Projektowanie Efektywnych Algorytmów

Projekt

21.12.2021

256423 Maciej Radziszewski

(3) Symulowane wyżarzanie

1. Treść zadania

Zadane polega na implementacji i zbadaniu efektywności algorytmu opartego o metodę symulowanego wyżarzania rozwiązującego problem komiwojażera. Polega on na odnalezieniu minimalnego lub maksymalnego (w zależności od wymagań) cyklu Hamiltona w grafie. Graf, który jest brany pod uwagę musi być pełny, oraz jego krawędzie muszą mieć wagi.

2. Metoda

Problem przedstawiany w treści zadania rozwiązywany jest metodą symulowanego wyżarzania. Metoda wywodzi się z algorytmu Metropolisa opracowanego w okolicach 1953r. przez Nicolasa Metropolisa, współtwórcę komputerów z serii MANIAC, które były najpotężniejszymi w swoich czasach. Nicolas był również członkiem legendarnego zespołu badawczego Projektu Manhattan. Algorytm nawiązuje do faktu, że substancje w wyższej temperaturze są bardziej plastyczne i podatne na modyfikacje. Sam algorytm w oryginale odnosił się do układów termodynamicznych i miał następujący przebieg:

- Wybór dowolnego stanu początkowego *i*
- Dla danego stanu wykonywany jest ruch cząstki następuje przejście do stanu *j*
- Jeżeli $E_j - E_i \leq 0$ następuje bezwarunkowa zmiana stanu na stan w przeciwnym przypadku przechodzimy do stanu z prawdopodobieństwem $exp\ (-\frac{Ej - Ei}{k_b T})$

Algorytm opracowany przez Metropolisa po pewnych utożsamieniach oraz modyfikacji został dostosowany do użycia w rozwiązywaniu problemów optymalizacyjnych. W ten sposób był używany i niezależnie przedstawiany kilkanaście razy przez badaczy na przestrzeni lat. W 1983r. został wykorzystany do rozwiązania problemu komiwojażera i to właśnie wtedy nazwany był pierwszy raz algorytmem symulowanego wyżarzania. W problemach optymalizacyjnych algorytm ma przebieg:

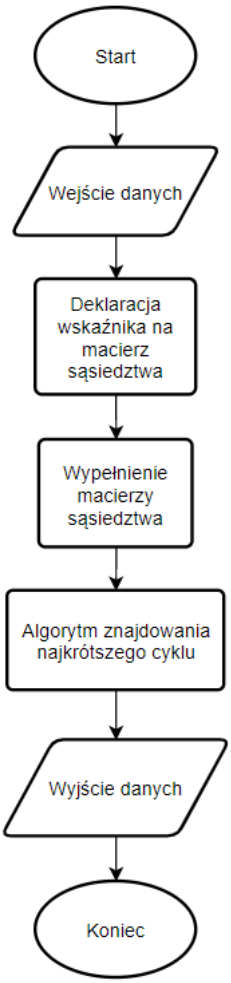
- Wybierz losowe rozwiązanie *x*
- Powtarzaj do określonego warunku zatrzymania
    - wybierz z sąsiedztwa rozwiązanie *y*
    - jeżeli jest *y* lepsze niż przyjmij *x* jako rozwiązanie bieżące
    - w przeciwnym przypadku jeżeli:

$$random \leq exp\ (-\frac{|f(y) - f(x)|}{T})$$

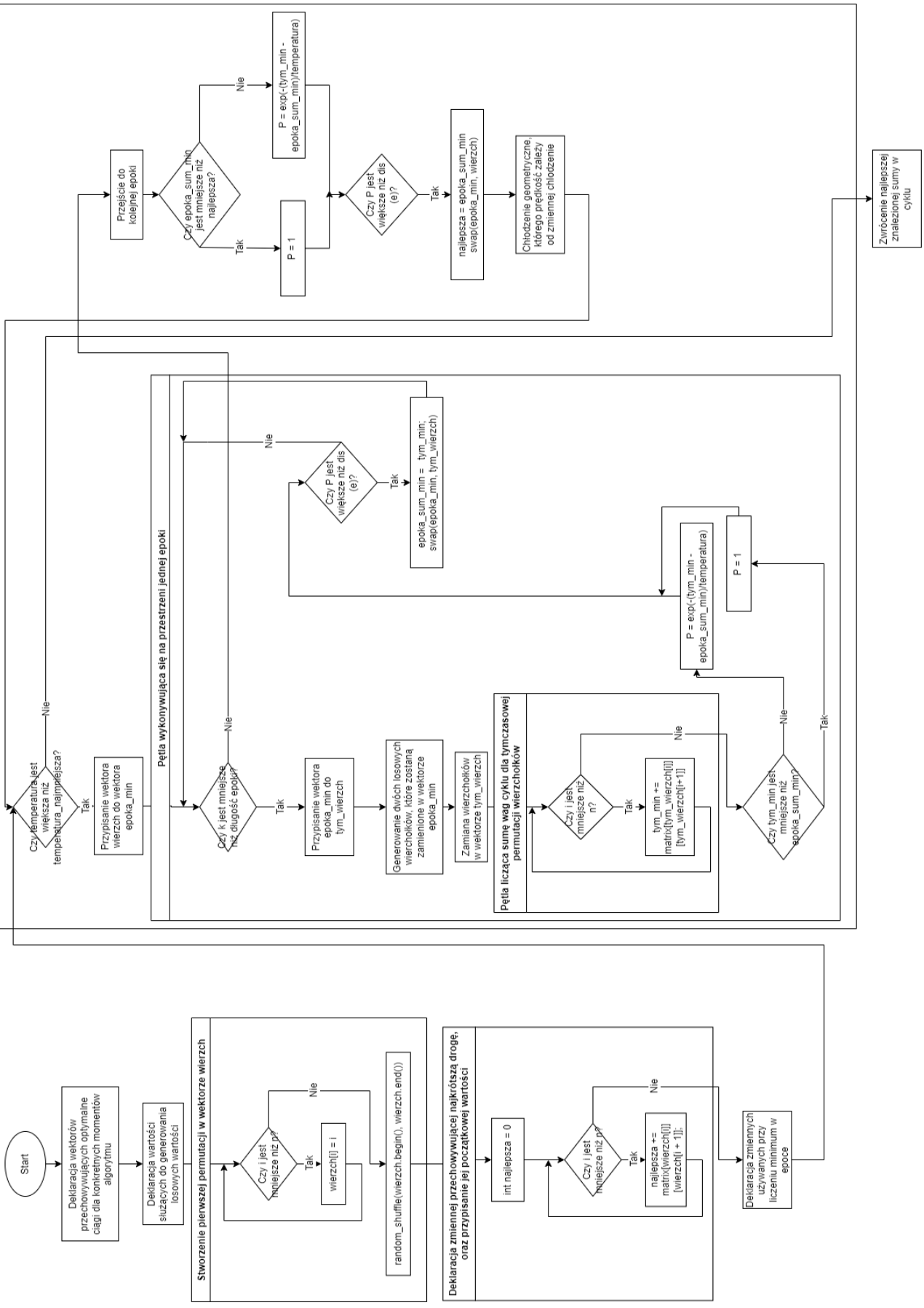
przyjmij rozwiązanie *y* pomimo, że jest gorsze

3. Algorytm

W implementacji algorytmu użyto schematu chłodzenia geometrycznego, przeglądu sąsiedztwa typu greedy, oraz wyboru rozwiązania w sąsiedztwie typu 2-zamiany. Funkcja realizująca algorytm przyjmuje na wejściu macierz sąsiedztwa, ilość wierzchołków w instancji, temperaturę początkową oraz zmienną, która jest czynnikiem determinującym prędkość chłodzenia. Algorytm generuje trzy wektory, odpowiednio: dotychczasowy najkrótszy cykl, najkrótszy cykl w epoce i tymczasowa permutacja. Gdy temperatura jest większa od przewidzianej temperatury minimalnej program przechodzi od epoki do epoki. Na przestrzeni jednej epoki algorytm losuje dwa wierzchołki w najkrótszym cyklu w epoce, a następnie zamienia je miejscami tworząc nową permutację. Jeśli suma wag cyklu jest mniejsza niż najlepszy dotychczasowy wynik w epoce, lub jeśli warunek przedstawiony w punkcie drugim jest spełniony wynik nowej permutacji staje się najlepszym w epoce. Po zakończeniu epoki, jej najlepszy wynik porównywany jest z najlepszym dotychczasowym wynikiem. Jeśli najlepszy wynik w epoce jest mniejszy niż najlepszy dotychczasowy wynik, lub jeśli warunek przedstawiony w punkcie drugim jest spełniony wynik nowej permutacji staje się najlepszym dotychczasowym wynikiem. Po całym przejściu następuje chłodzenie geometryczne za pomocą wzoru $T = \alpha^k$, gdzie k to numer epoki, następnie ma miejsce przejście do kolejnej epoki i kontynuacja działania algorytmu, aż do momentu osiągnięcia temperatury minimalnej.



Rys. 1. – Schemat blokowy programu



Rys. 2. – Schemat blokowy algorytmu

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:

1.fri26.txt, war. optymalna: 937; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

2.ftv33.txt, war. optymalna: 1286; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

3.ftv35.txt, war. optymalna: 1473; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

4.ftv38.txt, war. optymalna: 1530; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

5.dantzig42, war. optymalna: 699; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

6.ftv44.txt, war. optymalna: 1520; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

7.berlin52.txt, war. optymalna: 7542; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

8.gr96.txt, war. optymalna: 55209; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

9.kroA100.txt, war. optymalna: 21282; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

10.ch130.txt, war. optymalna: 6110; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

11.kroB150.txt, war. optymalna:26130; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

12.ftv170.txt, war. optymalna: 2755; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

13.rbg323.txt, war. optymalna: 1326; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

14.rbg358.txt, war. optymalna: 1163; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

15.rbg443.txt, war. optymalna: 2720; http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

5. Procedura badawcza

W przypadku rozwiązywania problemu komiwojażera za pomocą algorytmu symulowanego wyżarzania pojawiają się parametry, których zmiany wpływają na efektywność działania programu. Są nimi:

- Temperatura początkowa
- Sposób chłodzenia
- Parametr α chłodzenia
- Sposób generowania początkowej permutacji
- Długość epoki

W przypadku implementacji opisanej wcześniej zastosowano sposób chłodzenia geometryczny. Pomiary zostały wykonane dla różnych temperatur oraz różnych α chłodzenia. Zależność temperatury była badana dla wartości: 100, 1000, 10000 oraz $n \cdot 100$, gdzie n to liczba wierzchołków w instancji. Zależność α chłodzenia została zbadana dla 0.99, 0.999 oraz 0.9999. Dla każdej wartości parametrów zostały wykonane 4 pomiary.

Pomiary zostały wykonane na komputerze ze specyfikacją: Procesor AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz. Zainstalowana pamięć RAM 8,00 GB (dostępne: 7,37 GB). Typ systemu 64-bitowy system operacyjny, procesor x64. Do pomiaru czasu została użyta biblioteka <chrono>, wraz z narzędziem high_resolution_clock. Fragment kodu w języku C++ wykonujący pomiar czasu:

```
auto start = high_resolution_clock::now();

suma_cyklu = heldkarp(num_wierzch, matrix);

auto stop = high_resolution_clock::now();

auto duration = duration_cast<nanoseconds>(stop - start);

float time = (duration.count())/(1e+9);

cout << "Time [ms] : " << std::fixed << std::setprecision(3) << time*1000
```

6. Wyniki

Wyniki zgromadzone zostały w pliku: Wykres.xlsx. Wyniki pomiarów dla α chłodzenia 0.999 i temperatur:

- 100

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	7.02	24.756
ftv33.txt	12.80	33.635
ftv35.txt	11.25	36.765
ftv38.txt	13.22	40.520
dantzig42.txt	11.77	40.016
ftv44.txt	13.96	40.681

- 1000

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	1.77	30.510
ftv33.txt	12.62	42.770
ftv35.txt	15.46	45.493
ftv38.txt	18.30	50.521
dantzig42.txt	10.26	50.857
ftv44.txt	20.23	50.573

- 10000

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	4.48	37.765
ftv33.txt	16.82	51.715
ftv35.txt	16.77	55.018
ftv38.txt	9.26	60.019
dantzig42.txt	13.48	60.443
ftv44.txt	9.01	60.665

- $n \cdot 50$

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	3.77	31.762
ftv33.txt	14.15	44.760
ftv35.txt	12.32	50.522
ftv38.txt	12.08	55.200
dantzig42.txt	11.09	54.960
ftv44.txt	12.81	54.429

Wyniki pomiarów dla temperatury $n \cdot 50$ i α chłodzenia:

- 0.99

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	11.61	3.501
ftv33.txt	27.31	5.005
ftv35.txt	22.20	5.254
ftv38.txt	23.87	6.008
dantzig42.txt	25.89	6.503
ftv44.txt	24.69	6.003

- 0.999

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	3.77	31.762
ftv33.txt	14.15	44.760
ftv35.txt	12.32	50.522
ftv38.txt	12.08	55.200
dantzig42.txt	11.09	54.960
ftv44.txt	12.81	54.429

- 0.9999

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	4.38	336.756
ftv33.txt	11.53	478.272
ftv35.txt	11.17	510.697
ftv38.txt	6.93	565.756
dantzig42.txt	5.44	566.862
ftv44.txt	8.67	566.549

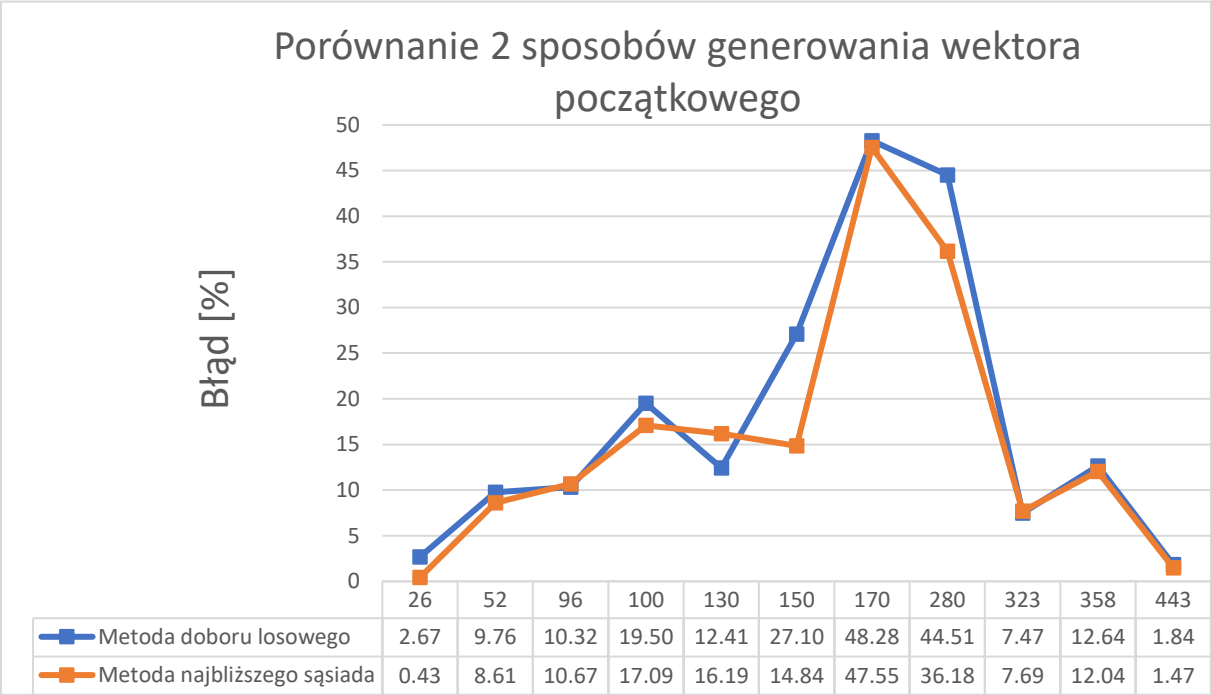
Wyniki pomiarów dla temperatury $n \cdot 50$ i α chłodzenia 0.9999 :

- Dobór wektora początkowego metodą losowej permutacji

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	2.67	345.856
berlin52.txt	9.76	812.374
gr96.txt	10.32	2097.682
kroa100.txt	19.50	2182.669
ch130.txt	12.41	3427.420
krob150.txt	27.10	4301.998
ftv170.txt	48.28	5432.598
a280.txt	44.51	14102.816
rbg323.txt	7.47	19240.451
rbg358.txt	12.64	23568.023
rbg443.txt	1.84	36471.590

- Dobór wektora początkowego metodą najbliższego sąsiada

Nazwa pliku	Błąd [%]	Średnia czasu [ms]
fri26.txt	0.43	293.584
berlin52.txt	8.61	743.998
gr96.txt	10.67	1962.581
kroa100.txt	17.09	2111.128
ch130.txt	16.19	3290.905
krob150.txt	14.84	4136.380
ftv170.txt	47.55	5237.050
a280.txt	36.18	13660.471
rbg323.txt	7.69	18466.256
rbg358.txt	12.04	22911.850
rbg443.txt	1.47	36098.801



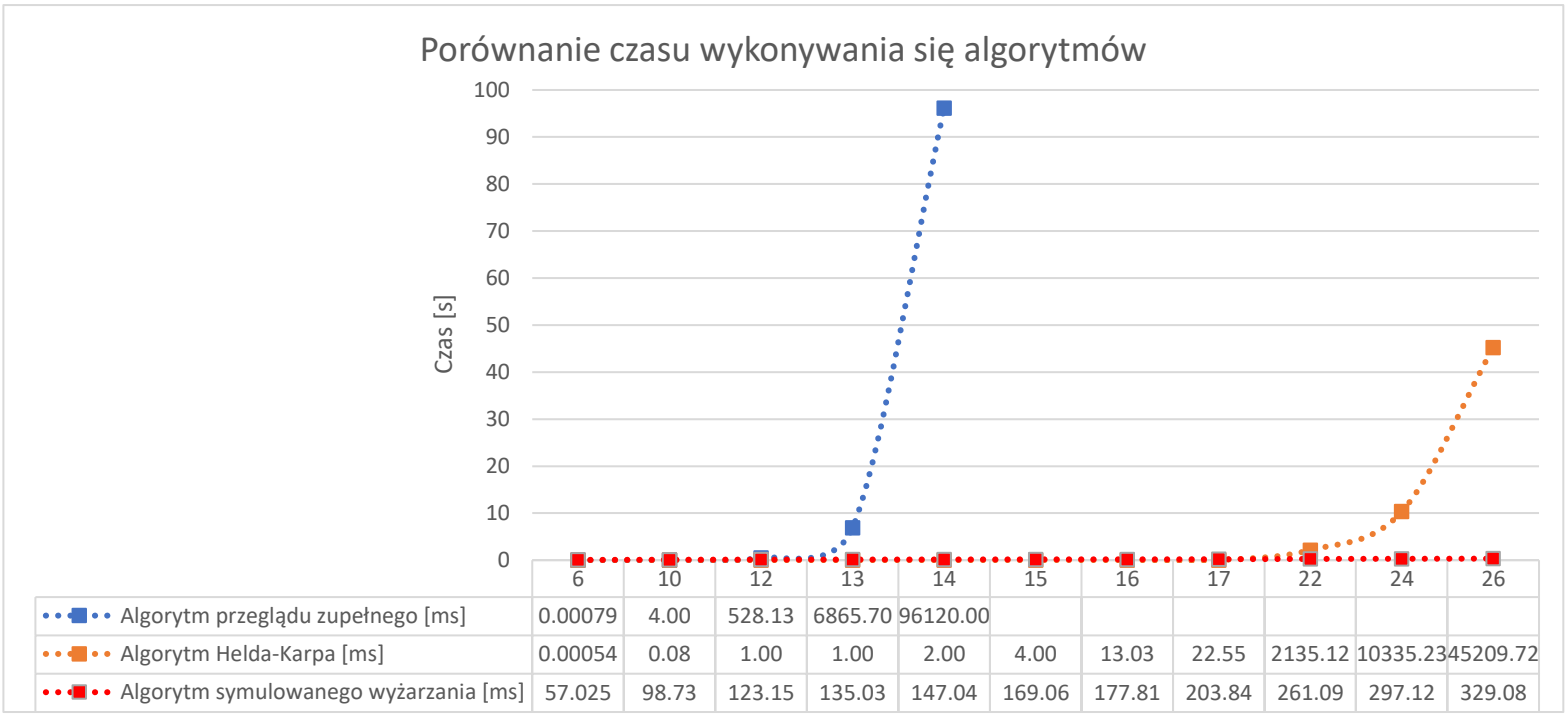
	26	52	96	100	130	150	170	280	323	358	443
Metoda doboru losowego	2.67	9.76	10.32	19.50	12.41	27.10	48.28	44.51	7.47	12.64	1.84
Metoda najbliższego sąsiada	0.43	8.61	10.67	17.09	16.19	14.84	47.55	36.18	7.69	12.04	1.47

Rys 3. – Wykres porównujący procent błędu dla 2 sposobów generowania początkowej permutacji

Po analizie wyników przy pomiarach zmieniających temperaturę wywnioskowano, że temperatura z najlepszym stosunkiem czasu do poprawności to ta wykorzystująca liczbę wierzchołków jako parametr. Wynika z tego, że najlepiej uzależnić startową wielkość temperatury od liczby wierzchołków. W analizie wyników przy pomiarach zmieniających α chłodzenia zaobserwowano wzrost dokładności wyników wraz ze wzrostem α. Wartością, która ma najlepszy stosunek czasu do poprawności jest 0.999 i 0.9999. Zostały przeprowadzone pomiary dla dwóch sposobów generowania startowej permutacji: permutacja wierzchołków w kolejności od 1 do n i permutacja wierzchołków w kolejności losowej. Po testach stwierdzono jednak brak różnicy w dokładności dla większości instancji. Następnie porównano generowanie losowe z metodą najbliższego sąsiada. Zauważalne są różnice na korzyść wyżej wymienionej metody jednak nie są one znaczące. Wynika to z faktu, że wybór lepszej formy generowania startowej permutacji związany jest z tym, jak blisko startowego wektora jest optymalne rozwiązanie dla instancji.

7. Analiza wyników i wnioski

W celu porównania wyników zadania trzeciego zostały zamieszczone wyniki uzyskane w zadaniu pierwszym i drugim:



	6	10	12	13	14	15	16	17	22	24	26
Algorytm przeglądu zupełnego [ms]	0.00079	4.00	528.13	6865.70	96120.00						
Algorytm Helda-Karpa [ms]	0.00054	0.08	1.00	1.00	2.00	4.00	13.03	22.55	2135.12	10335.23	45209.72
Algorytm symulowanego wyżarzania [ms]	57.025	98.73	123.15	135.03	147.04	169.06	177.81	203.84	261.09	297.12	329.08

Rys 4. – Wykres porównujący czasy wykonywania algorytmów w zależności od wielkości instancji

Na powyższym wykresie zauważalny jest fakt, że algorytm symulowanego wyżarzania działa znacząco wolniej niż algorytm Helda-Karpa dla instancji, które mają mniej niż 22 wierzchołków. Jednak w przypadku instancji o większej ilości wierzchołków symulowane wyżarzanie działa zauważalnie szybciej niż programowanie dynamiczne. Algorytm symulowanego wyżarzania deklasuje algorytm przeglądu zupełnego w kwestii czasu. Największym problemem algorytmu symulowanego wyżarzania jest to, że nie znajduje on rozwiązania optymalnego dla instancji większych niż 20, tylko takie które jest blisko optymalnego. Algorytm jest w stanie znaleźć rozwiązanie optymalne, ale nie jest to gwarantowane. Dla każdej instancji istnieje możliwość dopasowania parametrów takich jak temperatura, długość epoki, rodzaj chłodzenia oraz permutacja startowa w taki sposób, że wielkość błędu można sprowadzić do wartości bliskich zera, jednak ciągle nie można być pewnym, że program zwróci wynik optymalny. Złożoność wykorzystanej implementacji algorytmu to $O = n \times d \times log_{\propto}(\frac{T_{min}}{T})$, gdzie n to ilość wierzchołków w instancji, d to długość epoki, a $log_{\propto}(\frac{T_{min}}{T})$, to ilość epok. Sam algorytm można usprawnić zwiększając precyzyjność wyników oraz zmniejszając czas, co możliwe jest za pomocą min. implementacji innych metod chłodzenia, oraz doboru parametrów początkowych w zależności od instancji za pomocą algorytmów.