Sprawozdanie

Projektowanie Efektywnych Algorytmów P

Etap 2

Opracował: Patryk Uzarowski

Informatyka Techniczna, Wydział Informatyki i Telekomunikacji

Termin oddania projektu: 15 grudzień 2022

1. Wstęp

Celem drugiego zadania projektowego było opracowanie algorytmów heurystycznych rozwiązujących przybliżenie problem Komiwojażera (Travelling Salesman Problem). Problem ten opiera się na znalezieniu najbardziej optymalnej drogi z i do wierzchołka początkowego poprzez wszystkie pozostałe wierzchołki. Problem polega więc na odnalezieniu cyklu Hamiltona o najmniejszej sumie wag krawędzi.
Założenia dotyczące problemu Komiwojażera:

a) Asymetryczny – drogi docelowe oraz powrotne nie muszą być sobie równe ((i => j) != (j => i)).
b) Droga z miasta (i => i) może być reprezentowana jako 0 / 9999 / -1.
c) Wierzchołek 0 jest punktem startowym.

Założenia dotyczące struktury projektu:

a) Program napisany jest w podejściu Obiektowym wykorzystując język C++.
b) Program wyposażony jest w Meny Główne, w którym można dokonać wyboru algorytmu, wczytać macierz sąsiedztwa rozpatrywanego grafu lub wygenerować ją losowo. W pod menu algorytmów można również dokonać wyboru kryterium stopu (czasu maksymalnego egzekucji algorytmu) oraz wyboru sposobu generacji sąsiedztwa.
c) Algorytmy egzekwowane są przez klasy:
   Wyzarzanie – realizacja algorytmu Symulowanego Wyzarzania,
   TabuSearch – realizacja algorytmu Tabu Search.
   Dodatkowo do najważniejszych klas w projekcie należy również klasa **Menu**, implementująca prosty interfejs użytkownika w konsoli, **Matrix** reprezentująca strukturę macierzy sąsiedztwa rozważanego grafu oraz **Result**, pomocnicza klasa przechowująca pary: <Koszt_najlepszego_rozwązania, ścieżka_najlepszego_rozwiązania>
d) W przypadku każdego algorytmu argumentem wejściowym jest macierz sąsiedztwa reprezentująca rozpatrywany graf np. :

wierzchołek	0	1	2	3
0	INF	3	4	7
1	4	INF	6	9
2	2	5	INF	3
3	8	1	6	INF

2. Algorytm Symulowanego Wyżarzania

Metoda Symulowanego wyżarzania opiera swoje działanie na głównych składowych algorytmu jakimi jest temperatura algorytmu oraz wzór określający prawdopodobieństwo wyżarzone dopuszczające gorsze wyniki do puli rozważań promując tym samym eksplorację szerszego przedziału rozwiązań. Eksploatacja przedziału jest z kolei szeregiem przeszukiwań określonego zakresu dziedziny w celu znalezienia najlepszego wyniku lokalnego. Algorytm symulowanego wyżarzania jest więc zbiorem przeszukiwań lokalnych połączonych z możliwym „przejściem” do

kolejnych przedziałów lokalnych (eksploracja) uzależnionych od aktualnej temperatury algorytmu.
Uwzględnienie sąsiada rozważanego w przypadku lepszego wyniku wykonywane było automatycznie, lecz w przypadku „przeskoku" do sąsiada dającego gorszy wynik, jego akceptacja odbywała się z prawdopodobieństwem uwarunkowanym aktualną temperaturą algorytmu oraz różnicy kosztów rozwiązania i jego sąsiada:

$$annealing_{prob} = \mathrm{e}^{(\frac{-(różnica-kosztów)}{aktualna-temperatura})}$$

Następnie należało wygenerować zmienną pseudo-losową z przedziału <0.0, 1.0>,
W przypadku gdy stadnard_prob < annealing_prob, akceptowaliśmy gorszy wynik i promowaliśmy eksplorację nowego przedziału. Im większy koszt przejścia, tym mniejsze prawdopodobieństwo eksploracji, im wyższa temperatura tym większe prawdopodobieństwo akceptacji. Temperatura spada wraz z każdą iteracją algorytmu uwzględniając współczynnik chłodzenia „cooler". Temperatura maksymalna (początkowa) algorytmu została zaimplementowana poprzez maksymalną różnicę kosztów 120 losowych sąsiadów.
Adnotacja: W algorytmie została również zaimplementowana epoka oddzielająca właściwe procedury algorytmu od manipulacji temperaturą, epoka została zaimplementowana poprzez wyznaczenia eksperymentalne, w których zauważono, że nawet bardzo małe wartości epoki wpływają na polepszenie jakości wyników przy większych grafach (powyżej ok. 120-130 wierzchołków).

3. Algorytm Przeszukiwań Tabu Search

Kolejnym podejściem rozwiązania problemu TSP była implementacja metody Tabu Search. Jej działanie opiera się na przeszukiwaniu dziedziny sąsiadów rozważanej ścieżki w celu odnalezienia najlepszego (w przypadku TSP najmniejszego) kosztu ścieżki (sąsiada). Każde zaakceptowane rozwiązanie dodatkowo rozbudowywało tabelę ruchów zakazanych tabuList, w przypadku akceptacji danego przejścia, składowe (numery wierzchołków przejścia) trafiały na listę tabu wraz z wartością – kadencją określającą na ile iteracji algorytmu zadane przejście jest zabronione. Każda pełna iteracja algorytmu stopniowo zmniejszała kadencję wszystkich elementów, w przypadku jeżeli kadencja danego przejścia osiągnęła zero, dane połączenie usuwano z listy tabu oraz ponownie akceptowano sąsiadów z rozważanym przejściem.
Dodatkowo w algorytmie została uwzględniona strategia aspiracji (funkcja aspiracji) akceptująca danego sąsiada (pomimo przejścia znajdującego się na liście tabu) w wypadku gdy przejście jest lepsze niż najbardziej optymalne.
Lista tabu została określona jako macierz przejść tabu_list[źródło_przejścia][cel_przejścia]. Macierz jest wypełniona wartościami „-1" co implikuje, że dane przejście i -> j nie jest zabronione, w przeciwnym przypadku na adekwatnych indeksach znajduje się aktualny stan kadencji danego przejścia tabu.
Adnotacja: Dodatkowo w przypadku mniejszych grafów (np. br17), zauważono, że zaimplementowany algorytm bez strategii dywersyfikacji szybko znajduje otoczenie rozwiązania, jednak nie jest się w stanie z niego „uwolnić" i tym samym rozpocząć

eksploatacji innego przedziału (zbioru) sąsiadów. Do eksperymentu napisano pomocniczy, bardzo uproszczony model dywersyfikacji, który poprawił wyniki dla pomniejszych grafów, model jest zaimplementowany dla grafów poniżej 25 wierzchołków w kodzie aplikacji, jest to bardzo uproszczona wersja, w której funkcja restartu generuje nowy przedział poszukiwań przez losową generację rozwiązania (podobnie jak w przypadku rozruchu algorytmu).

4. Generacja sąsiedztwa

Rozważane algorytmy implementują trzy sposoby generacji sąsiedztwa:

4.1 Sąsiedztwo Swap

Najprostszy typ sąsiedztwa generowany poprzez zamianę dwóch wierzchołków rozważanej ścieżki, przykładowo dla cyklu {1,3,5,7,9} => swap(1,7) tworzy sąsiada dla rozważanego rozwiązania => {7,3,5,1,9}

4.2 Sąsiedztwo Insert

Generacja sąsiada polega na wstawieniu wierzchołka i w miejsce wierzchołka j oraz adekwatnego przesunięcia reszty cyklu, przykładowo dla cyklu {1,3,5,7,9} => Insert(1,7) tworzy sąsiada rozważanego rozwiązania {3,5,1,7,9}.

4.3 Sąsiedztwo Invert

Generacja sąsiada polega na zestawie swapów całego przedziału pomiędzy wierzchołkiem I oraz J adekwatnie dla kolejnych elementów swap((i+1)(j-1)), Swap((i+2)(j-2)) itp. Przykładowo dla cyklu {1,3,5,7,9} => Invert(3,9), tworzy sąsiada {1,9,7,5,3}.

5. Plan eksperymentu

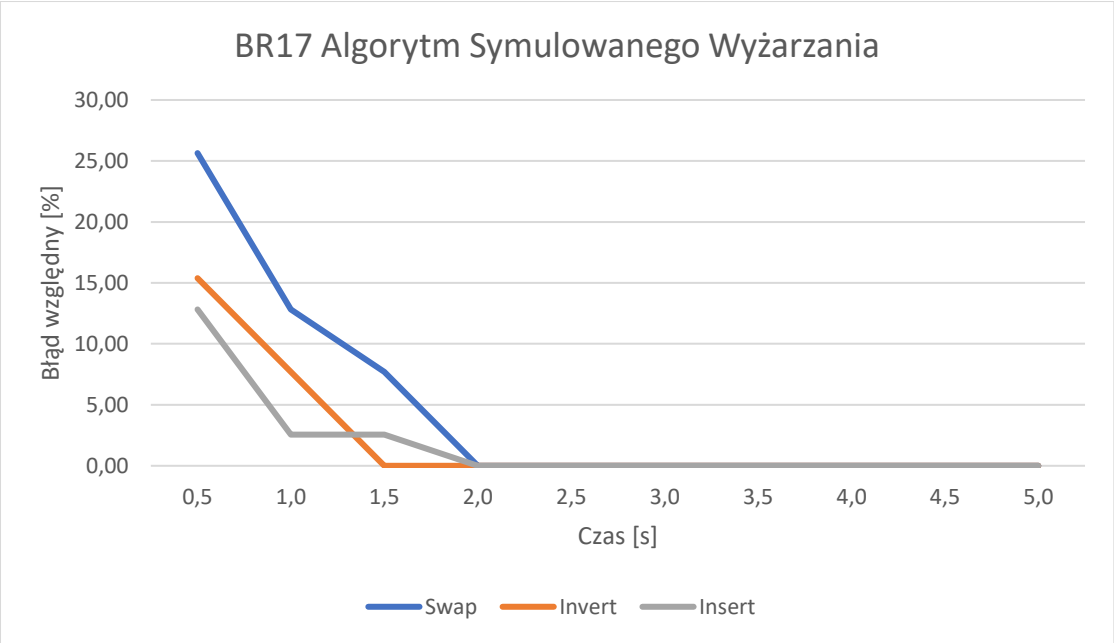
Zgodnie z założeniami projektowymi testy zostały przeprowadzone dla trzech grafów: Br17, Ftv55, Ftv170. Temperatura minimalna symulowanego wyżarzania została ustawiona na 0.01 , ze współczynnikiem chłodzenia 0.999999. Epoka minimalna ustawiona na 2 iteracje. W tabuSearch kadencja każdego elementu z listy wynosi number_of_vertices – ilość wierzchołków rozważanego grafu. Każdy z pomiarów trwał 60 [s]. Poniższe tabele prezentują wyniki pomiarów dla obydwu algorytmów z różnymi metodami generacji sąsiedztwa. Dla grafu br17 w przypadku algorytmu Tabu Search został zaimplementowany uproszczony model dywersyfikacji, spełniający jednak swoje założenie dla mniejszych problemów. Maksymalna liczba iteracji w podejściu z dywersyfikacją dla br17 wynosiła 20000 iteracji (wartość eksperymentalna).

Graf BR17:

Najlepszy znany wynik: 39

Symulowane Wyżarzanie:

	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
0,5	49	25,64	45	15,38	44	12,82
1,0	44	12,82	42	7,69	40	2,56
1,5	42	7,69	39	0,00	40	2,56
2,0	39	0,00	39	0,00	39	0,00
2,5	39	0,00	39	0,00	39	0,00
3,0	39	0,00	39	0,00	39	0,00
3,5	39	0,00	39	0,00	39	0,00
4,0	39	0,00	39	0,00	39	0,00
4,5	39	0,00	39	0,00	39	0,00
5,0	39	0,00	39	0,00	39	0,00



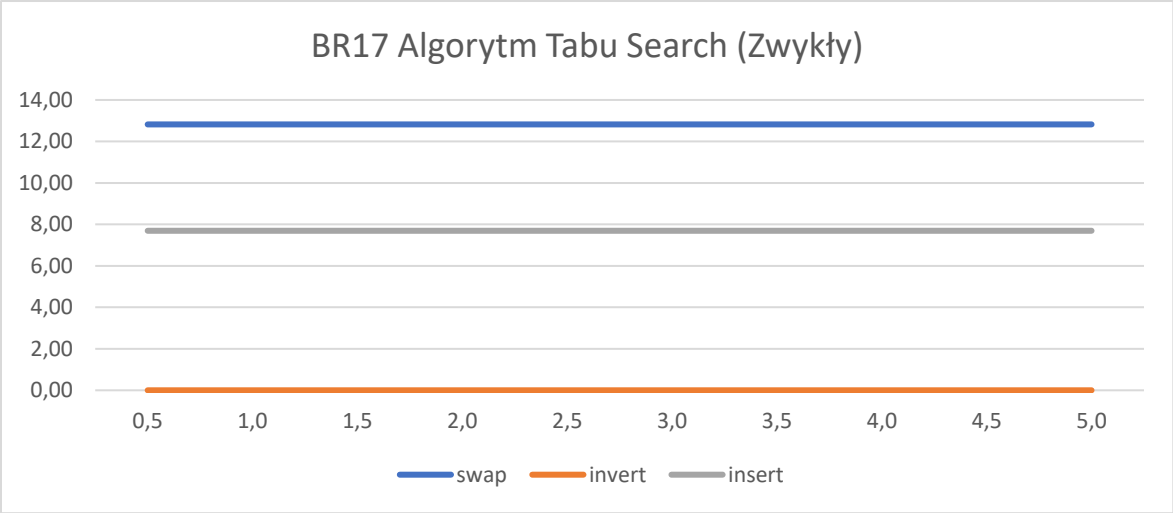
Tabu Search:

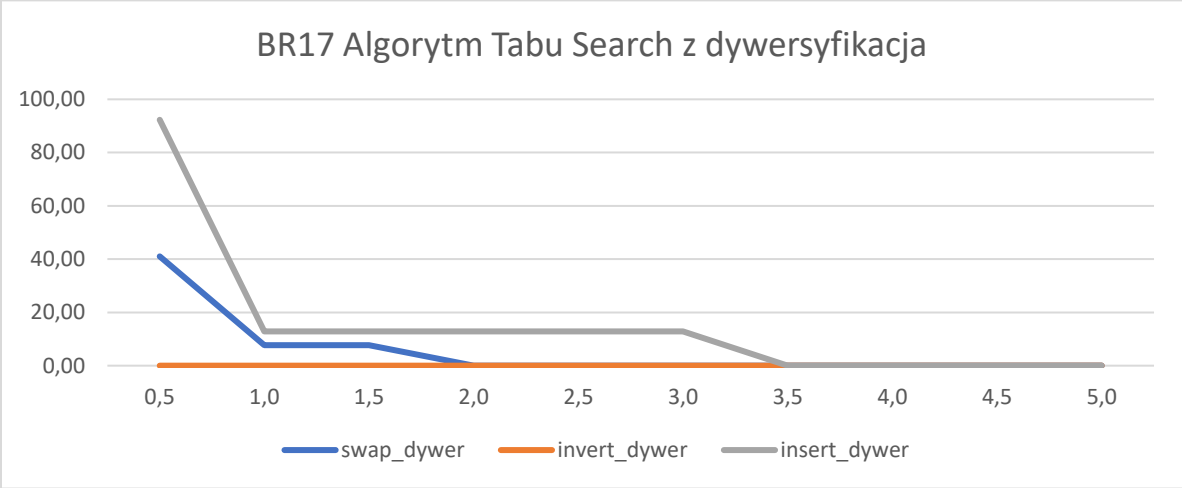
Bez dywersyfikacji:

	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
0,5	44	12,82	39	0	42	7,69
1,0	44	12,82	39	0	42	7,69
1,5	44	12,82	39	0	42	7,69
2,0	44	12,82	39	0	42	7,69
2,5	44	12,82	39	0	42	7,69
3,0	44	12,82	39	0	42	7,69
3,5	44	12,82	39	0	42	7,69
4,0	44	12,82	39	0	42	7,69
4,5	44	12,82	39	0	42	7,69
5,0	44	12,82	39	0	42	7,69

Z prostą dywersyfikacją:

	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
0,5	55	41,03	39	0	75	92,31
1,0	42	7,69	39	0	44	12,82
1,5	42	7,69	39	0	44	12,82
2,0	39	0,00	39	0	44	12,82
2,5	39	0,00	39	0	44	12,82
3,0	39	0,00	39	0	44	12,82
3,5	39	0,00	39	0	39	0,00
4,0	39	0,00	39	0	39	0,00
4,5	39	0,00	39	0	39	0,00
5,0	39	0,00	39	0	39	0,00





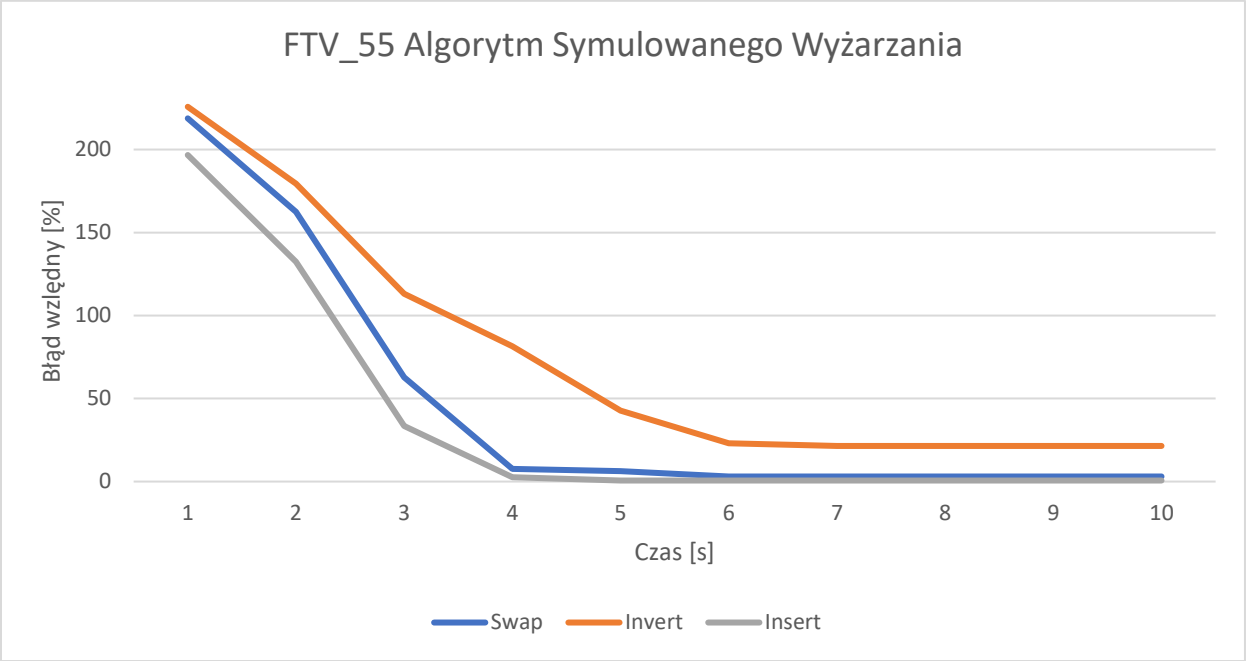
Graf FTV55:

Najlepsze znane rozwiązanie: 1608

Symulowane Wyżarzanie:

	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
1,0	5126	218,78	5238	225,75	4771	196,70
2,0	4222	162,56	4494	179,48	3738	132,46
3,0	2616	62,69	3427	113,12	2146	33,46
4,0	1730	7,59	2917	81,41	1651	2,67

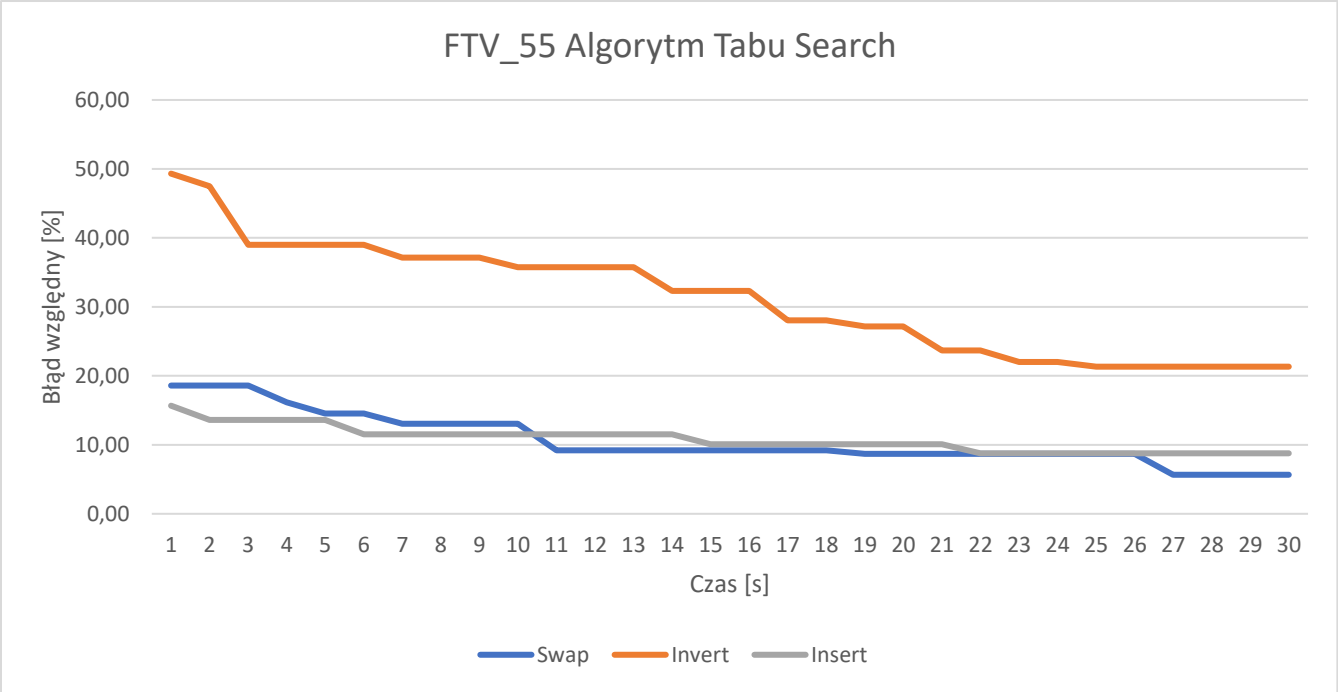
5,0	1709	6,28	2295	42,72	1618	0,62
6,0	1657	3,05	1980	23,13	1618	0,62
7,0	1657	3,05	1953	21,46	1618	0,62
8,0	1657	3,05	1953	21,46	1618	0,62
9,0	1657	3,05	1953	21,46	1618	0,62
10,0	1657	3,05	1953	21,46	1618	0,62



Tabu Search

	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
1,0	1907	18,59	2401	49,32	1860	15,67
2,0	1907	18,59	2372	47,51	1827	13,62
3,0	1907	18,59	2235	38,99	1827	13,62
4,0	1868	16,17	2235	38,99	1827	13,62

5,0	1842	14,55	2235	38,99	1827	13,62
6,0	1842	14,55	2235	38,99	1793	11,50
7,0	1818	13,06	2205	37,13	1793	11,50
8,0	1818	13,06	2205	37,13	1793	11,50
9,0	1818	13,06	2205	37,13	1793	11,50
10,0	1818	13,06	2183	35,76	1793	11,50
11,0	1756	9,20	2183	35,76	1793	11,50
12,0	1756	9,20	2183	35,76	1793	11,50
13,0	1756	9,20	2183	35,76	1793	11,50
14,0	1756	9,20	2128	32,34	1793	11,50
15,0	1756	9,20	2128	32,34	1770	10,07
16,0	1756	9,20	2128	32,34	1770	10,07
17,0	1756	9,20	2059	28,05	1770	10,07
18,0	1756	9,20	2059	28,05	1770	10,07
19,0	1748	8,71	2045	27,18	1770	10,07
20,0	1748	8,71	2045	27,18	1770	10,07
21,0	1748	8,71	1989	23,69	1770	10,07
22,0	1748	8,71	1989	23,69	1749	8,77
23,0	1748	8,71	1962	22,01	1749	8,77
24,0	1748	8,71	1962	22,01	1749	8,77
25,0	1748	8,71	1951	21,33	1749	8,77
26,0	1748	8,71	1951	21,33	1749	8,77
27,0	1699	5,66	1951	21,33	1749	8,77
28,0	1699	5,66	1951	21,33	1749	8,77
29,0	1699	5,66	1951	21,33	1749	8,77
30,0	1699	5,66	1951	21,33	1749	8,77

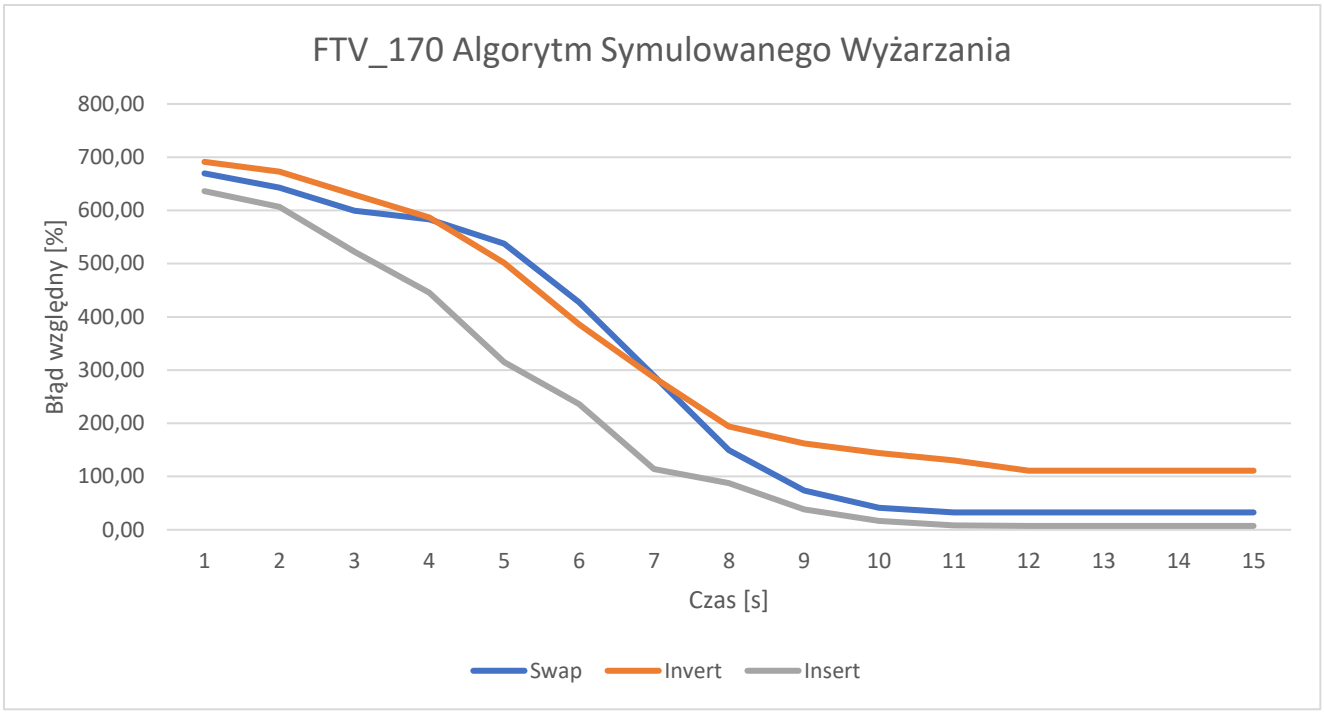


Graf FTV170:

Najlepsze znane rozwiązanie: 2755

Symulowane Wyżarzanie:

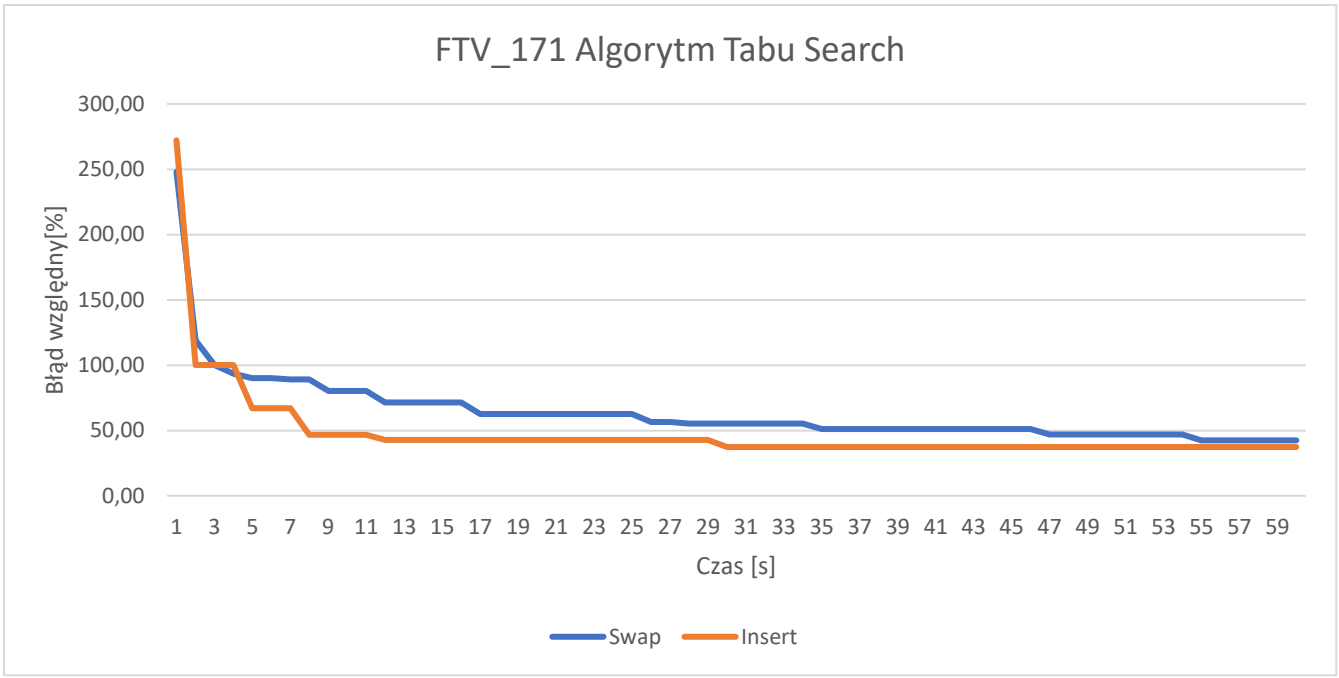
	Sąsiedztwo Swap		Sąsiedztwo Invert		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
1,0	21201	669,55	21798	691,22	20282	636,19
2,0	20461	642,69	21298	673,07	19473	606,82
3,0	19271	599,49	20093	629,33	17149	522,47
4,0	18826	583,34	18920	586,75	15025	445,37
5,0	17558	537,31	16567	501,34	11425	314,70
6,0	14537	427,66	13393	386,13	9263	236,23
7,0	10727	289,36	10638	286,13	5905	114,34
8,0	6877	149,62	8098	193,94	5170	87,66
9,0	4792	73,94	7221	162,11	3816	38,51
10,0	3887	41,09	6725	144,10	3214	16,66
11,0	3654	32,63	6347	130,38	2975	7,99
12,0	3654	32,63	5813	111,00	2955	7,26
13,0	3654	32,63	5813	111,00	2955	7,26
14,0	3654	32,63	5813	111,00	2955	7,26
15,0	3654	32,63	5813	111,00	2955	7,26



Tabu Search:

	Sąsiedztwo Swap		Sąsiedztwo Insert	
Czas [s]	Wynik	Błąd względny [%]	Wynik	Błąd względny [%]
1,0	9587	247,99	10252	272,12
2,0	6032	118,95	5514	100,15
3,0	5514	100,15	5514	100,15
4,0	5328	93,39	5514	100,15
5,0	5234	89,98	4603	67,08
6,0	5234	89,98	4603	67,08
7,0	5209	89,07	4603	67,08
8,0	5209	89,07	4040	46,64
9,0	4968	80,33	4040	46,64
10,0	4968	80,33	4040	46,64
11,0	4968	80,33	4040	46,64
12,0	4724	71,47	3934	42,79
13,0	4724	71,47	3934	42,79
14,0	4724	71,47	3934	42,79
15,0	4724	71,47	3934	42,79
16,0	4724	71,47	3934	42,79
17,0	4480	62,61	3934	42,79
18,0	4480	62,61	3934	42,79
19,0	4480	62,61	3934	42,79
20,0	4480	62,61	3934	42,79

21,0	4480	62,61	3934	42,79
22,0	4480	62,61	3934	42,79
23,0	4480	62,61	3934	42,79
24,0	4480	62,61	3934	42,79
25,0	4480	62,61	3934	42,79
26,0	4311	56,48	3934	42,79
27,0	4311	56,48	3934	42,79
28,0	4276	55,21	3934	42,79
29,0	4276	55,21	3934	42,79
30,0	4276	55,21	3784	37,35
31,0	4276	55,21	3784	37,35
32,0	4276	55,21	3784	37,35
33,0	4276	55,21	3784	37,35
34,0	4276	55,21	3784	37,35
35,0	4165	51,18	3784	37,35
36,0	4165	51,18	3784	37,35
37,0	4165	51,18	3784	37,35
38,0	4165	51,18	3784	37,35
39,0	4165	51,18	3784	37,35
40,0	4165	51,18	3784	37,35
41,0	4165	51,18	3784	37,35
42,0	4165	51,18	3784	37,35
43,0	4165	51,18	3784	37,35
44,0	4165	51,18	3784	37,35
45,0	4165	51,18	3784	37,35
46,0	4165	51,18	3784	37,35
47,0	4051	47,04	3784	37,35
48,0	4051	47,04	3784	37,35
49,0	4051	47,04	3784	37,35
50,0	4051	47,04	3784	37,35
51,0	4051	47,04	3784	37,35
52,0	4051	47,04	3784	37,35
53,0	4051	47,04	3784	37,35
54,0	4051	47,04	3784	37,35
55,0	3924	42,43	3784	37,35
56,0	3924	42,43	3784	37,35
57,0	3924	42,43	3784	37,35
58,0	3924	42,43	3784	37,35
59,0	3924	42,43	3784	37,35
60,0	3924	42,43	3784	37,35



6. Wnioski

6.1 Omówienie przeprowadzonego eksperymentu

Algorytm symulowanego wyżarzania jest bardzo dobrym sposobem na wyznaczenie bardzo jakościowego wyniku w niedługim czasie działania algorytmu. Podczas eksperymentów, manipulowano składowymi algorytmu takimi jak parametrem chłodzenia, zwiększeniem epoki czy coraz większego kryterium stopu. Powyżej przedstawione wyniki, zostały dobrane dla parametrów, dzięki którym w relatywnie niedługim czasie algorytm był w stanie podać wyniki nawet dla dużych instancji problemów na poziomie poniżej 8%. Najlepszym wyborem strategii generacji sąsiedztwa okazał się typ Insert, który już dla średniej wielkości problemu znacząco przewyższał sąsiedztwo Swap i Invert względem jakości końcowego wyniku. Sąsiedztwo Invert chociaż dużo szybciej rozwiązywało problem TSP dla najmniejszej instancji, to jednak im większy był graf wejściowy, tym jakość rozwiązania końcowego zaczęła drastycznie spadać. Na podstawie powyższych eksperymentów została również opracowane dodanie epoki do kodu Algorytmu Symulowanego Wyżarzania, które bardzo wpływało na jakość rozwiązania zwłaszcza dla sąsiedztwa swap oraz insert, epoka jest bardzo mała przez co nie wpływa znacząco na czas egzekucji algorytmu, z kolei jej efekty można dopiero zauważyć przy większych instancjach problemu.

Algorytm Tabu Search został zaimplementowany w dosyć uproszczonej wersji, w której nie występuje ogólna implementacja metody dywersyfikacji wyników. Algorytm dobrze radził sobie ze średnimi instancjami problemu, w których jakość wyników dla sąsiedztwa swap oraz insert wynosiła 5-9 %, podobnie jak w przypadku algorytmu Symulowanego Wyżarzania sąsiedztwo Invert wypadło tutaj najgorzej, szczególnie w przypadku grafu FTV170, w którym zdecydowanie go nie

uwzględniać ze względu na zbyt duże rozrzuty rezultatów (algorytm szybko spadał do otoczenia +/- 200% gorszego, oraz nie mógł go opuścić, przykładowa metoda dywersyfikacji nie została dobrze zoptymalizowana pod większe instancje problemów i nie wpływała na polepszenie wyników). Sąsiedztwo Invert znalazło jednak inne zastosowanie, dla mniejszych instancji problemów znajduje optymalny wynik dużo szybciej (W większości przeprowadzonych testów było to jeszcze podczas 1 sekundy egzekucji algorytmu), kiedy sąsiedztwo Swap oraz Insert bardzo często blokowały się w konkretnym przedziale eksploatacji, w przypadku sąsiedztwa Invert ten problem niemalże nigdy nie wystąpił. Warto mieć to na uwadze w przypadku kiedy chcielibyśmy zastosować metodę Tabu Search również dla mniejszych instancji problemów.

6.2 Wnioski

Ciężko porównać bezpośrednio oba rozwiązania, gdyż zaimplementowano jedynie uproszczoną wersję algorytmu podejścia Tabu Search. Biorąc pod uwagę jednak rozrzuty wyników jednakowej metody względem dobranego sposobu generowania sąsiadów, bardzo możliwe, że istnieje dużo bardziej optymalny sposób generacji sąsiedztwa dla algorytmu Tabu Search. Sąsiedztwo Insert wypada bardzo dobrze przy implementacji z algorytmem symulowanego wyżarzania, nie udało się jednak odnaleźć adekwatnego odpowiednika dla algorytmu Tabu Search. Wyniki dla dużych instancji problemu jak graf FTV170 są jednak bardzo zadowalające, uwzględniając przede wszystkim złożoność czasową algorytmów deterministycznych, które nie są w stanie rozwiązać tak dużych problemów w rozsądnym czasie.

Etap 2 projektu nauczył podstawowych zagadnień na temat algorytmów heurystycznych oraz zaoferował sporo satysfakcji związanej z ich optymalizacją.