Projektowanie Efektywnych Algorytmów

Projekt

22/12/2022

numer indeksu	imię i nazwisko
259190	Jędrzej Czykier

(3) Symulowane wyżarzanie

Spis treści strona

1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmów rozwiązujących problem komiwojażera, nazywanego również TSP *(eng. Traveling Salesman Problem)* w wersji optymalizującej. Problem ten, polega na znalezieniu minimalnego cyklu Hamiltona czyli znając zbiór wierzchołków grafu oraz wagi połączeń między nimi, należy znaleźć drogę w której każdy wierzchołek jest odwiedzany tylko raz, z wyjątkiem pierwszego, do którego algorytm powinien wrócić na końcu działania.

Problem ten jest NP-trudny, co oznacza że jego rozwiązanie jest co najmniej tak trudne, jak rozwiązanie innych problemów należących do klasy NP. Przykładami innych problemów należących do tej klasy są między innymi: problem plecakowy i problem zbioru niezależnego.

Jednym ze sposobów rozwiązania TSP jest metoda Brute-force. Algorytm ten sprawdza po kolei wszystkie możliwości, co czyni go relatywnie prostym do implementacji, lecz powolnym w działaniu. Kolejnym jest algorytm Helda-Karpa, nazywany też algorytmem Bellmana-Helda-Karpa. Wykorzystuje on programowanie dynamiczne aby rozwiązać problem komiwojażera. Oba te rozwiązania są algorytmami dokładnymi, co oznacza że zawsze znajdą optymalne rozwiązanie.

Do rozwiązywania problemów NP-trudnych często stosuje się algorytmy heurestyczne, czyli takie, które nie zapewniają optymalnych rozwiązań. Ze względu na wysoką złożoność obliczeniową algorytmów dokładnych, czasami godzi się na rozwiązanie wystarczająco dobre, które można otrzymać w krótszym czasie. Do takich algorytmów zalicza się symulowane wyżarzanie, które można wykorzystać do rozwiązania problemu komiwojażera.

2. Metoda

Metoda symulowanego wyżarzania należy do rodziny algorytmów heurystycznych. Cechuje się występowaniem parametru sterującego zwanego „temperaturą”, który zmniejsza się podczas wykonywania się algorytmu. Odpowiada on za chaotyczność zmian – im większy jest, tym zmiany są bardziej chaotyczne.

Algorytm ten i jego nazwę, zainspirowała obserwacja z dziedziny metalurgii w której zauważono, że im większa jest temperatura metalu, tym bardziej jest od plastyczny czyli podatny na zmiany.

Przy rozwiązywaniu problemu komiwojażera metoda symulowanego wyżarzania, polega na otrzymaniu w jakiś sposób początkowego rozwiązania np. metodą najbliższego sąsiada, następnie iteracyjnym wyznaczaniu rozwiązania z sąsiedztwa aktualnie rozpatrywanego rozwiązania. Jeżeli nowe rozwiązanie jest lepsze od aktualnego zostaje one zapisane i rozpatrywane jest jego sąsiedztwo. W celu uniknięcia utknięcia w minimum lokalnym, istnieje szansa na rozpoczęcie rozpatrywania gorszego rozwiązania. Prawdopodobieństwo zaistnienia takiego zdarzenia jest zależne od temperatury oraz wartości nowego i rozpatrywanego rozwiązania. Wyliczane jest ze wzoru:

$$p = exp(-|\frac{f(new) - f(old)}{T_k}|)$$

Z każdą iteracją algorytmu temperatura jest zmniejszana, a wraz z nią, zmniejsza się prawdopodobieństwo przyjęcia gorszego rozwiązania. Najpopularniejszymi funkcjami zmiany temperatury, nazywanymi także schematami chłodzenia są np. schemat geometryczny i schemat logarytmiczny (Boltzmana).

Schemat logarytmiczny (Boltzmana)

$$\alpha(T) = \frac{T}{a + b \log k}$$

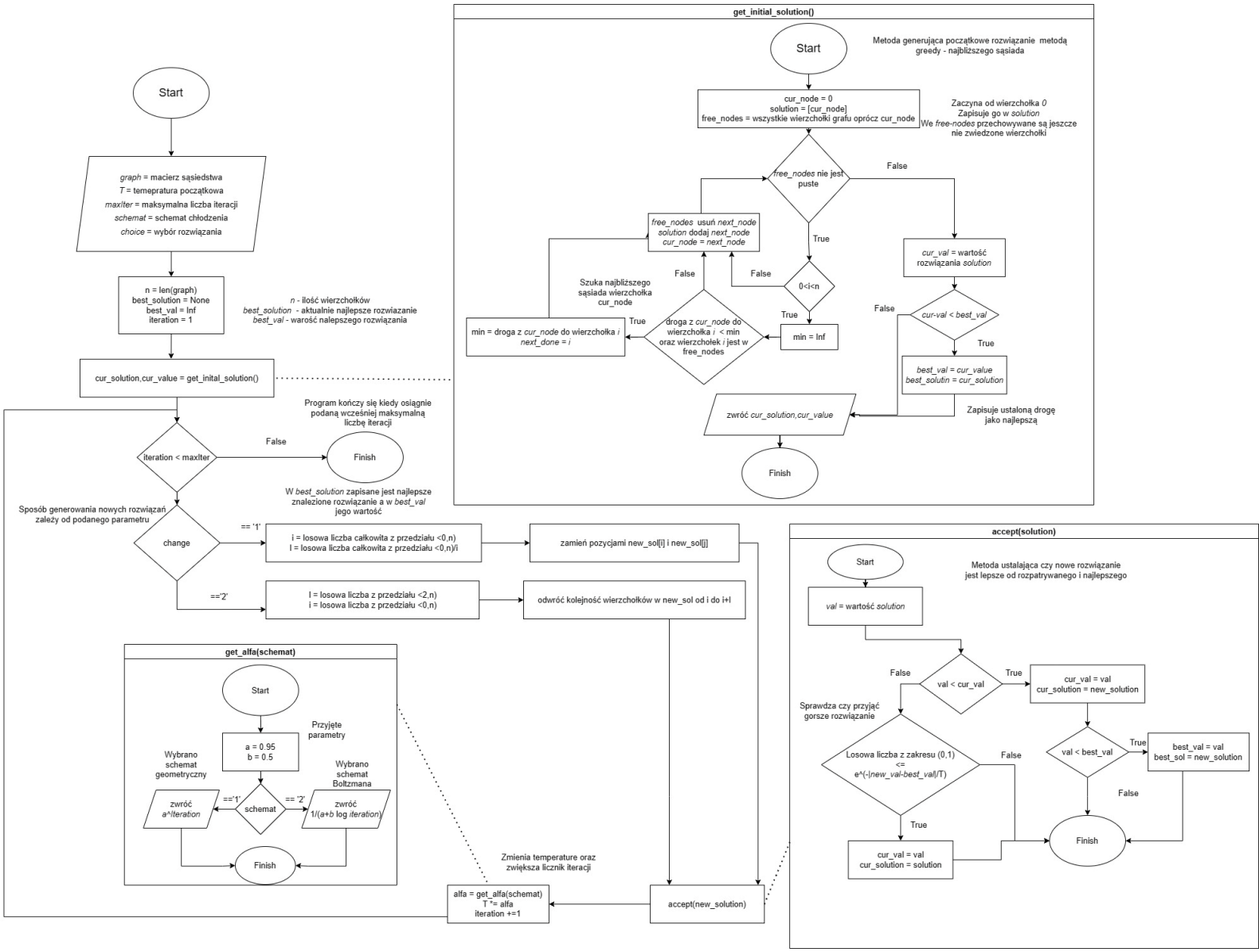
gdzie: a,b pewne współczynniki, k zmienna np. numer epoki

Schemat geometryczny

$$\alpha(T) = a^k T$$

gdzie: 0 < a < 1

3. Opis algorytmu



Rysunek 1 Przedstawienie algorytmu za pomocą schematu blokowego

4. Dane testowe

Dane wykorzystane do sprawdzenia poprawności algorytmu oraz poprawne wyniki:

Nazwa pliku, optymalna ścieżka, waga optymalnej ścieżki

1. tsp_6_1.txt, [0, 1, 2, 3, 4, 5], 132,

2. tsp_6_2.txt , [0, 5, 1, 2, 3, 4], 80,

3. tsp_13.txt, [0, 12, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10], 269

Powyższe pliki można znaleźć po adresem:

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Dane wykorzystane do badań:

Nazwa pliku, waga najlepszej znanej ścieżki

1. tsp_10.txt, 212

2. gr24.txt, 1272

3. gr48.txt, 5046

4. gr96.txt, 55209

5. gr120.txt, 6942

6. gr229.txt, 134602

Powyższe pliki można znaleźć pod adresami:

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Na potrzebę testów, pliki zaczynające się na „gr” zostały edytowane aby ich treść przyjmowała postać macierzy sąsiedztwa.

5. Procedura badawcza

Należało zbadać wpływ temperatury początkowej T_0, przyjętego schematu chłodzenia, długość epoki *L* oraz sposobu generowania rozwiązania w sąsiedztwie rozwiązania bieżącego na jakość uzyskanych rozwiązań, czas uzyskiwania rozwiązań oraz zużycie pamięci.

Rozwiązanie początkowe generowane było metodą greedy – najbliższego sąsiada, poczynając od wierzchołka *0.* Zbadana została zależność błędu względnego od liczby epok oraz temperatury początkowej dla kombinacji schematów chłodzenia: logarytmicznego i geometrycznego oraz sposobu generowania sąsiednich rozwiązań metodą 2-zmianową oraz odwrócenia ścieżki między wierzchołkami polegającej na wylosowaniu *i* z przedziału <0,n) oraz *l* z przedziału <2,n), gdzie n to ilość wierzchołków w grafie, oraz odwrócenie ścieżki między wierzchołkami *i* oraz *i+l*.

Treść pliku .ini:

[section_a]

file0 = tsp_10.txt 50 212 1000 100000 2 2

file1 = gr24.txt 30 1272 1000 100000 2 2

file2 = gr48.txt 10 5046 1000 100000 2 2

file3 = gr96.txt 10 55209 10000 100000 2 2

file4 = gr120.txt 10 6942 10000 100000 2 2

file5 = gr229.txt 5 134602 10000 100000 2 2

[section_b]

outputFile = outputSA.csv

Każda z instancji przetestowana została, podaną obok nazwy pliku ilość razy, np. tsp_10.txt wykonana została 50 razy. Kolejne liczby oznaczają po kolei: wartość najlepszej znanej ścieżki, temperaturę początkową, liczbę epok, wybrany schemat chłodzenia (1 – logarytmiczny, 2 – geometryczny) oraz sposób generowania sąsiednich rozwiązań (1 – 2-zmian, 2 – odwrócenie ścieżek). Do pliku wyjściowego outputSA.csv zapisywany był czas wykonania się algorytmu, otrzymane rozwiązanie (koszt ścieżki), ścieżka (numery kolejnych węzłów), ilość pamięci użytej przy wykonywaniu się algorytmu, otrzymany błąd względny oraz zastosowane w algorytmie parametry. Plik wyjściowy zapisywany był w formacie csv.

Błąd obliczany był ze wzoru:

$$err = \frac{wartość\ otrzymana\ -\ wartość\ optymalna}{wartość\ optymalna} * 100\%$$

Czas wykonywania się algorytmu był mierzony za pomocą funkcji z biblioteki *time*, a użyta pamięć za pomocą funkcji z biblioteki *tracemalloc.* Obie biblioteki wbudowane są w język Python.

Poniżej przedstawiono fragment zawartości pliku wyjściowego.

#file,path,value,T,maxIter,schemat,wybor,error,time,memory

tsp_10.txt,"[7,6,9, 1, 5, 0, 3, 4,2,8]", 212, 1000, 100000, 2, 2, 0.0, 0.919482946395874, 0.007859230041503906

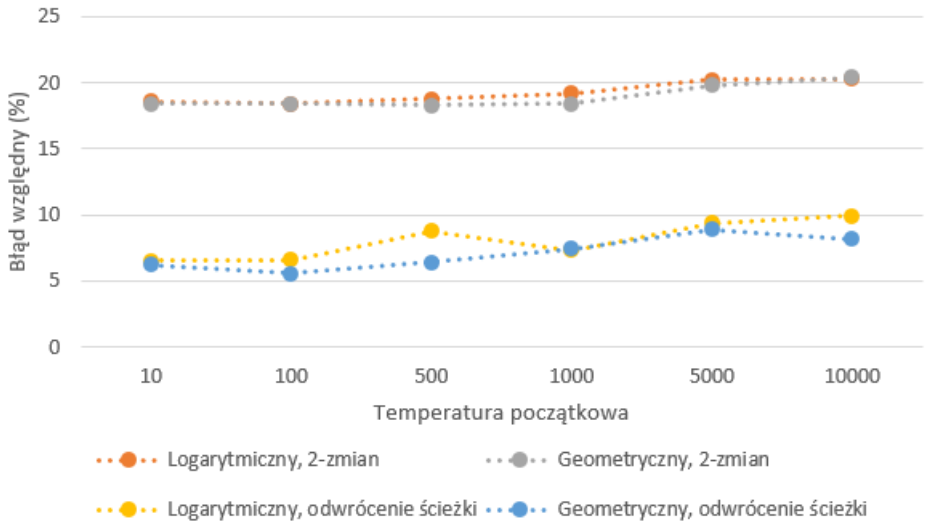
tsp_10.txt,"[1, 6, 7, 5, 0, 3, 4, 8, 2, 9]", 253, 1000, 100000, 2, 2, 19.339622641509436, 0.8642683029174805, 0.00089263916015625

….

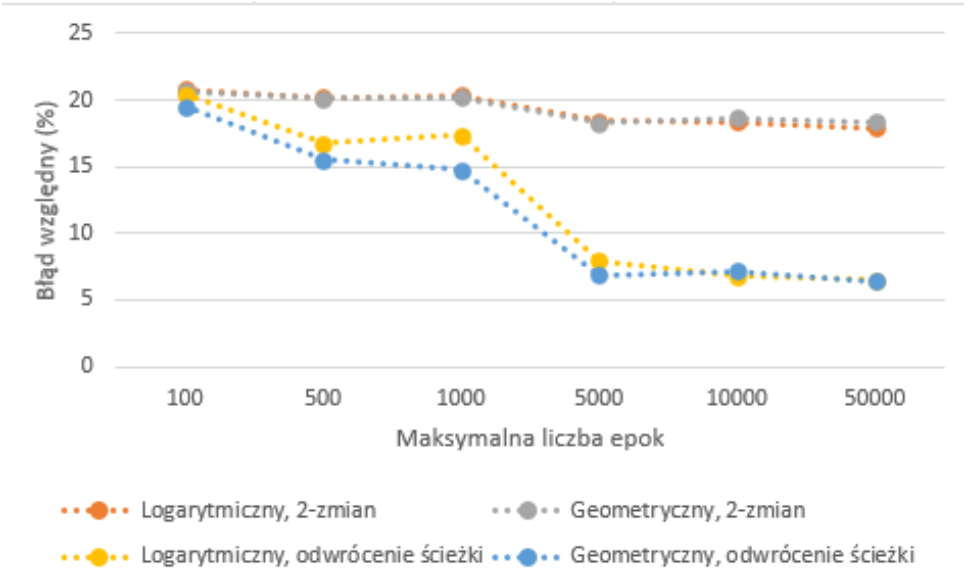
gr229.txt,"[1, 0, 27, 28, 41, 44, 43, 42, 223, 224, 225, 228, 227, 226, 217, 216, 218, 219, 215, 214, 213, 212, 207, 206, 211, 204, 205, 208, 209, 210, 203, 149, 154, 148, 147, 146, 145, 144, 143, 142, 141, 138, 139, 140, 151, 152, 153, 150, 156, 157, 202, 201, 220, 221, 222, 199, 200, 155, 158, 159, 160, 161, 162, 171, 163, 172, 164, 198, 197, 196, 177, 175, 174, 173, 176, 167, 169, 168, 170, 127, 129, 131, 130, 132, 134, 133, 136, 135, 137, 128, 126, 125, 165, 124, 123, 109, 110, 113, 116, 119, 122, 120, 121, 118, 117, 115, 114, 111, 112, 108, 107, 103, 104, 106, 105, 100, 101, 102, 73, 74, 75, 76, 69, 72, 71, 70, 68, 67, 66, 77, 78, 87, 79, 86, 80, 81, 82, 60, 62, 65, 64, 63, 61, 59, 58, 57, 53, 54, 56, 55, 23, 22, 83, 24, 84, 85, 88, 89, 90, 91, 92, 93, 99, 98, 96, 97, 95, 94, 36, 35, 34, 33, 37, 166, 178, 180, 179, 181, 182, 184, 185, 186, 195, 194, 193, 192, 191, 190, 189, 188, 187, 48, 46, 47, 49, 183, 45, 40, 39, 38, 31, 30, 32, 29, 26, 25, 14, 13, 12, 11, 15, 16, 20, 21, 19, 18, 52, 51, 50, 10, 17, 9, 8, 5, 6, 7, 4, 3, 2]", 142102, 10000, 100000, 2, 2, 5.571982585697092, 13.139859914779663, 0.0100250244140625

6. Wyniki

Wyniki zgromadzone w pliku saPomiary.xlsx. Wszystkie pliki wynikowe zostały przesłane na Eportal.

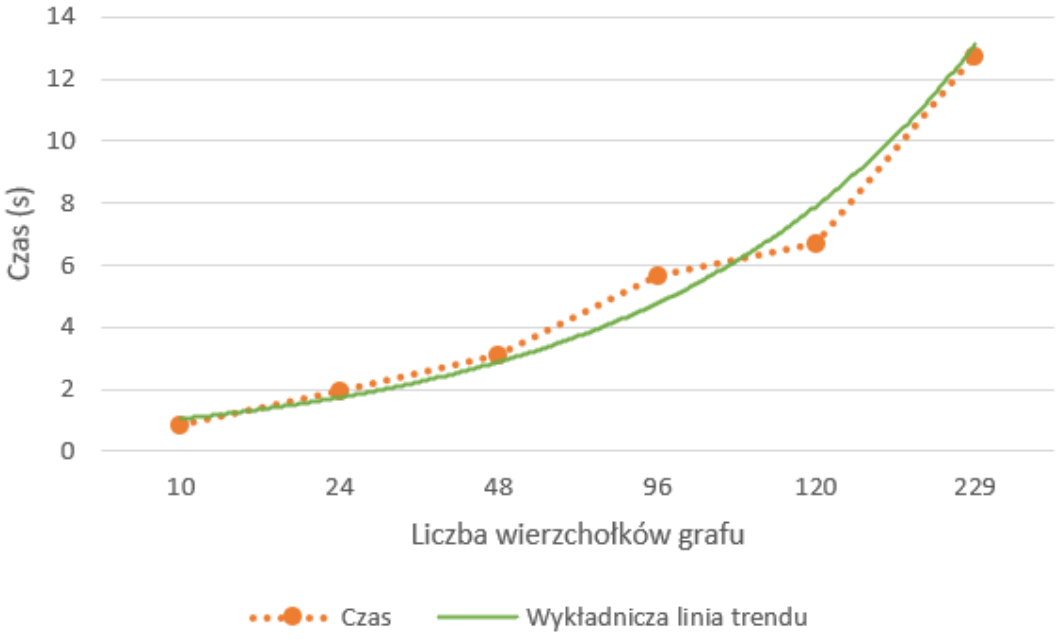


Rysunek 3 Zależność błędu względnego od temperatury początkowej dla kombinacji schematów chłodzenia i sposobu generowania sąsiednich rozwiązań, dla maksymalnej liczby epok równej 5000 w grafie o 48 wierzchołkach

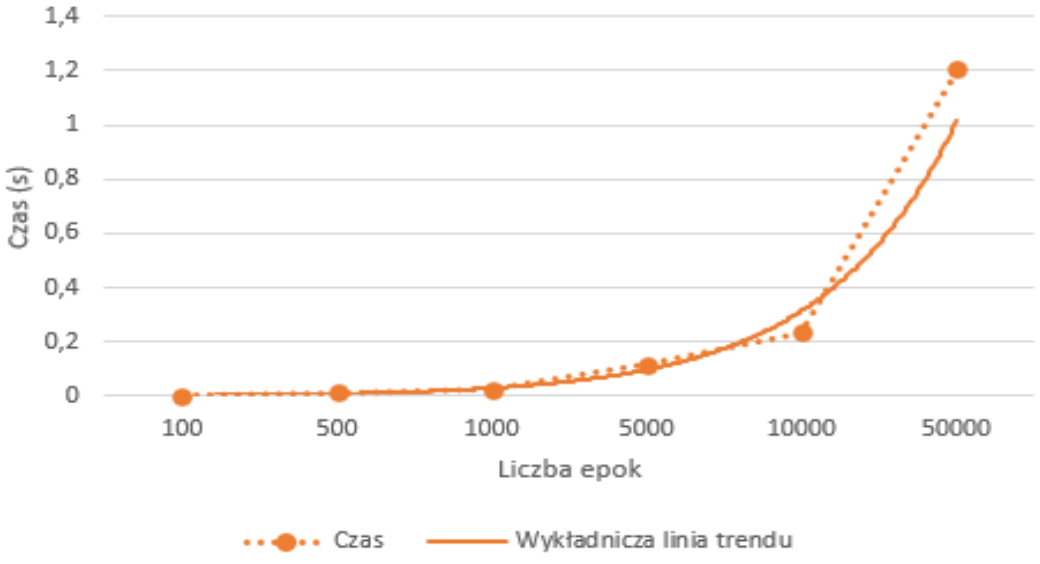


Rysunek 2 Zależność błędu względnego od maksymalnej liczby epok dla kombinacji schematów chłodzenia i sposobu generowania sąsiednich rozwiązań, dla temperatury początkowej 10000 w grafie o 48 wierzchołkach

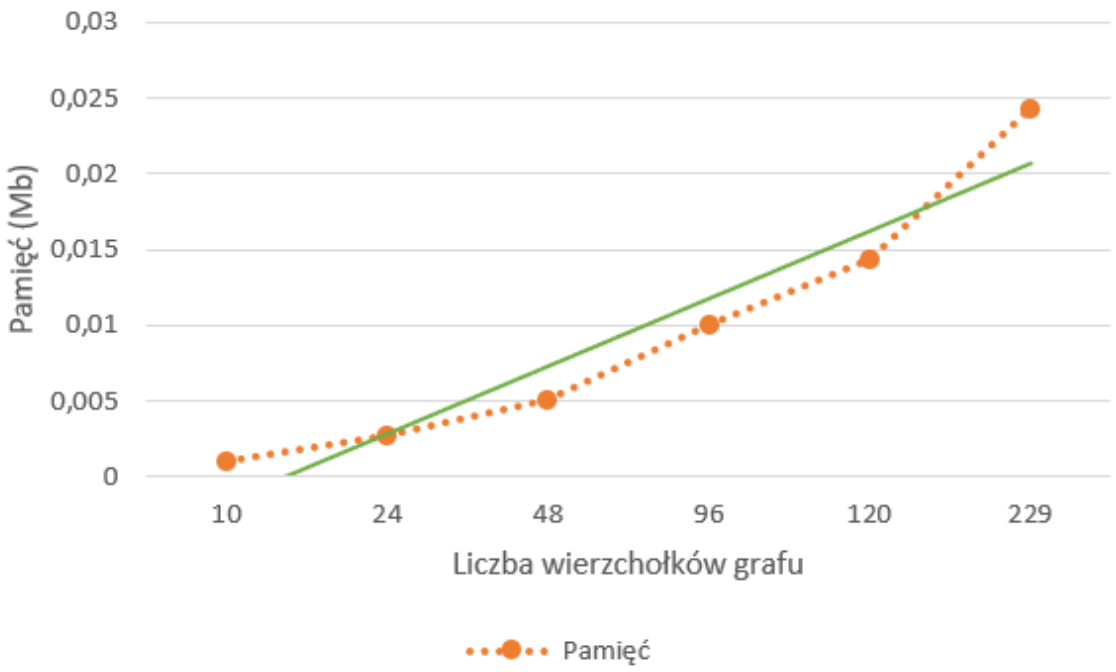
Poniższe testy zostały wykonane używając geometrycznego schematu chłodzenia oraz generowania sąsiednich rozwiązań za pomocą odwracania ścieżek, ponieważ kombinacja ta, skutkowała najmniejszym błędem względnym w poprzednich testach.



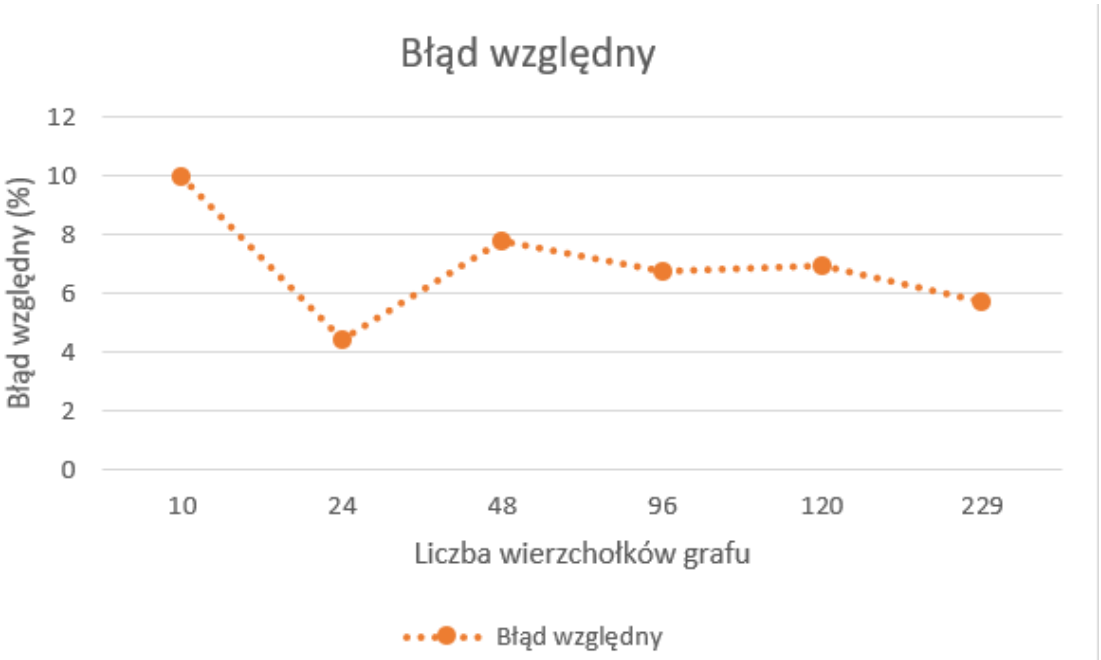
Rysunek 4 Zależność czasu wykonywania się algorytmu od liczby wierzchołków grafu. Dla parametru liczby epok równemu 10000.



Rysunek 5 Zależność czasu wykonywania się algorytmu od wartości parametru liczby epok



Rysunek 6 Zależność używanej przez algorytm pamięci od liczby wierzchołków w grafie.



Rysunek 7 Zależność wartości błędu względnego od liczby wierzchołków w grafie (dla stałych parametrów temperatura początkowa i schemat chłodzenia)

7. Analiza wyników i wnioski

Dobór odpowiedniej temperatury początkowej jest skomplikowany i różny dla każdej instancji. Jeżeli będzie za duża, algorytm może stać się zbyt losowy i chaotyczny, jeżeli będzie za mała algorytm może utknąć w minimum lokalnym.

Podobnie jest z doborem liczby epok. Na wykresie [Rysunek 3] widać zależność między wzrostem liczby epok, a spadkiem wartości błędu względnego. Liczba epok ma również wpływ na czas wykonywania się algorytmu [Rysunek 5]. Jeżeli parametr ten będzie zbyt duży, algorytm będzie wykonywał się niepotrzebnie długo, jeżeli zbyt mały – otrzymamy gorsze rozwiązanie. Aby otrzymać optymalne wyniki w stosunku do czasu pracy algorytmu, ilość epok powinna zależeć od ilości wierzchołków w grafie.

Po zbadaniu wpływu schematu chłodzenia i generacji sąsiednich rozwiązań [Rysunek 2][Rysunek 3], okazało się że schemat geometryczny generuje minimalnie lepsze rozwiązania od logarytmicznego, lecz może być to kwestia badanej instancji. Generowanie rozwiązań sąsiednich za pomocą odwracania ścieżek między wierzchołkami dało za to znacząco lepsze wyniki niż metoda 2-zmian.

Na czas wykonywania się algorytmu, oprócz ilości epok, wpływa również ilość wierzchołków w grafie. Linia trendu na wykresie [Rysunek 5] sugeruje pewną zależność wykładniczą. Mimo to algorytm wykonuje się w niedużym czasie nawet dla instancji z wieloma wierzchołkami.

Pamięć użyta podczas wykonywania się algorytmu rośnie liniowo [Rysunek 6]. Spowodowane jest to że algorytm nie potrzebuje zapamiętywać wartości innych niż reprezentacja grafu, kilku ścieżek oraz parametrów.

Błąd względny rozwiązania uzyskanego przez algorytm, nie wydaje się mieć powiązania z ilością wierzchołków w grafie [Rysunek 7]. Zależy od dobranych parametrów algorytmu oraz przypadku.