Projektowanie efektywnych algorytmów
Etap (2)

Asymetryczny problem komiwojażera

Jakub Kapłonek 263908

18 grudnia 2023

Spis treści

1 Wstęp Teoretyczny

1.1 Opis Problemu TSP

Problem Komiwojażera (TSP) polega na znalezieniu najkrótszej ścieżki przechodzącej przez wszystkie miasta dokładnie raz i wracającej do punktu początkowego. Jest to problem optymalizacyjny, który ma zastosowanie w wielu dziedzinach, takich jak logistyka czy trasowanie.

1.2 Cel projektu

Celem tego projektu jest wprowadzenie oraz ocena dwóch zaawansowanych metod heurystycznych: Tabu Search (TS) oraz Symulowane Wyżarzanie (SA), mających zastosowanie w efektywnym rozwiązaniu problemu Komiwojażera. Badania te mają na celu zanalizowanie skuteczności i efektywności tych algorytmów w kontekście optymalizacji tras podróży.

2 Algorytmy

2.1 Generacja sąsiadów

Oba algorytmy mają ceche wspólną którą jest potrzeba generowania rozwiązań znajdujących się w małej odległości od siebie (rozwiązania mało się od siebie różnią). W programie zaimplementowano trzy możliwości generowania sąsiadów.

1. Przez losową zamiane węzłów
2. Przez insercje losowego węzła w losowe miejsce
3. Przez odwrócenie losowej podścieżki

```
public class SwapRandomNodesNeighbors implements NeighborGeneration {
    @Override
    public List<List<Integer>> generateNeighbors(
        List<Integer> solution,
        int numberOfNeightbors
    ) {
        List<List<Integer>> neighbors = new ArrayList<>();
        Random random = new Random();

        for (int i = 0; i < numberOfNeightbors; i++) {
            int index1 = random.nextInt(solution.size());
            int index2 = random.nextInt(solution.size());

            List<Integer> neighbor = new ArrayList<>(solution);
            Collections.swap(neighbor, index1, index2);

            neighbors.add(neighbor);
        }

        return neighbors;
    }

    @Override
    public String getName() {
        return "Zamiana losowych miast";
    }
}
```

Rysunek 1: Metoda przez losową zamiane dwóch miast

2.2 Generacja początkowego rozwiązania

Generowanie rozwiązania początkowego stanowi kluczowy krok w algorytmach optymalizacyjnych, w tym także w przypadku Tabu Search. Proces ten ma istotny wpływ na jakość i skuteczność działania algorytmu. W kontekście Tabu Search, generowanie początkowego rozwiązania może odbywać się na kilka sposobów.

Jednym podejściem jest losowe wygenerowanie początkowego rozwiązania, co zapewnia różnorodność w przeszukiwanej przestrzeni. Innym podejściem jest zastosowanie heurystyki lub algorytmu przybliżonego do uzyskania początkowego rozwiązania, które może stanowić dobry punkt wyjścia dla procesu optymalizacji.

Kluczowe jest również dostosowanie generowania początkowego rozwiązania do specyfiki problemu. W przypadku problemów tras podróży, może to obejmować skonstruowanie początkowej trasy z uwzględnieniem lokalnych warunków czy ograniczeń.

Dobrze wygenerowane początkowe rozwiązanie może skrócić czas zbiegania algorytmu do optymalnego lub satysfakcjonującego rozwiązania. Jednakże, proces generowania początkowego rozwiązania wymaga uwzględnienia kontekstu danego problemu oraz cech przeszukiwanej przestrzeni rozwiązań, aby zapewnić skuteczność działania algorytmu Tabu Search.

W programie zaimplementowano generowanie rozwiązania początkowego metodą zachłanną oraz losową

```
public class GreedyFirstSolutionGeneration implements FirstSolutionGeneration {
    @Override
    public List<Integer> generateSolution(Matrix matrix) {
        ArrayList<Integer> solution = new ArrayList<>();
        int[][] costMatrix = matrix.getDistanceMatrix();

        solution.add(0);
        for(int i = 0 ; i < costMatrix.length - 1; i++) {
            int bestBranch = Integer.MAX_VALUE;
            int bestIndex = -1;
            for (int j = 0; j < costMatrix.length; j++) {
                if (
                    !solution.contains(j) &&
                    bestBranch > costMatrix[solution.get(solution.size() - 1)][j]
                ) {
                    bestBranch = costMatrix[solution.get(solution.size() - 1)][j];
                    bestIndex = j;
                }
            }
            solution.add(bestIndex);
        }

        return solution;
    }
}
```

Rysunek 2: Metoda zachłanna

2.3 Tabu search

Algorytm Tabu Search to zaawansowana metoda optymalizacyjna stosowana do rozwiązywania problemów optymalizacyjnych. Jego główną ideą jest przeszukiwanie przestrzeni rozwiązań, przy czym unika się powtarzania pewnych ruchów, co zapobiega utknięciu w lokalnych optimum.

Tabu Search wykorzystuje listę tabu do śledzenia niedozwolonych ruchów, co sprawia, że algorytm ma zdolność eksploracji różnych obszarów przestrzeni rozwiązań. Poprzez iteracyjne przeszukiwanie sąsiadów, algorytm stara się znaleźć globalne optimum w skomplikowanych problemach optymalizacyjnych.

Algorytm jest szeroko stosowany w różnych dziedzinach, takich jak planowanie tras, optymalizacja układów produkcyjnych czy rozkładanie zadań. Jego skuteczność wynika z równowagi między intensyfikacją, czyli dążeniem do znalezienia najlepszego rozwiązania w danym obszarze, a dywersyfikacją, która pomaga unikać utknięcia w lokalnych optimum.

2.3.1 Lista tabu

W algorytmie Tabu Search lista tabu jest wykorzystywana do monitorowania i kontrolowania ruchów w przestrzeni rozwiązań. Każdy wykonany ruch jest tymczasowo uznawany za niedozwolony przez określony czas lub liczbę iteracji, co zapobiega powtarzaniu tych samych kroków. To podejście umożliwia algorytmowi unikanie lokalnych optimum poprzez zmuszanie go do eksplorowania różnych obszarów przestrzeni rozwiązań, co przyczynia się do bardziej efektywnego znajdowania optymalnych rozwiązań problemu optymalizacyjnego.

W programie lista tabu została stworzona jako lista list wierzchołków, gdzie jedna lista wierzchołków tworzy jedno rozwiązanie problemu TSP. Przy każdej generacji sąsiadów wybiera się takich którzy nie znajdują sie na liście tabu, oraz takich którzy

2.3.2 Dywersyfikacja

W algorytmie Tabu Search, dywersyfikacja odgrywa kluczową rolę w zapobieganiu utknięciu w lokalnych optimum poprzez zachęcanie do eksploracji różnorodnych obszarów przestrzeni rozwiązań. Mechanizmy dywersyfikacyjne, takie jak lista tabu, wprowadzają zakazy na powtarzanie pewnych ruchów przez określony czas lub liczbę iteracji. To zmusza algorytm do poszukiwania alternatywnych ścieżek i pomaga unikać cyklicznego powtarzania tych samych kroków. Dywersyfikacja jest szczególnie istotna w przypadku problemów optymalizacyjnych, gdzie istnieje wiele lokalnych optimum, a skuteczna eksploracja przestrzeni rozwiązań jest kluczowa dla znalezienia globalnie optymalnego rozwiązania. Algorytm Tabu Search osiąga sukces, utrzymując równowagę między intensyfikacją, czyli skupieniem się na aktualnie najlepszym rozwiązaniu, a dywersyfikacją, która zachęca do poszukiwania nowych, potencjalnie lepszych ścieżek.

W programie zaimplementowano dywersyfikacje poprzez wygenerowanie sąsiada, pomijając liste tabu.

2.4 Symulowane wyżarzanie

Symulowane Wyżarzanie (SA) to metaheurystyczna technika optymalizacyjna, zainspirowana procesem fizycznego wyżarzania metali. Algorytm ten został skonstruowany w taki sposób, aby skutecznie przeszukiwać przestrzeń rozwiązań, także w przypadku problemów optymalizacyjnych trudnych do rozwiązania tradycyjnymi metodami.

Główną ideą Symulowanego Wyżarzania jest symulowanie procesu wyżarzania metalu, w którym stopniowe schładzanie stopu prowadzi do zmniejszenia energii. W kontekście optymalizacji, temperatura reprezentuje poziom akceptacji gorszych rozwiązań, umożliwiając algorytmowi unikanie utknięcia w lokalnych optimum.

Podczas działania SA, rozwiązania są iteracyjnie modyfikowane na podstawie temperatury, co pozwala na akceptację gorszych rozwiązań z pewnym prawdopodobieństwem. W miarę postępu algorytmu temperatura maleje, co zmniejsza szanse na akceptację gorszych rozwiązań, skupiając się na poszukiwaniu optymalnego rozwiązania.

Algorytm Symulowanego Wyżarzania znajduje zastosowanie w różnych dziedzinach, takich jak planowanie tras, optymalizacja grafów czy projektowanie układów energetycznych. Jego zdolność do eksploracji przestrzeni rozwiązań oraz znajdowania globalnych optimum sprawiają, że jest ceniony w rozwiązywaniu problemów optymalizacyjnych.

2.4.1 Temperatura początkowa

Wartość temperatury początkowej istotnie wpływa na inicjalną zdolność algorytmu do badania przestrzeni rozwiązań. W naszym zaimplementowanym algorytmie proces określania tej temperatury polega na wielokrotnym przeglądaniu przestrzeni rozwiązań w celu oceny różnicy kosztu pomiędzy wybraną próbką a jej sąsiadem. Ten proces ma na celu dokładne zobrazowanie krajobrazu przestrzeni rozwiązań. Ostateczna temperatura jest ustalana na podstawie mediany zebranych różnic kosztów, wykorzystując wzór:

$$T_0 = -\frac{\Delta}{\ln P} \tag{1}$$

```
private double calculateInitialTemperature(
    List<Integer> initialSolution,
    int[][] distanceMatrix
) {
    int numSteps = 1000;
    double totalDeltaCost = 0.0;

    for (int i = 0; i < numSteps; i++) {
        List<Integer> newSolution = neighborGeneration
                .generateNeighbors(initialSolution, 1)
                .get(0);

        double currentCost = calculateTotalDistance(initialSolution, distanceMatrix);
        double newCost = calculateTotalDistance(newSolution, distanceMatrix);
        totalDeltaCost += Math.abs(newCost - currentCost);
    }

    double averageDeltaCost = totalDeltaCost / numSteps;
    return -averageDeltaCost / Math.log(0.99); //wartość dobrana empirycznie
}
```

Rysunek 3: Obliczanie początkowej temperatury

gdzie:
δ - wartość mediany różnic kosztów ścieżek,
P - poziom akceptacji gorszego rozwiązania dostosowany eksperymentalnie.

W poniższym fragmencie kodu obliczamy ostateczną temperaturę, wykorzystując powyższy wzór.

2.4.2 Współczynnik schładzania

W algorytmie Symulowanego Wyżarzania (SA), współczynnik schładzania jest kluczowym parametrem, który wpływa na proces akceptacji gorszych rozwiązań w celu uniknięcia utknięcia w lokalnych optimum. W miarę postępu algorytmu, temperatura systematycznie maleje, co odzwierciedla stopniowe wygaszanie procesu "wyżarzania". Współczynnik schładzania determinuje szybkość, z jaką temperatura maleje, a zatem wpływa na prawdopodobieństwo akceptacji gorszych rozwiązań w późniejszych iteracjach. Optymalny dobór tego parametru jest kluczowy dla efektywnego działania algorytmu, umożliwiając równowagę między intensyfikacją poszukiwań w początkowej fazie a dywersyfikacją w późniejszych etapach, co sprzyja znalezieniu globalnego optimum. W praktyce, dostosowanie współczynnika schładzania jest często kwestią doświadczalną i zależy od charakterystyki konkretnego problemu optymalizacyjnego.

3 Strategia testowania

3.1 Tabu search

Dla algorytmu tabu search zostały przeprowadzone testy dla 3 plików: ftv47.atsp, ftv170.atsp, rbg403.atsp, każdy z plików został przepuszczony przez algorytm 10 razy, te operację będą powtórzone dla wszystkich metod generacji sąsiadów, z wygenerowanych danych zostały wykonane porównanie różnych metod generacji.

3.2 Symulowane wyżarzanie

Dla algorytmu SA zostały przeprowadzone testy dla 3 plików: ftv47.atsp, ftv170.atsp, rbg403.atsp, każdy z plików został przepuszczony przez algorytm 10 razy, te operację będą powtórzone dla trzech

współczynników schładzania - 0.99, 0.999 oraz 0.9999, z wygenerowanych danych zostały wykonane porównanie różnych metod generacji.

3.3 Ogólne

Zgodnie z poleceniem czas stopu dla plików został przyjęty następująco: ftv47 - 2min, ftv170 - 4min, rbg403 - 6min. Po zakończeniu serii uruchomień algorytmów, dla uzyskanych wyników został obliczony błąd względny przy użyciu następującego wzoru:

$$relative\_error = \frac{|f_{zn} - f_{opt}|}{f_{opt}} \tag{2}$$

gdzie:
f_{zn} wartość otrzymana przez algorytm.
f_{opt} najlepsze znane rozwiązanie dla danego pliku

W poniższej tabeli zostały zawarte najlepsze znane rozwiązania dla każdego z plików.

Plik	Najmniejszy koszt
ftv47.atsp	1776
ftv170.atsp	2755
rbg403.atsp	2465

Tabela 1: Najlepsze znane rozwiązania dla danego pliku.

W trakcie testów algorytm systematycznie monitorował oraz rejestrował najlepsze koszty ścieżek wraz z czasem, w którym te optymalne rozwiązania zostały osiągnięte. Poniżej przedstawione są zebrane dane, dokumentujące kroki algorytmów w kierunku uzyskiwania optymalnych rozwiązań dla konkretnych plików wejściowych i rodzajów sąsiedztwa.

4 Wyniki

4.1 Tabele z wynikami

4.1.1 Uśrednione wyniki

Plik	Metoda generowania sąsiadów	Średni wynik	Średni czas wyszukiwania	Średni błąd względny
ftv47.atsp	Zamiana wierzchołków	1798,4	72099,5	1,26
ftv47.atsp	Insercja wierzchołka	1776	6629,3	0
ftv47.atsp	Odwrócenie poddrogi	1985,5	47541,4	11,80
ftv170.atsp	Zamiana wierzchołków	3577,2	17252,4	29,84
ftv170.atsp	Insercja wierzchołka	3123,4	78511,1	13,37
ftv170.atsp	Odwrócenie poddrogi	3884,4	354,5	40,99
rbg403.atsp	Zamiana wierzchołków	2516,9	196362,9	2,11
rbg403.atsp	Insercja wierzchołka	2486,5	185775,6	0,87
rbg403.atsp	Odwrócenie poddrogi	3430,6	113605,9	39,17

Tabela 2: Tabu Search: Wyniki uśrednione

Z uśrednionych wyników możemy wywnioskować, że dla różnych plików algorytm TabuSearch radzi sobie w różny sposób, gdzie najgorzej wypada plik ftv47.atsp, można również zauważyć że najlepszą w kwestii błędu względnego dla każdego pliku okazała się metoda generacji sąsiadów poprzez insercje wierzchołka, a najgorszą, przez odwrócenie losowej podścieżki.

Plik	Współczynnik schładzania	Średni wynik	Średni czas wyszukiwania	Średni błąd względny
ftv47.atsp	0.99	2017,7	288,3	13,61
ftv47.atsp	0.999	1917,4	2651,6	7,96
ftv47.atsp	0.9999	1820,9	26551,9	2,53
ftv47.atsp	0.99	4400,5	437,6	59,73
ftv47.atsp	0.999	3941,7	8405,9	43,07
ftv47.atsp	0.9999	3444,8	80392,2	25,04
ftv47.atsp	0.99	2519,9	118239,5	59,73
ftv47.atsp	0.999	2490,1	144764,2	43,07
ftv47.atsp	0.9999	2472,9	80392,2	25,04

Tabela 3: Symulowane wyżarzanie: Wyniki uśrednione

Z tabeli uśrednionych wyników dla symulowanego wyżarzania możemy odczytać najlepszy współczynnik schładzania. Najlepszy okazuje się jak najwolniejszy współczynnik czyli 0,9999 gdzie 0,99 wypada najgorzej.

4.1.2 Najlepsze wyniki

Plik	Metoda generowania sąsiadów	Najlepszy wynik	Czas wyszukiwania najlepszego wyniku	Najlepszy błąd względny
ftv47.atsp	Zamiana sąsiadów	1786	52733	0,56
ftv47.atsp	Insercja sąsiada	1776	851	0
ftv47.atsp	Odwrócenie poddrogi	1946	43664	9,57
ftv170.atsp	Zamiana sąsiadów	3449	17882	25,19
ftv170.atsp	Insercja sąsiada	3029	128615	9,95
ftv170.atsp	Odwrócenie poddrogi	3847	2040	39,64
rbg403.atsp	Zamiana sąsiadów	2499	227081	1,38
rbg403.atsp	Insercja sąsiada	2480	174179	0,61
rbg403.atsp	Odwrócenie poddrogi	3406	70934	38,17

Tabela 4: Tabu Search: Najlepsze wyniki

Po tabeli najlepszych wyników widać czysto po raz kolejny najlepszą metode generacji sąsiadów jaką jest Insercja losowego wierzchołka, i adekwatnie najgorszą metode poprzez odwrócenie podścieżki

Najlepsze ścieżki dla plików:

ftv47.atsp:
35, 14, 15, 16, 45, 39, 19, 44, 21, 40, 47, 26, 42, 28, 3, 24, 4, 29, 30, 31, 5, 6, 8, 11, 10, 0, 25, 1, 9, 33, 27, 2, 41, 43, 22, 20, 37, 38, 18, 17, 12, 32, 7, 23, 34, 13, 46, 36

ftv170.atsp:

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

rbg403.atsp:
133, 196, 0, 193, 38, 127, 333, 267, 317, 23, 14, 62, 13, 205, 204, 142, 32, 274, 327, 33, 376, 135, 270, 19, 18, 402, 287, 107, 61, 281, 304, 394, 225, 58, 8, 6, 64, 3, 2, 386, 47, 112, 251, 322, 272, 28, 340, 182, 202, 151, 11, 177, 131, 150, 145, 102, 95, 55, 85, 106, 359, 152, 310, 29, 232, 217, 9, 397, 260, 312, 35, 108, 93, 84, 77, 31, 52, 40, 39, 78, 226, 147, 81, 22, 21, 360, 82, 247, 249, 86, 101, 153, 75, 169, 27, 170, 284, 184, 4, 79, 96, 66, 60, 44, 166, 67, 94, 88, 353, 263, 90, 72, 57, 163, 25, 130, 92, 51, 49, 37, 36, 301, 120, 392, 351, 395, 87, 56, 180, 165, 364, 303, 71, 91, 139, 239, 70, 387, 349, 73, 253, 99, 389, 111, 369, 373, 355, 115, 114, 69, 246, 244, 80, 307, 7, 213, 41, 201, 118, 352, 124, 269, 372, 278, 122, 119, 168, 242, 117, 128, 255, 209, 356, 104, 384, 383, 34, 361, 146, 144, 208, 105, 391, 154, 136, 207, 45, 374, 363, 289, 265, 275, 210, 109, 326, 258, 254, 223, 214, 192, 189, 162, 48, 264, 358, 83, 335, 125, 74, 216, 200, 198, 126, 10, 331, 323, 305, 215, 309, 252, 191, 187, 42, 234, 228, 224, 167, 240, 219, 237, 285, 282, 357, 328, 46, 241, 113, 293, 318, 132, 243, 231, 288, 97, 175, 291, 315, 161, 148, 325, 388, 134, 306, 319, 195, 280, 279, 336, 292, 329, 294, 110, 236, 227, 381, 188, 173, 344, 141, 116, 24, 54, 238, 229, 393, 137, 50, 149, 347, 401, 354, 342, 266, 286, 273, 257, 190, 100, 98, 178, 138, 297, 332, 296, 65, 171, 63, 337, 268, 16, 176, 156, 390, 181, 346, 338, 129, 155, 143, 159, 378, 398, 396, 350, 233, 324, 320, 157, 339, 140, 334, 158, 1, 316, 256, 314, 59, 311, 271, 370, 330, 290, 103, 276, 298, 248, 348, 343, 160, 15, 366, 321, 379, 230, 382, 212, 211, 194, 380, 362, 218, 203, 295, 179, 368, 183, 17, 12, 123, 399, 377, 164, 365, 172, 26, 174, 185, 186, 53, 400, 222, 250, 206, 30, 245, 220, 197, 221, 300, 235, 121, 262, 261, 68, 302, 89, 283, 277, 341, 43, 308, 5, 313, 20, 345, 299, 259, 367, 76, 375, 371, 385, 199

Plik	Współczynnik schładzania	Najlepszy wynik	Czas wyszukiwania najlepszego wyniku	Najleszy błąd względny
ftv47.atsp	0.99	1946	281	9,57
ftv47.atsp	0.999	1860	2822	4,73
ftv47.atsp	0.9999	1789	26101	0,73
ftv170.atsp	0.99	4104	3	48,97
ftv170.atsp	0.999	3699	8296	34,26
ftv170.atsp	0.9999	3224	81192	17,02
rbg403.atsp	0.99	2489	155191	0,97
rbg403.atsp	0.999	2474	216246	0,36
rbg403.atsp	0.9999	2465	211685	0

Tabela 5: Symulowane wyżarzanie: Najlepsze wyniki

Pomimo złych jakby się wydawało wyników uśrednionych, w najlepszym przejściu algorytm symulowanego wyżarzania zdołał osiągnąć dobre wyniki dla plików ftv47 oraz rbg403, gdzie dla pliku rbg zostało znalezione optimum, po raz kolejny widać najlepszy współczynnik schładzania na podsawie wyników błędu względnego

Najlepsze ścieżki dla plików:

ftv47.atsp:
18, 17, 13, 46, 36, 35, 14, 34, 23, 12, 32, 7, 31, 30, 5, 24, 4, 29, 3, 6, 8, 11, 10, 0, 25, 47, 26, 1, 9, 33, 27, 28, 2, 41, 43, 42, 22, 19, 44, 15, 16, 45, 39, 21, 40, 20, 38, 37

ftv170.atsp
91, 94, 165, 163, 99, 100, 102, 103, 104, 114, 164, 130, 131, 132, 111, 110, 109, 107, 106, 105, 98, 95, 96, 97, 113, 127, 126, 125, 129, 128, 135, 136, 137, 138, 139, 140, 141, 134, 6, 152, 142, 149, 161, 160, 14, 151, 7, 8, 9, 10, 76, 74, 75, 11, 12, 13, 18, 19, 29, 30, 28, 27, 26, 23, 24, 15, 159, 16, 17, 32, 158, 36, 157, 33, 31, 155, 41, 42, 45, 47, 48, 49, 170, 168, 72, 73, 77, 78, 82, 79, 80, 81, 0, 1, 2, 3, 4, 5, 133, 169, 112, 115, 116, 117, 118, 119, 122, 101, 123, 162, 120, 121, 124, 146, 145, 144, 143, 147, 148, 150, 25, 22, 21, 20, 37, 38, 39, 35, 34, 156, 40, 44, 46, 59, 61, 68, 67, 167, 70, 69, 83, 84, 71, 60, 50, 51, 52, 53, 43, 55, 54, 58, 57, 62, 66, 63, 64, 56, 65, 153, 87, 85, 86, 93, 166, 108, 92, 154, 88, 89, 90

rbg403.atsp
200, 122, 40, 263, 166, 67, 189, 77, 156, 154, 174, 250, 402, 287, 272, 278, 266, 239, 117, 307, 143, 361, 146, 381, 23, 225, 150, 349, 400, 179, 368, 245, 201, 89, 196, 0, 105, 391, 210, 106, 359, 129, 110, 275, 258, 217, 384, 383, 84, 142, 97, 91, 286, 60, 44, 136, 176, 31, 29, 318, 158, 1, 285, 282, 357, 358, 46, 345, 173, 214, 192, 66, 273, 207, 45, 141, 116, 213, 28, 93, 233, 380, 204, 79, 62, 242, 70, 38, 19, 18, 335, 332, 37, 54, 132, 243, 88, 96, 114, 69, 231, 306, 168, 374, 363, 315, 167, 36, 124, 269, 33, 376, 68, 385, 300, 340, 237, 264, 113, 317, 64, 365, 130, 190, 100, 98, 140, 334, 183, 17, 12, 178, 138, 377, 311, 271, 350, 125, 74, 52, 119, 319, 367, 75, 222, 185, 81, 22, 21, 372, 356, 202, 32, 274, 163, 3, 331, 323, 73, 137, 13, 205, 379, 155, 7, 161, 148, 147, 208, 276, 49, 215, 216, 342, 398, 396, 83, 206, 30, 232, 134, 288, 218, 203, 43, 34, 157, 339, 65, 236, 227, 212, 182, 252, 371, 370, 328, 257, 92, 347, 341, 95, 55, 171, 63, 297, 82, 247, 249, 211, 313, 20, 253, 375, 191, 187, 169, 27, 172, 26, 364, 303, 337, 160, 15, 240, 219, 336, 292, 290, 220, 197, 230, 226, 325, 388, 373, 246, 283, 8, 6, 397, 5, 316, 235, 121, 387, 145, 244, 80, 24, 238, 229, 14, 353, 39, 177, 265, 326, 390, 194, 324, 320, 180, 165, 262, 261, 149, 51, 298, 291, 289, 139, 4, 251, 322, 107, 386, 47, 330, 221, 199, 104, 9, 42, 131, 355, 344, 111, 369, 127, 184, 90, 72, 224, 128, 255, 103, 360, 78, 382, 144, 50, 126, 10, 152, 76, 333, 267, 293, 99, 389, 87, 56, 312, 35, 304, 394, 393, 308, 260, 209, 41, 133, 123, 399, 57, 175, 234, 228, 71, 162, 48, 241, 112, 2, 61, 327, 102, 301, 256, 135, 270, 86, 101, 186, 53, 309, 109, 151, 11, 302, 118, 254, 223, 305, 296, 295, 248, 348, 343, 366, 321, 362, 268, 16, 195, 280, 279, 108, 120, 392, 351, 395, 188, 299, 259, 85, 159, 378, 198, 314, 59, 164, 25, 401, 115, 94, 193, 181, 346, 338, 170, 284, 329, 294, 281, 58, 277, 352, 153, 310, 354

4.2 Porównanie najlepszych wyników algorytmów

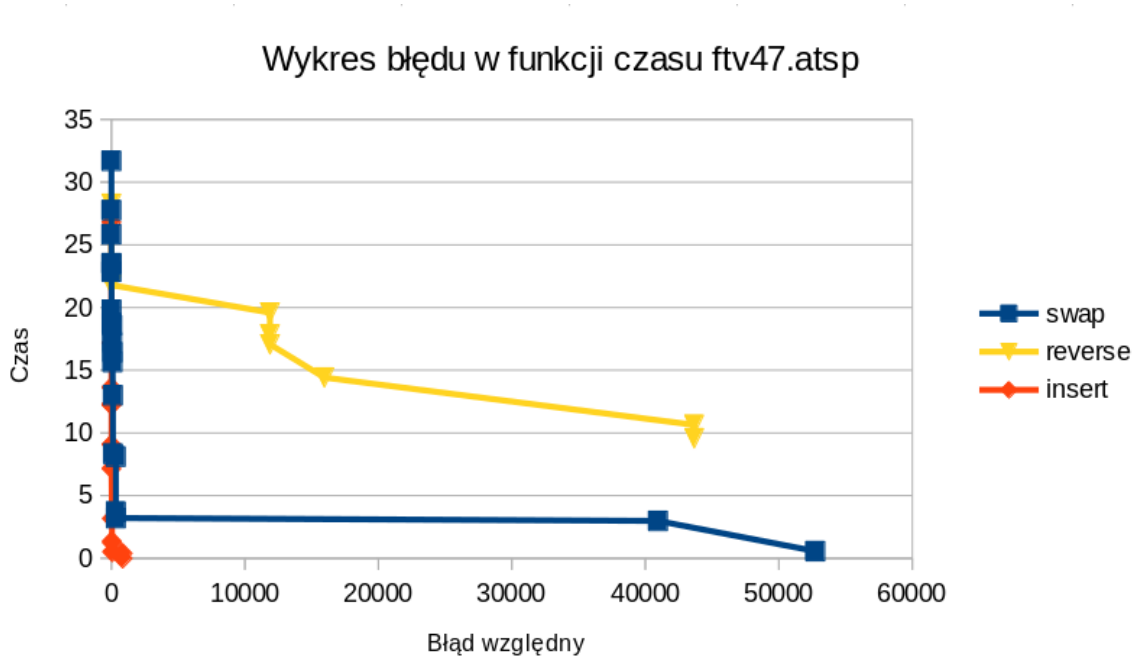
Plik	Tabu Search	Symulowane wyżarzanie
ftv47.atsp	1776 (0%)	1789 (0,73%)
ftv170.atsp	3029 (9,95%)	(17,02%)
rbg403.atsp	2480 (0,61%)	(0%)

Tabela 6: Porównanie najlepszych wyników z działania algorytmów TS oraz SA

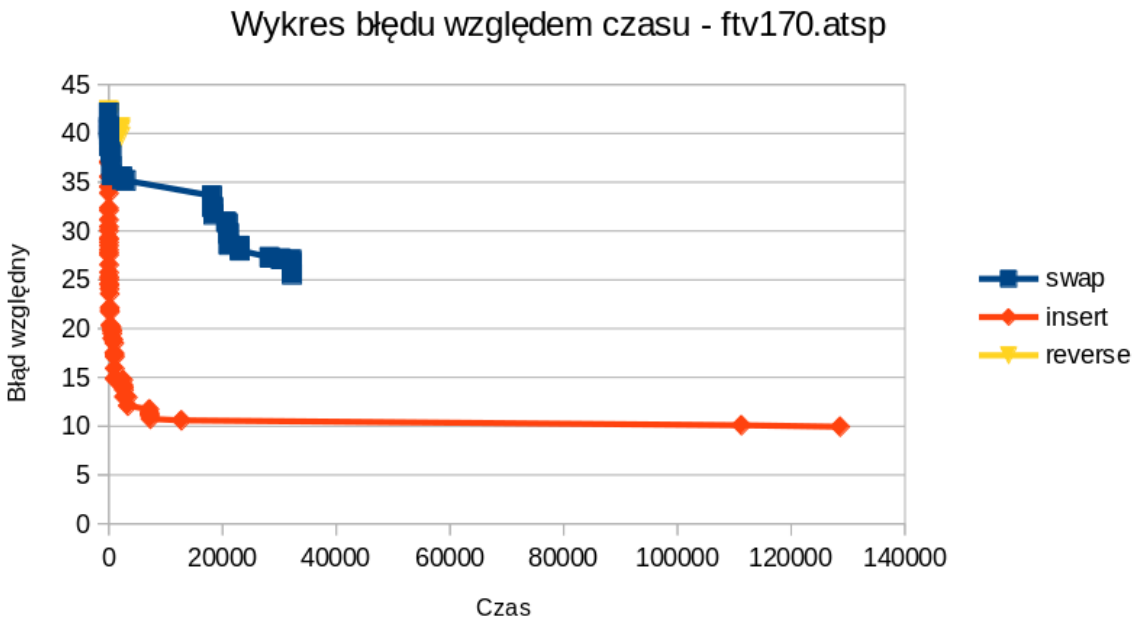
Porównując wyniki możemy zauważyć że obydwa algorytmy poradziły sobie dobrze ze znajdowaniem ścieżki w tych plikach, obydwa przynajmniej w jednym pliku znalazły optimum, w dwóch plikach były bardzo blisko optimum, błąd względny wynosił $<1\%$, problematycznym plikiem okazał się ftv170.atsp, dla którego błąd względny był wysoki porównując do pozostałych plików, dla niego lepiej poradził sobie algorytm Tabu Search.

4.3 Wykres błędu względnego w funkcji czasu

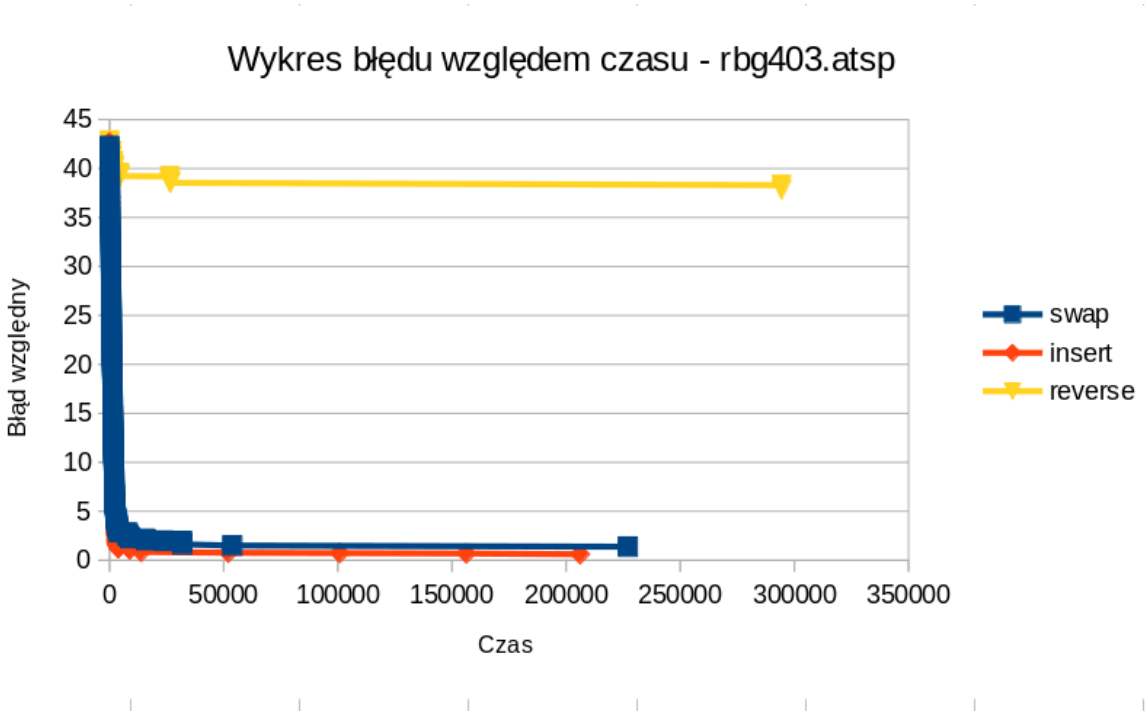
Na podstawie wykresów możemy ocenić jak algorytm stopniowo znajdował najlepsze rozwiązanie oraz w którym momencie zostało ono znalezione, sygnalizuje to zakończenie wykresu i brak dalszego polepszenia błędu względnego.



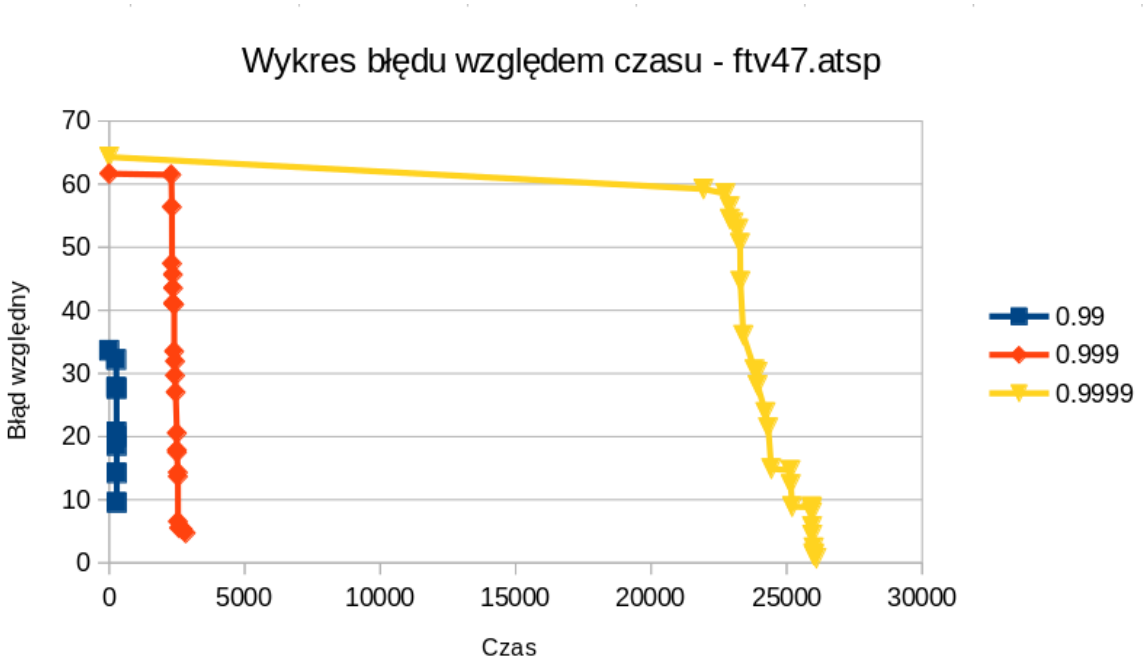
Rysunek 4: Tabu Search: ftv47.atsp



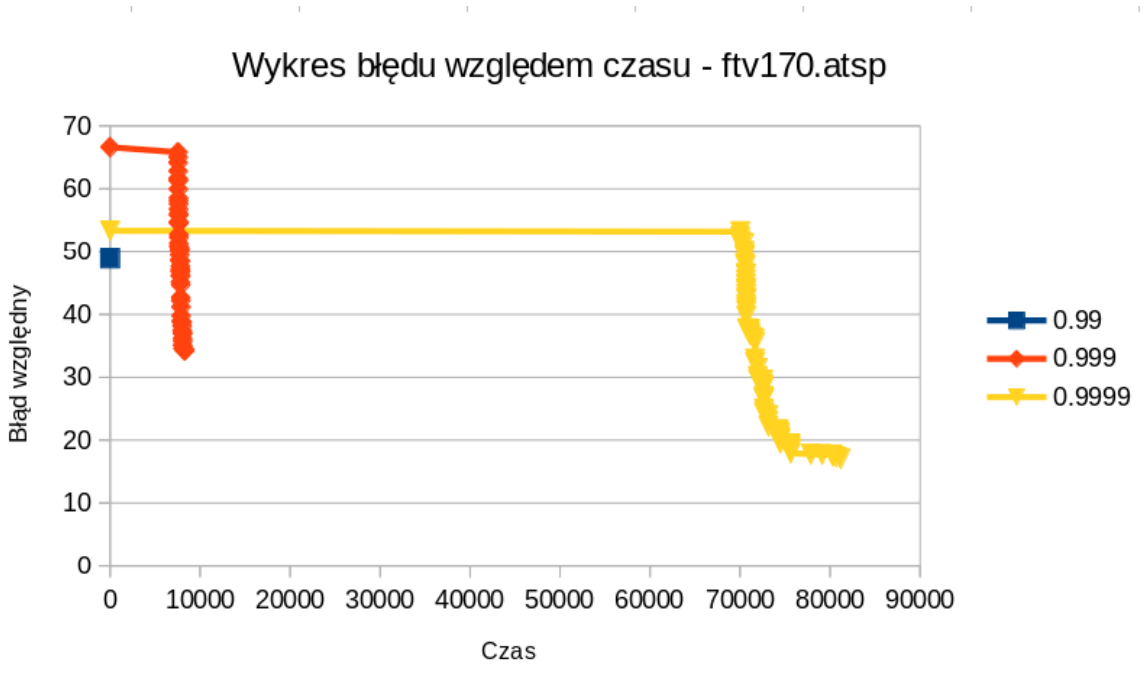
Rysunek 5: Tabu Search: ftv170.atsp



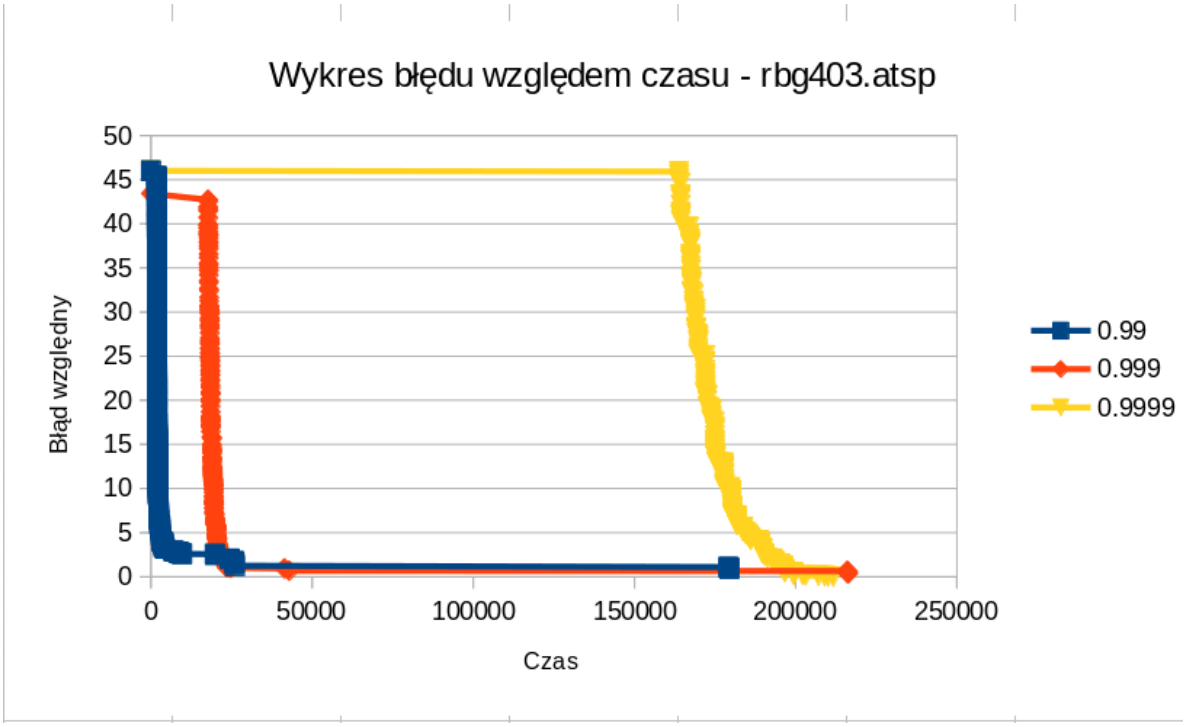
Rysunek 6: Tabu Search: rbg403.atsp



Rysunek 7: Symulowane Wyżarzanie: ftv47.atsp



Rysunek 8: Symulowane Wyżarzanie: ftv170.atsp



Rysunek 9: Symulowane Wyżarzanie: rbg403.atsp

4.4 Dokładne wyniki dla każdego uruchomienia

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	1795	110321	1,07
2	1786	52733	0,57
3	1797	59620	1,18
4	1795	83888	1,06
5	1807	44126	1,75
6	1807	9648	1,75
7	1798	91612	1,24
8	1805	97633	1,63
9	1790	59009	0,79
10	1804	112405	1,57

Tabela 7: : Zamiana losowych miast - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	3586	8914	30,16
2	3621	6291	31,43
3	3508	10723	27,33
4	3459	32203	25,55
5	3688	36297	33,86
6	3449	17882	25,19
7	3667	9498	33,10
8	3617	26554	31,29
9	3554	6535	29,00
10	3623	17627	31,50

Tabela 8: Tabu Search: Zamiana losowych miast - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2516	196019	2,07
2	2499	227081	1,38
3	2503	307161	1,54
4	2521	154284	2,27
5	2514	219289	1,99
6	2522	116651	2,31
7	2542	212881	3,12
8	2518	125827	2,15
9	2513	113840	1,94
10	2521	290596	2,27

Tabela 9: Tabu Search: Zamiana losowych miast - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	1776	6347	0
2	1776	3769	0
3	1776	11884	0
4	1776	9519	0
5	1776	851	0
6	1776	7083	0
7	1776	21514	0
8	1776	1516	0
9	1776	957	0
10	1776	2853	0

Tabela 10: Tabu Search: Insercja losowego miasta - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	3185	6728	15,61
2	3249	5229	17,93
3	3069	4729	11,39
4	3266	115958	18,55
5	3029	128615	9,95
6	3072	112953	11,51
7	3200	53487	16,15
8	3052	49557	10,78
9	3073	112836	11,54
10	3039	195019	10,31

Tabela 11: Tabu Search: Insercja losowego miasta - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2495	48252	1,22
2	2491	104242	1,05
3	2487	274919	0,89
4	2486	118606	0,85
5	2481	161917	0,65
6	2487	287317	0,89
7	2489	294763	0,97
8	2480	206174	0,61
9	2480	174179	0,61
10	2489	187387	0,97

Tabela 12: Tabu Search: Insercja losowego miasta - rbg403.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2036	62339	14,64
2	2026	48963	14,07
3	1946	43664	9,57
4	1953	42414	9,97
5	2011	60483	13,23
6	1952	37847	9,91
7	1996	36139	12,39
8	1973	14137	11,09
9	1996	59795	12,39
10	1966	69633	10,70

Tabela 13: Tabu Search: Odwrócenie losowej ścieżki - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	3859	1401	40,07
2	3901	65	41,59
3	3887	4	41,09
4	3887	4	41,09
5	3887	6	41,09
6	3887	6	41,09
7	3901	4	41,60
8	3901	7	41,60
9	3847	2040	39,64
10	3887	8	41,09

Tabela 14: Tabu Search: Odwrócenie losowej ścieżki - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	3406	70934	38,17
2	3452	96627	40,04
3	3410	125656	38,34
4	3398	294456	37,85
5	3478	10353	41,10
6	3410	330106	38,34
7	3445	12078	39,76
8	3423	115498	38,86
9	3481	60660	41,22
10	3403	19691	38,05

Tabela 15: Tabu Search: Odwrócenie losowej ścieżki - rbg403.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2051	328	
2	1992	279	
3	2069	288	
4	1946	281	
5	1993	275	
6	1984	283	
7	2049	294	
8	2074	287	
9	1995	280	
10	2024	288	

Tabela 16: Symulowane wyżarzanie: 0.99 - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	4559	1418	
2	4174	2	
3	4226	1278	
4	4443	4	
5	4110	3	
6	4578	2	
7	4544	3	
8	4631	1660	
9	4636	3	
10	4104	3	

Tabela 17: Symulowane wyżarzanie: 0.99 - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2506	29339	
2	2525	278809	
3	2541	32752	
4	2516	158486	
5	2489	155191	
6	2488	179468	
7	2548	93335	
8	2505	190878	
9	2559	10834	
10	2522	53303	

Tabela 18: Symulowane wyżarzanie: 0.99 - rbg403.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	1934	2509	
2	1868	2575	
3	1860	2822	
4	1861	2553	
5	1934	2690	
6	1936	2740	
7	1902	2623	
8	1931	2689	
9	2023	2723	
10	1925	2592	

Tabela 19: Symulowane wyżarzanie: 0.999 - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	4070	8774	
2	3699	8296	
3	3930	8081	
4	3808	8405	
5	3973	2	
6	3985	8587	
7	3865	8526	
8	4135	8513	
9	3962	16461	
10	3990	8414	

Tabela 20: Symulowane wyżarzanie: 0.999 - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2487	266793	
2	2497	222616	
3	2486	61972	
4	2486	48529	
5	2474	216246	
6	2501	31674	
7	2493	79758	
8	2495	89381	
9	2496	358816	
10	2486	71857	

Tabela 21: Symulowane wyżarzanie: 0.999 - rbg403.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	1790	26849	0,79
2	1800	26445	1,35
3	1804	26532	1,57
4	1789	26101	0,73
5	1832	26562	3,15
6	1871	25692	5,34
7	1842	27692	3,71
8	1852	26074	4,27
9	1833	26851	3,21
10	1796	26721	1,13

Tabela 22: Symulowane wyżarzanie: 0.9999 - ftv47.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	3381	82438	22,72
2	3524	80980	27,91
3	3556	79242	29,07
4	3475	80759	26,13
5	3356	83018	21,81
6	3455	80393	25,41
7	3500	75537	27,04
8	3655	79560	32,66
9	3224	81192	17,02
10	3322	80803	20,58

Tabela 23: Symulowane wyżarzanie: 0.9999 - ftv170.atsp

Uruchomienie	Najlepszy koszt	Czas osiągniecia[ms]	Błąd względny[%]
1	2476	225117	0,45
2	2470	215242	0,20
3	2465	211685	0
4	2470	231415	0,20
5	2473	214459	0,32
6	2469	234381	0,16
7	2476	215441	0,45
8	2471	213575	0,24
9	2486	221872	0,85
10	2473	211761	0,32

Tabela 24: Symulowane wyżarzanie: 0.9999 - rbg403.atsp

5 Wnioski

Podczas implementacji algorytmów Tabu Search i Symulowanego Wyżarzania do rozwiązania problemu Komiwojażera (TSP), zauważono skuteczność obu podejść w minimalizacji długości tras podróży. Algorytm Tabu Search koncentrował się głównie na intensyfikacji przeszukiwania, co pozwalało na systematyczne doskonalenie najlepszego znalezionego rozwiązania. Z kolei Symulowane Wyżarzanie wykazywało zdolność do dynamicznego dostosowywania się poprzez schładzanie, co przyczyniało się do unikania utknięcia w lokalnych optimum.

W trakcie eksperymentów zauważono, że skuteczność obu algorytmów była ściśle związana z odpowiednim dostosowaniem parametrów. Optymalne rezultaty uzyskiwano poprzez staranne kalibrowanie parametrów, takich jak temperatura początkowa czy współczynnik schładzania.

Podczas analizy wyników testów, oba algorytmy wykazywały się zdolnością do znajdowania efektywnych tras podróży, choć w niektórych przypadkach mogły zachodzić pewne różnice w skuteczności w zależności od charakterystyki problemu. Zastosowanie listy tabu w algorytmie Tabu Search skutecznie zapobiegało cyklicznemu powtarzaniu tych samych ruchów, co wpływało pozytywnie na różnorodność eksploracji przestrzeni rozwiązań.

Wnioski z implementacji algorytmów Tabu Search i Symulowanego Wyżarzania dla problemu TSP sugerują, że oba podejścia są potężnymi narzędziami do rozwiązywania problemów optymalizacyjnych. Jednakże, dokładne dostosowanie parametrów oraz uwzględnienie specyfiki problemu są kluczowe dla osiągnięcia satysfakcjonujących rezultatów.

Bibliografia

[1] AGH. "Heurytyki i metaheurystyki". W: (). URL: http://www.pi.zarz.agh.edu.pl/intObl/notes/IntObl_w2.pdf.

[2] Francis Allanah. "Travelling Salesman Problem Using Simulated Annealing". W: (). URL: https://medium.com/@francis.allanah/travelling-salesman-problem-using-simulated-annealing-f547a71ab3c6.

[3] ChatGPT.

[4] Farhad Kolahan i in. "Analysis of neighborhood generation and move selection strategies on the performance of Tabu Search". W: (sty. 2006).

[5] Shigeru Tsubakitani i James R. Evans. "Optimizing tabu list size for the traveling salesman problem". W: *Computers Operations Research* 25.2 (1998), s. 91–97. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(97)00030-0. URL: https://www.sciencedirect.com/science/article/pii/S0305054897000300.

[6] Paweł Zieliński. "Metody optymalizacji". W: (). URL: https://cs.pwr.edu.pl/zielinski/lectures/om/localsearch.pdf.