Projektowanie efektywnych algorytmów

Projekt 2

Asymetryczny problem komiwojażera

Mateusz Tesarewicz 272909

16.12.2024 Politechnika Wrocławska

1 Wstęp

Asymetryczny problem komiwojażera (ATSP, z ang. Asymmetric Traveling Salesman Problem) stanowi jedno z kluczowych wyzwań optymalizacyjnych w teorii grafów oraz badań operacyjnych. Zadanie to polega na wyznaczeniu najkrótszej możliwej trasy, która pozwala komiwojażerowi odwiedzić wszystkie zadane miasta dokładnie raz i powrócić do punktu początkowego, przy czym odległości między miastami mogą się różnić w zależności od kierunku przejścia (asymetria). ATSP znajduje szerokie zastosowanie w logistyce, planowaniu tras oraz optymalizacji produkcji, a także w systemach komunikacji miejskiej i transporcie.

Ze względu na swoją złożoność obliczeniową (problem należy do klasy NP-trudnych), rozwiązywanie ATSP wymaga zastosowania specjalistycznych algorytmów, takich jak algorytmy przeglądu zupełnego, podziału i ograniczeń lub programowania dynamicznego. Celem niniejszego badania jest ocena efektywności wybranych algorytmów stosowanych do rozwiązywania ATSP, poprzez analizę ich wyników w kontekście jakości uzyskanych rozwiązań.

2 Badane algorytmy

2.1 Algorytm Tabu Search

Tabu Search to metaheurystyczny algorytm optymalizacyjny, który wykorzystuje strategię pamięci, aby unikać lokalnych minimów i przeszukiwać przestrzeń rozwiązań bardziej efektywnie. Proces rozpoczyna się od wyboru początkowego rozwiązania, które może być wybrane losowo lub na podstawie prostszej heurystyki, na przykład metody najbliższego sąsiada. Następnie algorytm generuje sąsiednie rozwiązania, czyli niewielkie modyfikacje bieżącego rozwiązania. Spośród nich wybierane jest najlepsze rozwiązanie, przy czym algorytm unika ruchów, które są zapisane na liście

tabu. Lista tabu pełni funkcję pamięci, która blokuje niedawno wykonane ruchy na określony czas, aby uniknąć powtarzania tych samych ścieżek poszukiwań. Dodatkowo stosowane są mechanizmy aspiracji, które pozwalają na złamanie reguł tabu, jeśli prowadzi to do istotnie lepszego rozwiązania. W przypadku długiego braku poprawy w wynikach algorytm wprowadza mechanizm dywersyfikacji, który pozwala na eksplorację nowych, odległych obszarów przestrzeni rozwiązań.

2.2 Algorytm Symulowanego Wyżarzania

Symulowane Wyżarzanie to metaheurystyka inspirowana procesem wyżarzania metali, w którym materiał jest powoli schładzany w kontrolowany sposób, aby osiągnąć minimalną energię układu. Algorytm rozpoczyna się od początkowego rozwiązania i ustalenia wysokiej temperatury początkowej. Dla każdego rozwiązania generowane jest sąsiednie rozwiązanie poprzez niewielką modyfikację obecnego. Jeśli nowe rozwiązanie jest lepsze, algorytm je akceptuje, a jeśli jest gorsze, zostaje zaakceptowane z pewnym prawdopodobieństwem zależnym od różnicy jakości rozwiązań i aktualnej temperatury. Wysoka temperatura na początku działania algorytmu umożliwia akceptowanie gorszych rozwiązań, co pozwala na wychodzenie z lokalnych minimów. W miarę postępu algorytmu temperatura jest stopniowo obniżana zgodnie z funkcją chłodzenia, co zmniejsza prawdopodobieństwo akceptacji gorszych rozwiązań i powoduje konwergencję do optymalnego lub blisko optymalnego rozwiązania. Proces kończy się, gdy temperatura osiągnie zadany poziom lub upłynie określony czas.

3 Opisy działania algorytmów

3.1 Pseudokod Tabu Search

```
initializeTabuList()
bestRoute = generateGreedyRoute()
bestCost = calculateCost(bestRoute)
currentRoute = copy(bestRoute)
startTime = getCurrentTime()
noImprove = 0

while not timeExceeded():
    bestNeighbor = findNeighbor(currentRoute)
    neighborCost = calculateCost(bestNeighbor)

    if neighborCost < bestCost:
        updateBestSolution(bestNeighbor, neighborCost)
        noImprove = 0
    else:
        noImprove++

    decrementTabuList()
    markLastMoveAsTabu(currentRoute, bestNeighbor)
    updateCurrentRoute(currentRoute, bestNeighbor)

    if shouldDiversify(noImprove):
        diversifyRoute(currentRoute)
        noImprove = 0

return bestRoute
```

3.2 Pseudokod Symulowanego Wyżarzania

```
initializeSolution(currentSolution)
shuffleSolution(currentSolution)
bestRoute = copy(currentSolution)
bestCost = calculateCost(bestRoute)
currentCost = bestCost

calculateInitialTemperature()

iteration = 0
while not timeExceeded() && temperature > 1e-8

    for i in range(maxIterations):
        neighbor = generateNeighbor(currentSolution)
        neighborCost = calculateCost(neighbor)

       delta = neighborCost - currentCost
        acceptanceProbability = calculateAcceptanceProbability(currentCost,
neighborCost, temperature, delta)

        if shouldAcceptSolution(acceptanceProbability):
            updateCurrentSolution(currentSolution, neighbor)
            currentCost = neighborCost
```

```
        if currentCost < bestCost:
            updateBestSolution(bestRoute, currentSolution, currentCost)
            bestFindTime = elapsedTime

    updateTemperature(temperature, iteration, coolingScheme)
    iteration++

return bestRoute
```

3.3 Metody wyboru sąsiedztwa - Tabu Search

3.3.1 Metoda zamiany (Swap Neighborhood)

Ta metoda polega na zamianie miejscami dwóch miast w obecnej trasie. Iteracyjnie wybierane są wszystkie możliwe pary miast (z wyjątkiem pierwszego i ostatniego, ponieważ trasa musi być cykliczna), a następnie zamieniane są ich pozycje. Po każdej zamianie obliczany jest koszt nowej trasy, a najlepsze rozwiązanie, które spełnia warunki listy tabu, jest zapamiętywane jako najlepszy sąsiad. Metoda ta jest efektywna dla małych zmian w trasie, ale może szybko przechodzić do lokalnych minimów.

3.3.2 Metoda wstawiania (Insert Neighborhood)

W tej metodzie wybierane jest miasto z trasy, które następnie jest usuwane z obecnej pozycji i wstawiane w innym miejscu w trasie. Wszystkie możliwe pary pozycji są rozważane (z wyjątkiem sytuacji, gdzie miasto jest wstawiane na swoją pierwotną pozycję). Działanie tej metody pozwala na większe modyfikacje trasy niż w metodzie zamiany, co może pomóc w eksploracji odleglejszych rozwiązań w przestrzeni sąsiedztwa. Podobnie jak w przypadku zamiany, najlepsze rozwiązanie, spełniające kryteria tabu, jest wybierane.

3.3.3 Metoda k-zamiany (k-Swap Neighborhood)

Ta metoda wprowadza bardziej złożone modyfikacje poprzez zamianę pozycji trzech miast w trasie. Iteracyjnie wybierane są trzy różne miasta w trasie, a ich pozycje są zamieniane w sposób określony przez dwie kolejne operacje swap.

Ten sposób modyfikacji trasy generuje bardziej różnorodne sąsiedztwo i pozwala na uniknięcie lokalnych minimów, szczególnie w porównaniu do prostszych metod, takich jak zamiana dwóch miast. Jednak złożoność obliczeniowa tej metody jest wyższa ze względu na większą liczbę możliwych kombinacji.

3.4 Metody schładzania temperatury - Symulowane Wyżarzanie

Metoda wykładnicza

W tej metodzie temperatura jest zmniejszana w sposób wykładniczy poprzez przemnożenie jej przez stałą wartość, zwaną współczynnikiem schładzania (coolingRate). Zwykle wartość coolingRate jest mniejsza niż 1 (np. 0.95), co powoduje stopniowe, ale regularne zmniejszanie temperatury w miarę postępu iteracji. Metoda ta jest prosta w implementacji i efektywna w problemach, gdzie szybkie schładzanie jest korzystne. Jednak przy zbyt małym współczynniku schładzania może nie być wystarczająco eksploracyjna.

Formuła:

$T_{i+1} = T_i \times a$, gdzie a to współczynnik schładzania

Metoda adaptacyjna

Ta metoda schładzania zmniejsza temperaturę w sposób adaptacyjny, dzieląc jej wartość przez wyrażenie zależne od jej bieżącej wartości. Dzięki temu spadek temperatury jest szybszy na początku, gdy wartości są większe, a następnie staje się wolniejszy w dalszych iteracjach. Taka dynamika pozwala na bardziej efektywne eksplorowanie rozwiązań w początkowych fazach, a jednocześnie precyzyjniejszą optymalizację w końcowych etapach.

Formuła:

$$T_{i+1} = \frac{T_i}{1 + 0.001 \times T_i}$$

Metoda logarytmiczna

W tej metodzie temperatura zmniejsza się zgodnie z odwrotnością logarytmu liczby iteracji. Jest to najwolniejsza z zaprezentowanych metod schładzania, co pozwala algorytmowi dłużej eksplorować przestrzeń rozwiązań przy wyższych temperaturach. Metoda ta jest zgodna z teorią symulowanego wyżarzania, według której bardzo wolne schładzanie gwarantuje znalezienie rozwiązania globalnie optymalnego (pod warunkiem nieskończonej liczby iteracji). Sprawdza się w problemach wymagających intensywnej eksploracji.

Formuła:

$$T_{i+1} = \frac{T_i}{\log(i+2)}$$

4 Plan przeprowadzenia badania

Poprawnie zaplanowanie przebiegu badania jest bardzo kluczowe, aby uniknąć ewentualnych komplikacji lub co gorsza, niepoprawnych wyników przeprowadzonego badania. Badaniu poddamy wszystkie algorytmy dla wyznaczonych ilości miast. Odległości między miastami zostaną przedstawione w postaci macierzy kwadratowej. Odległości będą dodatnimi liczbami całkowitymi.

4.1 Środowisko i specyfikacja sprzętu

Do przeprowadzenia badania został napisany obiektowy program w języku C++ kompilowany w środowisku CLion. Badania zostaną przeprowadzone na komputerze o podanej specyfikacji:

Procesor :AMD Ryzen 7 5700X 8-Core Processor 3.40 GHz

Pamięć RAM: 16 GB 3200 MHz

System operacyjny: Windows 10 Home

Typ systemu: 64-bitowy system operacyjny, procesor x64

4.2 Przedstawienie odległości między miastami

Odległości między miastami będą zapisane w kwadratowej macierzy, gdzie numer wiersza będzie oznaczał miasto początkowe, a numer kolumny miasto końcowe. Wartością w macierzy będzie odległość między miastem początkowym a końcowym. Drogi między miastami będą zazwyczaj asymetryczne. Po przekątnej macierzy wpisane zostaną wartości -1 lub 0, żeby wykluczyć drogi z miast do siebie nawzajem. W innych komórkach będą mogły się znajdować tylko liczby naturalne dodatnie z przedziału ustalanego w configu. Aby wczytać macierz, należy przygotować ją w pliku o rozszerzeniu .atsp.

4.3 Rozmiar problemu – czas działania algorytmów

Badanie zostanie przeprowadzone na 3 rozmiarach problemu: **55, 170 i 358**
Dla tych wartości zostanie zbadany każdy algorytm pod względem wydajności. Dla 55 miast czas trwania algorytmu ustawimy na 2 minuty, dla 170 miast 4 minuty i dla 358 - 6 minut.

4.4 Przebieg eksperymentu

Zaimplementowany program pozwala testować algorytmy Tabu Search i Simulated Annealing. Dane mogą zostać wczytane z pliku rozszerzeniu atsp. i w ten sposób wczytamy trzy macierze o rozmiarach podanych wyżej. Ustawimy odpowiednie czasy działania algorytmów i będziemy obserwować otrzymane wyniki wraz z czasem znalezienia tego wyniku. Każdy test zostanie przeprowadzony 10 razy oraz sprawdzimy wpływ zmiany parametrów algorytmów.

5 Wyniki badań dla Tabu Search

5.1 Tabela z wynikami dla ftv55.atsp

ftv55.atsp 56 miast									
Tabu Search									
lp	Swap			Insert			kSwap		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	1798	0,048	11,82%	1661	45,797	3,30%	1827	90,084	13,62%
2	1868	104,507	16,17%	1652	96,754	2,74%	1846	92,086	14,80%
3	1845	98,81	14,74%	1663	30,705	3,42%	1807	34,777	12,38%
4	1828	104,517	13,68%	1704	55,798	5,97%	1869	31,195	16,23%
5	1794	44,18	11,57%	1640	0,688	1,99%	1791	102,268	11,38%
6	1805	85,56	12,25%	1670	2,678	3,86%	1790	0,3	11,32%
7	1798	0,033	11,82%	1629	58,668	1,31%	1807	38,958	12,38%
8	1755	49,33	9,14%	1694	13,698	5,35%	1739	40,045	8,15%
9	1812	32,491	12,69%	1653	37,67	2,80%	1849	108,084	14,99%
10	1829	49,81	13,74%	1702	56,628	5,85%	1829	16,143	13,74%

Podczas przeprowadzania testów dla liczby miast 56 ustawiłem wartość tabuTenure na 3 oraz maxNoImprove na 40. TabuTenure oznacza ilość iteracji trwania zakazu na liście tabu. maxNoImprove to maksymalna ilości iteracji bez poprawy wyniku, po tym następuje wylosowanie innej trasy.

5.2 Tabela z wynika dla ftv170.atsp

ftv170.atsp 171 miast									
Tabu Search									
lp	Swap			Insert			kSwap		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	3348	9,625	21,52%	3405	12,353	23,59%	3451	0,279	25,26%
2	3382	0,015	22,76%	3430	1,343	24,50%	3457	0,208	25,48%
3	3368	10,656	22,25%	3365	54,544	22,14%	3385	3,543	22,87%
4	3310	56,644	20,15%	3285	86,446	19,24%	3378	6,435	22,61%
5	3385	75,745	22,87%	3320	90,455	20,51%	3471	0,139	25,99%
6	3402	12,234	23,48%	3343	26,643	21,34%	3517	0,0643	27,66%
7	3343	86,457	21,34%	3412	0,553	23,85%	3369	4,463	22,29%
8	3310	34,346	20,15%	3382	68,674	22,76%	3412	47,575	23,85%
9	3389	97,345	23,01%	3298	110,423	19,71%	3428	75,464	24,43%
10	3412	4,464	23,85%	3301	96,435	19,82%	3390	45,564	23,05%

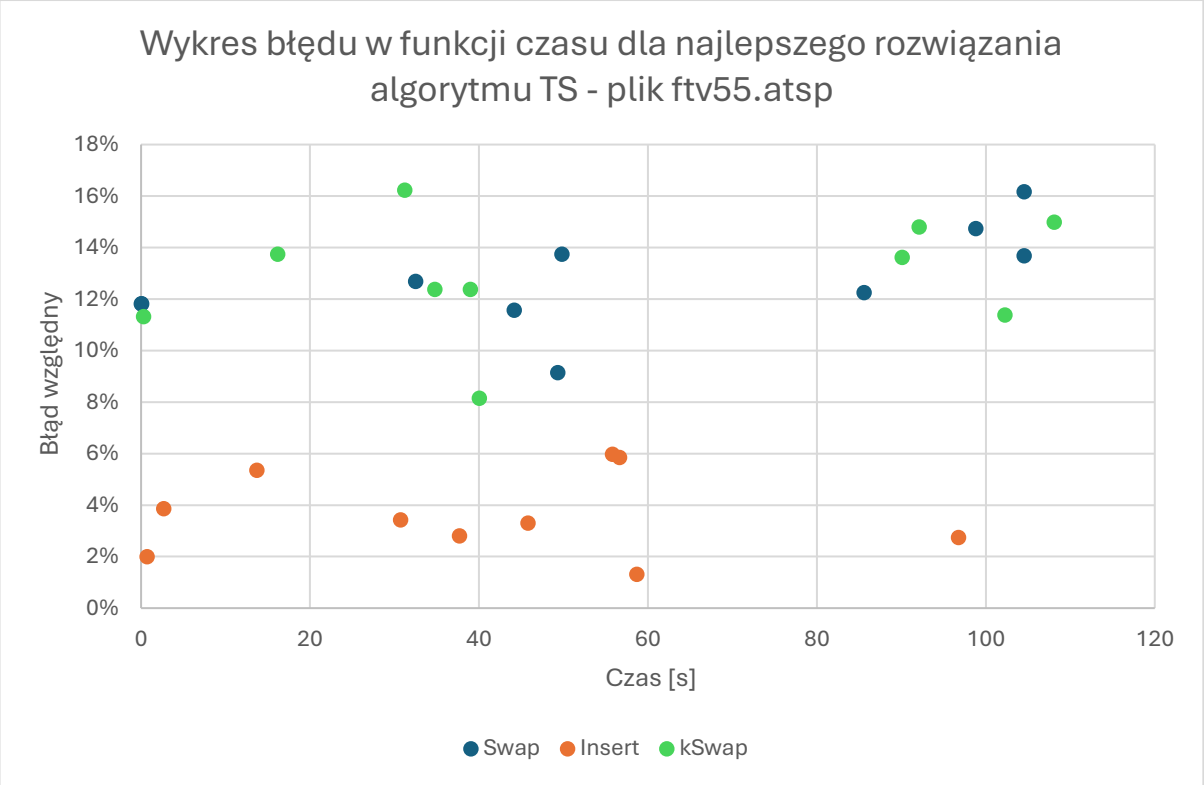
Dla rozmiaru 171 miast użyłem tabuTenure o wartości 10 i maxNoImprowe równe 80.

5.3 Tabela z wynikami dla rbg358.atsp

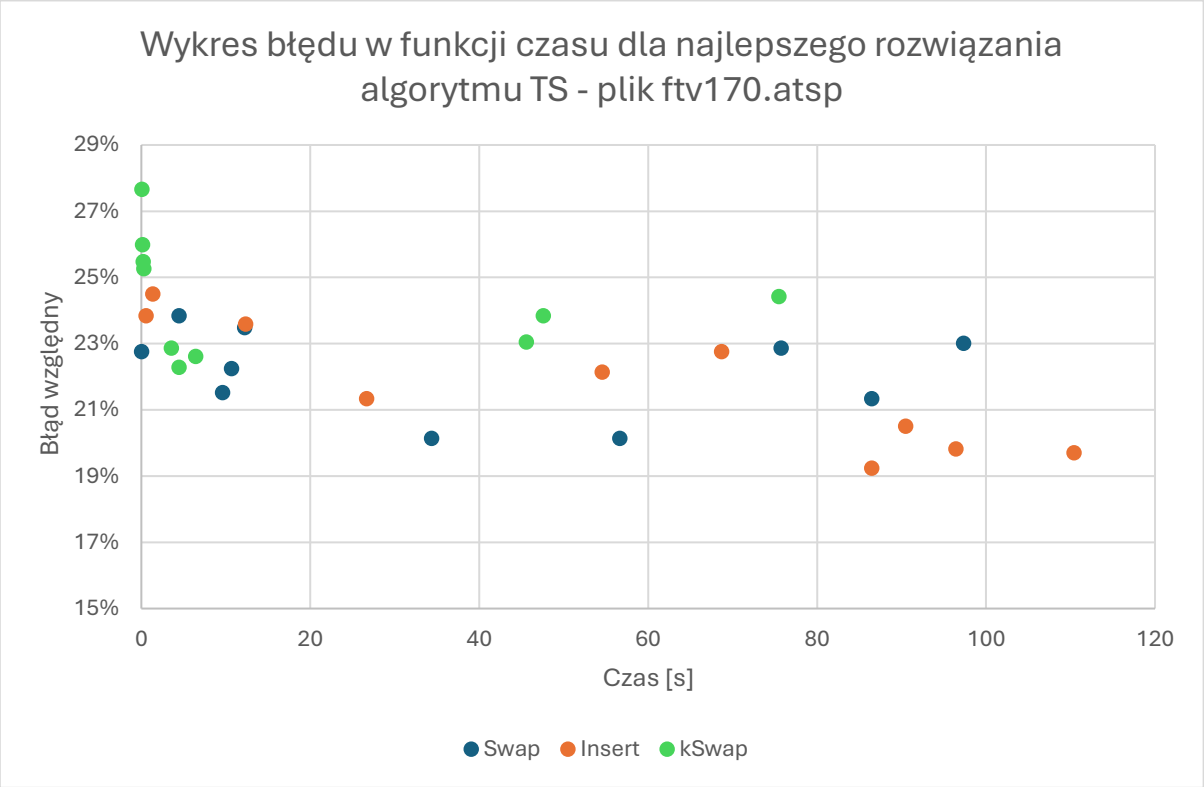
rbg358.atsp 359 miast									
Tabu Search									
lp	Swap			Insert			kSwap		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	1356	214,878	16,60%	1369	26,861	17,71%	1417	33,696	21,84%
2	1364	317,576	17,28%	1419	258,570	22,01%	1472	65,298	26,57%
3	1393	0,814	19,78%	1362	50,935	17,11%	1432	285,780	23,13%
4	1356	138,645	16,60%	1384	167,753	19,00%	1451	18,571	24,76%
5	1360	120,428	16,94%	1326	1,759	14,02%	1432	283,026	23,13%
6	1352	8,045	16,25%	1340	10,088	15,22%	1430	347,376	22,96%
7	1391	1,543	19,60%	1406	126,091	20,89%	1387	92,572	19,26%
8	1373	32,234	18,06%	1362	24,666	17,11%	1354	278,881	16,42%
9	1358	265,353	16,77%	1330	196,848	14,36%	1344	213,826	15,56%
10	1389	4,353	19,43%	1340	76,429	15,22%	1367	34,285	17,54%

Dla przykładu z 358 miastami użyłem wartości tabuTenure = 15 i maxNoImprove = 150.

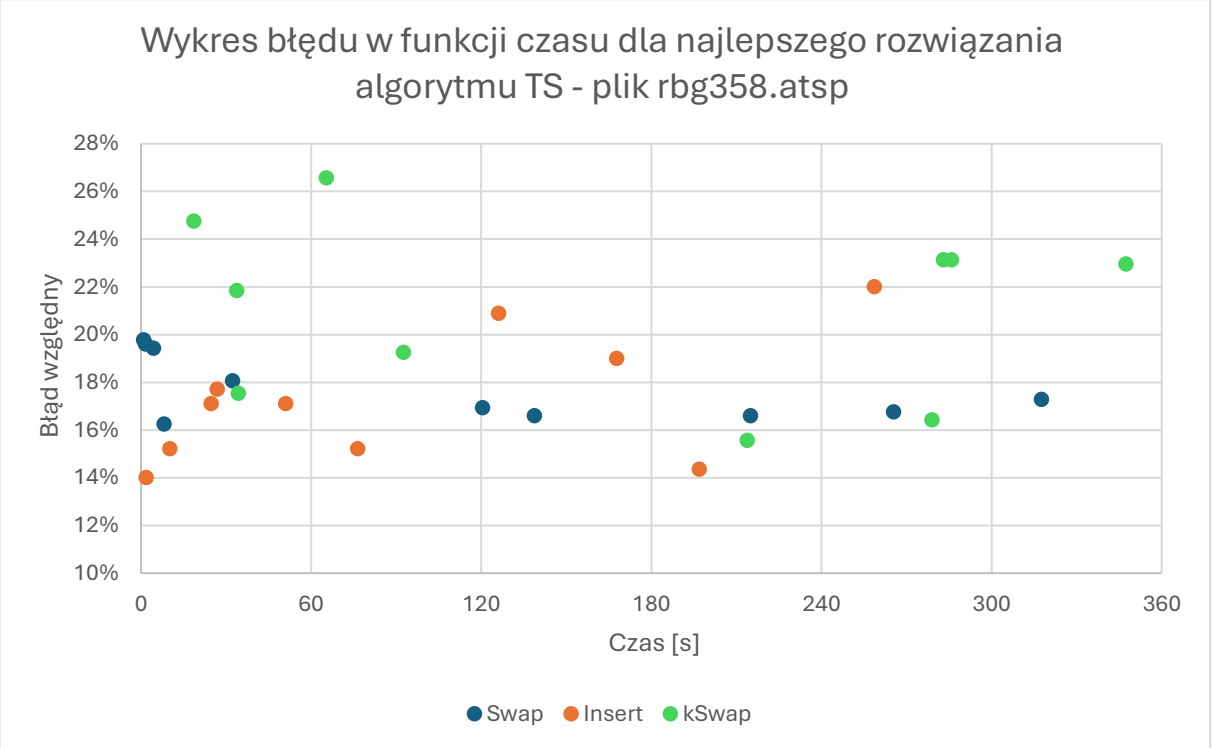
5.4 Wykres błędu w funkcji czasu dla pliku ftv55.atsp



5.5 Wykres błędu w funkcji czasu dla pliku ftv170.atsp



5.6 Wykres błędu w funkcji czasu dla pliku rbg358.atsp



6 Wyniki badań dla Symulowanego wyżarzania

6.1 Tabela z wynikami dla ftv55.atsp

ftv55.atsp 56 miast									
Simulated Annealing									
lp	T[i+1] = T[i] * a			T[i+1] = T[i] / (1 + 0.001 * T[i])			T[i+1] = T[i] / log(i + 2)		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	1724	0,036	7,21%	1677	0,119	4,29%	1771	0,005	10,14%
2	1721	0,029	7,03%	1675	0,06	4,17%	1801	0,005	12,00%
3	1690	0,035	5,10%	1608	0,08	0,00%	1791	0,005	11,38%
4	1651	0,039	2,67%	1679	0,034	4,42%	1871	0,006	16,36%
5	1709	0,03	6,28%	1685	0,031	4,79%	1824	0,005	13,43%
6	1706	0,038	6,09%	1678	0,043	4,35%	1668	0,005	3,73%
7	1725	0,034	7,28%	1649	0,05	2,55%	1837	0,013	14,24%
8	1679	0,034	4,42%	1695	0,066	5,41%	1797	0,004	11,75%
9	1706	0,026	6,09%	1690	0,065	5,10%	1745	0,004	8,52%
10	1689	0,028	5,04%	1683	0,049	4,66%	1892	0,006	17,66%
Tp	324			324			324		
Tk	9,91E-09			0,00227			6,53E-09		

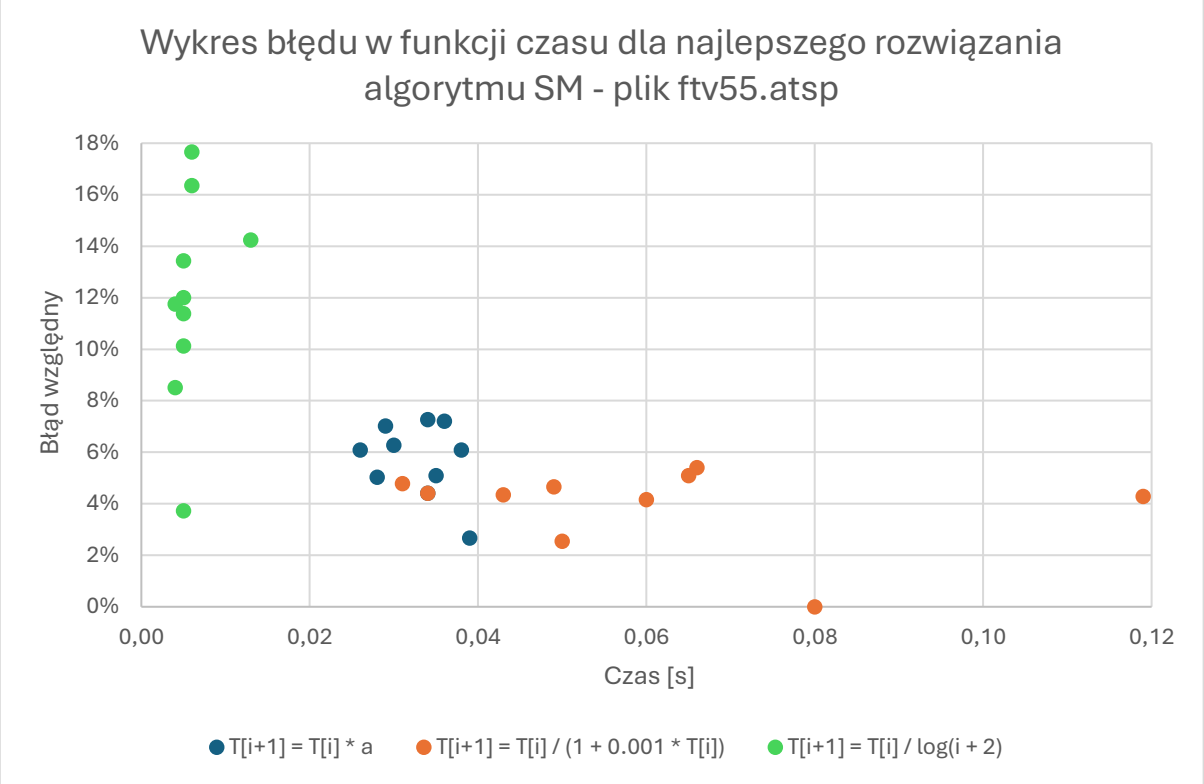
6.2 Tabela z wynikami dla ftv170.atsp

ftv170.atsp 171 miast									
Simulated Annealing									
lp	T[i+1] = T[i] * a			T[i+1] = T[i] / (1 + 0.001 * T[i])			T[i+1] = T[i] / log(i + 2)		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	3310	0,856	20,15%	2972	1,216	7,88%	3946	0,092	43,23%
2	3207	1,366	16,41%	3023	1,214	9,73%	3544	0,13	28,64%
3	3069	0,548	11,40%	3173	1,477	15,17%	3391	0,106	23,09%
4	3205	0,47	16,33%	2818	1,27	2,29%	3412	0,107	23,85%
5	3045	0,513	10,53%	3004	1,093	9,04%	3618	0,088	31,32%
6	3361	0,484	22,00%	2892	1,064	4,97%	3704	0,131	34,45%
7	3440	0,479	24,86%	3134	1,44	13,76%	3708	0,13	34,59%
8	3295	0,56	19,60%	2916	1,066	5,84%	3849	0,102	39,71%
9	3126	0,54	13,47%	2934	1,799	6,50%	4087	0,094	48,35%
10	3384	0,689	22,83%	2927	1,24	6,24%	3533	0,132	28,24%
Tp	368			368			368		
Tk	9,65E-09			0,25108			7,42E-09		

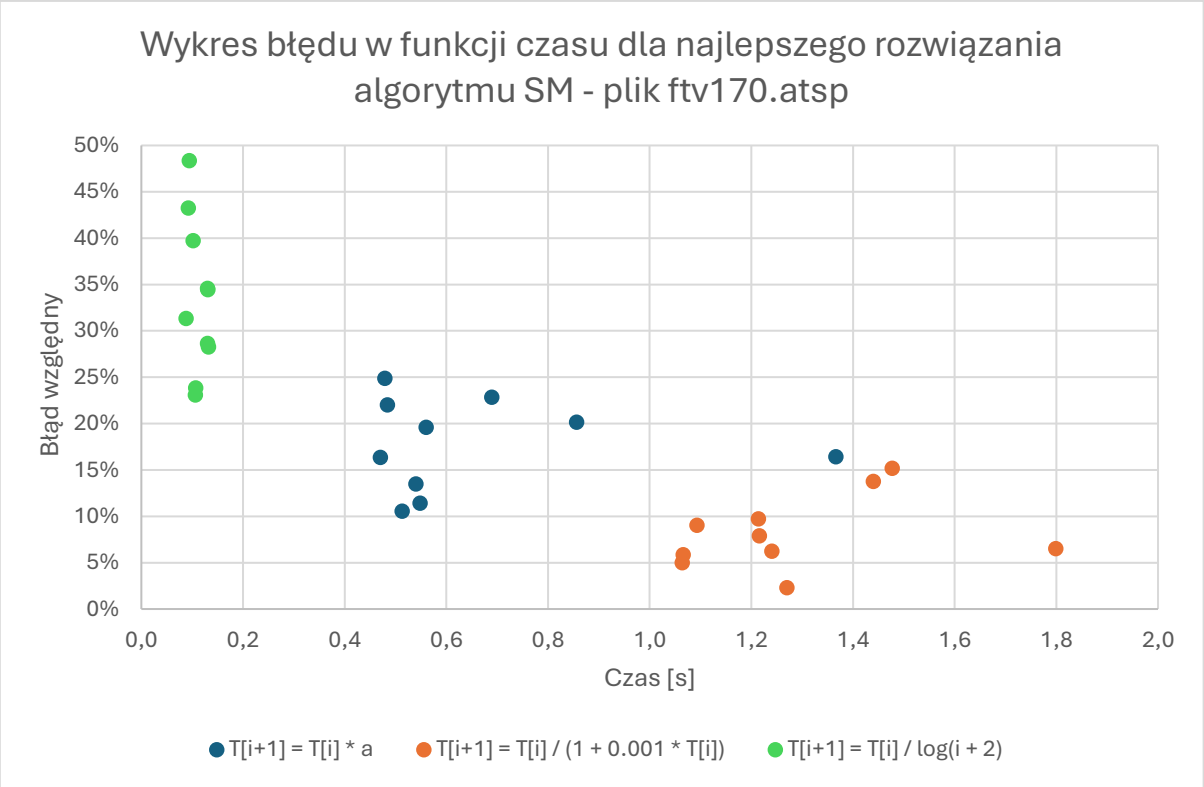
6.3 Tabela z wynikami dla rbg358.atsp

rbg358.atsp 359 miast									
Simulated Annealing									
lp	1			2			3		
	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny	najlepsze rozwiązanie	znaleziono w [s]	błąd względny
1	1252	6,914	7,65%	1174	105,569	0,95%	1335	0,888	14,79%
2	1237	4,534	6,36%	1172	142,394	0,77%	1386	0,788	19,17%
3	1258	8,292	8,17%	1178	189,482	1,29%	1367	0,78	17,54%
4	1227	3,904	5,50%	1173	115,177	0,86%	1322	0,609	13,67%
5	1256	3,875	8,00%	1170	114,309	0,60%	1322	0,729	13,67%
6	1267	4,336	8,94%	1177	95,718	1,20%	1370	0,928	17,80%
7	1222	4,289	5,07%	1170	119,414	0,60%	1324	0,791	13,84%
8	1236	4,522	6,28%	1172	238,888	0,77%	1377	0,952	18,40%
9	1218	7,957	4,73%	1171	172,83	0,69%	1353	0,846	16,34%
10	1210	4,706	4,04%	1178	170,224	1,29%	1374	0,839	18,14%
Tp	718			718			718		
Tk	9,64E-09			0,11179			7,47E-09		

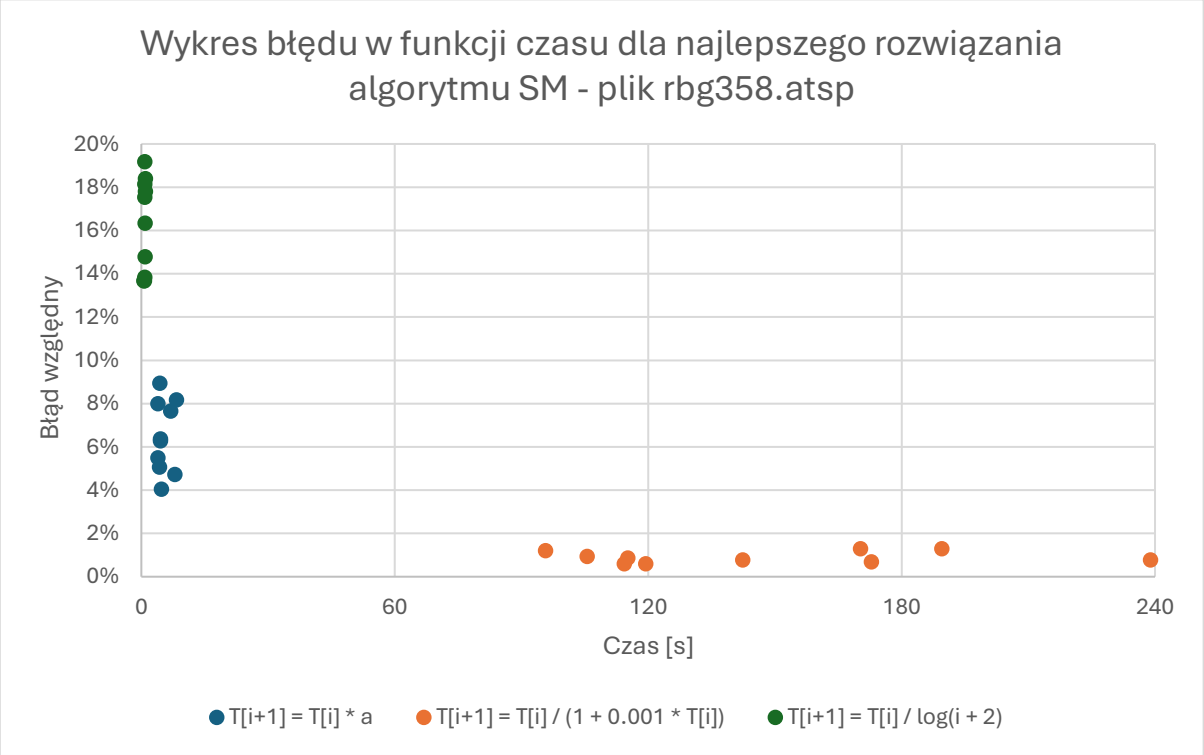
6.4 Wykres błędu w funkcji czasu dla pliku ftv55.atsp

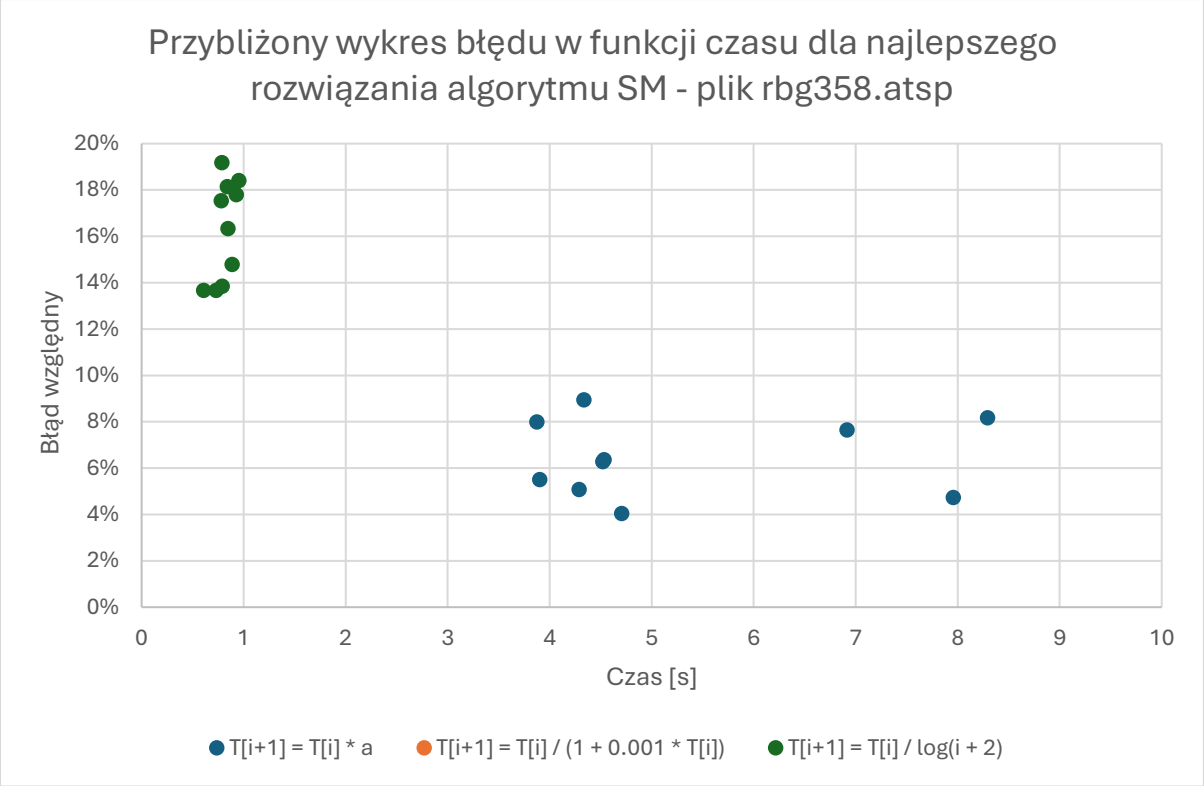


6.5 Wykres błędu w funkcji czasu dla pliku ftv170.atsp



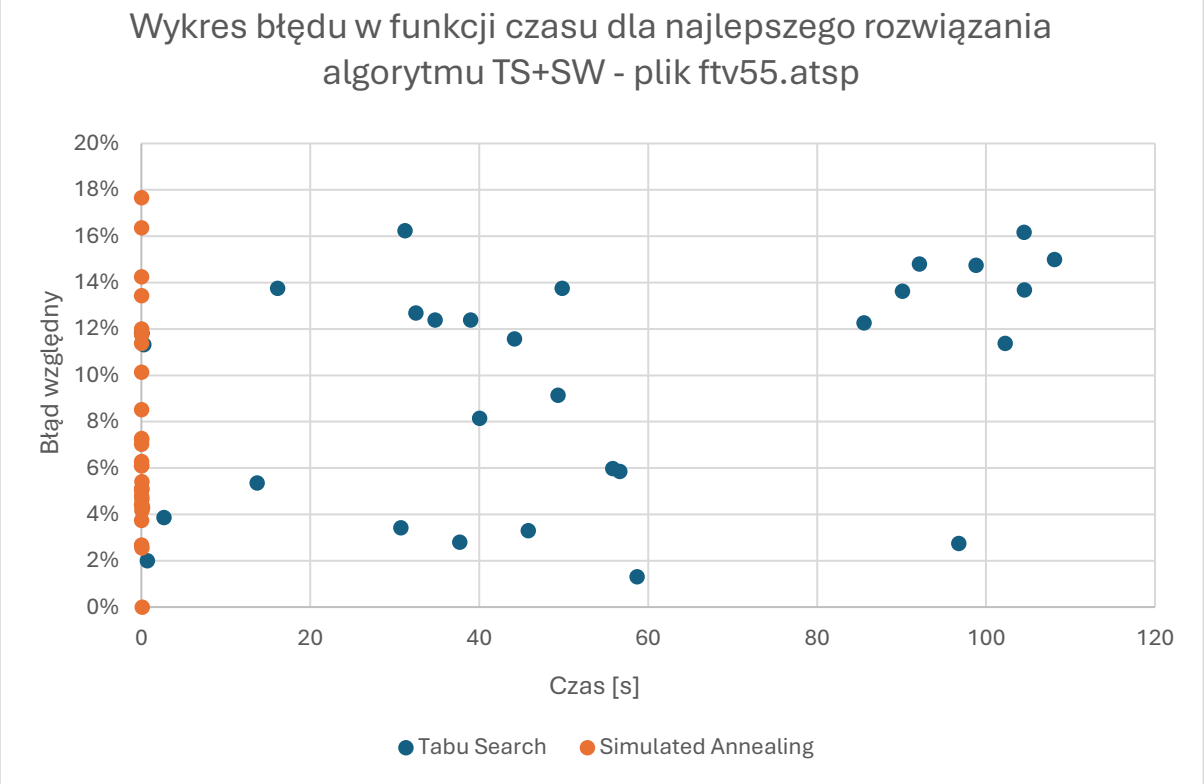
6.6 Wykresy błędu w funkcji czasu dla pliku rbg358.atsp

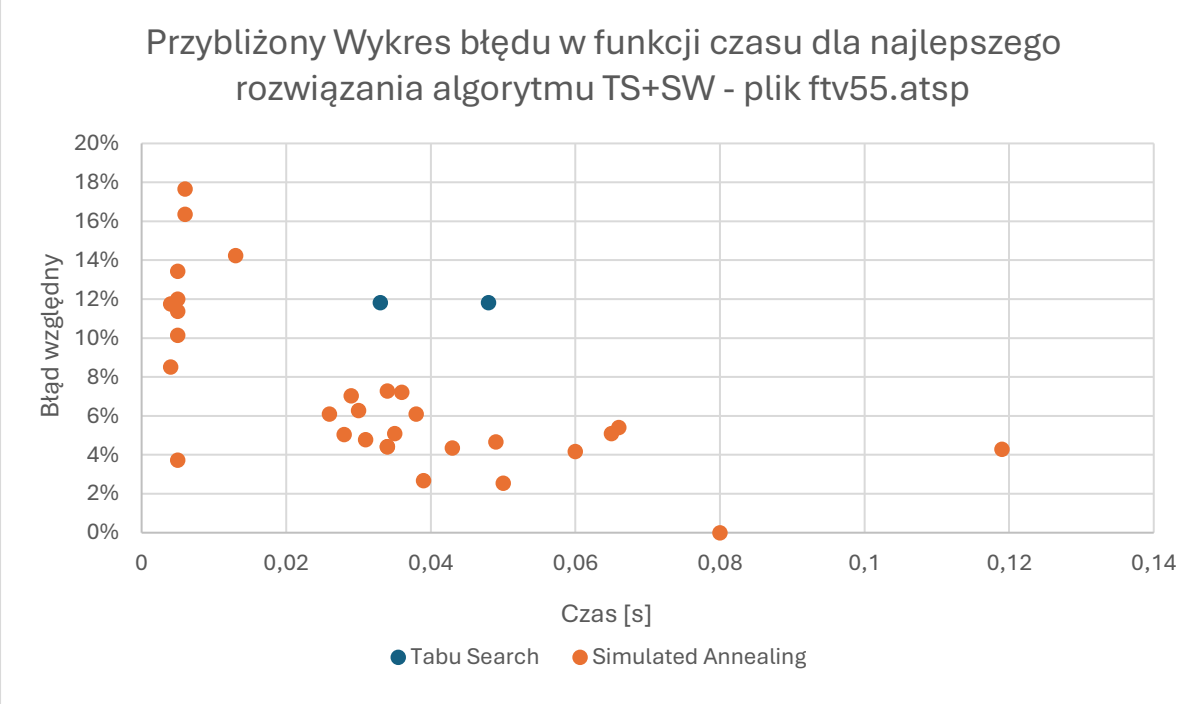



Przybliżony wykres błędu w funkcji czasu dla najlepszego rozwiązania algorytmu SM - plik rbg358.atsp
Błąd względny
20%
18%
16%
14%
12%
10%
8%
6%
4%
2%
0%
0
1
2
3
4
5
6
7
8
9
10
Czas [s]
T[i+1] = T[i] * a
T[i+1] = T[i] / (1 + 0.001 * T[i])
T[i+1] = T[i] / log(i + 2)

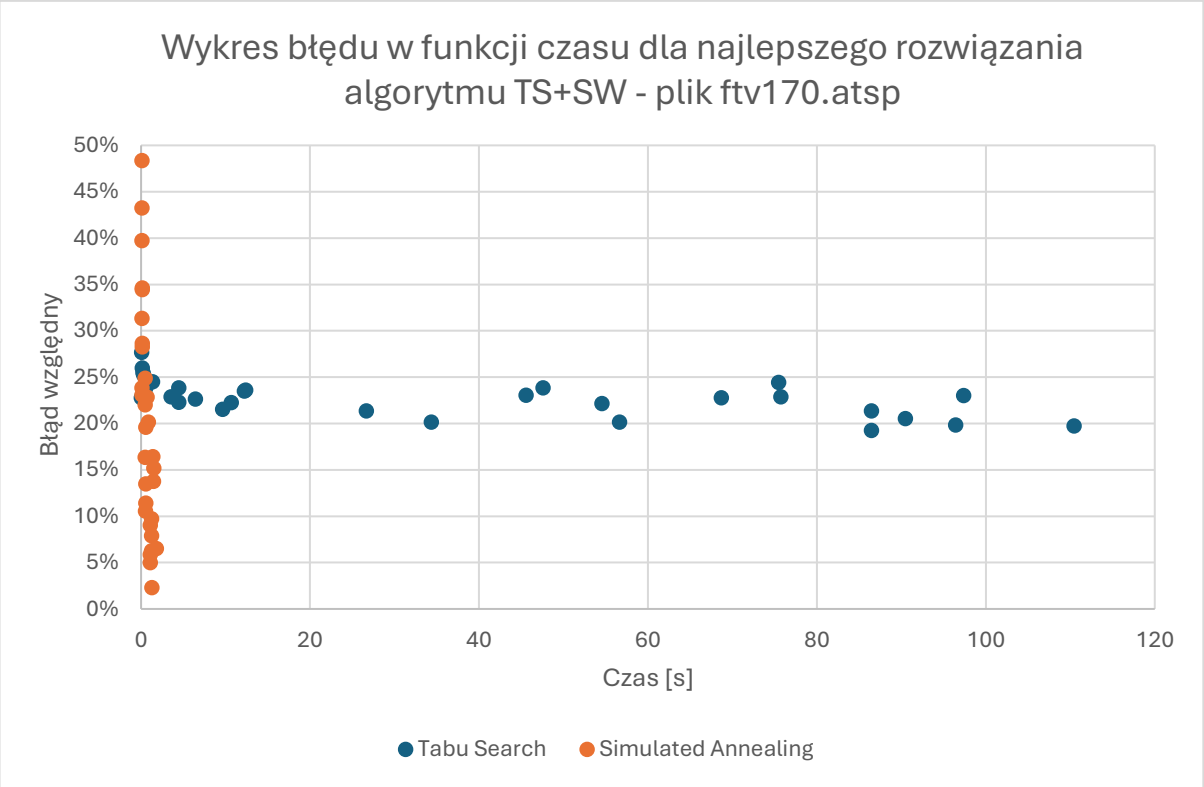
7 Porównanie wyników Tabu Search i Symulowanego wyżarzania

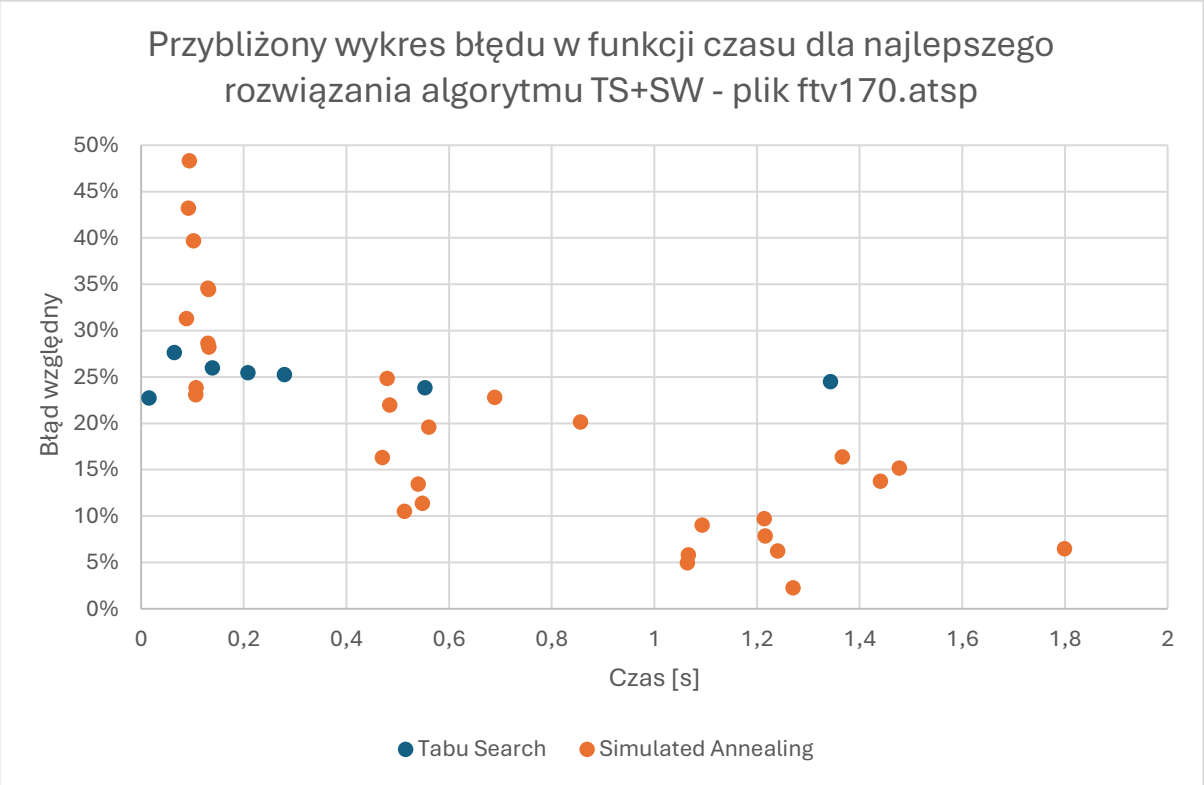
7.1 Wykresy błędu w funkcji czasu dla pliku ftv55.atsp



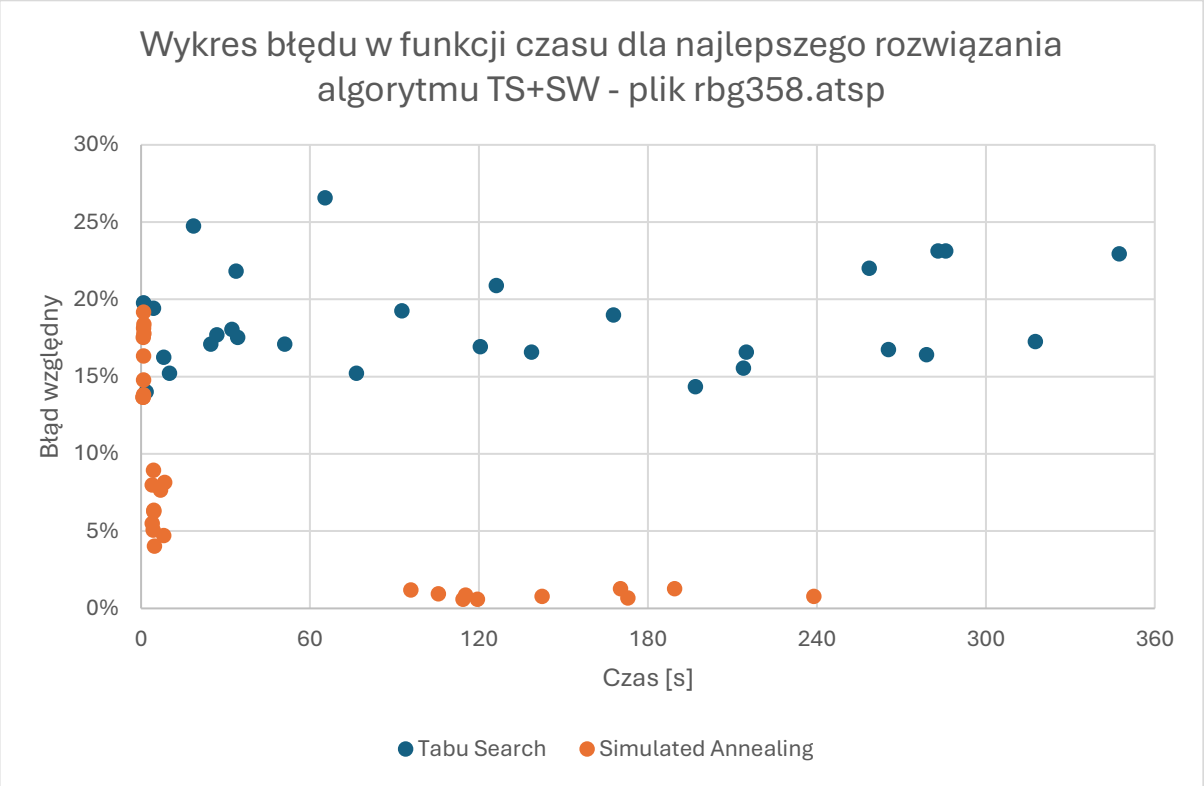


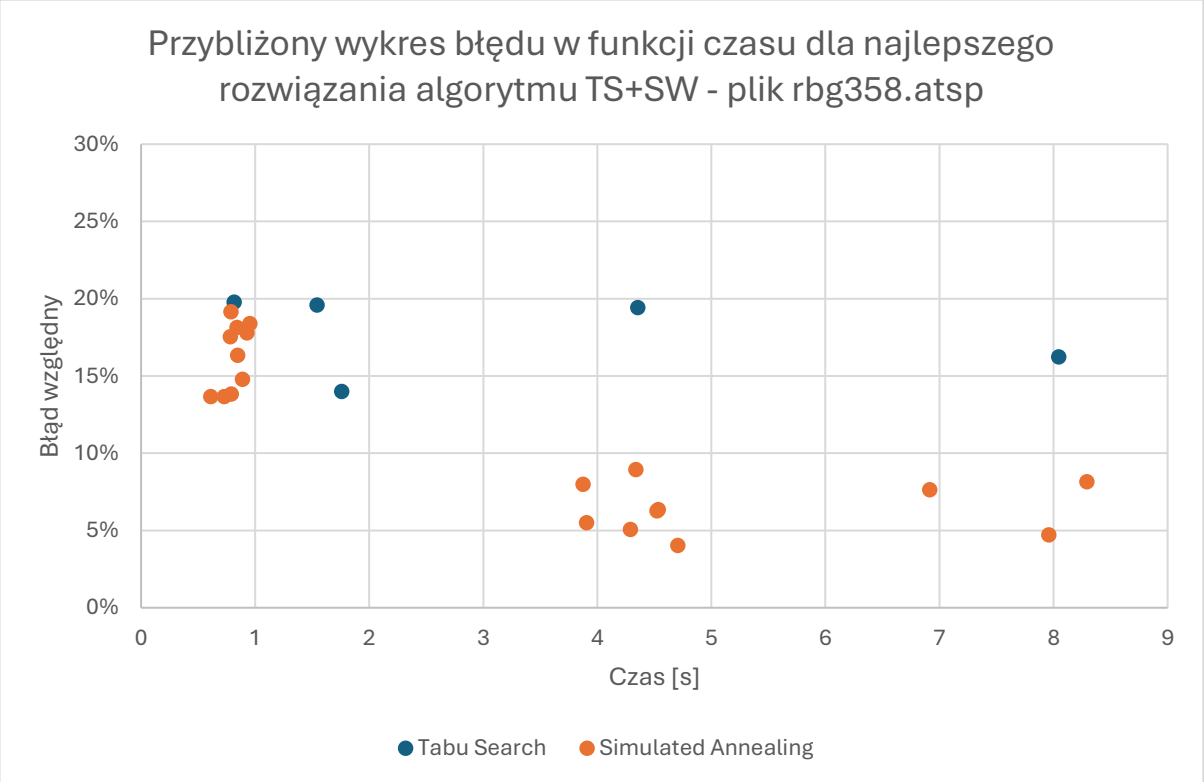
7.2 Wykresy błędu w funkcji czasu dla pliku ftv170.atsp





7.3 Wykresy błędu w funkcji czasu dla pliku rbg358.atsp





8 Podsumowanie

Podsumowując przeprowadzone badania stwierdzam, że algorytmy cechują się dużą losowością i nie ma żadnej zależności między czasem trwania algorytmów, a otrzymanym wynikiem. Zmienianie parametrów nie wpływało na otrzymywane wyniki. W Tabu Search najlepszą metodą wyboru sąsiedztwa okazała się metoda Insert, ale znaczną różnicę widać tylko dla pliku ftv55.atsp. Dla pozostałych plików wszystkie metody radziły sobie podobnie źle, ponieważ algorytm nie mógł znaleźć lepszego rozwiązania niż 13% błędu względnego, a dla pliku ftv170 było to aż 19%, dodatkowo algorytm, mimo wykonywania się 4 minuty, znajdywał rozwiązania w przedziale dwóch pierwszych minut. Nie widać tutaj poprawy wyników wraz z wydłużaniem czasu działania algorytmu. Algorytm Symulowanego Wyżarzania również był odporny na wydłużanie czasu działania i zwracał wyniki w pierwszych sekundach i jeszcze szybciej. Wyjątkiem jest 2 metoda schładzania dla największego pliku, która okazała się być najlepszą metodą dla działania algorytmu, ponieważ wyniki wychodziły zawsze najlepsze właśnie dla tej metody. Najgorszą metodą okazała się być ta ostatnia, ponieważ schładzanie działo się za szybko i przez to nie było możliwości większego przeszukiwania rozwiązań. Porównując oba algorytmy, moim zdaniem Symulowane wyżarzanie poradziło sobie o wiele lepiej niż Tabu Search. Szczególnie dla plików ftv170 i rbg358. Algorytm Tabu Search nie był w stanie znaleźć tak dobrych wyników, jak Algorytm Wyżarzania i dodatkowo zajmowało mu to o wiele więcej czasu. Szkoda, że nie udało się zauważyć zależności otrzymanych wyników od czasu działania algorytmów.

9 Źródła

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

www.eduinf.waw.pl

www.geeksforgeeks.org

www.stackoverflow.com

www.tutorialspoint.com

www.medium.com