# Problem Komiwojażera

## Spis Treści

Spis treści

[Problem Komiwojażera 1](#_Toc211800265)

[Spis Treści 1](#_Toc211800266)

[Wstęp 2](#_Toc211800267)

[Testowane Algorytmy i Dane 2](#_Toc211800268)

[Kluczowe Elementy Implementacji 2](#_Toc211800269)

[Sekwencje Ruchu (Operatorzy Sąsiedztwa) 2](#_Toc211800270)

[Parametryzacja i Cel Projektu 3](#_Toc211800271)

[Podsumowanie i Cel Końcowy 3](#_Toc211800272)

[Rozwiązania Solver (Excel) 3](#_Toc211800273)

[Algorytm Najbliższego Sąsiada (Nearest Neighbor - NN) 3](#_Toc211800274)

[Zasada Działania 3](#_Toc211800275)

[Parametryzacja 4](#_Toc211800276)

[Testowanie 4](#_Toc211800277)

[Wnioski 4](#_Toc211800278)

[Algorytm Wspinaczki Wzgórz (Hill Climbing - HC) 4](#_Toc211800279)

[Zasada Działania 4](#_Toc211800280)

[Parametryzacja i Protokół Testowania 4](#_Toc211800281)

[Protokół Testowania 5](#_Toc211800282)

[Testowanie 5](#_Toc211800283)

[Wnioski 5](#_Toc211800284)

[Algorytm Symulacji Wyżarzania (Simulated Annealing - SA) 5](#_Toc211800285)

[Zasada Działania 5](#_Toc211800286)

[Parametryzacja i Protokół Testowania 6](#_Toc211800287)

[Protokół Testowania 6](#_Toc211800288)

[Testowanie 6](#_Toc211800289)

[Wnioski 6](#_Toc211800290)

## Wstęp

Niniejsze sprawozdanie prezentuje analizę i porównanie różnych algorytmów heurystycznych oraz metaheurystycznych zaprojektowanych do rozwiązania **Problemu Komiwojażera (TSP - Traveling Salesperson Problem)**. TSP, będący klasycznym przykładem problemu optymalizacji kombinatorycznej, polega na znalezieniu najkrótszej możliwej trasy, która łączy zbiór miast, przy czym każde miasto jest odwiedzane dokładnie raz, a trasa kończy się w mieście startowym.

### Testowane Algorytmy i Dane

W ramach projektu zaimplementowano i przetestowano szereg algorytmów optymalizacyjnych. Ich skuteczność i wydajność zostały ocenione na podstawie trzech dostarczonych nam zbiorów danych TSP: **TSP\_127**, **TSP\_48** oraz **TSP\_76**. Różnica w liczbie węzłów w tych zbiorach danych pozwala na wszechstronną ocenę skalowalności i efektywności zastosowanych metod.

### Kluczowe Elementy Implementacji

Wszystkie zaimplementowane algorytmy zostały wzbogacone o mechanizm **multistart (wielokrotnego startu)**. Multistart jest strategią mającą na celu zwiększenie szans na znalezienie globalnie optymalnego rozwiązania, zwłaszcza w przypadku problemów z wieloma lokalnymi minimami. Polega ona na **wielokrotnym uruchamianiu algorytmu z różnych, losowo lub heurystycznie wybranych, początkowych rozwiązań (tras)**. Najlepsze rozwiązanie spośród wszystkich uzyskanych w poszczególnych przebiegach jest traktowane jako ostateczny wynik. Dzięki temu mechanizmowi, algorytm ma możliwość eksploracji szerszego obszaru przestrzeni rozwiązań.

### Sekwencje Ruchu (Operatorzy Sąsiedztwa)

W celu eksploracji przestrzeni rozwiązań i poszukiwania lepszych tras, algorytmy lokalnego przeszukiwania wykorzystują różne **sekwencje ruchu**, czyli operatory sąsiedztwa, definiujące możliwe przejścia z obecnego rozwiązania do rozwiązań sąsiednich:

* **Swap (zamiana):** Polega na **zamianie miejscami dwóch losowo wybranych miast** w bieżącej trasie. Jest to najprostszy operator sąsiedztwa.
* **Insert (wstawianie):** Polega na **usunięciu jednego miasta z obecnej pozycji i wstawieniu go w inną, losowo wybraną pozycję** w trasie.
* **Reverse (odwrócenie), często znane jako 2-opt:** Polega na **wybraniu dwóch krawędzi w trasie, usunięciu ich, a następnie odwróceniu sekwencji miast między tymi punktami, aby utworzyć dwie nowe krawędzie**. Ta operacja jest często bardzo efektywna w znajdowaniu lepszych lokalnie rozwiązań, ponieważ efektywnie minimalizuje przecinające się krawędzie. W naszej pracy zastosowaliśmy różne podejścia do Reverse nie tylko klasyczne ale również różniące się (w zależności od algorytmu)

### Parametryzacja i Cel Projektu

Algorytmy optymalizacyjne są **sparametryzowane**, co oznacza, że ich działanie jest regulowane przez zestaw specyficznych wartości (np. liczba iteracji, temperatura początkowa w Simulated Annealing, wielkość populacji w Algorytmach Ewolucyjnych itp.). W ramach projektu przetestowane zostały różne **sekwencje parametrów** i ich kombinacje, których szczegółowy opis i analiza wpływu na wyniki zostaną przedstawione w dalszych sekcjach sprawozdania.

### Podsumowanie i Cel Końcowy

Głównym celem niniejszego projektu jest **wskazanie, który z zaimplementowanych algorytmów (wraz z jego optymalnymi parametrami)** zapewnia najlepsze rozwiązania dla każdego z testowanych zbiorów danych (**TSP\_127, TSP\_48, TSP\_76**). Końcowa analiza ma na celu identyfikację optymalnego algorytmu i zestawu parametrów dla każdego przypadku, co pozwoli na wyciągnięcie wniosków dotyczących efektywności różnych strategii heurystycznych w kontekście Problemu Komiwojażera.

## Rozwiązania Solver (Excel)

## Algorytm Najbliższego Sąsiada (Nearest Neighbor - NN)

Algorytm Najbliższego Sąsiada (NN) jest najprostszą **heurystyką konstrukcyjną** wykorzystywaną do szybkiego generowania początkowego rozwiązania dla TSP.

### Zasada Działania

Algorytm NN buduje trasę na zasadzie **zachłannego wyboru**:

1. Zaczyna od wybranego miasta startowego.
2. Zawsze wybiera jako następne to miasto, które jest **najbliżej** obecnego, spośród miast jeszcze nieodwiedzonych.
3. Proces jest kontynuowany do momentu odwiedzenia wszystkich miast, a trasa zamyka się powrotem do miasta startowego.

Ze względu na swoją naturę, NN nie gwarantuje globalnego optimum, ale jest szybkim sposobem na uzyskanie rozwiązania bazowego.

### Parametryzacja

W przypadku algorytmu Najbliższego Sąsiada, **jedynym parametrem** wpływającym na wynik jest wybór **miasta startowego**.

Aby zminimalizować wpływ tego wyboru na jakość trasy, testy zostały przeprowadzone dla **każdego możliwego miasta startowego** w danym zbiorze danych. Ostatecznym wynikiem algorytmu NN jest **najkrótsza trasa** uzyskana ze wszystkich tych symulacji. Technika ta jest specyficzną formą **multistartu** zastosowaną do algorytmu konstrukcyjnego.

### Testowanie

### Wnioski

### Algorytm Wspinaczki Wzgórz (Hill Climbing - HC)

Algorytm Wspinaczki Wzgórz (HC) jest podstawową **heurystyką lokalnego przeszukiwania (local search)**, której celem jest znalezienie najlepszego rozwiązania w bezpośrednim sąsiedztwie obecnego.

### Zasada Działania

Algorytm zaczyna od rozwiązania początkowego i w każdej iteracji przechodzi do **lepszego rozwiązania sąsiedniego**. Generowanie rozwiązań sąsiednich odbywa się za pomocą operatorów ruchu: **SWAP, INSERT lub REVERSE (2-opt)** (szczegółowo opisanych we Wstępie). Algorytm zatrzymuje się, gdy nie jest w stanie znaleźć lepszego rozwiązania w swoim sąsiedztwie (utknięcie w **lokalnym minimum**).

### Parametryzacja i Protokół Testowania

Aby zwiększyć szansę na ucieczkę z lokalnych minimów i wszechstronnie ocenić algorytm, zastosowano mechanizm **multistart** (jak wyjaśniono we Wstępie), a jego działanie regulują następujące parametry, testowane w różnych konfiguracjach:

| **Parametr** | **Zbiór testowanych wartości** | **Opis** |
| --- | --- | --- |
| **maxIterations** (Maksymalna Liczba Iteracji) | 1000, 5000, 10000, 20000 | Kryterium zatrzymania oparte na liczbie kroków. |
| **maxStagnation** (Maksymalna Stagnacja) | 250, 500, 1000, 2000 | Kryterium zatrzymania oparte na liczbie iteracji bez poprawy. |
| **methods** (Operator Ruchu) | "SWAP", "INSERT", "REVERSE" | Definiuje, który z operatorów sąsiedztwa jest używany w danej serii testów. |

Protokół Testowania

Każda unikalna kombinacja tych parametrów jest testowana w **10 niezależnych uruchomieniach (multistart = 10)**. Ostatecznym wynikiem dla danej konfiguracji jest **najlepsza (najkrótsza) trasa** uzyskana ze wszystkich 10 startów.

### Testowanie

### Wnioski

## Algorytm Symulacji Wyżarzania (Simulated Annealing - SA)

Algorytm Symulacji Wyżarzania (SA) to **metaheurystyka**, która rozszerza przeszukiwanie lokalne, wprowadzając element losowości pozwalający na **akceptowanie gorszych ruchów** i tym samym na **ucieczkę z lokalnych minimów**.

### Zasada Działania

W przeciwieństwie do HC, SA pozwala na gorsze ruchy z prawdopodobieństwem zależnym od aktualnej **temperatury ($T$)** – im wyższa temperatura, tym większa szansa na akceptację gorszej trasy (eksploracja). W miarę postępu algorytmu temperatura jest stopniowo obniżana (**chłodzenie**), co sprawia, że algorytm staje się coraz bardziej zachłanny (eksploatacja). Ruchy są generowane za pomocą operatorów sąsiedztwa (**SWAP, INSERT, REVERSE**).

### Parametryzacja i Protokół Testowania

Wydajność SA jest silnie zależna od odpowiedniego doboru schematu chłodzenia, kontrolowanego przez następujące parametry, testowane w różnych konfiguracjach:

| **Parametr** | **Zbiór testowanych wartości** | **Opis** |
| --- | --- | --- |
| **Initial Temperature ($T\_0$)** | 1000.0, 500.0, 200.0 | Kontroluje początkowy stopień eksploracji (prawdopodobieństwo akceptacji gorszych ruchów). |
| **Cooling Rate ($\alpha$)** | 0.99, 0.999, 0.95 | Wpływa na tempo obniżania temperatury. Wartość bliższa 1 oznacza wolniejsze chłodzenie i dokładniejsze przeszukiwanie. |
| **Solutions Per Temperature ($L$)** | 10, 50, 100 | Liczba prób zmian (ruchów) wykonywanych przy stałej temperaturze. |
| **Methods** (Operator Ruchu) | "SWAP", "INSERT", "REVERSE” | Definiuje, który operator sąsiedztwa jest wykorzystywany. |

### Protokół Testowania

Podobnie jak w przypadku HC, **każda unikalna kombinacja** parametrów SA jest testowana w **10 niezależnych uruchomieniach (multistart = 10)**, startując z różnych rozwiązań początkowych. Najkrótsza trasa z tych 10 prób jest traktowana jako ostateczny wynik danej konfiguracji.

### Testowanie

### Wnioski