Projektowanie efektywnych algorytmów
Projekt

Projekt nr 2 - Symulowane wyżarzanie

Autorka: Daria Jeżowska 252731
Prowadzący: mgr inż. Anotni Sterna
Dzień zajęć: Czwartek 15^{15}

1 Wstęp

Problem komiwojażera polega na znalezieniu minimalnego cyklu Hamiltona, czyli najkrótszej drogi pomiędzy wszystkimi wierzchołkami grafu, przy czym każdy wierzchołek może zostać odwiedzony tylko jeden raz i podróż musi się skończyć w punkcie startowym - rozwiązaniem problemu komiwojażera jest cykl o najmniejszej sumie wag krawędzi. Problem ten został sformułowany w 1859 roku i jest to NP-trudny problem.

Symulowane wyżarzanie to algorytm metaheurystyczny nawiązujący do procesu wyżarzania - nagrzaniu materiału do danej temperatury, wygrzewaniu go w określonych warunkach oraz powolnym studzeniu. Celem tego jest otrzymanie pewnych właściwości materiału (np. twardość, wytrzymałość). Symulowane wyżarzanie polega na akceptowaniu lepszego rozwiązania problemu oraz podjęciu decyzji czy należy zaakceptować gorsze. To, czy gorsze rozwiązanie zostanie zaakceptowane zależy w głównej mierze od temperatury - im niższa, tym prawdopodobieństwo zaakceptowania będzie mniejsze. W Umożliwia to wyjście poza obszar minimum lokalnego, co pozwala na znalezienie minimum globalnego. W przypadku symulowanego wyżarzania nie da się także dobrać zawsze idealnych parametrów przez co algorytm nie zapewnia znalezienia rozwiązania danego problemu. Poza parametrami jest także duża losowość w wybieraniu i zamianie ścieżek, co dodatkowo wpływa na jego wyniki. Dzięki ograniczeniom czasowym pomaga za to znaleźć jak najlepsze rozwiązanie w jak najkrótszym czasie.

2 Opis działania algorytmu

- Wybór rozwiązania początkowego S i obliczenie wartości ścieżki
- Dopóki T > $T_{końcowe}$:
  - Dopóki N < max_iteracji
    - Wygeneruj rozwiązanie S' w sąsiedztwie S poprzez zamianę dwóch losowych elementów w S
    - Obliczenie wartości ścieżki dla nowego rozwiązania
    - Jeśli S' < S lub $e^{\frac{-\Delta}{T}}$ > random(0,1) to S = S',
  - Obniżenie temperatury T

3 Implementacja algorytmu

Algorytm został zaimplementowany w za pomocą języka C++.
Program składa się z dwóch klas - *Menu*, która służy do wczytywania i generowania danych oraz wyświetlania menu:

- *displayMenu* odpowiada za wyświetlanie menu, do którego zostały użyte instrukcje switch ... case.
- *readFile* wczytuje dane z pliku wybranego przez użytkownika do dwuwymiarowego wektora, który następnie jest przekazywany do funkcji *algorithm* klasy *SimulatedAnnealing*
- *displayReadedData* wypisuje na konsoli wczytane dane
- *generateRandom* tworzy losowe dane o wybranym rozmiarze

Druga klas *SimulatedAnnealing* odpowiada za algorytm symulowanego wyżarzania i posiada następujące funkcje:

- *swapCities* odpowiada za stworzenie nowej ścieżki poprzez zamianę miejscami ze sobą dwóch miast

- *randomPath* odpowiada za stworzenie nowej ścieżki poprzez wymieszanie kolejności miast. Funkcja ta działa na bazie funkcji *swapCities*
- *calculatePathCost* oblicza koszt ścieżki
- *printPath* wyświetla ścieżkę
- odpowiada za działanie algorytmu

Parametry:

- temperature - temperatura
- *iterations* - odpowiada za ilość iteracji wewnątrz pętli for wykonującej się w funkcji *algorithm*
- *coolingRate* - przy każdej iteracji pętli while temperatura jest mnożona przez właśnie tą wartość, co pozwala na stopniowe zmniejszanie temperatury
- *maxTime* - maksymalny czas (w sekundach) wykonywania się algorytmu

Algorytm w pierwszej kolejności tworzy losową ścieżkę za pomocą funkcji *randomPath*, którą przechowujemy w wektorze *localBestPath* oraz liczymy jej koszt, który przechowujemy w zmiennej *localBestPathCost*. Następnie jest tworzony wektor *nextPath* oraz zmienna typu int *nextPathCost*, które są nową, sąsiedzką ścieżką oraz kosztem tej ścieżki. Kolejnym krokiem jest pętla jest while, której warunkami są $temperature > 0.0001$ *i* $maxTime > elapsedMs$, przy czym *maxTime* jest ustawiane przez użytkownika i mnożone przez 1000, aby obliczanie czasu było dokładniejsze. Natomiast temperatura domyślnie wynosi 1000 i przy każdym obrocie pętli jest mnożona przez stałą *coolingRate*, która wynosi 0.99. W tej pętli jest zawarta pętla for, której ilość iteracji określa zmienna *iterations*. W środku pętli for tworzymy nową ścieżkę dla wektora *nextPath* używając funkcji *swapCities* oraz obliczamy jej koszt. W zmiennej *delta* przechowujemy różnicę między wartością zmiennej *nextPathCost* oraz *localBestPathCost*, której używa się do stwierdzenia czy nowa ścieżka jest lepsza oraz do obliczenia prawdopodobieństwa ze wzoru $e^{\frac{-\Delta}{T}}$, gdzie Δ to różnica między kosztami dwóch ścieżek. Zmienna *prob* przechowuje właśnie to prawdopodobieństwo. Pierwsza instrukcja warunkowa if sprawdza czy *delta* jest mniejsze od zera. Jeśli któryś z tych warunków jest spełniony to nasza aktualna ścieżka sąsiedzka staje się główną ścieżką *globalBestPath*. Analogicznie dzieje się dla kosztów tych ścieżek. W drugim ifie sprawdzane jest czy prawdopodobieństwo jest większe niż losowa liczba z przedziału (0,1). Jeśli tak jest do wektora *localBestPath* przypisywana jest kolejność miast z wektora *nextPath* oraz wartość *localBestPathCost* zmienia się na wartość *nextPathCost*.

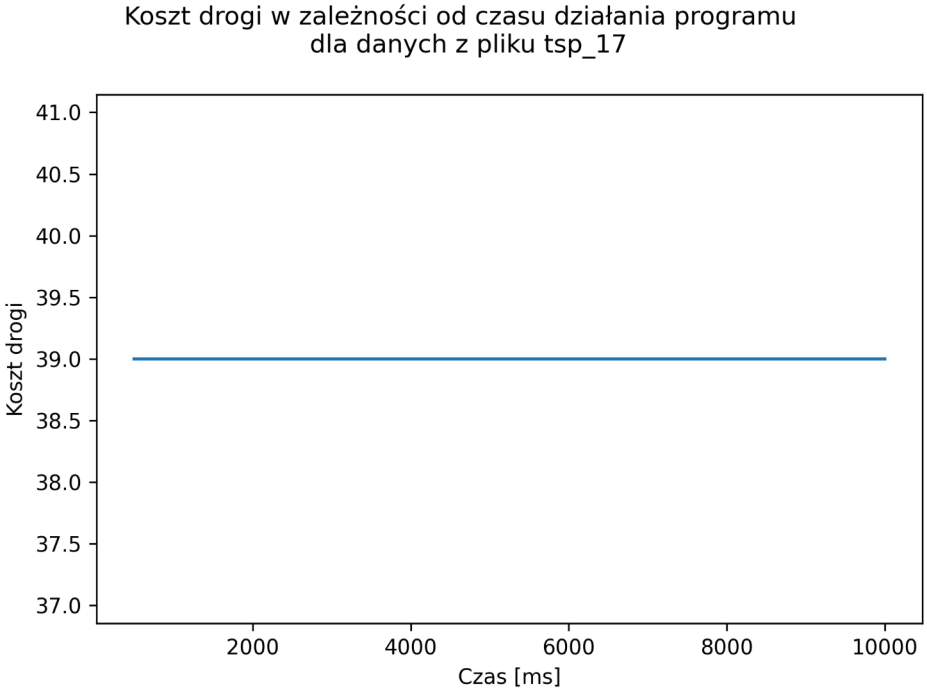
4 Skuteczność algorytmu i analiza wyników

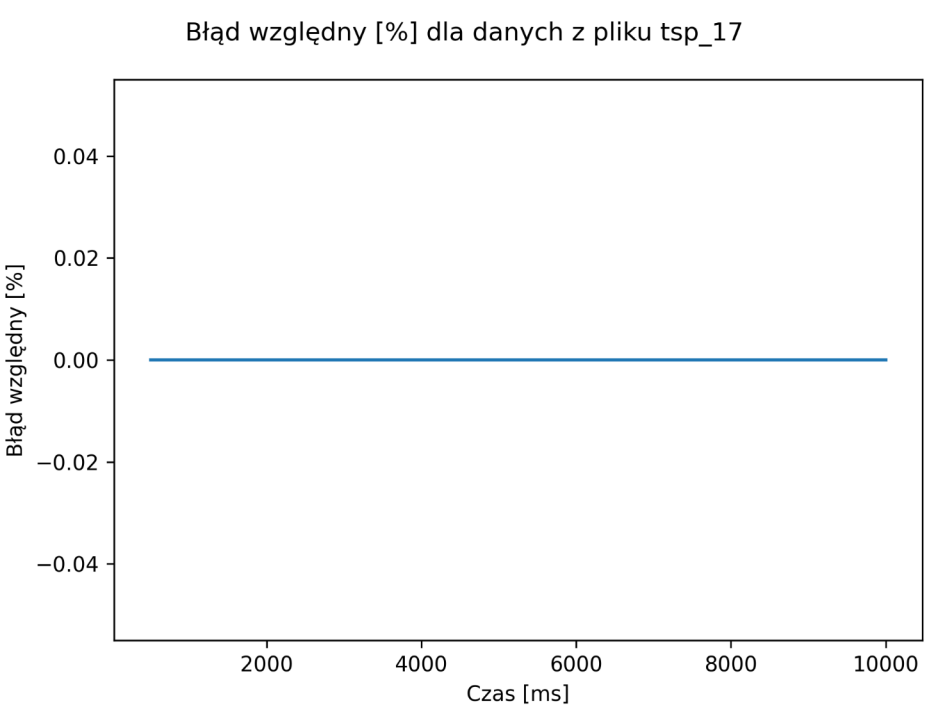
W zależności od ustawionego czasu algorytm działa dokładniej lub mniej. Im większy jest czas wykonywania się algorytmu tym można oczekiwać lepszych wyników. Jak widać w tabeli oraz na wykresach wraz ze wzrostem czasu spada koszt drogi oraz zmniejsza się błąd względny. Kolejnym czynnikiem wpływającym na wyniki jest temperatura. Algorytm może nie dawać lepszych wyników, gdy temperatura zmniejsza się - dotyczy to prawdopodobieństwa zaakceptowania gorszej ścieżki - wraz ze zmniejszającą się temperatury zmniejsza się i ono. Jest większe prawdopodobieństwo, że nie wyjdziemy z lokalnego minimum i nie dostaniemy się do globalnego minimum. Dla każdego testu temperatura została ustawiona na INT_MAX, aby algorytm nie zakończył działania przed upływem danego czasu.

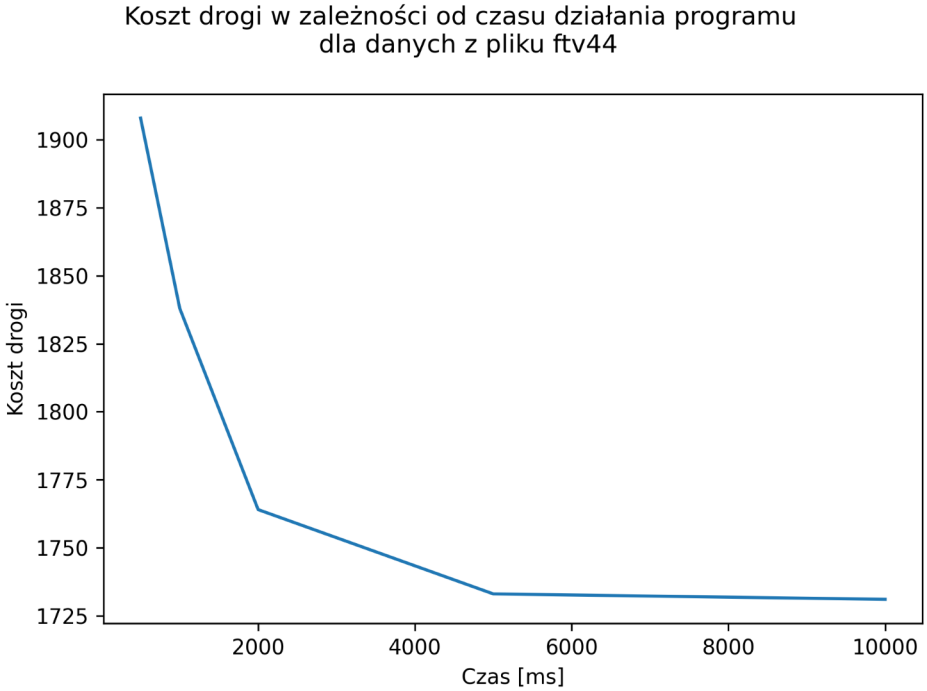
Nazwa pliku	Najniższy koszt ścieżki
tsp_17	39
ftv44	1613
ft70	38673
bier127	118282
rbg403	2465

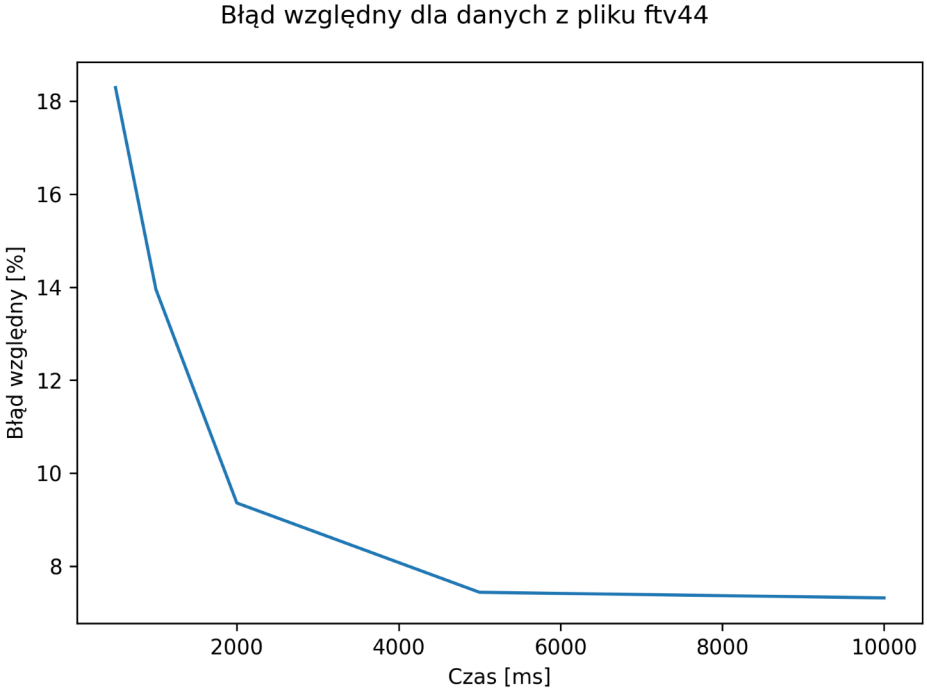
Nazwa pliku	Najlepszy wynik	Czas [s]	Błąd względny [%]
tsp_17	39	0,5	0%
	39	1	0%
	39	2	0%
	39	5	0%
ftv44	1908	0,5	18,29%
	1838	1	13,95%
	1764	2	9,36%
	1733	5	7,44%
	1731	10	7,32%
ft70	42707	0,5	10,43%
	41936	1	8,4%
	41546	2	7,43%
	40797	5	5,49%
	40673	10	5,17%
bier127	148613	0,5	25,64%
	141212	1	19,39%
	139550	2	17,98%
	138665	5	17,23%
	138056	10	16,72%
	137993	20	16,66%
a280	31975	0,5	1139,82%
	28714	1	1013,38%
	20794	2	706,28%
	4187	5	62,35%
	3857	10	49,55%
	3664	20	42,07%
	3640	30	41,14%
rbg403	7894	0,5	220,24%
	7495	1	204,06%
	7275	2	195,13%
	6422	5	160,53%
	2634	10	6,86%
	2503	20	1,54%
	2497	50	1,30%

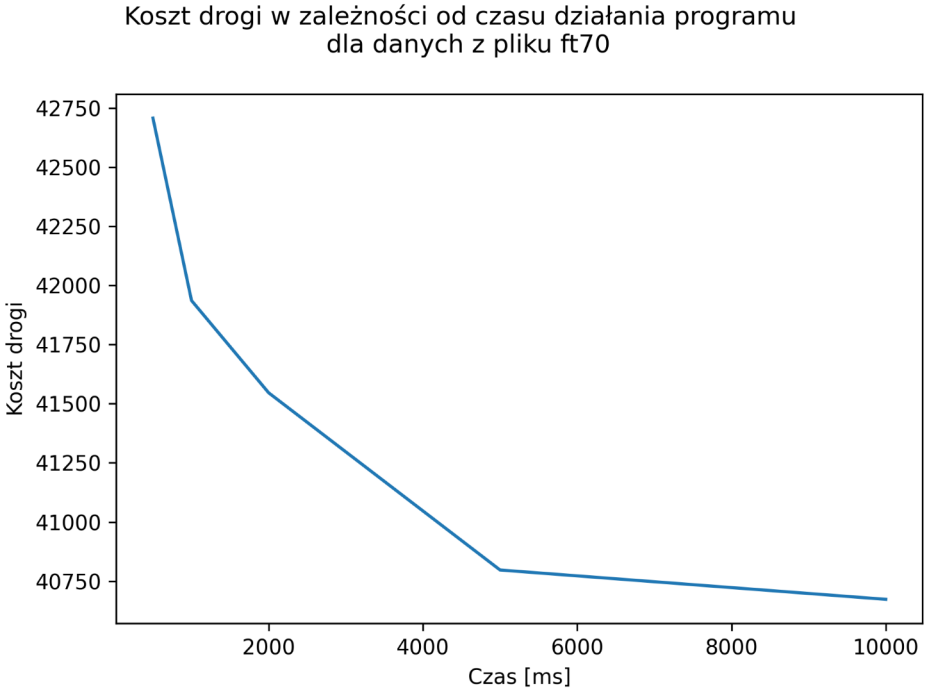
Nazwa pliku	Najlepszy wynik	Temperatura	Błąd względny [%]
tsp_17	39	100	0%
	39	1000	0%
	39	10000	0%
ftv44	1 818	100	12,71%
	1 748	1000	8,37%
	1 843	10000	14,26%
	1829	100000	13,39%
ft70	41 664	100	7,73%
	41 369	1000	6,97%
	40 919	10000	5,81%
bier127	150 914	100	27,59%
	138 273	1000	16,9%
	137 603	10000	16,33%
	131 409	100000	11,1%
a280	4008	100	55,41%
	3901	1000	51,26%
	3728	10000	44,55%
	3834	100000	48,66%
rbg403	2520	100	2,23%
	2506	1000	1,66%
	2507	10000	1,7%
	2505	100000	1,62%

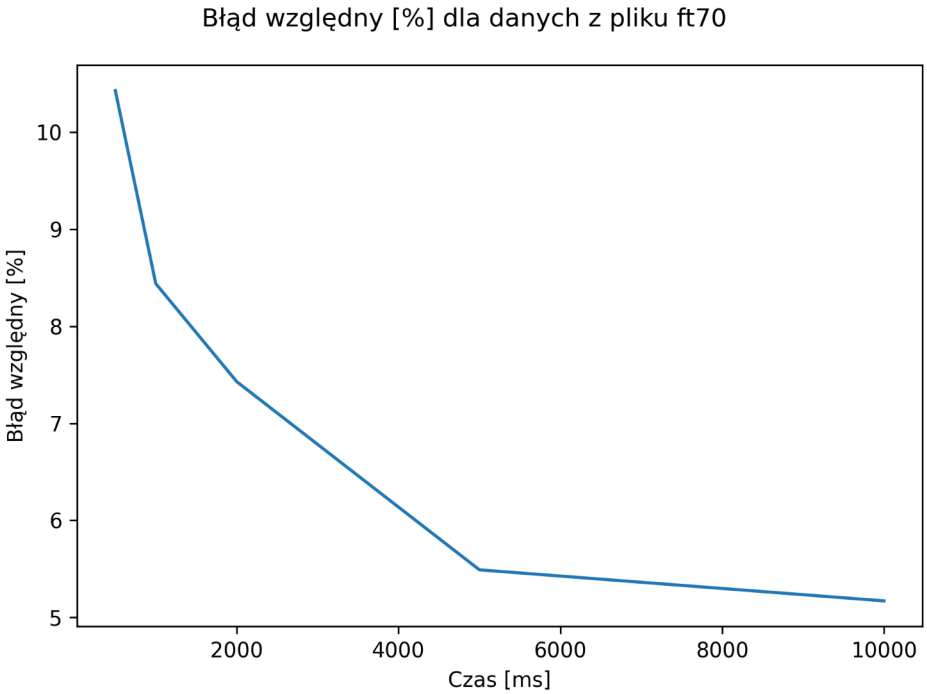

Koszt drogi w zależności od czasu działania programu
dla danych z pliku tsp_17
Koszt drogi
41.0
40.5
40.0
39.5
39.0
38.5
38.0
37.5
37.0
2000
4000
6000
8000
10000
Czas [ms]

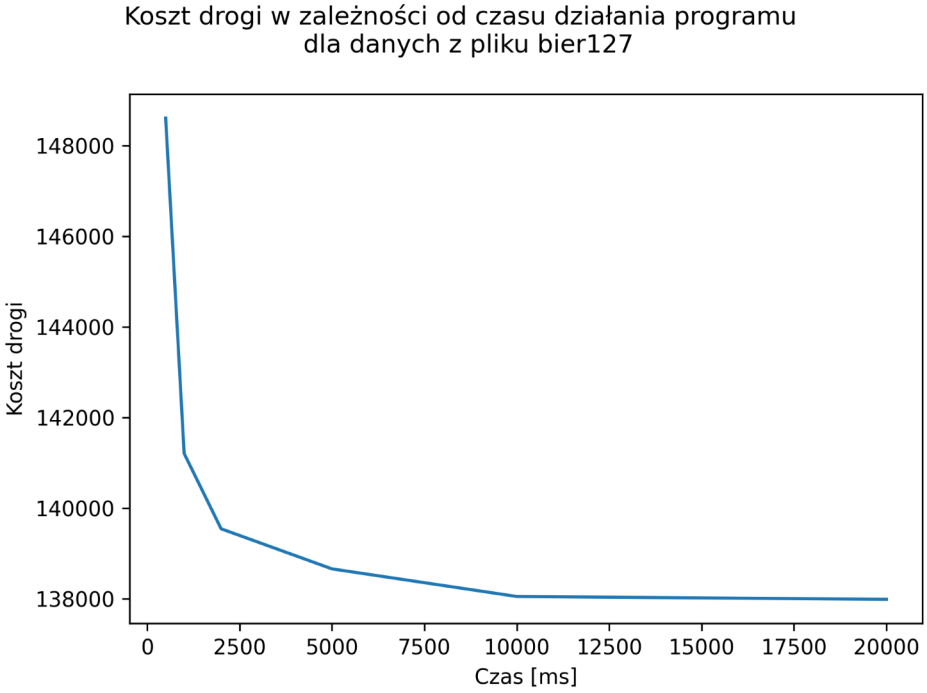

Błąd względny [%] dla danych z pliku tsp_17
Błąd względny [%]
0.04
0.02
0.00
−0.02
−0.04
2000
4000
6000
8000
10000
Czas [ms]

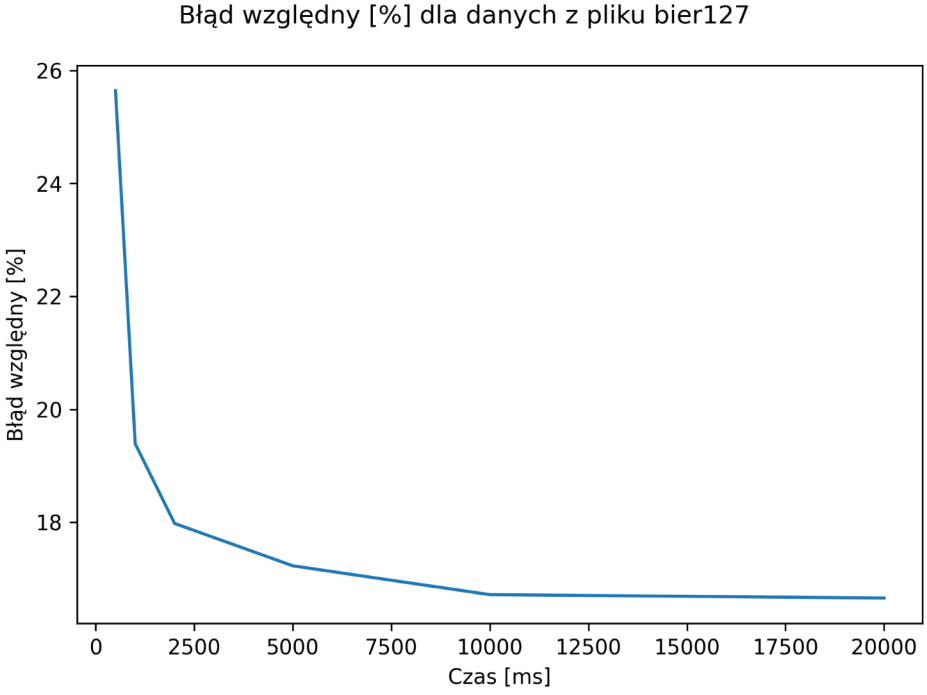

Koszt drogi w zależności od czasu działania programu
dla danych z pliku ftv44
Koszt drogi
1900
1875
1850
1825
1800
1775
1750
1725
2000
4000
6000
8000
10000
Czas [ms]

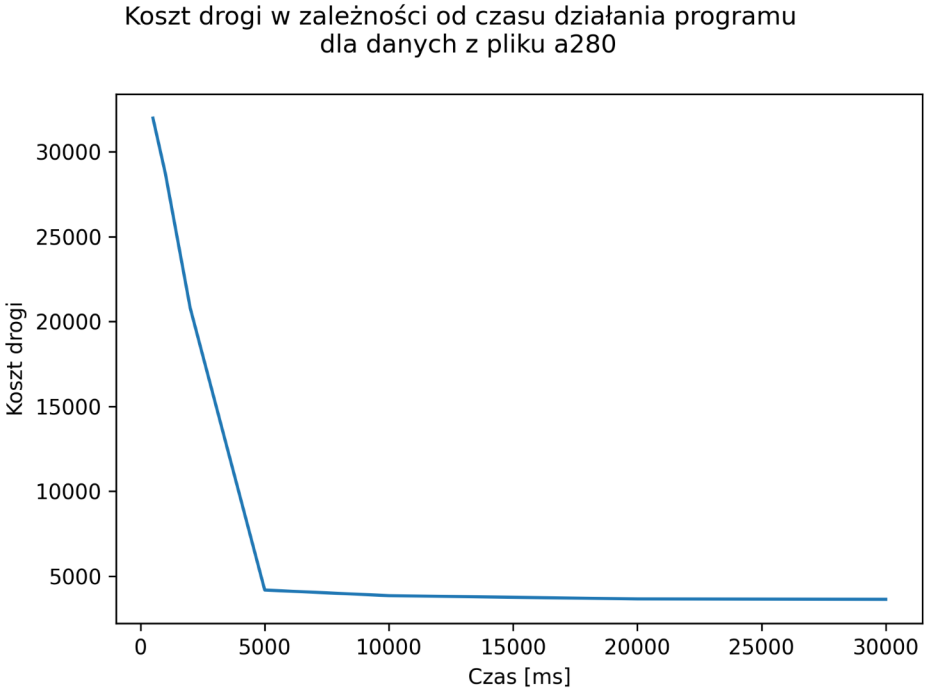

Błąd względny dla danych z pliku ftv44
Błąd względny [%]
18
16
14
12
10
8
2000
4000
6000
8000
10000
Czas [ms]

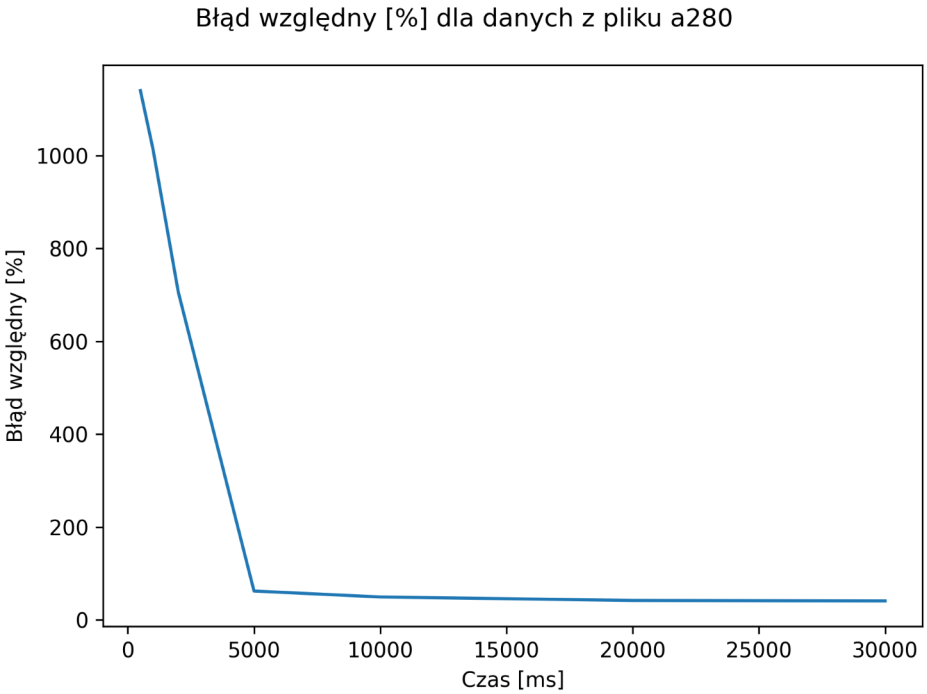

Koszt drogi w zależności od czasu działania programu
dla danych z pliku ft70
Koszt drogi
42750
42500
42250
42000
41750
41500
41250
41000
40750
2000
4000
6000
8000
10000
Czas [ms]

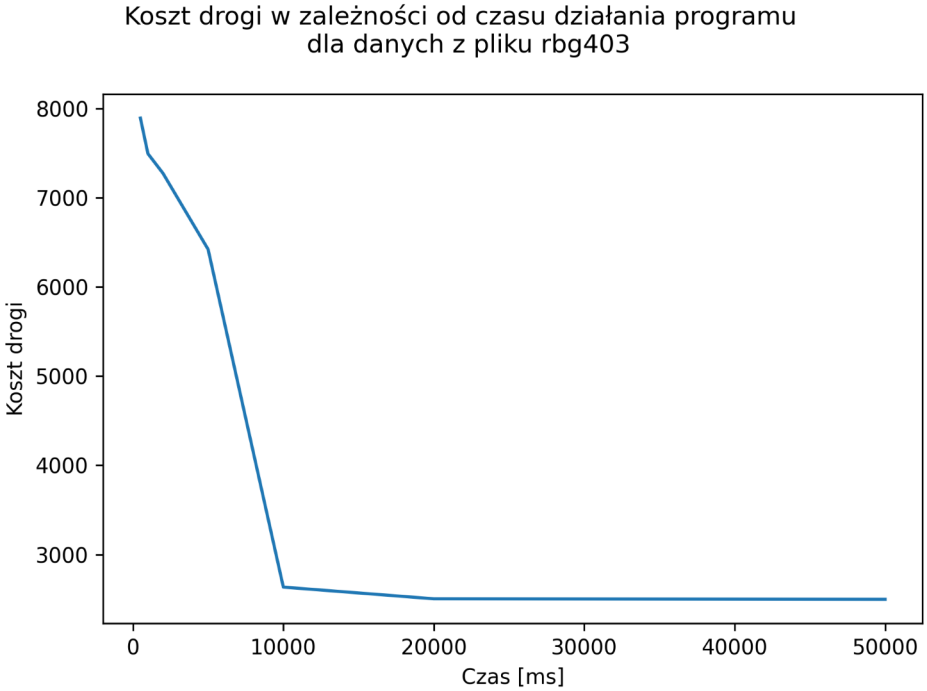

Błąd względny [%] dla danych z pliku ft70
Błąd względny [%]
10
9
8
7
6
5
2000
4000
6000
8000
10000
Czas [ms]

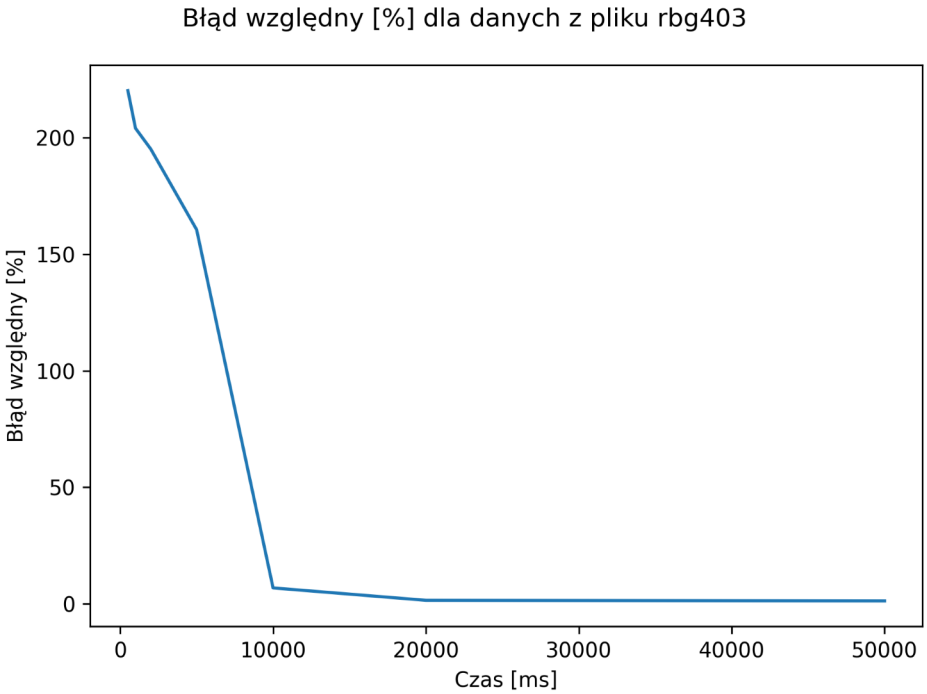

Koszt drogi w zależności od czasu działania programu
dla danych z pliku bier127
Koszt drogi
148000
146000
144000
142000
140000
138000
0
2500
5000
7500
10000
12500
15000
17500
20000
Czas [ms]


Błąd względny [%] dla danych z pliku bier127
Błąd względny [%]
26
24
22
20
18
0
2500
5000
7500
10000
12500
15000
17500
20000
Czas [ms]


Koszt drogi w zależności od czasu działania programu
dla danych z pliku a280
Koszt drogi
30000
25000
20000
15000
10000
5000
0
5000
10000
15000
20000
25000
30000
Czas [ms]


Błąd względny [%] dla danych z pliku a280
Błąd względny [%]
1000
800
600
400
200
0
0
5000
10000
15000
20000
25000
30000
Czas [ms]


Koszt drogi w zależności od czasu działania programu
dla danych z pliku rbg403
Koszt drogi
8000
7000
6000
5000
4000
3000
0
10000
20000
30000
40000
50000
Czas [ms]


Błąd względny [%] dla danych z pliku rbg403
Błąd względny [%]
200
150
100
50
0
0
10000
20000
30000
40000
50000
Czas [ms]

5 Wnioski

Przy problemie komiwojażera i próbie rozwiązania go za pomocą metody symulowanego wyżarzania można otrzymać dobre wyniki - jednakże trzeba dobrze dostosować czas wykonywania oraz temperaturę. Jeśli jest więcej dużo miast do odwiedzenia to czas działania algorytmu powinien być zdecydowanie większy niż przy np. kilkunastu miastach. Zaletą symulowanego wyżarzania jest to, że dostajemy stosunkowo dobrą ścieżkę w odpowiednio krótkim czasie. W przypadku pliku rbg403 dla czasu 50s błąd względny był rzędu *1,30%*, natomiast dla ft70 dla czasu 10s było to *5,15%*. Przy algorytmie Helda-Karpa dla większych liczb czas wykonywania algorytmu mocno się wydłużał i byłoby to niemożliwe doczekać się na wynik dla dwóch wymienionych przed chwilą danych. Mimo że wyniki symulowanego wyżarzania nie zawsze są blisko idealnych to w konkretnych przypadkach ten algorytm może być lepszym wyborem. Przy tym projekcie więcej czasu zajęło samo testowanie danych niż implementacja kodu. Powodem tego było dobieranie odpowiednich parametrów - czasu, temperatury czy współczynnika chłodzącego (był on na stałe ustawiony na liczbę 0,99). Trzeba było rozważyć różne czasy i temperatury, aby uzyskać możliwie najlepszy wynik i było to najtrudniejszą częścią projektu.

6 Bibliografia

- Wykłady
- https://www.mimuw.edu.pl/ grygiel/woen/woen4.pdf
- http://www.cs.put.poznan.pl/mkomosinski/lectures/optimization/SA.pdf
- Dane do testów zostały zaczerpnięte z
  - http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/wyniki.as
  - http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php