Jakub Superczyński 241381 18.12.2019

Projektowanie efektywnych algorytmów

Implementacja i analiza efektywności algorytmu symulowanego wyżarzania dla problemu komiwojażera.

Prowadzący zajęcia:

Dr inż. Dariusz Banasiak

1. Wstęp teoretyczny

Problem komiwojażera to zagadnienie optymalizacyjne z teorii grafów polegające na odnalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Dla łatwiejszego zobrazowania możemy przedstawić to w ten sposób: przedstawiciel handlowy ma listę pewnej liczby miast, które musi odwiedzić a następnie powrócić do miasta początkowego. Problem polega na tym, aby tak zaplanował swoją podróż aby przejazdy między miastami kosztowały go jak najmniej (np. aby odległości były jak najkrótsze lub jak najmniej zapłacił za paliwo). Co ważne: zakładamy, że graf może być asymetryczny, tzn. waga tej samej krawędzi (koszt tej samej trasy) może być różna w zależności w którą stronę chcemy przez nią przejść.

Jednym z algorytmów heurystycznych wykorzystywanych do rozwiązania tego problemu jest algorytm symulowanego wyżarzania. Jego działanie wzoruje się na metodę obróbki metalów w metalurgii polegającej na stopniowym ochładzaniu materiału. W dużej temperaturze cząsteczki metalu poruszają się bardzo chaotycznie, jednak wraz z ochładzaniem ich ruchy powoli maleją. Jeśli taki materiał zostanie ochłodzony zbyt szybko, cząsteczki zatrzymają się w praktycznie losowy, nieuporządkowany sposób. Gdyby jednak ochładzać go stopniowo, można doprowadzić materiał w kontrolowany sposób do stanu jaki nam odpowiada. Metoda ta znalazła swoją analogię w metodzie rozwiązywania problemu komiwojażera: pierwszym krokiem jest ustalenie początkowego rozwiązania. Jest to rozwiązanie „nieuporządkowane", tzn. losowe. Ustalamy również parametr, który przez analogię do metalurgii możemy nazwać temperaturą. Mając jedno rozwiązanie, losujemy inne rozwiązanie w jego sąsiedztwie a następnie je oceniamy. Jeśli nowe rozwiązanie jest lepsze, ustawiamy je jako rozwiązanie bieżące. Jeśli jest gorsze, następny krok jest sterowany przez instrukcję: możemy albo odrzucić to rozwiązanie albo przyjąć je jako bieżące. Im wyższa jest obecna temperatura, tym gorsze rozwiązanie może zostać przyjęte. Pseudokod tego algorytmu można przedstawić następująco:

1. Inicjuj T, T_{min}, α
2. Wybierz losowo rozwiązanie A
3. Dopóki $T > T_{min}$:
    a. Wygeneruj rozwiązanie B poprzez zamianę dwóch losowych elementów miejscami
    b. Jeśli koszt rozwiązania $f(A) > f(B)$: A = B
    c. W przeciwnym wypadku, jeśli losowa liczba $\epsilon\ (0, 1) < e^{-\frac{f(B)-f(A)}{T}}$: A = B
    d. $T = T * \alpha$
4. Zwróć rozwiązanie A

Gdzie $f(X)$ to ocena rozwiązania, tzn. koszt przejścia znalezionej ścieżki. Główną zaletą tego algorytmu jest niewielka złożoność czasowa, ma jednak poważną wadę: nie zawsze znajduje rozwiązanie optymalne. Podczas przeszukiwania przestrzeni algorytm może zatrzymać się w minimum lokalnym i nigdy nie osiągnąć szukanego w rzeczywistości minimum globalnego. W algorytmie istotną rolę odgrywa losowość, więc jest też mocno niestabilny.

2. Opis implementacji

Program składa się z dwóch klas: *TSP* odpowiedzialnej za przechowywanie informacji o instancji (tablica sąsiedztwa grafu oraz rozmiar problemu) , generowanie losowych instancji, wczytywanie instancji z pliku i wypisywanie obecnie wczytanego grafu; oraz *SA* odpowiedzialnej za znalezienie rozwiązania dla danej instancji.

Tablica sąsiedztwa grafu przechowywana jest w klasie *TSP* dwuwymiarowej tablicy alokowanej dynamicznie. Pamięć na nią alokowana jest zawsze przed wygenerowaniem lub wczytaniem nowej instancji poprzez wywołanie funkcji *Initialize()*. Dla ułatwienia implementacji klasa *SA* również posiada zapisane rozmiar problemu oraz wskaźnik na tablicę z reprezentacją grafu. Dodatkowo klasa *SA* posiada dynamicznie alokowaną tablicę *path[]* aby zapisywać obecnie sprawdzane rozwiązanie (pamięć jest rezerwowana na (rozmiar problemu + 1) miejsc, aby móc zapamiętać powrót z ostatniego do pierwszego wierzchołka). Program nie potrzebuje dwóch tablic na rozwiązania „stare i nowe", ponieważ wszystkie operacje zamiany elementów odbywają się na tej samej tablicy, a jeśli rozwiązanie zostaje odrzucone, następuje powrót do starszego rozwiązanie poprzez powrotną zamianę.

3. Plan eksperymentu

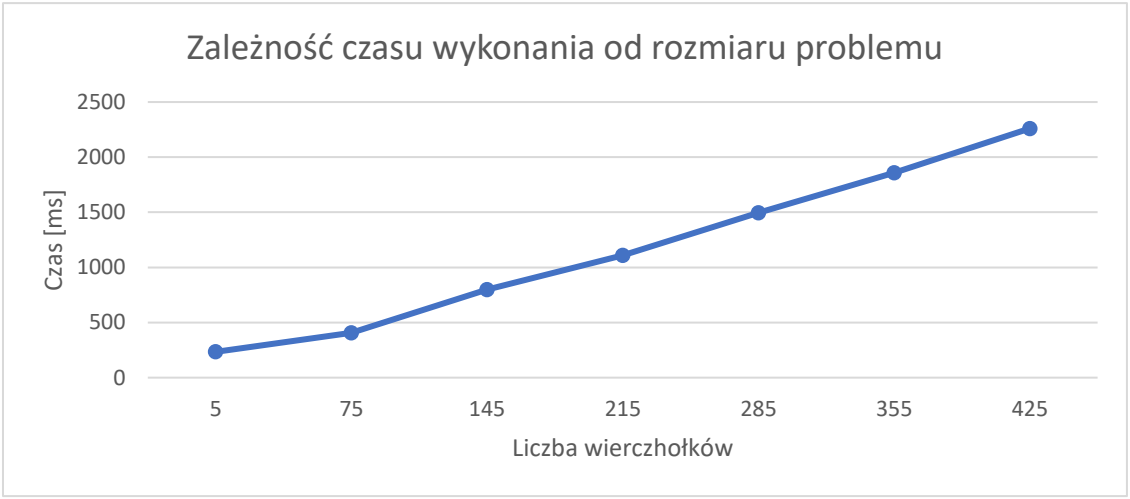
Jako że algorytm symulowanego wyżarzania nie daje optymalnych wyników, w eksperymencie należy sprawdzić dwie cechy algorytmu: czas wykonania oraz poprawność wyniku. Do sprawdzenia poprawności wykorzystano przykładowe instancje asymetrycznego problemu komiwojażera z biblioteki tsplib. Największa instancja w tej bibliotece ma rozmiar 443 wierzchołków (*rbg443.txt)*, dlatego założono że w eksperymencie nie przekroczy się tego rozmiaru. Pierwszym krokiem było poglądowe sprawdzenie dla jakiej wartości współczynnika chłodzenia (α) algorytm da akceptowalne wyniki dla instancji *rbg443.txt*. Okazało się, że akceptowalne wyniki przy również niedużym czasie realizacji pojawiały się już przy współczynniku chłodzenia równym 0,99999. Dlatego eksperyment sprawdzający zależność czasu rozwiązania od rozmiaru problemu postanowiono realizować przy następujących współczynnikach: temperatura początkowa = 10000, temperatura końcowa = 0.001, współczynnik chłodzenia = 0,99999. Rozmiary instancji dla których eksperyment był przeprowadzany są następujące: 5, 75, 145, 215, 285, 355, 425 miast. Dla każdego rozmiaru przeprowadzono 100 pomiarów (za każdym razem generując losową instancję o zadanym rozmiarze) i wyciągnięto średnią. Oprócz tego zaplanowano eks

W nieco inny sposób wykonano eksperyment dotyczący poprawności. Planem było sprawdzenie w jaki sposób średni błąd rozwiązania zależy od współczynnika chłodzenia dla różnych rozmiarów problemu. Dlatego dla 7 różnych instancji: *data5.txt, data10.txt, br17.atsp, ftv33.atsp, ft70.atsp, rbg323.atsp, rbg443.atsp* oraz dla 6 różnych wartości współczynnika: 0,9; 0,99; 0,999; 0,9999; 0,99999; 0,999999 przeprowadzono 50 pomiarów i wyciągnięto średnią. Następnie porównano średnią do znanego optymalnego rozwiązania.

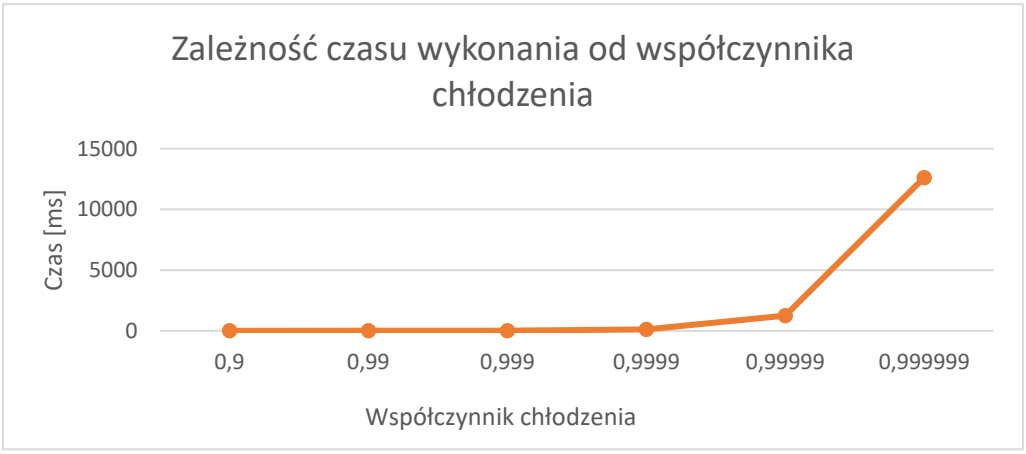
4. Wyniki eksperymentów

Sprawdzenie czasu wykonania

n	t[ms]
5	233,8693
75	407,0028
145	797,7865
215	1109,938
285	1495,158
355	1859,166
425	2259,939



α	t[ms]
0,9	0,665042
0,99	1,80089
0,999	13,72152
0,9999	127,7566
0,99999	1254,554
0,999999	12617,79



Sprawdzenie błędu

instancja	data5					
rozwiązanie idealne	22					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	25,32	22,48	22	22	22	22
średni błąd	15,09%	2,18%	0,00%	0,00%	0,00%	0,00%

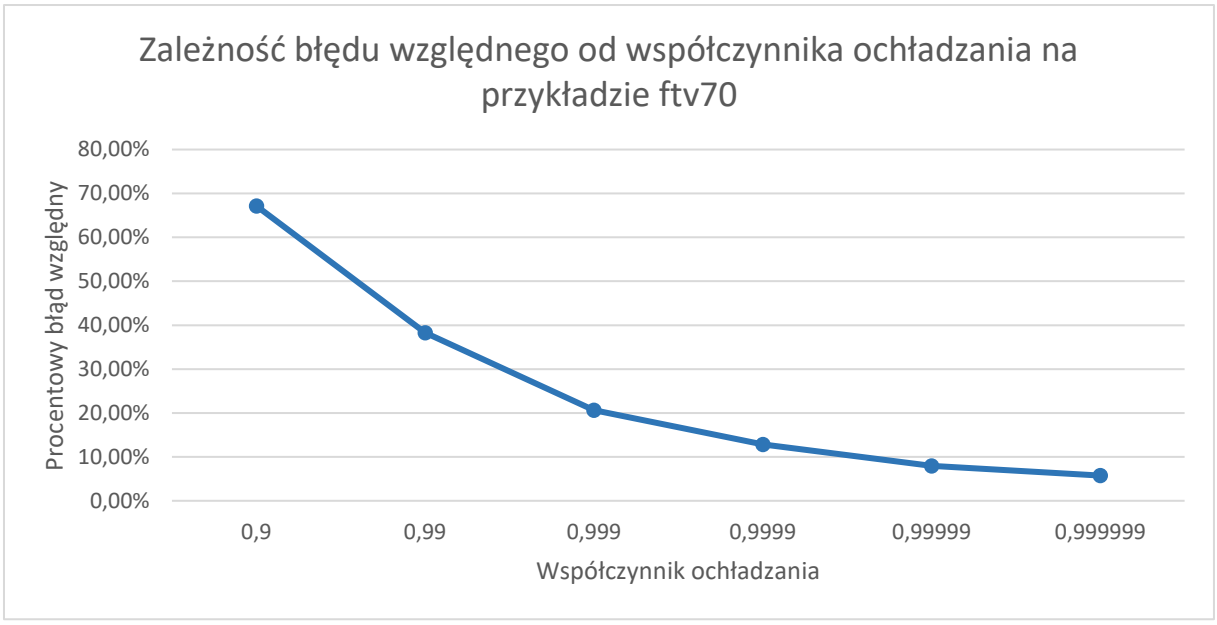
instancja	data10					
rozwiązanie idealne	212					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	304,6	239,92	228,18	236,18	239,74	239
średni błąd	43,68%	13,17%	7,63%	11,41%	13,08%	12,74%

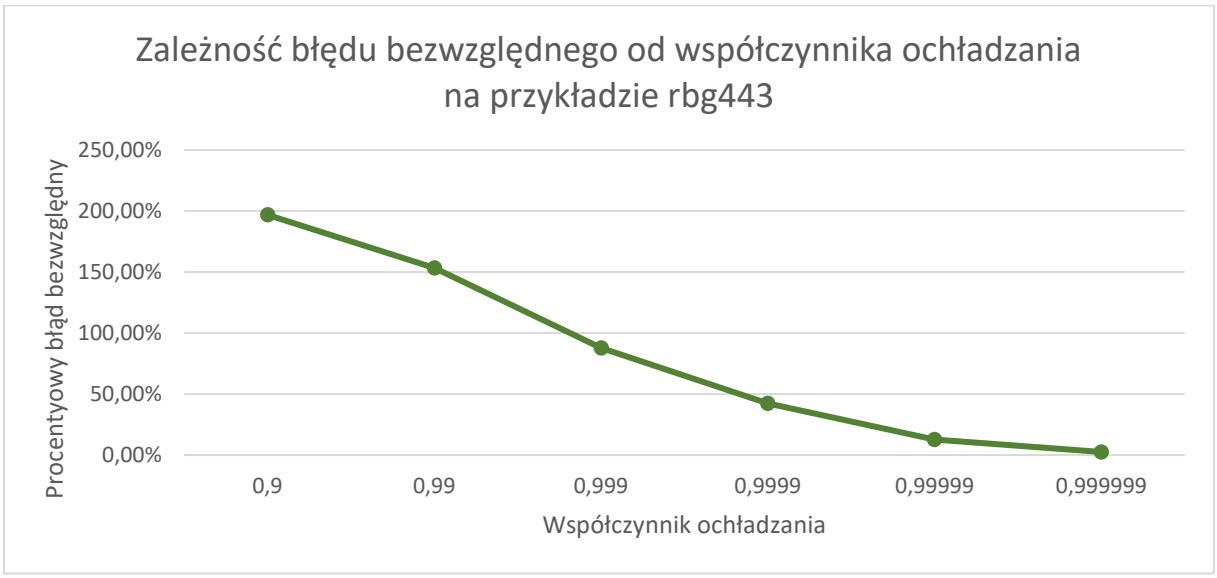
instancja	br17					
rozwiązanie idealne	39					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	100,22	50,34	41,02	39,84	39,86	39,9
średni błąd	156,97%	29,08%	5,18%	2,15%	2,21%	2,31%

instancja	ftv33					
rozwiązanie idealne	1286					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	3341,24	2229,7	1763,56	1543,2	1477,08	1438,14
średni błąd	159,82%	73,38%	37,14%	20,00%	14,86%	11,83%
instancja	ftv70					
rozwiązanie idealne	38673					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	64646,56	53480,46	46648,6	43633,44	41751,14	40894,46
średni błąd	67,16%	38,29%	20,62%	12,83%	7,96%	5,74%

instancja	rbg323					
rozwiązanie idealne	1326					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	5990,32	4918,42	3315,58	2206,24	1628,72	1441,76
średni błąd	351,76%	270,92%	150,04%	66,38%	22,83%	8,73%

instancja	rbg443					
rozwiązanie idealne	2720					
współczynnik ochładzania	0,9	0,99	0,999	0,9999	0,99999	0,999999
rozwiązanie średnie	8073,14	6889,44	5106,28	3875,38	3064,54	2788,86
średni błąd	196,81%	153,29%	87,73%	42,48%	12,67%	2,53%





5. Wnioski

Jak widać, największą zaletą algorytmu symulowanego wyżarzania jest dość krótki czas wykonywania algorytmu, który od rozmiaru problemu zależy liniowo. Niestety, poważną wadą jest brak dostarczenia rozwiązania optymalnego; co gorsza, często dla niekorzystnych instancji różnica między rozwiązaniem dostarczonym a optymalnym jest dość duża. Warto zwrócić uwagę na niestabilność algorytmu: jest on bardzo zależny nie tylko od rozmiaru instancji, ale samego ułożenia wierzchołków. Dla instancji z dużą różnicą między minimami lokalnymi a globalnymi ta różnica często jest duża. Oprócz tego algorytm czasami okazuje się działać inaczej nie moglibyśmy sobie wyobrażać: np. przy instancji *data10* daje gorsze wyniki dla większej wartości współczynnika wyżarzania.