Politechnika Wrocławska

Badania operacyjne i optymalizacja dyskretna

Sprawozdanie z wykonania zadania

Autorzy: Sebastian Żółkiewicz *259337* Piotr Kulczycki *259366*

Kod przedmiotu: W04ISA-SM0401G, Grupa: 2

Termin zajęć: czwartek 11:15 - 13:00

Politechnika
Wrocławska

Spis treści

1 Opis ćwiczenia **2**

2 Problem komiwojażera (TSP) **2**

3 Algorytmy rozwiązujące **2**
3.1 Algorytm symulowanego wyżarzania (SA) 2
3.2 Algorytm najbliższego sąsiada (NN) 2
3.3 Algorytm dalekiej insercji (FI) 3

4 Dane wejściowe i reprezentacja rozwiązania **3**
4.1 Macierz odległości 3
4.2 Reprezentacja permutacji 3

5 Dobieranie Hiperparametrów w Algorytmie Symulowanego Wyżarzania (SA) . . **4**
5.1 Hiperparametry Algorytmu SA 4
5.2 Optymalizacja Hiperparametrów za pomocą Optuna 4
5.2.1 Funkcja celu 4
5.2.2 Optymalizacja z użyciem Optuna 4
5.3 Wizualizacja Wyników 5
5.4 Podsumowanie strojenia hiperparametrów 6

6 Badania **7**
6.1 Metodyka badań 7
6.2 Wyniki 8

7 Wnioski **10**

8 Źródła **10**

1 Opis ćwiczenia

W ramach ćwiczenia zaimplementowano i zbadano algorytmy SA, NN i FI dla problemu TSP.

2 Problem komiwojażera (TSP)

Problem komiwojażera (ang. Traveling Salesman Problem, TSP) jest jednym z klasycznych problemów optymalizacyjnych w teorii grafów i badaniach operacyjnych. Zadanie polega na znalezieniu najkrótszej możliwej trasy, która odwiedza każde z n miast dokładnie raz i powraca do miasta początkowego. Formalnie można go przedstawić w postaci:

- Dane: Graf pełny $G = (V, E)$, gdzie V jest zbiorem wierzchołków (miast), a E zbiorem krawędzi o wagach d_{ij} reprezentujących odległości między miastami i i j.
- Cel: Znaleźć permutację P wierzchołków, która minimalizuje funkcję celu:

$$\text{Min} \sum_{i=1}^{n} d_{P(i),P(i+1)} + d_{P(n),P(1)}.$$

TSP jest problemem NP-trudnym, co oznacza, że nie istnieje znany algorytm rozwiązujący go dokładnie w czasie wielomianowym.

3 Algorytmy rozwiązujące

3.1 Algorytm symulowanego wyżarzania (SA)

Algorytm symulowanego wyżarzania (ang. Simulated Annealing, SA) jest metaheurystyką inspirowaną procesem fizycznym wyżarzania. Polega na stopniowym chłodzeniu układu w celu znalezienia rozwiązania o minimalnej energii (w kontekście TSP - minimalnej długości trasy). Kluczowe etapy algorytmu:

- Początkowe losowe rozwiązanie.
- Iteracyjne modyfikowanie trasy przez permutacje sąsiednich wierzchołków.
- Akceptowanie gorszych rozwiązań z prawdopodobieństwem zależnym od temperatury T i różnicy jakości ΔE:

$$P = e^{-\Delta E/T}.$$

- Stopniowe zmniejszanie temperatury T.

3.2 Algorytm najbliższego sąsiada (NN)

Algorytm najbliższego sąsiada (ang. Nearest Neighbor, NN) jest prostą heurystyką działającą według zasady chciwej. Rozpoczyna w losowo wybranym mieście, a następnie iteracyjnie wybiera najbliższe jeszcze nieodwiedzone miasto. Proces trwa, aż wszystkie miasta zostaną odwiedzone. Chociaż algorytm jest szybki, często prowadzi do rozwiązań dalekich od optymalnych.

3.3 Algorytm dalekiej insercji (FI)

Algorytm dalekiej insercji (ang. Farthest Insertion, FI) to heurystyka, która iteracyjnie buduje rozwiązanie, zaczynając od najmniejszego możliwego podzbioru wierzchołków. Algorytm działa według następujących kroków:

1. Rozpoczyna od początkowej trasy złożonej z dwóch najbliższych miast.
2. W każdej iteracji wybiera wierzchołek, który znajduje się najdalej od już odwiedzonych wierzchołków w bieżącej trasie.
3. Dodaje ten wierzchołek do trasy, wstawiając go w miejsce minimalizujące wzrost długości trasy.

4 Dane wejściowe i reprezentacja rozwiązania

Dane wejściowe dla problemu komiwojażera (TSP) opisują miasta, ich wzajemne położenie oraz odległości między nimi. Mogą być reprezentowane na różne sposoby w zależności od specyfiki problemu i zastosowanego algorytmu. W tym zadaniu zastosowano:

4.1 Macierz odległości

Macierz odległości jest jedną z najpopularniejszych reprezentacji danych wejściowych. Jest to macierz $D = [d_{ij}]$, gdzie d_{ij} oznacza odległość (lub koszt podróży) między miastami i i j.

- Jeśli miasta są połączone bezpośrednio, $d_{ij} > 0$.
- W przypadku grafów pełnych każda para miast ma przypisaną wartość d_{ij}.
- Dla grafów nieskierowanych $d_{ij} = d_{ji}$, a w przypadku grafów skierowanych wartości te mogą się różnić.

4.2 Reprezentacja permutacji

Rozwiązanie problemu TSP jest często reprezentowane jako permutacja wierzchołków grafu, np. $[v_1, v_2, \ldots, v_n]$, która określa kolejność odwiedzania miast. Długość trasy oblicza się wtedy jako suma kosztów krawędzi w tej permutacji:

$$\text{Długość trasy} = \sum_{i=1}^{n-1} d_{v_i,v_{i+1}} + d_{v_n,v_1}.$$

5 Dobieranie Hiperparametrów w Algorytmie Symulowanego Wyżarzania (SA)

W algorytmie symulowanego wyżarzania (SA), wybór odpowiednich hiperparametrów ma kluczowe znaczenie dla jakości i efektywności rozwiązywanego problemu, takiego jak problem komiwojażera (TSP). Aby znaleźć optymalne wartości hiperparametrów, można zastosować techniki optymalizacji, takie jak Optuna, która automatycznie dobiera najlepsze wartości dla zadanych parametrów. W tym rozdziale przedstawiono podejście do doboru hiperparametrów w algorytmie SA za pomocą biblioteki Optuna.

5.1 Hiperparametry Algorytmu SA

Algorytm symulowanego wyżarzania (SA) wykorzystuje kilka kluczowych hiperparametrów, które wpływają na jego działanie:

- **Temperatura początkowa (T0)**: Określa początkowy poziom "gorąca" w algorytmie, czyli szansę na akceptację gorszego rozwiązania. Wysoka wartość T0 pozwala na większą eksplorację przestrzeni rozwiązań, natomiast zbyt mała może prowadzić do zbyt wczesnego zatrzymania procesu.
- **Współczynnik chłodzenia (alpha)**: Określa, jak szybko temperatura będzie malała podczas procesu wyżarzania. Wartość ta zwykle mieści się w zakresie od 0.85 do 0.99. Mniejsze wartości oznaczają szybsze obniżenie temperatury, co może skutkować szybszym zbieżnością algorytmu.
- **Liczba iteracji na poziomie temperatury (L)**: Określa, ile iteracji algorytm wykona przy każdej ustalonej temperaturze. Większa liczba iteracji może zwiększyć dokładność algorytmu, ale także jego czas działania.
- **Minimalna temperatura (T_min)**: Określa, jaką minimalną temperaturę osiągnie algorytm przed zakończeniem procesu. Zbyt wysoka minimalna temperatura może prowadzić do nieoptymalnych rozwiązań.

5.2 Optymalizacja Hiperparametrów za pomocą Optuna

Optuna to biblioteka do optymalizacji hiperparametrów, która umożliwia automatyczne dostosowanie wartości parametrów w celu uzyskania najlepszych wyników w zadanym problemie. W tym przypadku wykorzystaliśmy Optuna do optymalizacji hiperparametrów algorytmu symulowanego wyżarzania (SA) dla rozwiązania problemu komiwojażera.

5.2.1 Funkcja celu

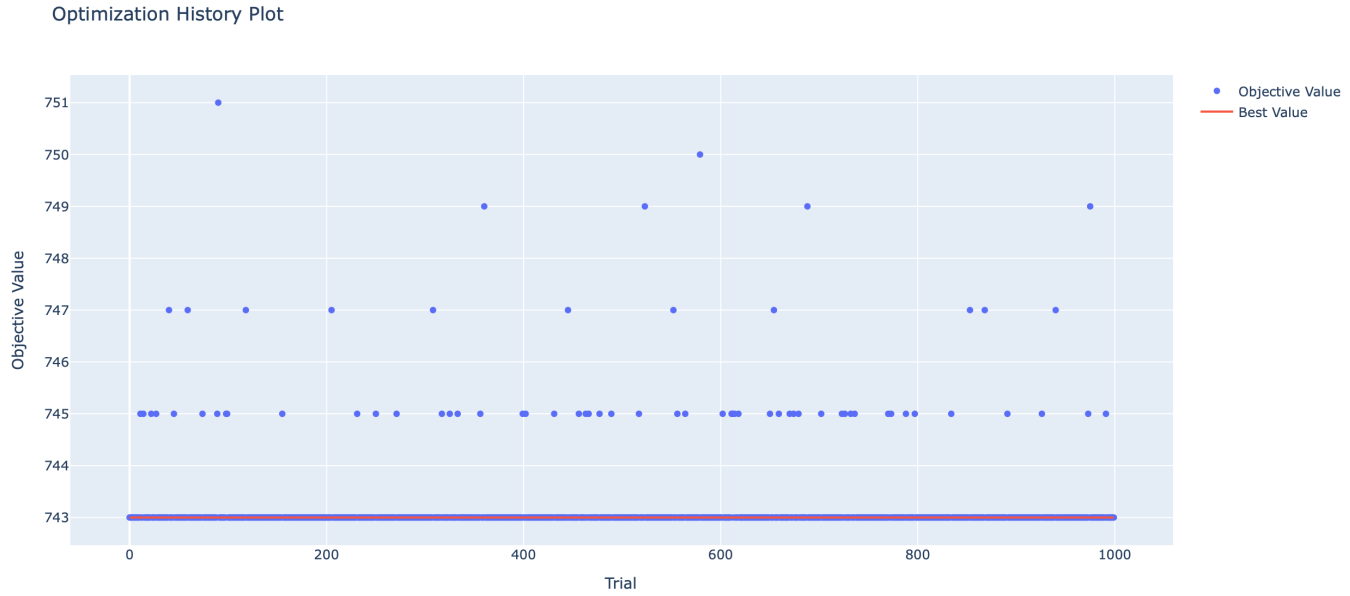
Aby dostosować parametry, tworzymy funkcję celu, która jest minimalizowana przez Optuna. Funkcja ta przyjmuje argument `trial`, który zawiera próbki wartości dla hiperparametrów. Funkcja celu wykorzystuje te parametry do uruchomienia algorytmu SA i zwrócenia wyniku w postaci wartości funkcji celu, którą Optuna stara się minimalizować.

5.2.2 Optymalizacja z użyciem Optuna

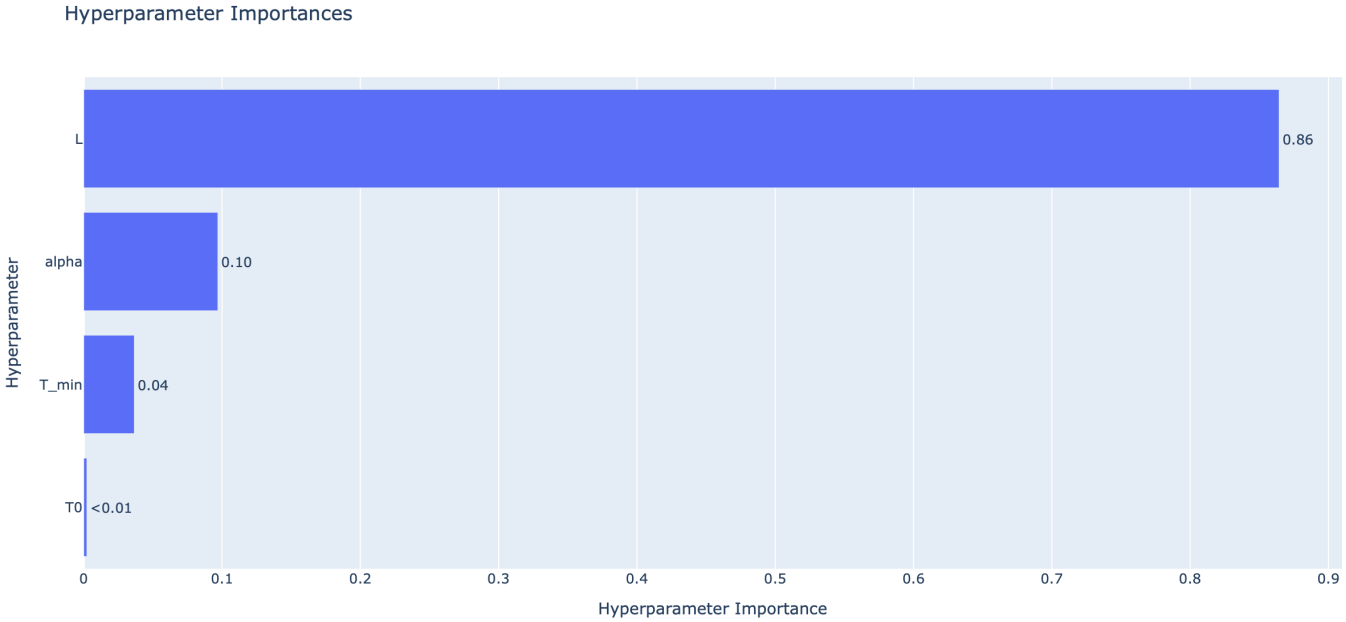
Po zdefiniowaniu funkcji celu, tworzymy obiekt `study`, który będzie zarządzał procesem optymalizacji. W tym przypadku Optuna będzie minimalizować funkcję celu przez 1000 prób, próbując różnych kombinacji wartości hiperparametrów.

5.3 Wizualizacja Wyników

Po zakończeniu procesu optymalizacji, Optuna umożliwia wizualizację wyników, co pozwala na lepsze zrozumienie, które hiperparametry mają największy wpływ na wynik algorytmu.



Rysunek 1: Historia optymalizacji



Rysunek 2: Ważność poszczególnych hiperparametrów w kontekście wyników optymalizacji

Pierwszy wykres pokazuje historię optymalizacji, a drugi wykres przedstawia ważność poszczególnych hiperparametrów w kontekście wyników optymalizacji.

5.4 Podsumowanie strojenia hiperparametrów

Optymalizacja hiperparametrów przy użyciu biblioteki Optuna pozwala na automatyczne dopasowanie wartości takich parametrów jak temperatura początkowa, współczynnik chłodzenia, liczba iteracji czy minimalna temperatura w algorytmie symulowanego wyżarzania. Dzięki temu możliwe jest uzyskanie lepszych wyników w krótszym czasie. Proces optymalizacji może być wizualizowany, co ułatwia analizę wpływu poszczególnych parametrów na wyniki algorytmu.

Najlepsze odnalezione hiperparametry:

- 'T0': 73.6418617987443,
- 'alpha': 0.9524385825925912,
- 'L': 55,
- 'T_min': 0.26818680277433227

Najlepsza odnaleziona wartość funkcji celu: 743.0, gdzie wykorzystaną instancją tsp była instancja o nazwie gr96.tsp ze znanej biblioteki tsplib.

6 Badania

W ramach przeprowadzonych badań dokonano porównania algorytmów: NN, FI i kilku wariantów SA. Przetestowano wariant SA startujący z permutacji losowej oraz jako algorytm poprawy dla algorytmów NN i FI.

6.1 Metodyka badań

- **Dane wejściowe:** Przeprowadzono testy na różnej wielkości instancjach tsp. Dane zostały wygenerowane losowo dla każdego powtórzenia. Dla SA zastosowano wcześniej znalezione hiperparametry.
- **Kryteria porównawcze:** Wynik funkcji celu znormalizowano według wzoru:

$$\frac{(A - A')}{A'},$$

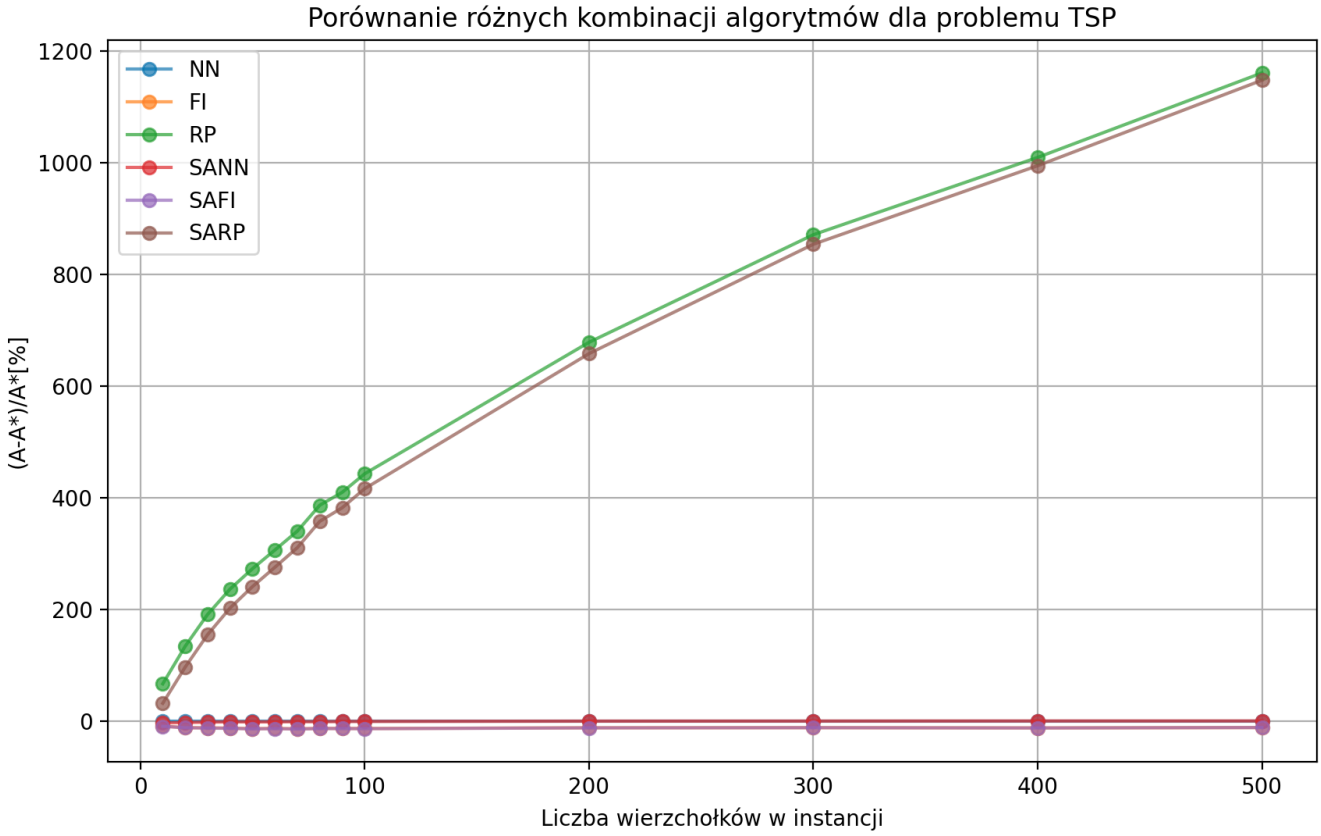
  gdzie:

  - A - wynik funkcji celu testowanego algorytmu,
  - A' - wynik funkcji celu algorytmu porównawczego, w tym przypadku NN.

  Wyniki uśredniono z 100 prób.
- **Środowisko testowe:** Pomiary przeprowadzono na laptopie MB Pro M1 2021.

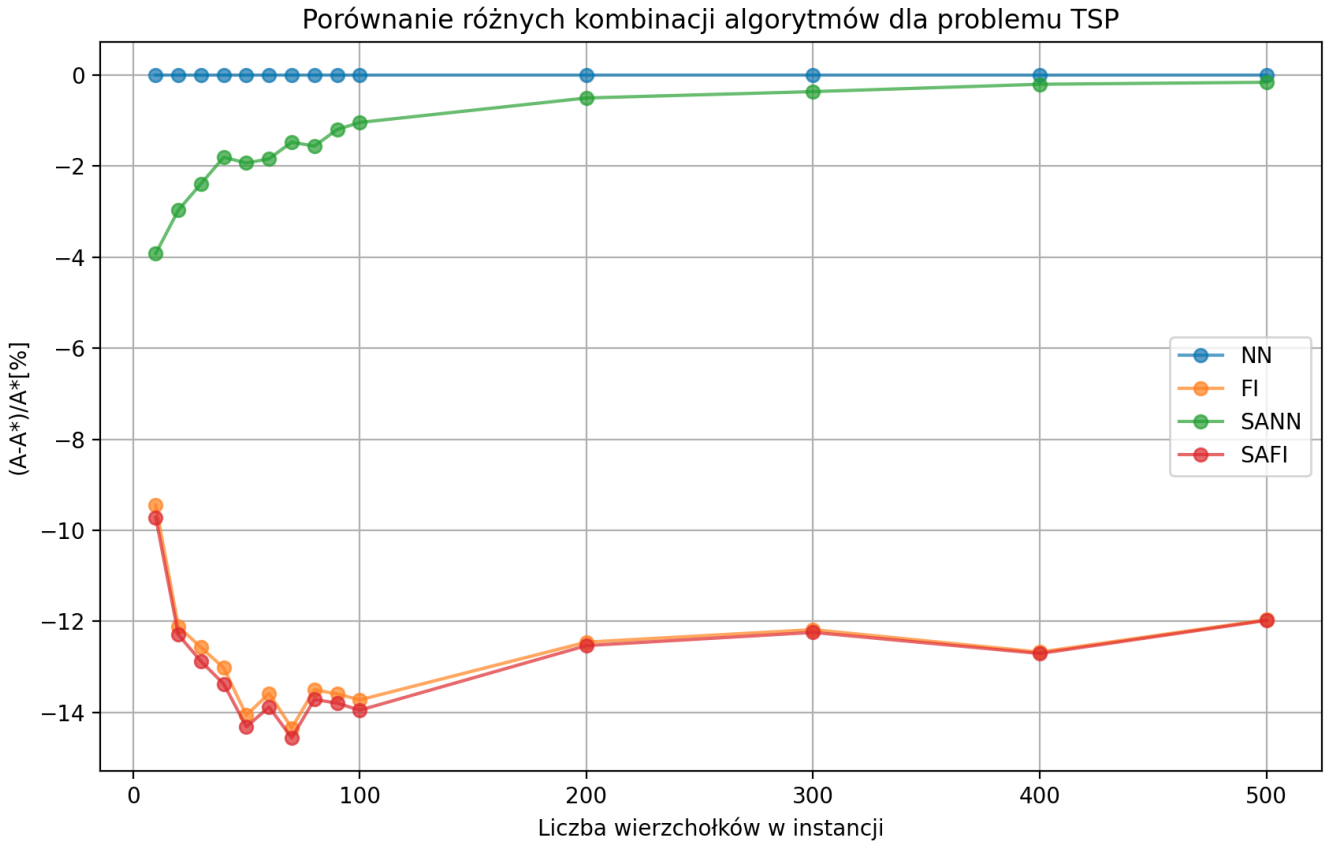
6.2 Wyniki

Na wykresie 3 widać wyniki znormalizowane wartości funkcji celu dla wszystkich algorytmów dla różnych rozmiarów instancji. Należy zauważyć, że wyniki dla SA startującego z permutacji losowej (SARP) znacznie odstają od pozostałych wartości.



Rysunek 3: Wyniki dla wszystkich algorytmów dla różnych rozmiarów instancji.

Na kolejnym wykresie nie przedstawiono wyników dla SARP oraz permutacji losowej (RP) w celu zwiększenia czytelności wykresu. Im mniejsza wartość tym lepiej. W tym badaniu widać, że algorytm FI oraz FI z SA jako algorytm poprawy sprawdziły się najlepiej.



Rysunek 4: Wyniki dla wybranych algorytmów dla różnych rozmiarów instancji.

7 Wnioski

- **Najlepsza wydajność algorytmu FI oraz FI z SA jako algorytm poprawy:** Z przeprowadzonych testów wynika, że algorytm FI oraz jego połączenie z algorytmem SA jako metodą poprawy (FI+SA) dały najlepsze wyniki w porównaniu z innymi algorytmami. Znormalizowane wyniki funkcji celu dla tych algorytmów były wyraźnie lepsze od wyników uzyskanych przez pozostałe algorytmy.

- **SA jako algorytm poprawy:** Algorytm symulowanego wyżarzania (SA) wykazał swoje zalety, gdy był zastosowany jako algorytm poprawy wyników uzyskanych przez inne algorytmy, takie jak FI i NN. W szczególności, po zastosowaniu SA jako metody poprawy, uzyskano lepsze wyniki niż w przypadku algorytmu NN, co sugeruje, że SA może efektywnie udoskonalić rozwiązania wygenerowane przez prostsze heurystyki.

- **Wydajność SA w zależności od rozmiaru instancji:** Jednym z ważniejszych spostrzeżeń jest fakt, że skuteczność algorytmu SA maleje wraz ze wzrostem rozmiaru instancji problemu TSP. Dla mniejszych instancji algorytm SA był w stanie znaleźć znacznie lepsze rozwiązania, jednak w przypadku większych instancji, czas obliczeniowy oraz trudność w optymalizacji powodowały, że SA nie był w stanie znaleźć rozwiązania o takiej jakości, jak dla mniejszych problemów. To może wskazywać na potrzebę dalszej optymalizacji samego algorytmu lub zastosowania bardziej zaawansowanych technik w przypadku dużych instancji.

- **Optymalizacja algorytmu dla większych instancji:** Wyniki sugerują, że konieczne może być zastosowanie dodatkowych strategii, takich jak zmiana parametrów SA (np. tempo chłodzenia, liczba iteracji na poziomie temperatury), aby poprawić jego efektywność w przypadku dużych problemów.

8 Źródła

- Materiały z wykładu i zajęć.

- Materiały dostarczone przez prowadzącego: http://mariusz.makuchowski.staff.iiar.pwr.wroc.pl