Projektowanie efektywnych algorytmów

Zadanie 2: Implementacja i analiza efektywności algorytmu Tabu Search i symulowanego wyżarzania dla problemu Komiwojażera

Autor: Michał Sikacki (259152)

Grupa projektowa: czwartek 13:15- 15:00

Prowadzący: mgr inż. Antoni Sterna

1. Wstęp teoretyczny

Algorytmy przedstawione w tym zadaniu mają za zadanie rozwiązać problem Komiwojażera. W przeciwieństwie jednak do algorytmów przedstawionych w zadaniu 1 tzn. B&B i DP, nie zwracają one zawsze najlepszego wyniku. Algorytmy symulowanego wyżarzania i tabu search stosują podejście probabilistyczne. Ich celem jest znalezienie rozwiązania, które będzie najbardziej bliskie idealnemu. Techniki te wykorzystywane są głównie dla problemów, dla których znalezienie rozwiązania najbardziej optymalnego ze wszystkich nie jest możliwe w rozsądnym czasie. Dla TSP próba znalezienia najkrótszej drogi dla 100 miast używając algorytmów tj. B&B jest niemożliwa. Używając algorytmów symulowanego wyżarzania i tabu search, możemy jednak zbliżyć się do optimum globalnego.

a) Symulowane wyżarzanie

Metoda ta inspirowana jest na wyżarzaniu w metalurgii. Polega ono na podgrzaniu materiału i następnie jego schładzaniu tak aby zmniejszyć jego defekty. Algorytm ten polega na losowaniu rozwiązań i szukaniu w każdej iteracji algorytmu lepszego niż dotychczasowe. Jeśli zostanie jednak znalezione gorsze rozwiązanie, to może one zostać przyjęte jako nowe rozwiązanie z pewnym prawdopodobieństwem. Algorytm symulowanego wyżarzania przyjmuje różne parametry. Najważniejszym z nich jest temperatura. Tym wyższa, tym większe prawdopodobieństwo, że zostanie zaakceptowane gorsze rozwiązanie. Z drugiej jednak strony istnieje również większa szansa na przemieszczanie się po całej przestrzeni rozwiązań. W algorytmie tym bierzemy również pod uwagę różnice pomiędzy dotychczasowym rozwiązaniem „x" a nowym znalezionym w danej iteracji pętli „y":

$$\delta = g(y) - g(x)$$

Tym wyższa jest ta różnica tym mniejsza jest szansa, że nowe, gorsze rozwiązanie „y" zostanie zaakceptowane.

Prawdopodobieństwo zaakceptowania gorszego rozwiązania prezentuje się więc następująco:

$$\mathrm{P} = \exp\left(-\frac{\delta}{T}\right)$$

Istotnym elementem wykorzystywanym w tym algorytmie jest również sąsiedztwo. Definiujemy je po to aby nie poruszać się po całej przestrzeni rozwiązań. Jego zdefiniowanie pozwala nam określić w jaki sposób będzie losowane kolejne nowe, potencjalne rozwiązanie. Istnieją trzy podstawowe typy sąsiedztwa, które można zaprezentować dla problemu TSP. Załóżmy, że mamy sekwencję wierzchołków:

$$\pi = < \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) >$$

W sąsiedztwie typu insert po wykonaniu ruchu (i, j), sekwencja będzie wyglądać następująco:

$$\pi =< \pi(1), \dots, \pi(i-1), \pi(j), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j+1), \dots, \pi(n) >$$

W sąsiedztwie typu swap:

$$\pi =< \pi(1), \dots, \pi(i-1), \pi(j), \pi(i+1), \dots, \pi(j-1), \pi(i), \pi(j+1), \dots, \pi(n) >$$

W sąsiedztwie typu invert:

$$\pi =< \pi(1), \dots, \pi(i-1), \pi(j), \pi(j-1), \dots, \pi(i+1), \pi(i), \pi(j+1), \dots, \pi(n) >$$

Wszystkie parametry, które są ustalane przed uruchomieniem algorytmu, należy dobrać w zależności od typu problemu i jego złożoności. W związku z tym musimy posiadać wiedze o problemie. Nie istnieje niestety jeden schemat doboru tych parametrów.

Ogólny schemat działania algorytmu przedstawia się następująco:

Wyznacz rozwiązanie początkowe x_0

Wyznacz temperaturę początkową T_0

Powtarzaj:

    Dla i = 0 to L:

        Wyznacz w sąsiedztwie x_0 rozwiązanie y

            $\delta = g(y) - g(x) < 0$ wtedy x = y

        W przeciwnym przypadku:

            Wylosuj s z zakresu (0,1)

            Jeżeli $s < e^{-\delta/T}$ wtedy x = y

    T = α(T)

Dopóki warunek zatrzymania

Zwróć rozwiązanie x

W powyższym algorytmie widzimy dodatkowe takie elementy jak α(T). Jest to schemat chłodzenia tzn. funkcja określająca, w jaki sposób ma obniżać się temperatura. Dodatkowo L oznacza tutaj długość epoki. Jest to parametr określający przez jak dużą ilość iteracji temperatura nie ma być zmniejszana. Warunek zatrzymania określa moment, w którym algorytm ma przerwać pracę. Mogą to być różne warunki np. czas wykonania algorytmu, ilość iteracji pętli, przekroczenie minimalnej temperatury albo osiągniecie satysfakcjonującego rozwiązania.

b) Tabu Search- przeszukiwanie z zakazami

Jest to metaheurystyka, którą można uznać za swojego rodzaju strategię przeszukiwania przestrzeni. Dokonuje się to za pomocą odpowiedniej sekwencji ruchów, które dają zwykle coraz lepsze rozwiązanie. Aby nie powtarzać dokonanych do tej pory ruchów, definiuje się tzn. listę tabu, która przechowuje ruchy zakazane, tzn. takie, które na danym etapie nie można wykonać.

Sposób działania tego algorytmu zależy od sposobu implementacji. W najprostszej postaci raczej nie daje wysoce optymalne wyniki.

W algorytmie tym głównym pojęciem jest tzn. lista tabu. Przechowuje ona listę ruchów zakazanych. Są to najczęściej ruchy, które dawały najbardziej optymalne rozwiązania w danym sąsiedztwie. Umieszczamy je na liście, żeby algorytm nie wpadł w cykl w okolicach lokalnego minimum. Kolejnym ważnym pojęciem jest kadencja, która określa, jak długo dany element będzie na liście tabu. Jej długość ma istotny wpływ na to, jak przeszukiwana będzie przestrzeń. Jeśli zależy nam na większej dokładności przeszukiwania, należy wybrać krótką kadencję. Niemniej trzeba pamiętać, że za zbyt krótka kadencja może czynić listę tabu bezużyteczną. Jeśli jednak kadencja będzie dłuższa, zwiększamy dywersyfikację tzn. zakres przeszukiwania się zwiększa. Tutaj jednak trzeba pamiętać, że tracimy przy tym jakość rozwiązania (zbyt intensywne skakanie po przestrzeni rozwiązań przypomina wybieranie rozwiązania losowo).

Innym istotnym pojęciem jest kryterium aspiracji. Określa ono, kiedy dany ruch może zostać zaakceptowany pomimo, że jest na liście tabu. Jest ono stosowane w sytuacjach nadzwyczajnych, najczęściej wtedy jeśli znalezione rozwiązanie po wykonaniu określonego ruchu jest najlepsze z dotychczas znalezionych.

Algorytm ten w celu znajdowania najlepszych ruchów przeszukuje sąsiedztwo danego rozwiązania. Tutaj ponownie jak w przypadku symulowanego wyżarzania mamy trzy typy sąsiedztwa: swap, insert oraz invert.

Ogólny schemat algorytmu przedstawia się następująco:

wybierz lub wylosuj punkt startowy $x0 \in X$

$xopt \leftarrow x0$

tabu list $\leftarrow \emptyset$

repeat

znajdź $x \in N(x0)$, dla którego mval(x0, x) jest największa

$x0 \leftarrow x$

if $f(x0) > f(xopt)$ then

```
            xopt ← x0
        zweryfikuj tabu list
        ∀element ∈ tabu list do
            − − kadencjai
            if kadencjai = 0 then
                usuń element(atrybuti, kadencjai) z tabu list
        if CriticalEvents() then
            x0 ← Restart() (Dywersyfikacja)
        if f(x0) > f(xopt) then
            xopt ← x0
until warunek zakończenia
```

2. Opis implementacji algorytmu

a) Symulowane wyżarzanie

Klasa w projekcie, w której zostały napisane wszystkie funkcje i procedury potrzebne dla algorytmu symulowanego wyżarzania znajduje się w plikach SimAnn.cpp oraz SimAnn.hpp. W konstruktorze tej klasy definiowana jest przede wszystkim początkowa temperatura. W funkcji setParameters() można określić przede wszystkim warunek zatrzymania (czas w sekundach), typ sąsiedztwa oraz współczynnik temperatury „a" w schemacie chłodzenia (użyty został schemat geometryczny, gdzie współczynnik „a" jest podniesiony do potęgi pierwszej). Funkcje calculateCurrentValue() oraz calculateCurrentValueForThis() obliczają dla danej sekwencji wierzchołków drogę. W funkcji neighbourhood() na postawie wybranego typu sąsiedztwa dokonywana jest transformacja. Zostały zaimplementowane dwa typy sąsiedztwa: „swap" oraz „invert". W funkcji cooling() jest wybrany odpowiedni schemat chłodzenia. Funkcja algorithm() realizuje algorytm symulowanego wyżarzania na podstawie podanego w części teoretycznej schematu. Do pomiaru czasu wykonywania algorytmu została wykorzystana biblioteka <chrono>. W trakcie wykonywania algorytmu w podanym przez użytkownika czasie sprawdzamy, jakie rozwiązanie jest na danym etapie. Funkcja showResult() zwraca wynik (znaleziona ścieżka i wartość drogi) i opcjonalnie pokazuje, jak wynik zmieniał się w poszczególnych chwilach pracy algorytmu (na potrzeby dalszego eksperymentu).

b) Tabu Search

Klasa w projekcie, w której zostały napisane wszystkie funkcje i procedury potrzebne dla algorytmu symulowanego wyżarzania znajduje się w plikach TabuSearch.cpp oraz TabuSearch.hpp. Ponownie jak w przypadku symulowanego wyżarzania definiujemy analogicznie działające funkcje takie jak: setParameters(), calculateCurrentValue(), calculateCurrentValueForThis() oraz neighbourhood(). Definiujemy nową funkcję o nazwie neighbourhoodLook(), która będzie przeszukiwać sąsiedztwo w poszukiwaniu najlepszego lokalnego rozwiązania oraz ruchu, który je wygenerował. Zostały zdefiniowane sąsiedztwa typu swap oraz invert. Funkcja algorithm() jest funkcją główną. Realizuje ona tabu search zgodnie ze schematem umieszczonym w części teoretycznej. Do pomiaru czasu wykonywania algorytmu została ponownie wykorzystana biblioteka <chrono>. W trakcie wykonywania algorytmu w podanym przez użytkownika czasie sprawdzamy, jakie rozwiązanie jest na danym etapie. Definiujemy analogicznie jak dla symulowanego wyżarzania funkcję showResult().

3. Plan eksperymentu.

W wybranych punktach pomiarowych (czasowych) będziemy zapisywali dotychczasową drogę jaką wyznaczył nasz algorytm. Dla każdego punktu pomiar wykonujemy 10 razy i obliczamy średnią arytmetyczną w celu poprawienia jakości uzyskanych wyników testów. Należy ją porównać z rozwiązaniem idealnym. W eksperymencie tym zostały wybrane więc trzy różne pliki zawierające graf tzn. br17.txt, ftv55.txt oraz ftv170.txt zawierające kolejno 17, 56 oraz 171 wierzchołków, dla których rozwiązanie optymalne jest znane. Porównanie to zostanie przedstawione jako błąd względny. Zatem podstawowe wykresy przedstawiać będą błąd względny w funkcji czasu wykonywania algorytmu.

Najlepsze znane rozwiązania dla plików:

- br17.txt – 39
- ftv55.txt – 1608
- ftv170.txt – 2755

a) Symulowane wyżarzanie:

Punkty pomiarowe zostały zdefiniowane następująco:
- dla pliku br17.txt: [0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.011, 0.012, 0.013, 0.014, 0.015, 0.016],
- dla pliku ftv55.txt: [0.015, 0.03, 0.045, 0.06, 0.075, 0.09, 0.105 0.12, 0.135, 0.15, 0.165, 0.18, 0.195, 0.21, 0.225, 0.24],
- dla pliku ftv170.txt: [0.32, 0.64, 0.96, 1.28, 1.6, 1.92, 2.24, 2.56, 2.88, 3.2, 3.52, 3.84, 4.16, 4.48, 4.8, 5.12]

Wybór różnych punktów pomiarowych dla plików wynikał z różnych czasów znajdowania optimum w zależności od ilości wierzchołków grafu.

Została zainicjalizowana temperatura początkowa, którą obliczamy na podstawie wzoru: droga_dla_początkowego_rozwiązania * 10. Została wybrana pewna liczba epok wynosząca (ilość_wierzchołków * (ilość_wierzchołków - 1)) / 2 * wybrany_współczynnik. Współczynnik ten został wybrany jako 0,25. Ustawienie długości epoki pozwala na znalezienia lepszego rozwiązania w sąsiedztwie. Współczynnik temperatury powinien być nie mniejszy niż 0,85, więc ustawiony on został na 0,99.
Na wykresach zostało pokazane porównanie wyników dla różnych użytych typów sąsiedztwa.

b) Tabu Search:

Punkty pomiarowe zostały zdefiniowane następująco:
- dla pliku br17.txt: [0.004, 0.008, 0.012, 0.016, 0.02, 0.024, 0.028, 0.032, 0.036, 0.04, 0.044, 0.048, 0.052, 0.056, 0.06, 0.064],
- dla pliku ftv55.txt: [0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6, 2.8, 3, 3.2],
- dla pliku ftv170.txt: [0.8, 1.6, 2.4, 3.2, 4, 4.8, 5.6, 6.4, 7.2, 8, 8.8, 9.6, 10.4, 11.2, 12, 12.8]
Powyżej wybranych punktów pomiarowych, algorytm zwykle nie zwracał już lepszego rozwiązania. Długość listy tabu została ustawiona jako ilość_wierzchołków / 2. Długość kadencji jest stała i wynosi ilość_wierzchołków * 2. Została wprowadzona dywersyfikacja, która uruchamia się jeśli po stu iteracjach od znalezienia rozwiązania, nadal nie doszło do jego poprawy. Przetestowane zostały sąsiedztwa typu swap i invert z dywersyfikacją i bez.

4. Wyniki eksperymentu.

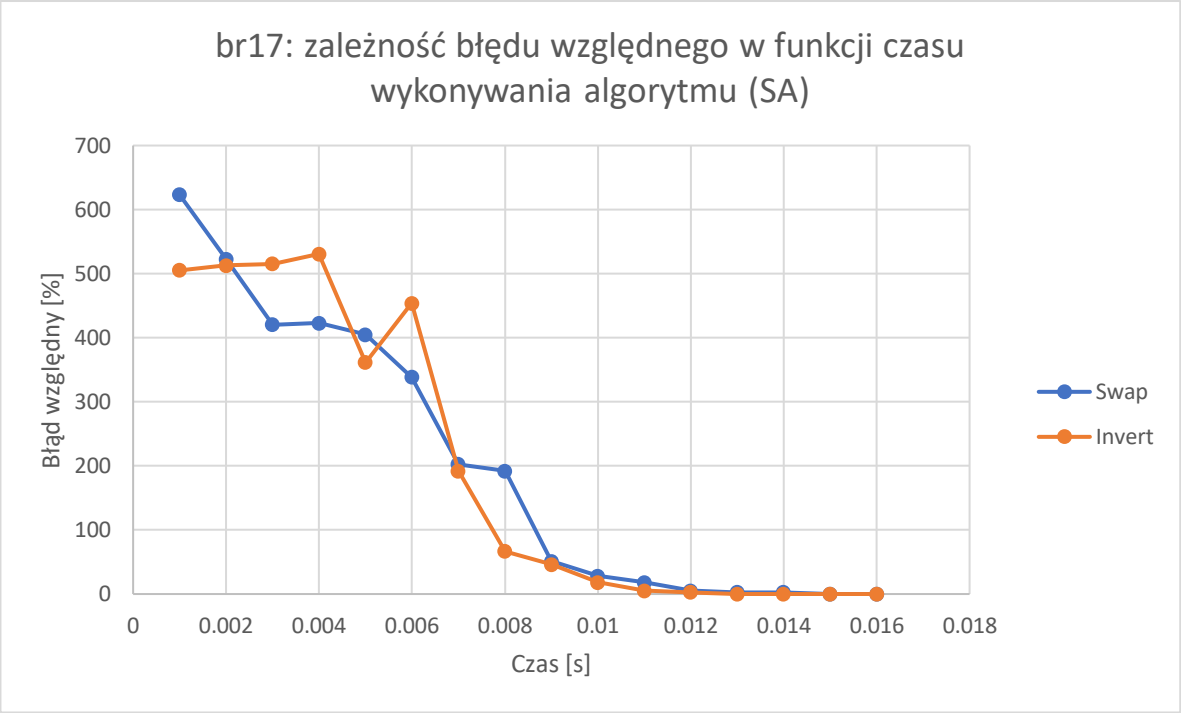
a) Symulowane wyżarzanie

Tabela 1. SA- Wyniki dla pliku testowego br17.txt (swap)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.001	282	623.0769231
2	0.002	243	523.0769231
3	0.003	203	420.5128205
4	0.004	204	423.0769231
5	0.005	197	405.1282051
6	0.006	171	338.4615385
7	0.007	118	202.5641026
8	0.008	114	192.3076923
9	0.009	59	51.28205128
10	0.01	50	28.20512821
11	0.011	46	17.94871795
12	0.012	41	5.128205128
13	0.013	40	2.564102564
14	0.014	40	2.564102564
15	0.015	39	0
16	0.016	39	0

Tabela 2. SA- Wyniki dla pliku testowego br17.txt (invert)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.001	236	505.1282051
2	0.002	239	512.8205128
3	0.003	240	515.3846154
4	0.004	246	530.7692308
5	0.005	180	361.5384615
6	0.006	216	453.8461538
7	0.007	114	192.3076923
8	0.008	65	66.66666667
9	0.009	57	46.15384615
10	0.01	46	17.94871795
11	0.011	41	5.128205128
12	0.012	40	2.564102564
13	0.013	39	0
14	0.014	39	0
15	0.015	39	0
16	0.016	39	0



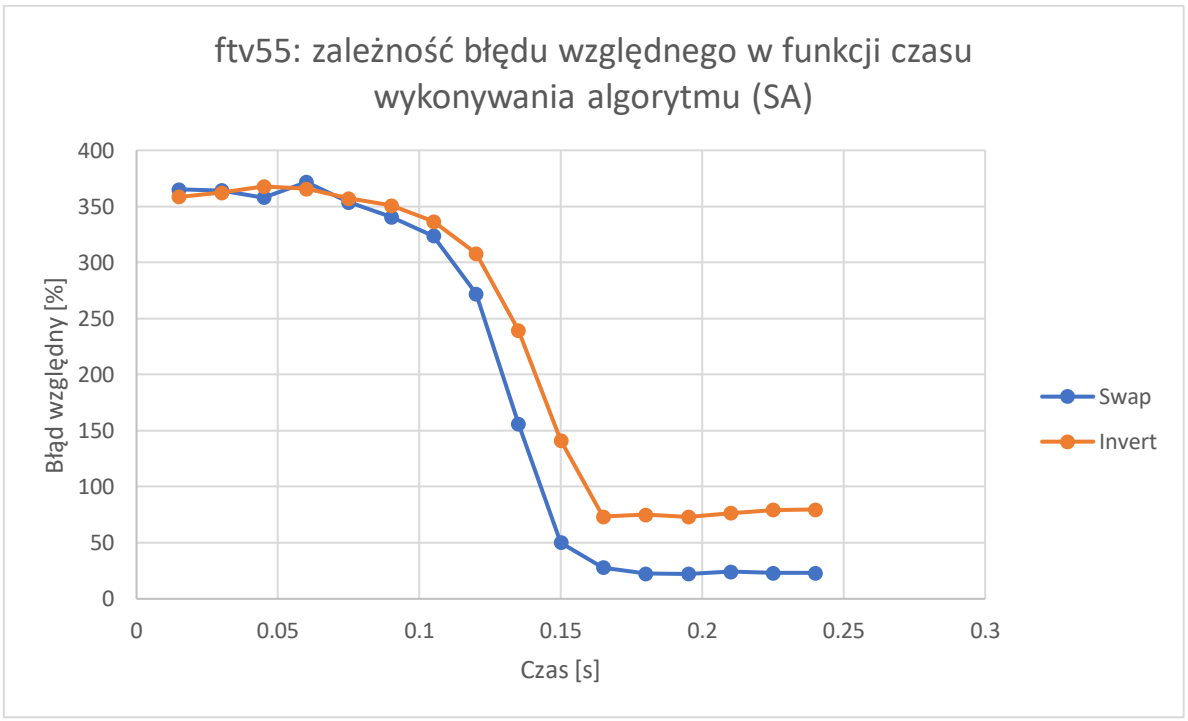
Rys. 1 SA- Wykres dla pliku testowego br17.txt

Tabela 3. SA- Wyniki dla pliku testowego ftv55.txt (swap)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.015	7477	364.9875622
2	0.03	7463	364.1169154
3	0.045	7365	358.0223881
4	0.06	7583	371.579602
5	0.075	7295	353.6691542
6	0.09	7085	340.6094527
7	0.105	6813	323.6940299
8	0.12	5980	271.8905473
9	0.135	4120	156.2189055
10	0.15	2413	50.06218905
11	0.165	2057	27.92288557
12	0.18	1967	22.32587065
13	0.195	1964	22.13930348
14	0.21	1995	24.06716418
15	0.225	1980	23.13432836
16	0.24	1980	23.13432836

Tabela 4. SA- Wyniki dla pliku testowego ftv55.txt (invert)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.015	7375	358.6442786
2	0.03	7434	362.3134328
3	0.045	7522	367.7860697
4	0.06	7488	365.6716418
5	0.075	7348	356.9651741
6	0.09	7247	350.6840796
7	0.105	7021	336.6293532
8	0.12	6558	307.8358209
9	0.135	5458	239.4278607
10	0.15	3879	141.2313433
11	0.165	2788	73.38308458
12	0.18	2812	74.87562189
13	0.195	2782	73.00995025
14	0.21	2836	76.3681592
15	0.225	2881	79.16666667
16	0.24	2886	79.47761194



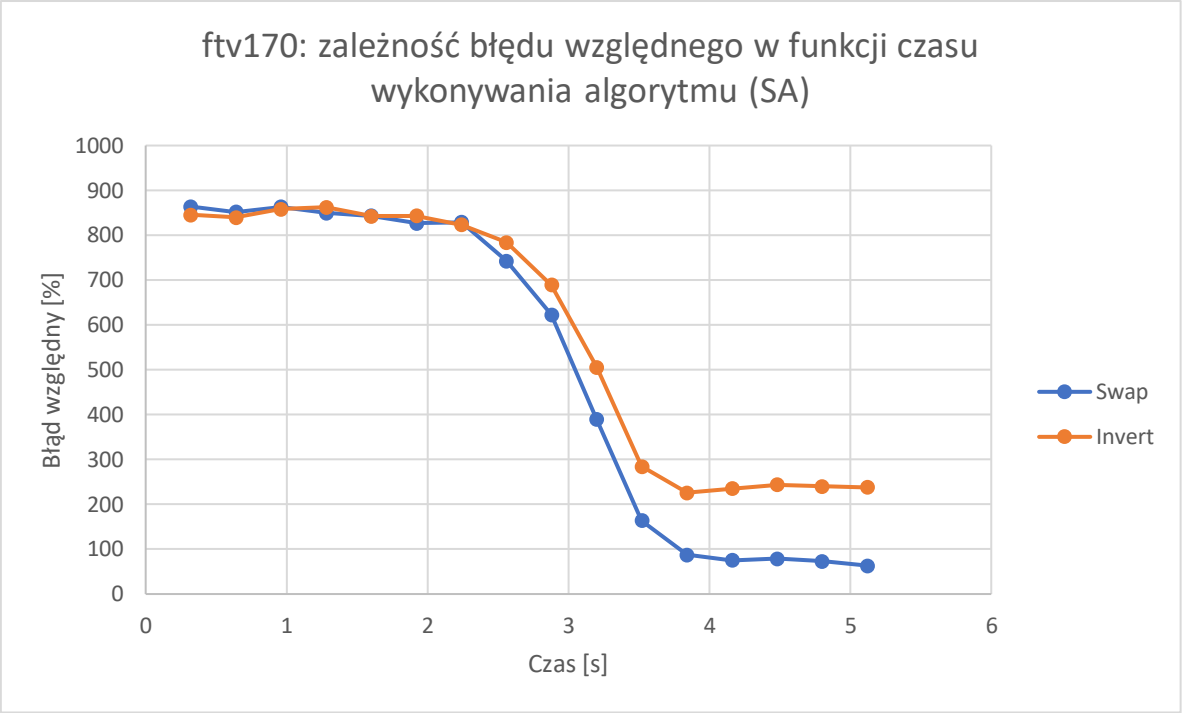
Rys. 2 SA- Wykres dla pliku testowego ftv55.txt

Tabela 5. SA- Wyniki dla pliku testowego ftv170.txt (swap)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.32	26557	863.9564428
2	0.64	26230	852.0871143
3	0.96	26544	863.4845735
4	1.28	26164	849.6914701
5	1.6	25996	843.5934664
6	1.92	25526	826.5335753
7	2.24	25593	828.9655172
8	2.56	23211	742.5045372
9	2.88	19914	622.831216
10	3.2	13478	389.2196007
11	3.52	7275	164.0653358
12	3.84	5158	87.22323049
13	4.16	4831	75.353902
14	4.48	4934	79.09255898
15	4.8	4760	72.77676951
16	5.12	4502	63.41197822

Tabela 6. SA- Wyniki dla pliku testowego ftv170.txt (invert)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.32	26041	845.2268603
2	0.64	25888	839.6733212
3	0.96	26394	858.0399274
4	1.28	26520	862.6134301
5	1.6	25969	842.6134301
6	1.92	25989	843.3393829
7	2.24	25431	823.0852995
8	2.56	24357	784.1016334
9	2.88	21749	689.4373866
10	3.2	16685	505.6261343
11	3.52	10582	284.1016334
12	3.84	8981	225.9891107
13	4.16	9238	235.3176044
14	4.48	9467	243.6297641
15	4.8	9373	240.2177858
16	5.12	9319	238.2577132



Rys. 3 SA- Wykres dla pliku testowego ftv170.txt

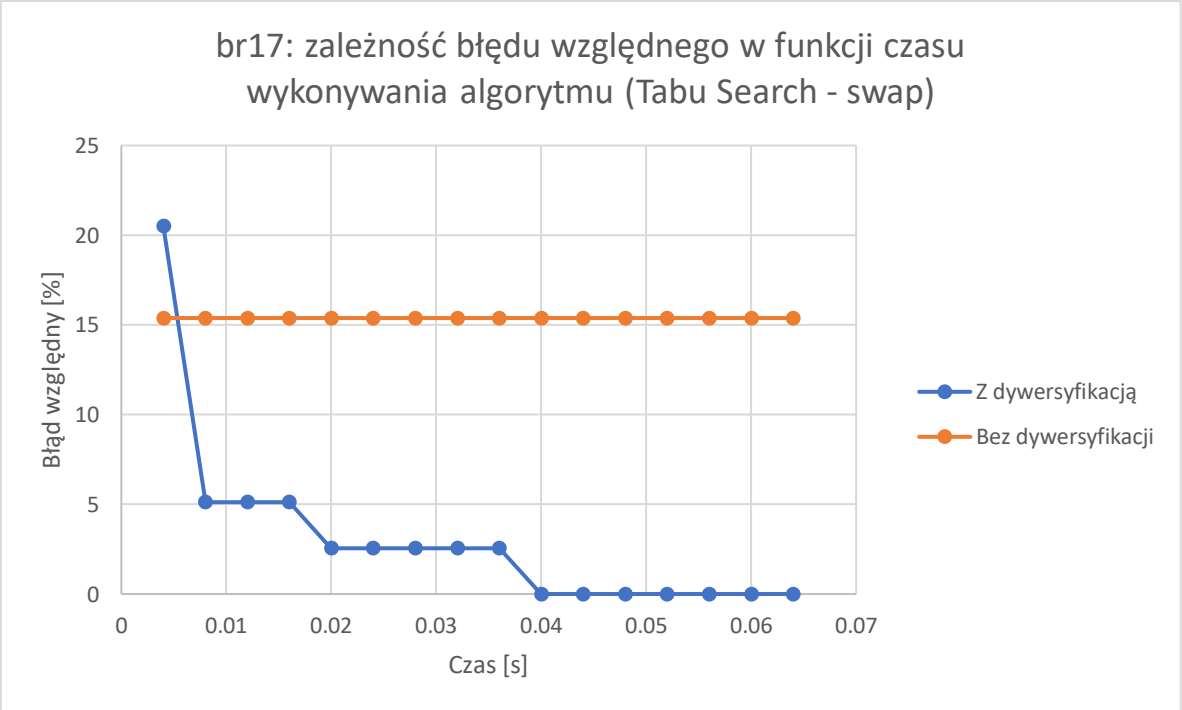
b) Tabu Search

Tabela 7. TS- Wyniki dla pliku testowego br17.txt (swap- z dywersyfikacją)

		Czas [s]	Znaleziona droga	Błąd względny[%]
1		0.004	47	20.51282051
2		0.008	41	5.128205128
3		0.012	41	5.128205128
4		0.016	41	5.128205128
5		0.02	40	2.564102564
6		0.024	40	2.564102564
7		0.028	40	2.564102564
8		0.032	40	2.564102564
9		0.036	40	2.564102564
10		0.04	39	0
11		0.044	39	0
12		0.048	39	0
13		0.052	39	0
14		0.056	39	0
15		0.06	39	0
16		0.064	39	0

Tabela 8. TS- Wyniki dla pliku testowego br17.txt (swap- bez dywersyfikacji

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.004	45	15.38461538
2	0.008	45	15.38461538
3	0.012	45	15.38461538
4	0.016	45	15.38461538
5	0.02	45	15.38461538
6	0.024	45	15.38461538
7	0.028	45	15.38461538
8	0.032	45	15.38461538
9	0.036	45	15.38461538
10	0.04	45	15.38461538
11	0.044	45	15.38461538
12	0.048	45	15.38461538
13	0.052	45	15.38461538
14	0.056	45	15.38461538
15	0.06	45	15.38461538
16	0.064	45	15.38461538



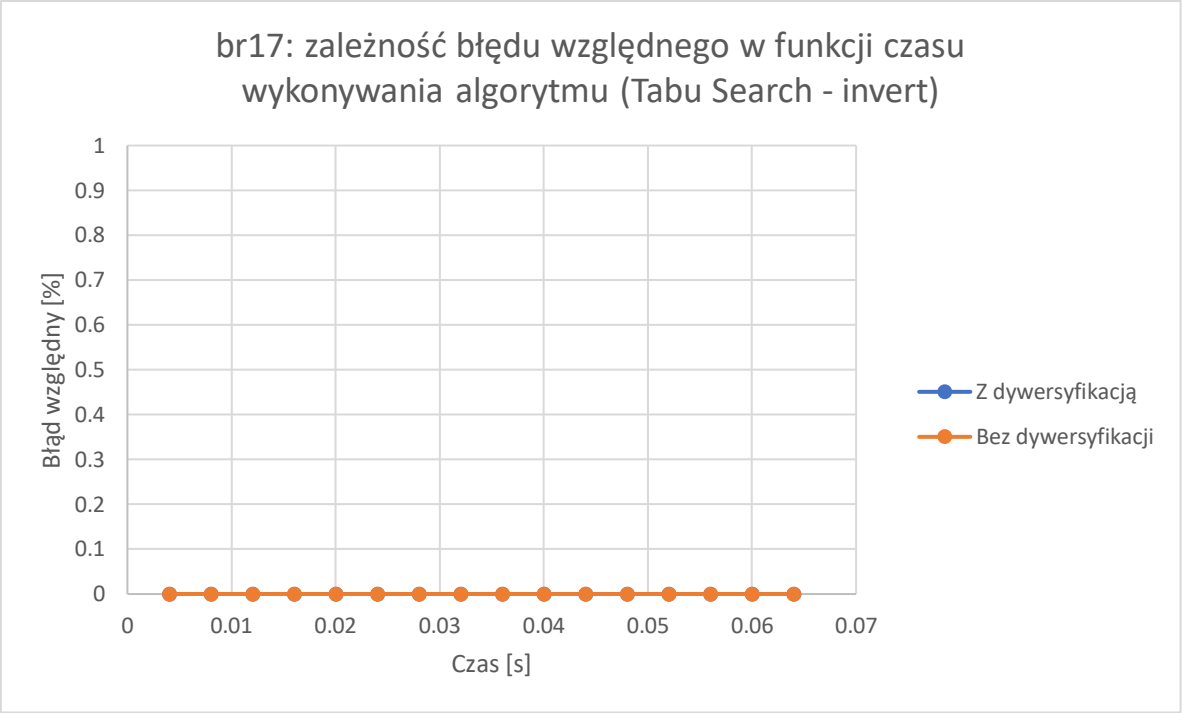
Rys. 4 TS- Wykres dla pliku testowego br17.txt (swap)

Tabela 9. TS- Wyniki dla pliku testowego br17.txt (invert- z dywersyfikacją)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.004	39	0
2	0.008	39	0
3	0.012	39	0
4	0.016	39	0
5	0.02	39	0
6	0.024	39	0
7	0.028	39	0
8	0.032	39	0
9	0.036	39	0
10	0.04	39	0
11	0.044	39	0
12	0.048	39	0
13	0.052	39	0
14	0.056	39	0
15	0.06	39	0
16	0.064	39	0

Tabela 10. TS- Wyniki dla pliku testowego br17.txt (invert- bez dywersyfikacji)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.004	39	0
2	0.008	39	0
3	0.012	39	0
4	0.016	39	0
5	0.02	39	0
6	0.024	39	0
7	0.028	39	0
8	0.032	39	0
9	0.036	39	0
10	0.04	39	0
11	0.044	39	0
12	0.048	39	0
13	0.052	39	0
14	0.056	39	0
15	0.06	39	0
16	0.064	39	0



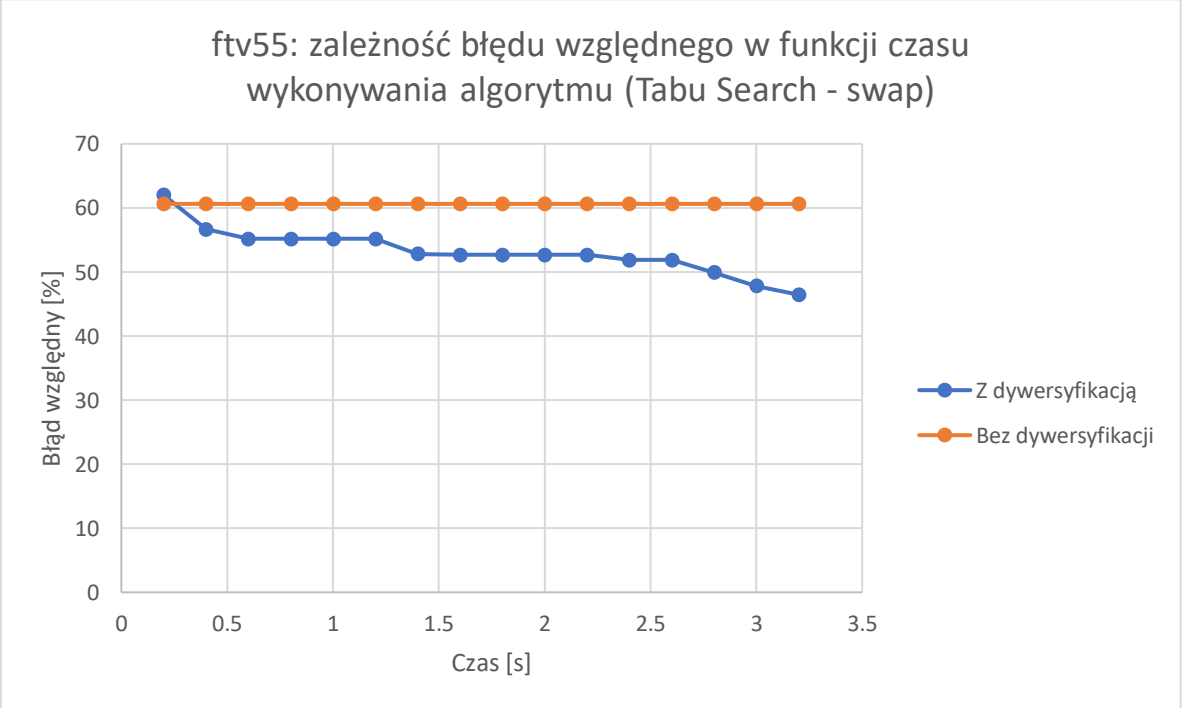
Rys. 5 TS- Wykres dla pliku testowego br17.txt (invert)

Tabela 11. TS- Wyniki dla pliku testowego ftv55.txt (swap- z dywersyfikacją)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.2	2605	62.00248756
2	0.4	2519	56.65422886
3	0.6	2495	55.16169154
4	0.8	2495	55.16169154
5	1	2495	55.16169154
6	1.2	2495	55.16169154
7	1.4	2457	52.79850746
8	1.6	2455	52.67412935
9	1.8	2455	52.67412935
10	2	2455	52.67412935
11	2.2	2455	52.67412935
12	2.4	2442	51.86567164
13	2.6	2442	51.86567164
14	2.8	2411	49.93781095
15	3	2377	47.82338308
16	3.2	2355	46.45522388

Tabela 12. TS- Wyniki dla pliku testowego ftv55.txt (swap- bez dywersyfikacji)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.2	2583	60.63432836
2	0.4	2583	60.63432836
3	0.6	2583	60.63432836
4	0.8	2583	60.63432836
5	1	2583	60.63432836
6	1.2	2583	60.63432836
7	1.4	2583	60.63432836
8	1.6	2583	60.63432836
9	1.8	2583	60.63432836
10	2	2583	60.63432836
11	2.2	2583	60.63432836
12	2.4	2583	60.63432836
13	2.6	2583	60.63432836
14	2.8	2583	60.63432836
15	3	2583	60.63432836
16	3.2	2583	60.63432836



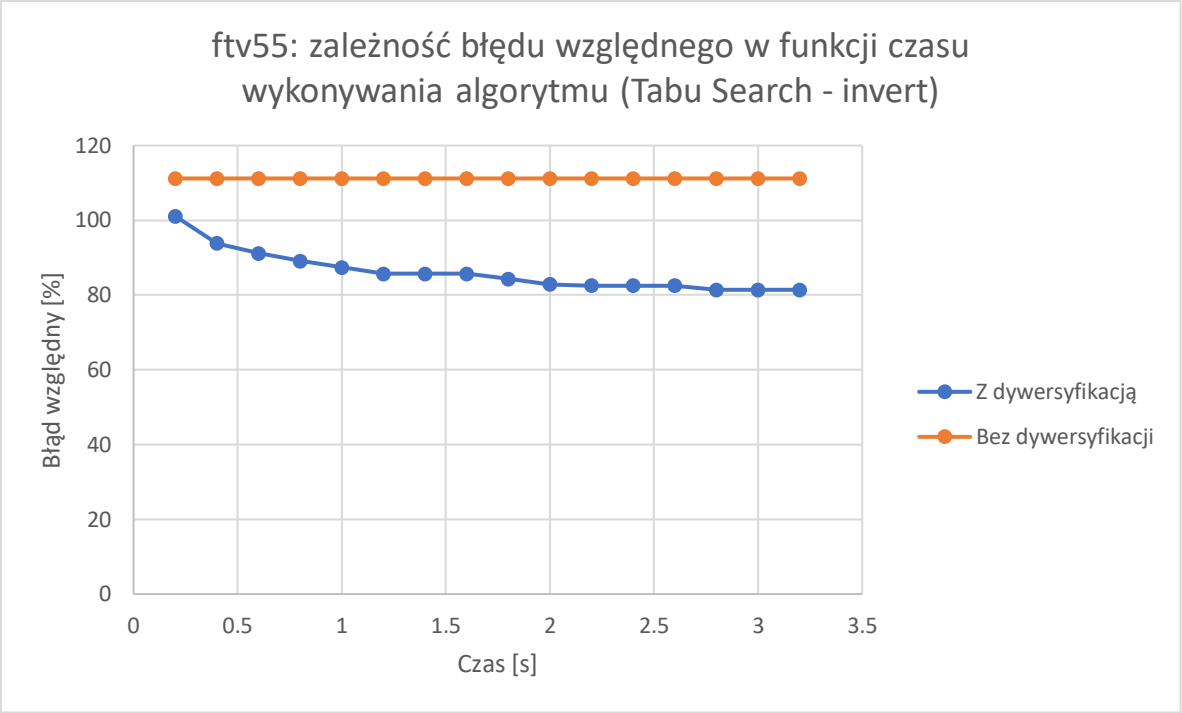
Rys. 5 TS- Wykres dla pliku testowego ftv55.txt (swap)

Tabela 13. TS- Wyniki dla pliku testowego ftv55.txt (invert- z dywersyfikacją)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.2	3234	101.119403
2	0.4	3117	93.84328358
3	0.6	3074	91.16915423
4	0.8	3041	89.11691542
5	1	3014	87.43781095
6	1.2	2986	85.69651741
7	1.4	2986	85.69651741
8	1.6	2986	85.69651741
9	1.8	2964	84.32835821
10	2	2940	82.8358209
11	2.2	2935	82.52487562
12	2.4	2935	82.52487562
13	2.6	2935	82.52487562
14	2.8	2917	81.40547264
15	3	2917	81.40547264
16	3.2	2917	81.40547264

Tabela 14. TS- Wyniki dla pliku testowego ftv55.txt (invert- bez dywersyfikacji)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.2	3396	111.1940299
2	0.4	3396	111.1940299
3	0.6	3396	111.1940299
4	0.8	3396	111.1940299
5	1	3396	111.1940299
6	1.2	3396	111.1940299
7	1.4	3396	111.1940299
8	1.6	3396	111.1940299
9	1.8	3396	111.1940299
10	2	3396	111.1940299
11	2.2	3396	111.1940299
12	2.4	3396	111.1940299
13	2.6	3396	111.1940299
14	2.8	3396	111.1940299
15	3	3396	111.1940299
16	3.2	3396	111.1940299



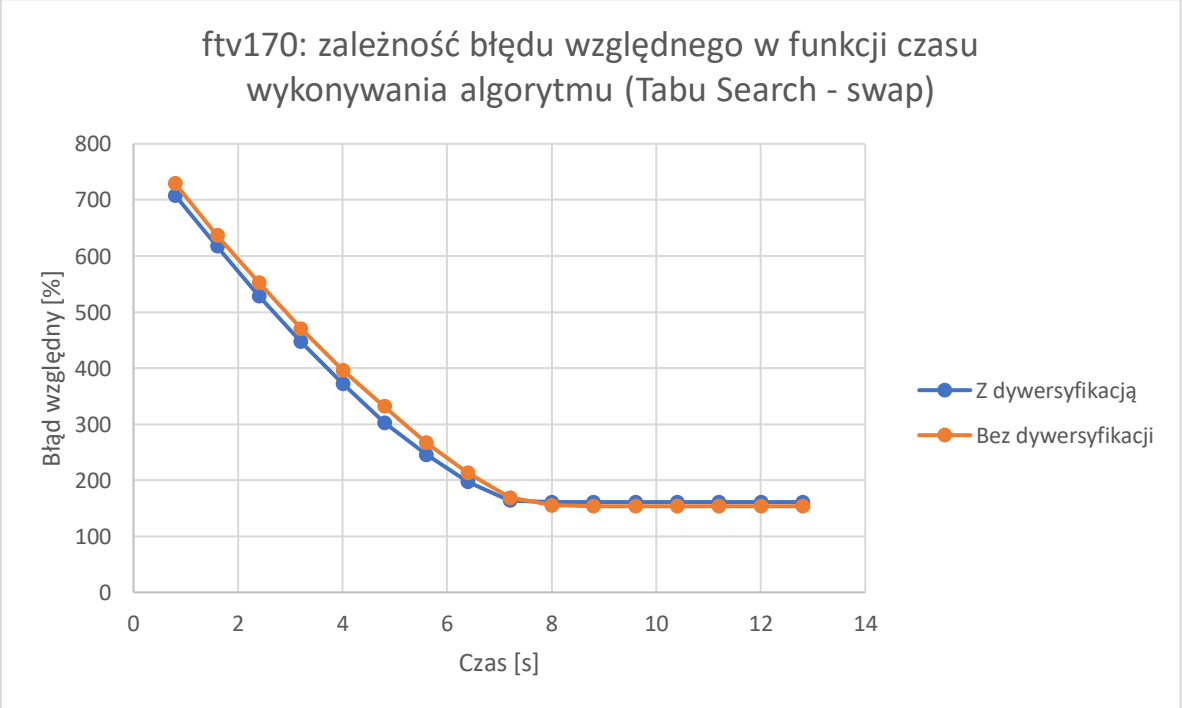
Rys. 6 TS- Wykres dla pliku testowego ftv55.txt (invert)

Tabela 15. TS- Wyniki dla pliku testowego ftv170.txt (swap- z dywersyfikacją)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.8	22259	707.9491833
2	1.6	19767	617.4954628
3	2.4	17333	529.1470054
4	3.2	15078	447.2958258
5	4	13016	372.4500907
6	4.8	11102	302.9764065
7	5.6	9515	245.3720508
8	6.4	8180	196.9147005
9	7.2	7268	163.8112523
10	8	7191	161.0163339
11	8.8	7191	161.0163339
12	9.6	7191	161.0163339
13	10.4	7191	161.0163339
14	11.2	7191	161.0163339
15	12	7191	161.0163339
16	12.8	7191	161.0163339

Tabela 16. TS- Wyniki dla pliku testowego ftv170.txt (swap- bez dywersyfikacji)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.8	22844	729.1833031
2	1.6	20302	636.9147005
3	2.4	17994	553.1397459
4	3.2	15721	470.6352087
5	4	13668	396.1161525
6	4.8	11895	331.7604356
7	5.6	10110	266.969147
8	6.4	8619	212.8493648
9	7.2	7408	168.892922
10	8	7038	155.4627949
11	8.8	6986	153.5753176
12	9.6	6986	153.5753176
13	10.4	6986	153.5753176
14	11.2	6986	153.5753176
15	12	6986	153.5753176
16	12.8	6986	153.5753176



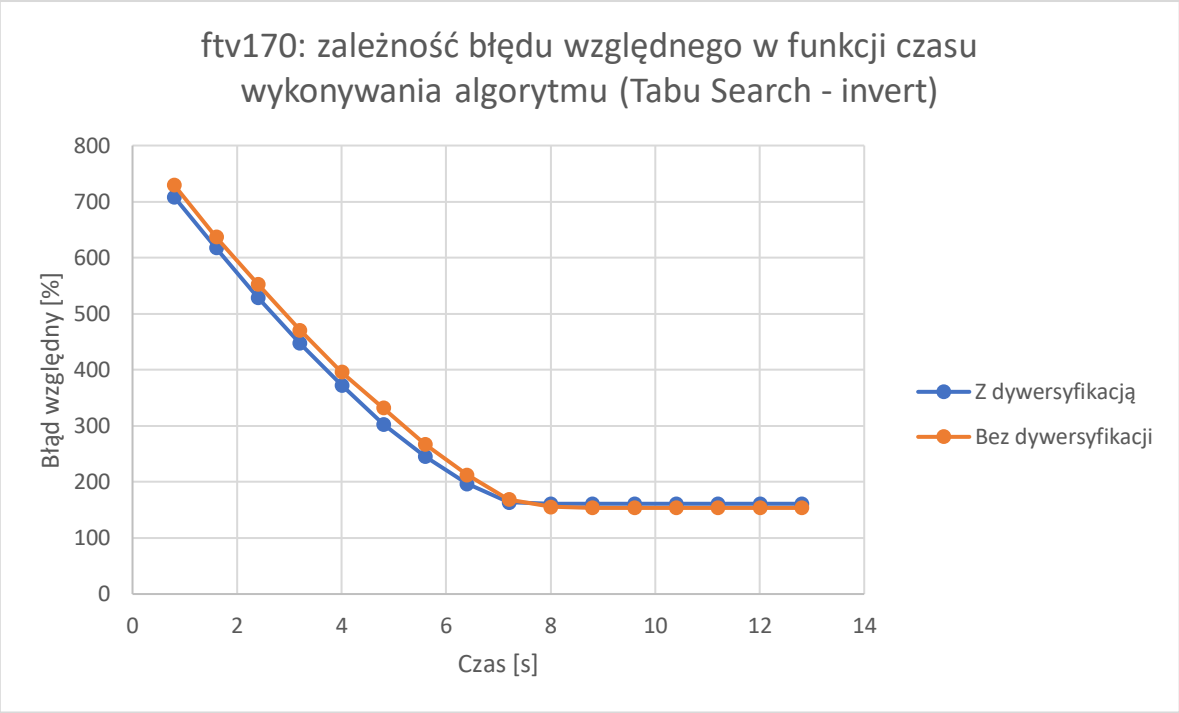
Rys. 7 TS- Wykres dla pliku testowego ftv170.txt (swap)

Tabela 17. TS- Wyniki dla pliku testowego ftv170.txt (invert- z dywersyfikacją)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.8	22709	724.2831216
2	1.6	19810	619.0562613
3	2.4	17105	520.8711434
4	3.2	14425	423.5934664
5	4	11894	331.7241379
6	4.8	10134	267.8402904
7	5.6	10036	264.2831216
8	6.4	10036	264.2831216
9	7.2	10036	264.2831216
10	8	10036	264.2831216
11	8.8	10036	264.2831216
12	9.6	10036	264.2831216
13	10.4	10036	264.2831216
14	11.2	10036	264.2831216
15	12	10036	264.2831216
16	12.8	10036	264.2831216

Tabela 18. TS- Wyniki dla pliku testowego ftv170.txt (invert- bez dywersyfikacji)

	Czas [s]	Znaleziona droga	Błąd względny[%]
1	0.8	22858	729.6914701
2	1.6	20031	627.0780399
3	2.4	16960	515.6079855
4	3.2	14173	414.446461
5	4	11532	318.584392
6	4.8	10439	278.9110708
7	5.6	10439	278.9110708
8	6.4	10439	278.9110708
9	7.2	10439	278.9110708
10	8	10439	278.9110708
11	8.8	10439	278.9110708
12	9.6	10439	278.9110708
13	10.4	10439	278.9110708
14	11.2	10439	278.9110708
15	12	10439	278.9110708
16	12.8	10439	278.9110708



Rys. 8 TS- Wykres dla pliku testowego ftv170.txt (invert)

5. Wnioski

a) Symulowane wyżarzanie:

Algorytm ten jest łatwy do implementacji, jednak do poprawnego działania wymaga on wyboru odpowiednich parametrów. W trakcie wykonywania eksperymentu wybierane były różne współczynniki „a", liczba epok, temperatura początkowa itp. Ostatecznie jednak parametry do wykonania eksperymentu zostały wybrane tak jak zostało to opisane w punkcie 3 sprawozdania. Dla tak wybranych parametrów program najczęściej działał stabilnie i zwracał optimum globalne (dla małych grafów), lub rozwiązanie bliskie optimum globalnego (dla dużych grafów). Nie była to jednak reguła (wszystko zależało od danego problemu).

Z otrzymanych wyników można dojść do wniosku, że tym większa macierz, tym zwykle trudniej jest o rozwiązanie z mniejszym błędem względnym. Oprócz tego potrzeba więcej czasu na uzyskanie najbardziej optymalnego wyniku. Dla testowanych grafów powyżej wyznaczonych punktów czasowych wynik zwykle nie wychodził lepszy, ze względu na za niską już temperaturę. Porównując sąsiedztwo typu „swap" i „invert" można dojść do wniosku, że lepsze wyniki wychodziły zwykle dla sąsiedztwa typu „swap". Nie do końca jest jasne z czego to może wynikać. Typy sąsiedztwa są często wybierane w zależności od problemu i niektóre potrafią być lepsze a drugie gorsze. Istnieje także szansa, że sąsiedztwo mogło zostać nie do końca poprawnie zdefiniowane.

Podsumowując, wyniki eksperymentu można uznać za udane. Trzeba jednak mieć na uwadze, że te rezultaty wyszły tylko dla trzech przykładowych testowanych grafów i nie ma gwarancji, że dla innych, podobnych przykładów, algorytm zachowa się w ten sam sposób. Wszystko zależy tutaj od doboru odpowiednich parametrów.

b) Tabu Search:

Dla pliku br17.txt dywersyfikacja musiała zostać zastosowana, jeśli chcieliśmy uzyskać najlepsze znane rozwiązanie. Bez jej zastosowania znajdywana droga nie mogła osiągnąć globalnego optimum. Dla sąsiedztwa typu invert najlepsze znane rozwiązanie było odnajdywane nawet bez zastosowania dywersyfikacji. Wydaje się, że nieco lepiej zadziałało w tym określonym przypadku sąsiedztwo typu invert.

Dla pliku ftv55.txt ponownie stosując dywersyfikację udało nam się nieco polepszyć rozwiązanie końcowe. Algorytm potrzebował jednak więcej czasu a rozwiązanie były znacznie mniej dokładnie niż w przypadku br17.txt. Globalnego optimum nie udało się osiągnąć. Sąsiedztwo typu invert zwracało gorsze rozwiązania niż swap, niemniej jednak zastosowanie dywersyfikacji ponownie polepszało nieco wyniki. Dla pliku ftv170.txt zastosowanie dywersyfikacji nie wiele zmieniło. Ponownie lepsze okazało się sąsiedztwo typu swap.

Ponownie jak w przypadku symulowanego wyżarzania lepsze okazało się sąsiedztwo typu swap.

Zauważono, że dla mniejszych grafów dywersyfikacja przynosi zwykle lepszy skutek niż dla dużych. W trakcie testowania dało się zauważyć, że bez zastosowania dywersyfikacji algorytm bardzo szybko zaczynał wpadać w lokalne minima. Doprowadziło to najprawdopodobniej do tego, że wyniki w tabelach opisujących brak dywersyfikacji są stałe dla wszystkich punktów pomiarowych. Dywersyfikacja sprawiła, że algorytm dłużej szukał optymalnego rozwiązania i nie wpadał tak szybko w minima lokalne. Nie działa on jednak idealnie, jako że nie ograniczył w pełni wpadania w cykl, a dla przykładu ftv170.txt praktycznie w ogóle nie odgrywał większej roli.

Niestety wyniki dla algorytmu Tabu Search wyszły gorsze niż dla symulowanego wyżarzania, a dla dużych plików grafów tj. ftv170.txt zwracał zwykle wynik z błędem wynoszącym powyżej 100%. Możliwe, że algorytm ten został zaimplementowany w nieco zbyt trywialnej wersji lub nie do końca zostały wybrane poprawnie niektóre parametry.

6. Bibliografia.

- Konspekty z wykładów z projektowania efektywnych algorytmów sporządzonych przez dr inż. Tomasza Kapłona
- Materiały dodatkowe zamieszczone w opisie do zadania 2 projektu z projektowania efektywnych algorytmów, który opublikował mgr inż. Antoni Sterna