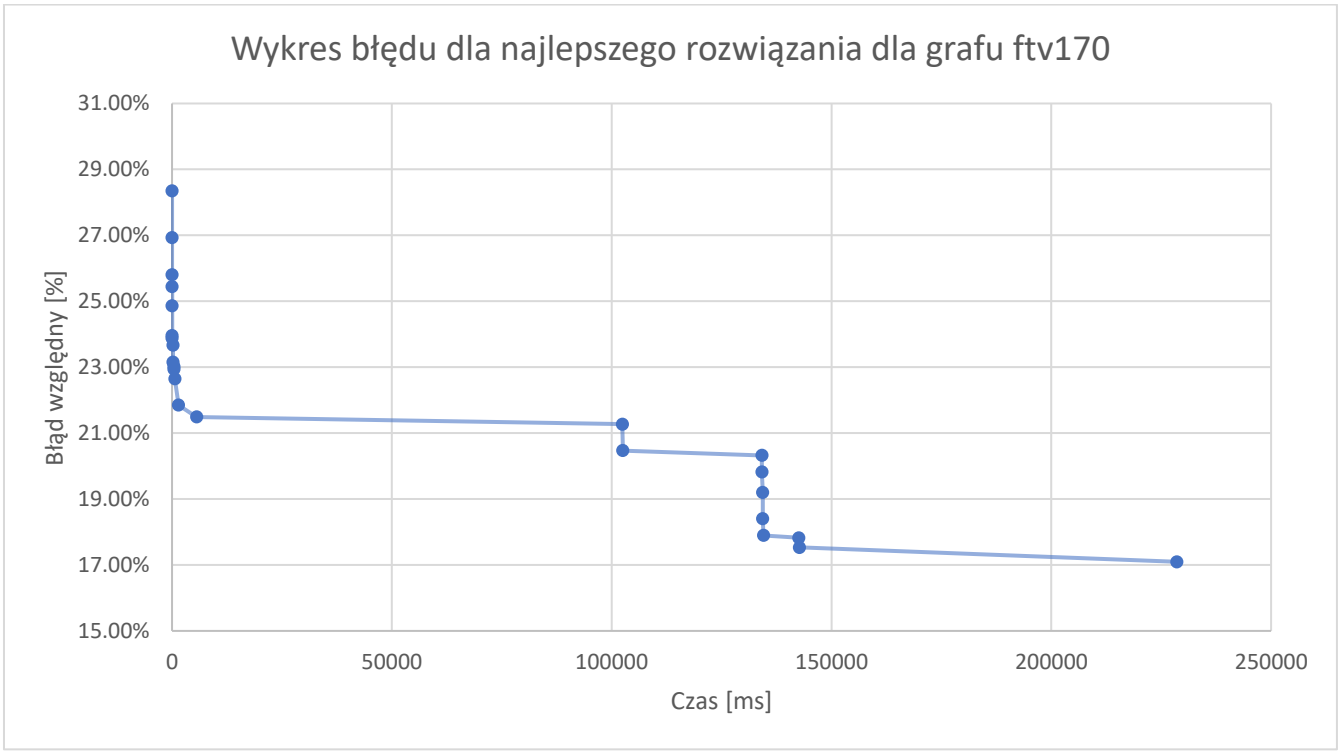
Czas działania algorytmu dla tego problemu ustawiono na 4 minuty.

	Rozmiar grafu: 171			
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm
1	178357	178.357	3582	**3241**
2	228578	228.578	3582	**3226**
3	134763	134.763	3582	**3253**
4	43063.3	43.0633	3582	**3265**
5	227569	227.569	3582	**3253**
6	78010.7	78.0107	3582	**3229**
7	141930	141.93	3582	**3226**
8	171518	171.518	3582	**3251**
9	143876	143.876	3582	**3249**
10	217242	217.242	3582	**3238**

Najlepszy znany koszt: 2755		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	3241	17.64%
2	3226	17.10%
3	3253	18.08%
4	3265	18.51%
5	3253	18.08%
6	3229	17.21%
7	3226	17.10%
8	3251	18.00%
9	3249	17.93%
10	3238	17.53%

Wykres błędu względnego w funkcji czasu:

- Najlepsze znalezione rozwiązanie: 3226, błąd względny: 17.10%



5.2.3 Graf rbg358

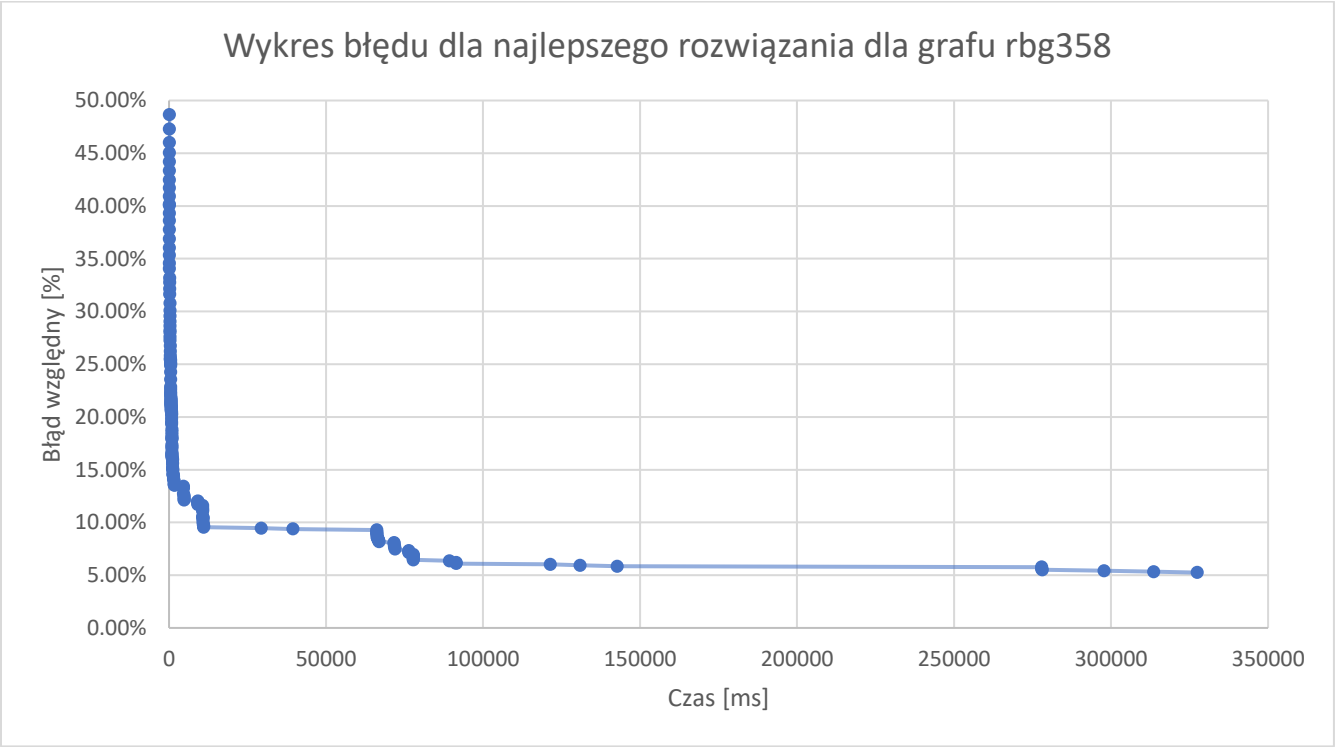
Czas działania algorytmu dla tego problemu ustawiono na 6 minut.

	Rozmiar grafu: 358			
Lp.	Czas znalezienia najlepszego rozwiązania [ms]	Czas znalezienia najlepszego rozwiązania [s]	Koszt ścieżki wyznaczonej z algorytmu zachłannego	Najlepsze znalezione rozwiązanie obliczone przez algorytm
1	224638	224.638	1747	**1250**
2	193086	193.086	1747	**1228**
3	290331	290.331	1747	**1257**
4	328282	328.282	1747	**1229**
5	329443	329.443	1747	**1240**
6	195926	195.926	1747	**1238**
7	176025	176.025	1747	**1232**
8	327392	327.392	1747	**1224**
9	310665	310.665	1747	**1251**
10	306775	306.775	1747	**1240**

Najlepszy znany koszt: 1163		
Lp.	Najlepsze znalezione rozwiązanie	Błąd względny
1	1250	7.48%
2	1228	5.59%
3	1257	8.08%
4	1229	5.67%
5	1240	6.62%
6	1238	6.45%
7	1232	5.93%
8	1224	5.25%
9	1251	7.57%
10	1240	6.62%

Wykres błędu względnego w funkcji czasu:

- Najlepsze znalezione rozwiązanie: 1224, błąd względny: 5.25%



5.3 Porównanie obu algorytmów

Poniższa tabela przedstawia porównanie najlepszych rezultatów obu algorytmów dla danego problemu (grafu):

	TS		SA	
	Najlepsze znalezione rozwiązanie	Błąd względny	Najlepsze znalezione rozwiązanie	Błąd względny
ftv55	1661	3.30%	1609	**0.06%**
ftv170	3226	**17.10%**	3298	19.71%
rbg358	1224	5.25%	1191	**2.41%**

Algorytm symulowanego wyżarzania okazał się bardziej skuteczny w przypadku problemów ftv55 oraz rbg358. Algorytm przeszukiwania tabu był zauważalnie bardziej skuteczny w przypadku problemu ftv170.

6. Wnioski

Na podstawie przeprowadzonych eksperymentów wywnioskować można, że algorytm przeszukiwania tabu jest bardziej dokładny. Na wykresach funkcji błędu względnego w dziedzinie czasu obserwujemy, że algorytm ten praktycznie od samego początku znajduje dużo lepsze rozwiązania od tego, które wyznacza algorytm zachłanny.

W przypadku algorytmu symulowanego wyżarzania, choć zależy to znacząco od współczynnika schładzania, znajdowanie rozwiązań lepszych od algorytmu zachłannego zachodzi później (mechanizm ucieczki z minimum lokalnego). Mimo że algorytm daje mniej spójne rezultaty, okazuje się lepszy w przypadku 2 z 3 badanych grafów.

Prawdopodobnie wynika to z faktu, że zaimplementowany algorytm przeszukiwania tabu charakteryzuje większa złożoność czasowa (wynika z regularnego przeglądu całego sąsiedztwa), przez co w tym samym czasie działania, algorytm nie jest w stanie przeszukać tak wielkiego obszaru rozwiązań, jak algorytm symulowanego wyżarzania. Być może zatem, gdyby udało się przyspieszyć działanie algorytmu, okazałoby się że przeszukiwanie tabu jest obiektywie lepszy.

Większa dokładność algorytmu Tabu Search ma jednak swoje zalety, ponieważ jak możemy zaobserwować, dla problemu ftv170 osiągnął lepszy rezultat.

7. Literatura i źródła

1. Symulowane wyżarzanie – Wykład autorstwa dr hab. inż. Macieja Komosińskiego

https://www.youtube.com/watch?v=gX-X85dCib0&t=1003s

https://www.cs.put.poznan.pl/mkomosinski/lectures/optimization/SA.pdf

2. Przeszukiwanie tabu – Wykład autorstwa dr hab. inż. Macieja Komosińskiego

https://www.youtube.com/watch?v=HsGJrSBFQNs&t=1914s

https://www.cs.put.poznan.pl/mkomosinski/lectures/optimization/TS.pdf

3. Metody przeszukiwania lokalnego

https://cs.pwr.edu.pl/zielinski/lectures/om/localsearch.pdf

4. Heurystyki i metaheurystyki – AGH

http://www.pi.zarz.agh.edu.pl/intObl/notes/IntObl_w2.pdf

5. Metody wyznaczania temperatury początkowej algorytmu symulowanego wyżarzania - Walid Ben-Ameur

https://www.researchgate.net/publication/227061666_Computing_the_Initial_Temperature_of_Simulated_Annealing

8. Najlepsze znalezione rozwiązania

8.1 Symulowane wyżarzanie

- ftv55 – koszt 1609

Ścieżka:

15 16 17 52 0 33 2 13 35 4 5 6 7 32 8 36 9 37 11 38 19 20 40 42 21 22 41 50 23 54 27 49 45 30 46 55 34 1 3 48 31 47 53 43 44 28 29 26 25 24 18 39 10 51 14 12 15

- ftv170 – koszt 3298

Ścieżka:

146 145 144 143 147 148 149 161 142 152 14 15 25 150 160 151 8 9 10 76 74 75 11 12 18 19 20 29 22 24 159 16 17 21 23 26 27 28 30 31 41 42 45 44 46 47 48 49 50 170 73 77 2 1 110 109 83 84 69 67 66 63 64 56 57 62 61 68 167 70 85 71 51 52 53 43 55 54 58 59 60 65 88 153 154 89 90 91 87 86 93 108 166 107 106 105 104 114 113 127 126 125 136 129 128 130 131 115 116 117 118 120 121 122 162 123 101 98 95 94 92 96 97 165 163 99 100 102 103 119 124 137 138 139 140 141 6 7 13 32 158 36 157 33 34 35 156 155 40 39 38 37 168 72 78 82 79 80 81 3 4 5 133 169 111 0 112 132 134 164 135 146

- rbg358 – koszt 1191

Ścieżka:

307 180 47 319 243 264 341 62 190 297 99 49 19 102 91 155 106 306 336 84 32 31 138 113 105 225 294 136 85 147 233 78 58 261 242 241 189 204 8 326 24 254 318 262 332 279 338 168 167 2 57 177 296 195 89 182 323 302 274 88 122 303 301 11 81 23 100 298 248 252 335 317 161 152 146 229 309 53 239 21 90 305 300 76 1 132 148 166 227 315 64 68 353 313 129 247 236 4 125 343 287 51 98 104 5 347 83 235 202 285 72 172 238 46 170 45 25 12 256 160 269 92 275 344 288 266 151 143 232 17 13 3 312 299 186 346 79 52 171 251 174 56 345 289 10 282 200 263 27 334 178 184 270 260 183 320 223 222 316 224 70 30 114 63 197 80 0 37 28 325 211 181 20 357 96 153 65 314 286 208 115 329 253 36 29 164 218 73 280 48 278 61 220 250 215 308 209 130 349 331 139 271 245 283 118 116 162 133 311 95 127 131 206 191 230 342 169 348 292 187 205 258 59 44 40 203 74 310 41 354 273 140 110 304 194 67 259 356 101 234 199 50 321 337 137 240 295 193 217 201 179 173 150 216 128 293 339 276 192 327 33 322 109 16 126 188 77 18 108 290 159 121 120 231 328 213 255 176 111 141 163 7 145 246 6 15 284 43 38 93 26 112 103 119 267 75 54 185 212 142 352 214 158 149 226 157 330 257 228 71 277 351 87 291 219 196 144 123 156 60 340 281 324 210 265 272 86 165 107 94 117 135 69 39 237 355 268 35 42 198 154 124 97 22 350 207 134 82 55 66 14 34 9 249 221 244 333 175 307

8.2 Przeszukiwanie tabu

- ftv55 – koszt 1661

Ścieżka:

15 16 17 52 0 33 2 13 35 4 6 5 47 30 46 55 34 1 3 48 31 29 26 25 42 21 22 41 50 54 27 49 45 44 28 53 43 23 24 20 40 19 18 39 37 36 9 11 38 10 32 8 7 51 14 12 15

- ftv170 – koszt 3226

Ścieżka:

24 25 150 160 151 152 141 134 131 132 111 110 109 115 116 117 118 119 120 122 162 123 101 102 121 124 129 128 130 135 138 139 140 6 7 8 9 10 76 74 75 11 12 18 19 20 21 29 22 26 27 28 30 31 33 35 34 156 40 39 38 37 49 170 168 73 77 1 0 81 80 79 82 78 72 71 60 50 51 52 53 43 55 54 58 59 61 68 67 167 70 87 85 86 83 84 69 66 63 64 56 57 62 65 88 153 154 89 90 91 94 96 97 99 98 95 92 93 166 108 107 106 105 165 163 100 103 104 114 113 127 126 125 136 137 147 148 149 161 14 13 17 32 158 36 157 155 41 42 44 45 46 47 48 2 3 4 5 133 169 112 164 146 145 144 143 142 15 159 16 23 24

- rbg358 – koszt 1224

Ścieżka:

266 43 12 4 44 137 263 27 16 17 13 3 164 218 321 329 310 48 354 24 35 18 346 55 36 29 278 251 37 326 314 187 205 258 167 25 38 225 11 246 6 81 15 284 56 23 185 75 209 51 78 345 289 52 104 353 287 264 341 280 76 1 357 61 227 315 290 236 180 165 170 194 67 253 93 30 99 59 125 294 66 14 80 0 88 58 257 68 124 135 298 26 112 248 155 151 143 154 322 352 183 176 311 223 77 42 198 90 305 181 171 213 286 22 114 63 97 208 131 189 192 295 193 92 275 276 191 238 204 182 91 70 32 31 127 98 347 291 119 199 212 249 87 83 122 303 200 233 19 211 210 265 226 41 74 64 234 188 120 45 7 145 28 96 130 111 216 222 316 224 134 82 146 136 85 84 33 156 60 108 95 138 113 317 139 271 245 140 110 304 142 109 168 186 107 94 153 65 273 72 172 292 174 8 115 319 243 129 247 21 157 103 158 149 241 206 10 282 274 159 121 285 161 152 79 162 133 335 269 235 202 179 173 349 331 254 160 105 169 348 343 313 126 267 50 337 175 307 207 177 296 195 255 320 214 252 312 141 283 118 259 260 270 356 101 219 106 262 332 272 203 73 62 86 47 340 301 324 279 308 54 100 333 220 46 150 299 144 123 232 217 201 166 339 230 71 334 240 250 215 338 277 302 281 344 288 221 244 328 325 350 40 231 20 132 148 228 327 330 261 242 323 336 147 117 163 2 57 256 318 306 351 128 293 229 309 53 239 69 39 237 355 268 89 342 190 297 197 49 5 196 178 184 116 34 9 102 300 266