Projektowanie Efektywnych Algorytmów

Projekt

20/12/2021

264067 Mateusz Waszczuk

(4) Simulated Annealing

1.Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności czasowej oraz jakości rozwiązań w zależności od różnych parametrów algorytmu symulowanego wyżarzania rozwiązującego problem komiwojażera w wersji optymalizacyjnej. Wspomniane wyżej parametry to:

- Długość epoki
- Sposób wyboru rozwiązania w sąsiedztwie (2-zamiana lub losowe przemieszczenie)

Zaimplementowany program ma mierzyć czas wykonania algorytmu SA dla każdej instancji oraz obliczać koszt znalezionego cyklu Hamiltona w celu obliczenia różnicy między rozwiązaniem optymalnym. Dane wejściowe dotyczące ilości powtórzeń, długości epoki, oraz sposobu wyboru rozwiązania w sąsiedztwie dla każdej instancji zostaną umieszczone w pliku inicjującym pomiary.ini. Wyniki działania algorytmu mają zostać zwrócone w pliku pomiary.csv. Po zaimplementowaniu algorytmu SA następnym zadaniem będzie przeprowadzanie testów poprawności jego działania oraz określenie liczby pomiarów dla wybranych instancji na podstawie czasu pojedynczego wykonania, co zamierzam zrobić jednym plikiem inicjującym wstępne_testy.ini. Ostatnią i najważniejszą częścią zadania jest zbadanie zależności między długością epoki i sposobem wyboru rozwiązania w sąsiedztwie, a czasem wykonania, dokładnością i zużyciem pamięci z uwzględnieniem ilości wierzchołków dla implementowanego przeze mnie algorytmu SA.

2.Metoda

Metoda symulowanego wyżarzania (SA) jest metodą heurystyczną i całkowicie różni się od wcześniej opracowanej metody siłowej oraz metody B&B, ponieważ co do zasady nie zwraca rozwiązania optymalnego, lecz dzięki temu wykonuje mniej operacji i rozwiązuje duże instancje znacznie szybciej. Oczywiście możliwe jest że znalezione rozwiązanie czasami okaże się być optymalne. Przy odpowiednim dostrojeniu (takim, aby algorytm wykonał przegląd zupełny) można nawet sprawić, że będzie on zawsze zwracał rozwiązania optymalne, jednak zgodnie z tym co wspomniałem wyżej, służy on w praktyce do szybkiego szukania dobrych, lecz nie idealnych rozwiązań, co znacznie skraca czas jego pracy w stosunku do metod B&B oraz BF. Dostrojenie zmuszające go do wykonania przeglądu zupełnego sprawi że czas pracy będzie niemniejszy niż algorytmu BF.

Metoda symulowanego wyżarzania (SA) w swoim działaniu wykorzystuje analogię do procesu wyżarzania metalu jako inspirację do poszukiwania rozwiązań problemów optymalizacyjnych. W procesie wyżarzania metalu jego kryształy są powoli chłodzone, a atomy mają szansę przyjąć bardziej uporządkowane struktury. Podobnie w SA, algorytm przeszukuje przestrzeń rozwiązań w poszukiwaniu coraz lepszych rozwiązań, przy czym akceptuje także gorsze rozwiązania w celu uniknięcia utknięcia w lokalnych minimach.

Kluczowym elementem metody SA jest wprowadzenie parametru temperatury, która odgrywa rolę kontrolującą proces przyjmowania gorszych rozwiązań. Na początku procesu temperatura jest wysoka (w stosunku do końcowej), co pozwala na akceptowanie rozwiązań słabej jakości. W miarę postępu algorytmu temperatura stopniowo maleje, tak jak przy schładzaniu metalu i prawdopodobieństwo akceptacji złych rozwiązań spada. Zmniejszanie temperatury odbywa się zgodnie z wybranym schematem chłodzenia, w moim przypadku geometrycznym o współczynniku alfa równym 0.99. W trakcie jednej epoki, algorytm SA przeszukuje sąsiedztwo aktualnego rozwiązania. Sposób wyboru rozwiązania w sąsiedztwie może się różnić, przykładowymi metodami są 2-zamiana, lub losowe przemieszczenie. Dzięki temu możliwa jest eksploracja różnych obszarów przestrzeni rozwiązań.

Głównym celem SA jest unikanie pułapek lokalnych minimów. To podejście stwarza możliwość przemieszczania się w przestrzeni rozwiązań, nawet jeśli tymczasowo prowadzi to do pogorszenia jakości rozwiązania. Dzięki temu, SA ma zdolność przeszukiwania przestrzeni rozwiązań w sposób bardziej elastyczny niż metody deterministyczne, takie jak algorytm brute force czy branch and bound.

W dalszej części tego opracowania badane będą wpływ długości epoki i sposób wyboru rozwiązania w sąsiedztwie na działanie algorytmu dla różnych instancji. Warto w tym miejscu przedstawić spodziewany wpływ tych parametrów na pomiary.

1. Długość epoki

Krótsze epoki skutkują większą ilością iteracji w jednej chłodzeniowej fazie. Oznacza to częstsze badanie nowych rozwiązań, co może prowadzić do lepszej eksploracji przestrzeni rozwiązań. Dłuższe epoki prowadzą do mniejszej liczby iteracji w jednej fazie chłodzenia. Może to prowadzić do głębszej eksploracji konkretnych obszarów przestrzeni rozwiązań, ale z mniejszą liczbą przeszukiwanych sąsiedztw.

2. Sposób Wyboru Rozwiązania w Sąsiedztwie (2-zamiana vs. Losowe Przemieszczenie)

Sposób 2-zamiany, czyli zamiana dwóch elementów w rozwiązaniu, może prowadzić do lokalnych zmian, co z kolei może poprawić zdolność algorytmu do unikania lokalnych minimów. Wymaga jednak więcej obliczeń. Losowe przemieszczenie może prowadzić do bardziej globalnych zmian w rozwiązaniu, co z kolei może pomóc w szybszym opuszczaniu lokalnych minimów. Może być bardziej efektywne dla problemów o większej złożoności.

W zaimplementowanym programie badaniu nie będą podlegać schemat chłodzenia, sposób wyboru temperatury początkowej oraz sposób w wyboru rozwiązania początkowego. Warto w tym miejscu przedstawić przyjęte w tych obszarach rozwiązania i krótko je uzasadnić.

1. Schemat chłodzenia

W zaimplementowanym programie algorytm używa geometrycznego schematu chłodzenia o współczynniku alfa równym 0.99. Oznacza to, że przy każdej iteracji temperatura aktualna przemnażana jest przez 0.99. W trakcie implementowania algorytmu rozważałem jeszcze użycie schematu liniowego i logarytmicznego. Schemat liniowy prowadzi jednak schładzanie ze stałą szybkością w każdej iteracji, co może prowadzić do szybkiego utknięcia w lokalnym minimum, zaś schemat logarytmiczny zaś gwarantuje znalezienie optimum globalnego, ale średni czas osiągnięcia rozwiązania porównywalny jest z tym dla przeglądu zupełnego. Schemat geometryczny nie posiada wyżej wymienionych wad, stąd też został on ostatecznie przyjęty. Przyjęty zaś współczynnik 0.99 to kompromis między zachowaniem możliwości eksploracyjnych a skierowaniem algorytmu w kierunku bardziej wyrafinowanych rozwiązań w miarę postępu procesu optymalizacji.

2. Sposób wyboru temperatury początkowej

W zaimplementowanym programie dla każdej instancji temperatura początkowa jest iloczynem współczynnika alfa i rozwiązania początkowego (zwróconego przez algorytm najbliższego sąsiada). Takie podejście dąży do zachowania stosunkowo wysokiego poziomu ruchliwości w początkowej fazie algorytmu. Z uwagi na to, że jest obliczana na podstawie początkowego rozwiązania, jest dostosowana dla każdej instancji.

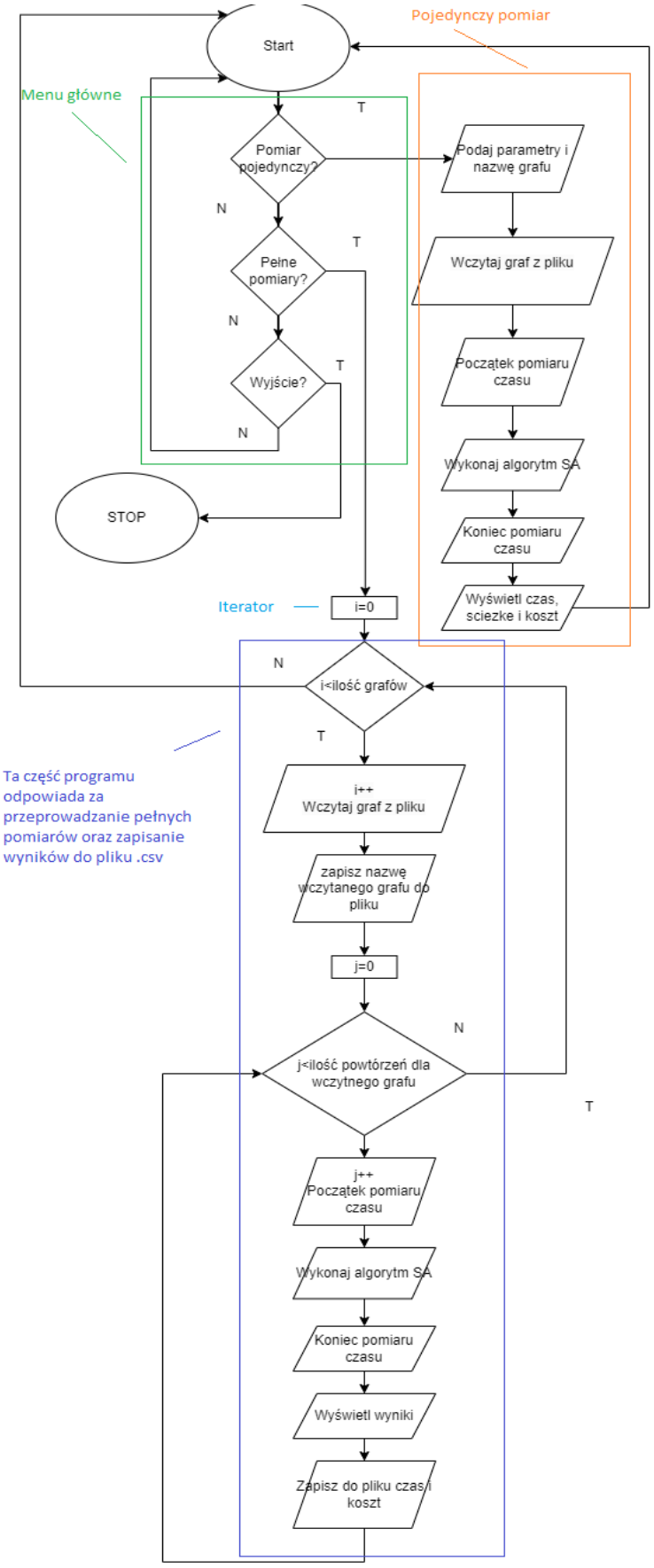
3. Wybór rozwiązania początkowego

W zaimplementowanym programie rozwiązanie początkowe wybierane jest poprzez algorytm najbliższego sąsiada. Jest to stosunkowo mało dokładny algorytm (porównując do SA), jednak jest on bardzo prosty i wyjątkowo szybki, co w mojej opinii jest dobrym kompromisem pomiędzy losowaniem jakiegoś cyklu Hamiltona, a bardziej skomplikowanym algorytmami heurystycznymi mogącymi istotnie wpłynąć na czas wykonania.

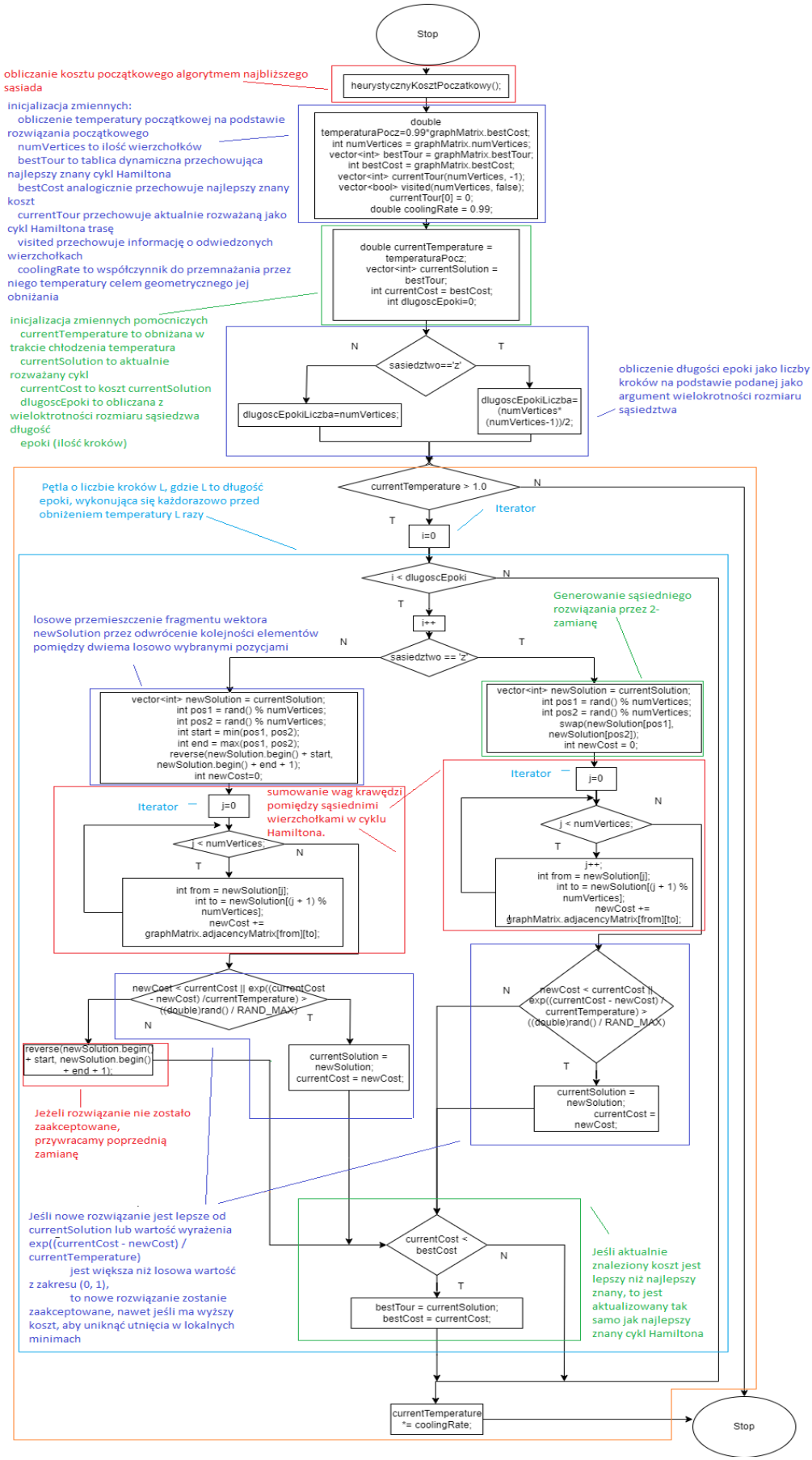
Po wykonaniu badań spodziewana jest kolosalnie niższa złożoność czasowa niż algorytmów B&B i BF dla wszystkich planowanych badań, przy utrzymanym wzroście złożoności czasowym wraz z wzrostem rozmiaru grafu. Rozwiązania co do zasady nie powinny być optymalne, gdyż jak już wspomniano SA jest algorytmem heurystycznym, a parametry w żadnym badaniu nawet nie będą zbliżone do takich, które zmuszą algorytm do przeprowadzenia przeglądu zupełnego. Spodziewany jest wzrost zużycia czasu wraz ze wzrostem długości epoki przy jednoczesnym wzroście dokładności. Spodziewany jest także znacznie dłuższy czas wykonania dla sposobu wyboru rozwiązania w sąsiedztwie przez 2-zamianę niż przez losowe przemieszczenie przy jednoczesnej gorszej jakości wyniku. Teoretycznie losowe przemieszczenie powinno pozwolić na bardziej skomplikowane i różnorodne eksplorowanie przestrzeni rozwiązań, co może prowadzić do znalezienia lepszych lokalnych minimów. Z uwagi na to, że algorytm nie alokuje skomplikowanych i dużych struktur, nie jest spodziewane wysokie zużycie pamięci, jednak z uwagi na to że będą rozwiązywane stosunkowo duże instancje (w porównaniu do BF i B&B) może być zauważalny wzrost pamięci wraz ze wzrostem wielkości instancji. Z uwagi na dokładniejszą eksplorację przestrzeni rozwiązań zużycie pamięci dla SA może być większe niż dla B&B.

3.Algorytm

Na rysunku 1 znajduje się ogólny schemat blokowy całego programu który realizuje pomiary, zaś na rysunku 2 znajduje się schemat blokowy części programu odpowiedzialnej za samą logikę algorytmu SA. Algorytm do znalezienia rozwiązania początkowego jak już wspomniano używa algorytmu najbliższego sąsiada. Funkcja heurystycznyKosztPoczatkowy która znajduje rozwiązanie początkowe wyżej wspomnianym sposobem jest na rysunku 2 jedynie wywoływana w odpowiednim miejscu, natomiast z uwagi że algorytm najbliższego sąsiada nie jest przedmiotem niniejszego opracowania nie posiad dokładnego schematu blokowego. Schemat blokowy pomija również nieistotne z punktu widzenia działania algorytmu fragmenty kodu, jak zabezpieczenia przed źle podanymi danymi.



Rysunek 1: Ogólny schemat blokowy całego programu



Rysunek 2: Schemat blokowy algorytmu SA

4.Dane testowe

Do sprawdzenia poprawności działania algorytmu oraz do określenia zestawu instancji do pomiarów użyto następujący zestaw instancji:

-tsp_10.txt, min. koszt: 212

-gr_21.txt, min koszt 2707

-bayg29.tsp, min. koszt 1610

-dantzig42.tsp, min. koszt 699

-berlin52.tsp, min. koszt 7542

-st70.tsp, min. koszt 675

-kroA100.tsp, min. koszt 21282

-pr136.tsp, min. koszt 96772

-brg180.tsp, min. koszt 1950

-gr229.tsp, min. koszt 134602

-a280.tsp, min. koszt 2579

-linhp318.tsp, min. koszt 41345

-pcb442.tsp, min. koszt 50778

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

https://github.com/PrzemekRychter/PEA/tree/main/sa/program/instancje

Po przetestowaniu programu za pomocą wszystkich podanych instancji koszty były zbliżone do podanych, choć poza tsp_10.txt wyższe niż najlepsze znane. Jest to całkowicie zgodne z oczekiwaniami wobec tego algorytmu. Pomiary były prowadzone dla długości epoki 5L, gdzie L jest rozmiarem sąsiedztwa oraz obu sposobów wyboru rozwiązania dla każdej instancji. 5L jest większa od najwyższej badanej długości epoki, zatem umożliwiło to wybranie zestawu instancji które na pewno wykonają się dla wszystkich właściwych już pomiarów. Okazało się, że udało się wykonać wszystkie testowane instancje w dopuszczalnym czasie (60 min), więc nie odrzucono instancji powyżej jakiejkolwiek ilości wierzchołków. Instancje do wyżej wspomnianych wstępnych pomiarów zawarto w pliku inicjującym wstępne_testy.ini Do przeprowadzenia właściwych pomiarów wybrano następujący zestaw instancji:

-tsp_10.txt, min. koszt: 212

-tsp_12.txt, min. koszt: 264

-tsp_13.txt, min. koszt: 269

-tsp_14.txt, min. koszt: 282

-tsp_15.txt, min. koszt: 291

-tsp_17.txt, min. koszt: 39

- gr17.tsp, min koszt 2085

- gr21.tsp, min koszt 2707

- bayg29.tsp, min koszt 1610

- dantzig42.tsp, min koszt 699

- berlin52.tsp, min koszt 7542

- brazil58.tsp, min koszt 25395

- st70.tsp, min koszt 675

- eil76.tsp, min koszt 538

- gr96.tsp, min koszt 55209

- pr144.tsp, min koszt 58537

- gr202.tsp, min koszt 40160

- pr299.tsp, min koszt 48191

- rd400.tsp, min koszt 15281

- pcb442.tsp, min koszt 50778

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

https://github.com/PrzemekRychter/PEA/tree/main/sa/program/instancje

5.Procedura badawcza

Należało zbadać wpływ długości epoki i sposobu wyboru rozwiązania w sąsiedztwie na czas wykonania, błąd oraz zużycie pamięci. W przypadku algorytmu realizującego algorytm SA dla przestrzeni rozwiązań dopuszczalnych nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu sterowanego plikiem inicjującym .ini: (format pliku : długość_epoki, sposób_wyboru_rozwiązania_w_sąsiedztwie, nazwa_pliku_z_grafem, ilość_powtórzeń). Wyniki były zapisywane w pliku pomiary.csv. Poniżej przedstawiono fragment zawartości jednego z plików inicjujących (wstępne_testy.ini):

5 l tsp_10.txt 1

5 z tsp_10.txt 1

5 l gr_21.txt 1

5 z gr_21.txt 1

5 l bayg29.tsp 1

5 z bayg29.tsp 1

Każda instancji rozwiązywana była zgodnie z liczbą jej wykonań, w tym przypadku każda raz. Do pliku wyjściowego pomiary.csv zapisywany był czas wykonania pomiarów, błąd (procentowo) jak bardzo obliczony koszt odbiega od najlepszego znanego oraz nazwa instancji której te pomiary dotyczą. Plik wyjściowy zapisywany był w formacie csv. Dane znajdują się w następującym formacie: Nazwa instancji; Czas[ms]; Błąd[%];. Dla każdej instancji dodatkowo przed pierwszym pomiarem zapisywane są nazwy kolumn zgodnie z wyżej podanym formatem. Poniżej przedstawiono fragment zawartości pliku wyjściowego:

gr17.tsp;Czas [ms];Blad [%]

;0;0,239808

;0;0,239808

;0;0,000000

;0;0,000000

Pomiary zużycia pamięci zostały przeprowadzone przy użyciu menedżera zadań systemu Windows 10. Z tego powodu każdy pomiar był przeprowadzany poprzez wczytanie pliku inicjującego nakazującego programowi wykonanie algorytmu dla danej instancji 100000 razy dla zadanych parametrów. Zadeklarowanie dużej liczby powtórzeń miało na celu zapewnienie czasu na wykonanie ręcznego odczytu używanej przez proces pamięci RAM. Poniżej przykład zawartości pliku inicjującego dla pr152.tsp do pomiarów zużycia pamięci:

0.5 z pr152.tsp 100000

Wyniki zostały opracowane w MS Excel.

6.Wyniki

Wyniki zgromadzone zostały w plikach: pomiary.xlsx, pamiec.xlsx Wszystkie ww. pliku zostały dołączone do raportu i znajdują się na dysku Google pod adresem:

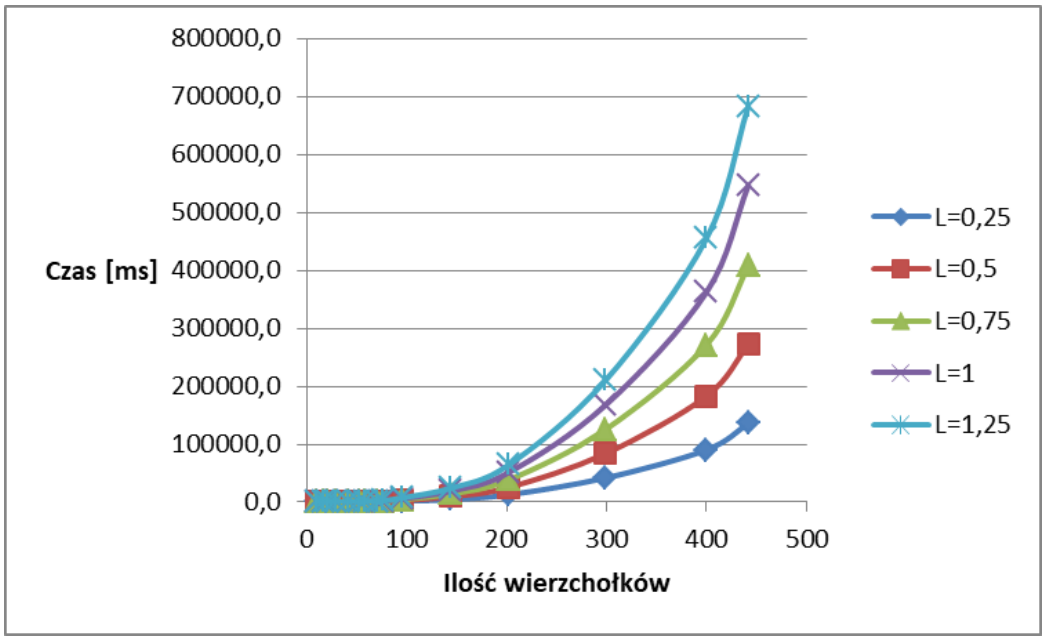
https://drive.google.com/drive/folders/113X275AvvonhyTK1MFhEiYMs25LdP0iI?usp=sharing

6.1 Wpływ długości epoki na jakość rozwiązania, czas oraz zużycie pamięci

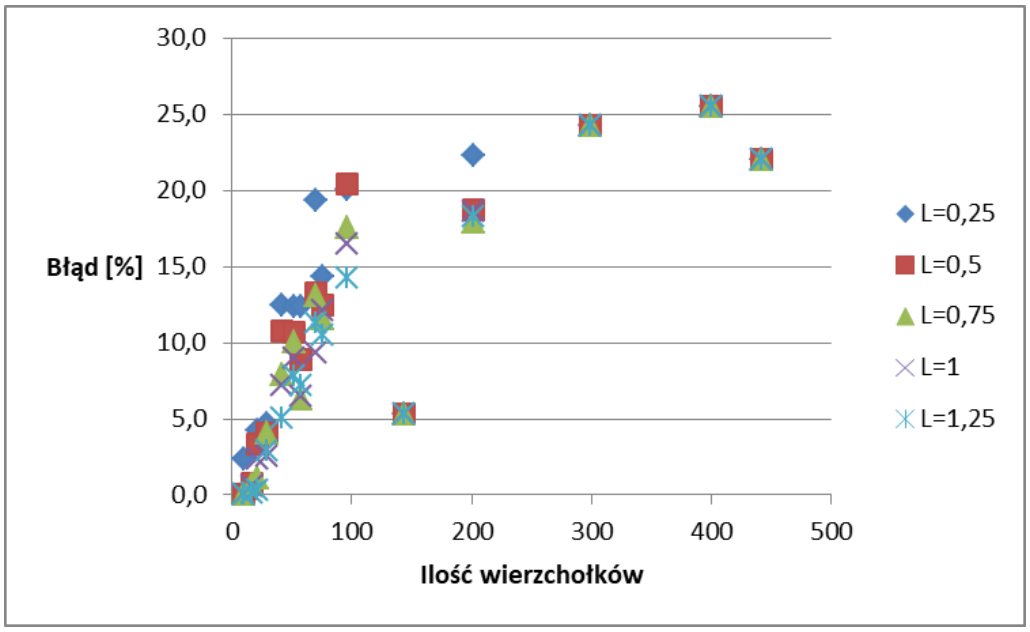
W pierwszej kolejności przeprowadzone zostały badania wpływu długości epoki na czas rozwiązania (tabela 1, rysunek 3), jego jakość (tabela 1, rysunek 4,) oraz zużycie pamięci (tabela 2) dla wyboru rozwiązania w sąsiedztwie przez 2- zamianę. Z uwagi na brak wpływu długości epoki na zużycie pamięci nie sporządzono wykresu dla tych danych. Dla każdej długości epoki oraz każdego sposobu wyboru rozwiązania w sąsiedztwie było to 0,7MB.

Tabela 1: Wpływ długości epoki na czas wykonania i jakość rozwiązania dla 2-zamiany

2-zamiana										
	L=0,25		L=0,5		L=0,75		L=1		L=1,25	
l. wierzchołków	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]
10	4,5	2,4	6,0	0,0	3,0	0,0	9,0	0,0	12,7	0,0
17	15,6	0,6	28,3	0,8	35,5	0,6	46,5	0,6	51,5	0,2
21	19,8	4,3	44,6	3,3	60,7	1,1	82,5	2,4	102,8	0,3
29	43,5	4,7	95,6	4,1	127,9	4,0	173,7	2,5	216,7	2,9
42	106,2	12,4	205,0	10,8	306,7	7,9	404,5	7,2	512,1	5,1
52	237,4	12,4	471,7	10,7	710,2	10,1	948,1	9,0	1176,4	7,8
58	360,6	12,4	718,7	8,9	1076,4	6,3	1438,2	6,5	1797,7	7,1
70	404,0	19,3	830,6	13,3	1181,7	13,1	1568,4	9,3	1987,9	11,3
76	473,5	14,3	944,9	12,4	1421,7	11,6	1899,2	12,1	2394,2	10,5
96	1559,3	20,1	3078,7	20,4	4654,3	17,5	6178,9	16,5	7757,3	14,3
144	4800,3	5,3	9582,4	5,3	14386,5	5,3	19255,8	5,3	24323,2	5,3
202	12932,5	22,3	25563,0	18,7	38547,3	17,9	51191,2	18,6	64338,6	18,3
299	41804,6	24,3	83874,2	24,3	125572,0	24,3	167459,8	24,3	210305,0	24,3
400	90303,6	25,5	183038,6	25,5	271202,8	25,5	362205,2	25,5	455035,4	25,5
442	135771,6	22,1	272233,2	22,1	407634,4	22,1	546990,8	22,1	683557,2	22,1
średnia	-	13,5	-	12,0	-	11,2	-	10,8	-	10,3



Rysunek 3: Wpływ długości epoki na czas wykonania dla 2-zamiany



Rysunek 4: Wpływ długości epoki na jakość rozwiązania dla 2-zamiany

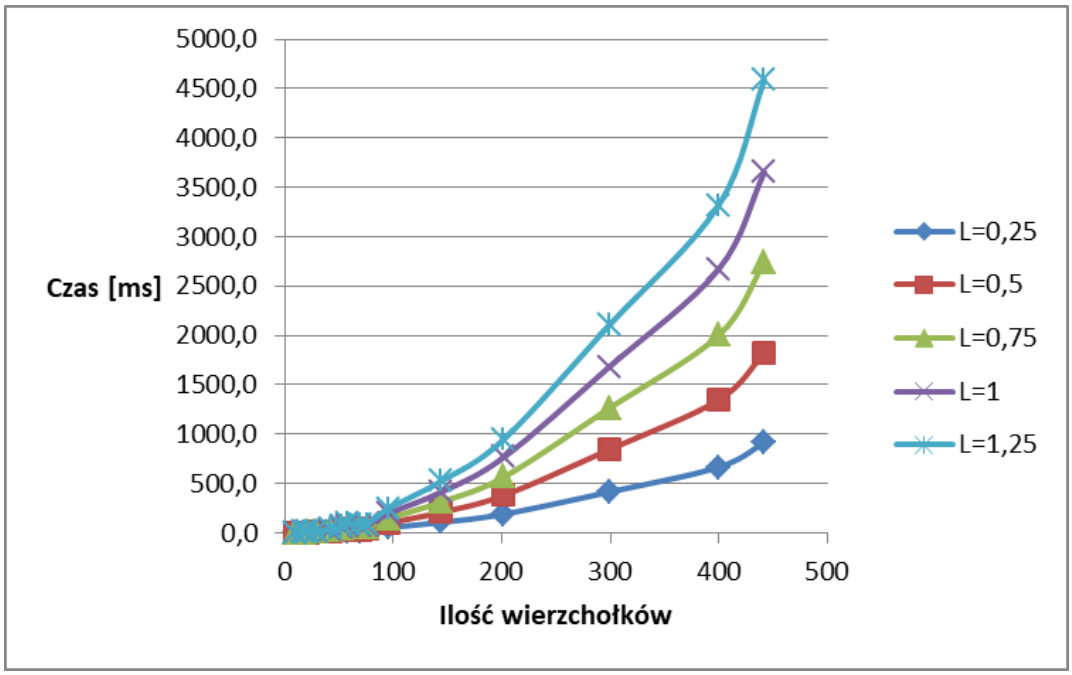
Tabela 2: Wpływ długości epoki na zużycie pamięci dla 2-zamiany

Pomiar pamięci: 2-zamiana (pomiary przedstawiono w MB)					
nr pomiaru/dl. Epoki, gdzie L to rozmiar sąsiedztwa	0,25L	0,5L	0,75L	1L	1,25L
1	0,7	0,7	0,7	0,7	0,7
2	0,7	0,7	0,7	0,7	0,7
3	0,7	0,7	0,7	0,7	0,7
4	0,7	0,7	0,7	0,7	0,7
5	0,7	0,7	0,7	0,7	0,7
srednia [MB]	0,7	0,7	0,7	0,7	0,7

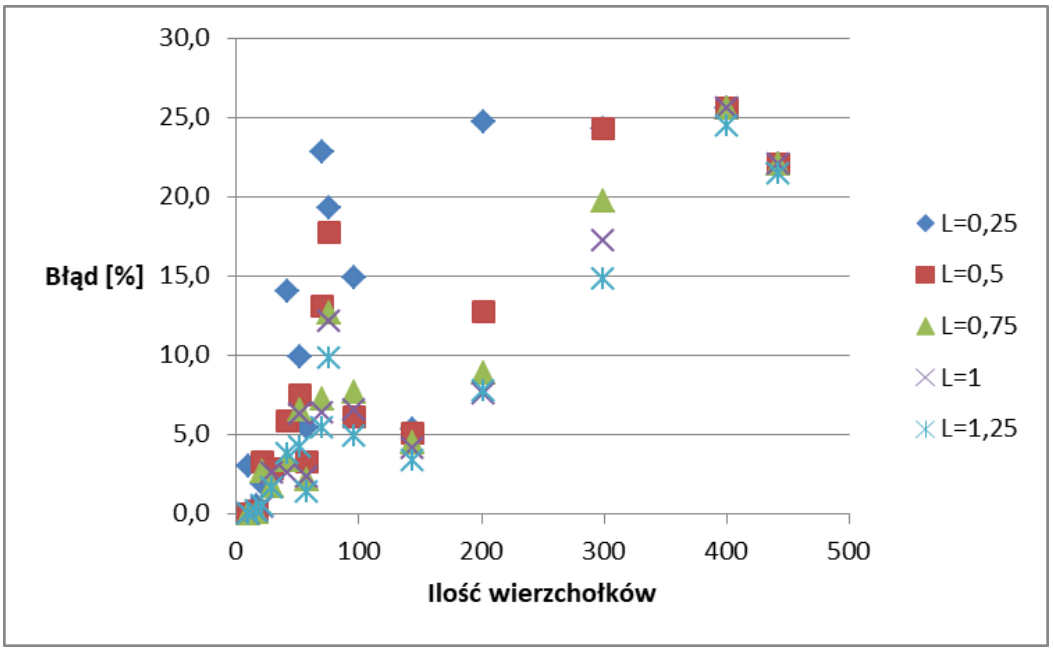
Następnie przeprowadzone zostały badania wpływu długości epoki na czas rozwiązania (tabela 3, rysunek 5), jego jakość (tabela 3, rysunek 6, rysunek 7) oraz zużycie pamięci (tabela 4) dla wyboru rozwiązania w sąsiedztwie przez losowe przemieszczenie. Po sporządzeniu wykresów z rysunków 4 i 6 można było zaobserwować, że długość epoki ma wpływ na jakość wyniku, zaś ilość wierzchołków go nie ma lub jest on marginalny. Z tego powodu obliczono średni błąd dla każdej długości epoki w tabelach 1 i 3 i sporządzono rysunek 7, który pozwala wyraźnie zaobserwować zależność między długością epoki oraz jakością rozwiązania. Z uwagi na brak wpływu długości epoki na zużycie pamięci także dla losowego przemieszczenia nie sporządzono wykresu dla tych danych, podobnie jak przy 2-zamianie zawsze było ono równe 0,7MB.

Tabela 3: Wpływ długości epoki na czas wykonania i jakość rozwiązania dla losowego przemieszczenia

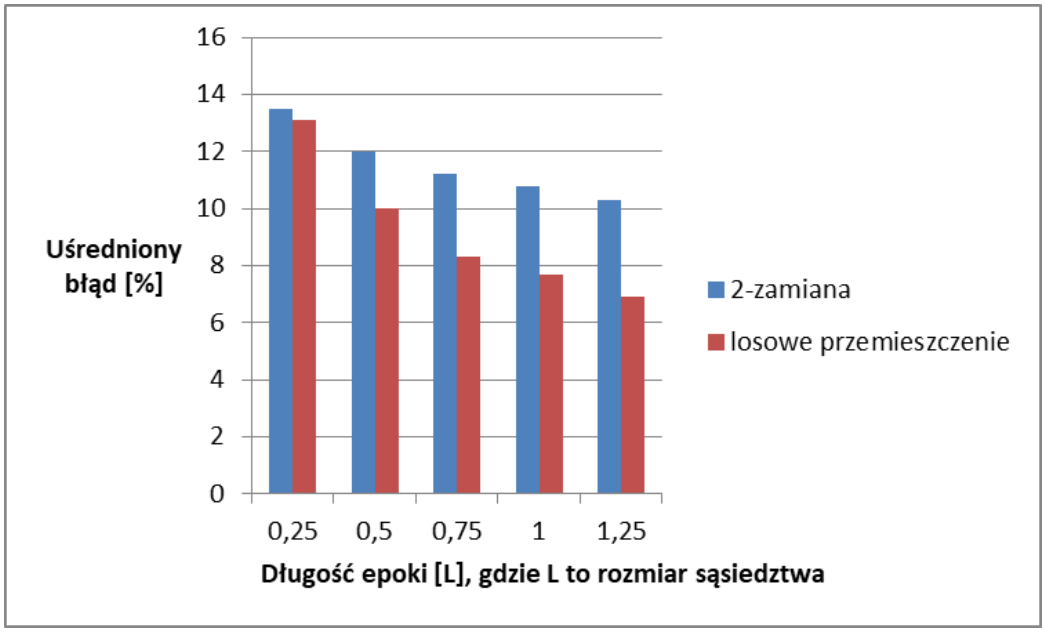
Losowe przemieszczenie										
	L=0,25		L=0,5		L=0,75		L=1		L=1,25	
l. wierzchołków	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]	czas [ms]	blad [%]
10	3,0	3,0	1,5	0,0	3,0	0,0	6,0	0,0	4,5	0,0
17	3,0	0,5	4,5	0,2	6,0	0,1	15,0	0,1	12,0	0,1
21	3,3	1,9	3,0	3,3	15,3	2,7	12,0	0,4	15,0	0,5
29	0,0	2,6	10,5	2,9	12,0	1,7	18,2	2,6	19,8	1,6
42	12,0	14,1	15,1	5,9	21,4	3,4	31,0	2,6	40,0	3,8
52	15,1	9,9	28,2	7,5	40,0	6,5	54,5	6,3	70,9	4,2
58	16,6	5,5	41,9	3,3	57,7	2,1	78,3	2,3	97,4	1,4
70	16,6	22,8	32,9	13,1	55,6	7,3	65,9	6,4	86,4	5,4
76	21,4	19,3	38,5	17,7	55,3	12,6	78,0	12,2	92,4	9,8
96	54,0	14,9	101,4	6,1	148,4	7,6	199,3	6,5	250,7	4,9
144	107,7	5,3	208,9	5,1	313,0	4,4	414,0	4,2	523,9	3,4
202	191,4	24,7	381,7	12,7	567,6	8,9	767,8	7,6	950,0	7,8
299	414,8	24,3	844,8	24,3	1262,2	19,7	1681,8	17,2	2103,8	14,8
400	666,6	25,5	1347,8	25,5	2002,4	25,5	2673,2	25,5	3321,2	24,5
442	912,4	22,1	1821,0	22,1	2737,2	22,1	3653,4	22,1	4589,6	21,5
średnia	-	13,1	-	10,0	-	8,3	-	7,7	-	6,9



Rysunek 5: Wpływ długości epoki na czas wykonania dla losowego przemieszczenia



Rysunek 6: Wpływ długości epoki na jakość rozwiązania dla losowego przemieszczenia



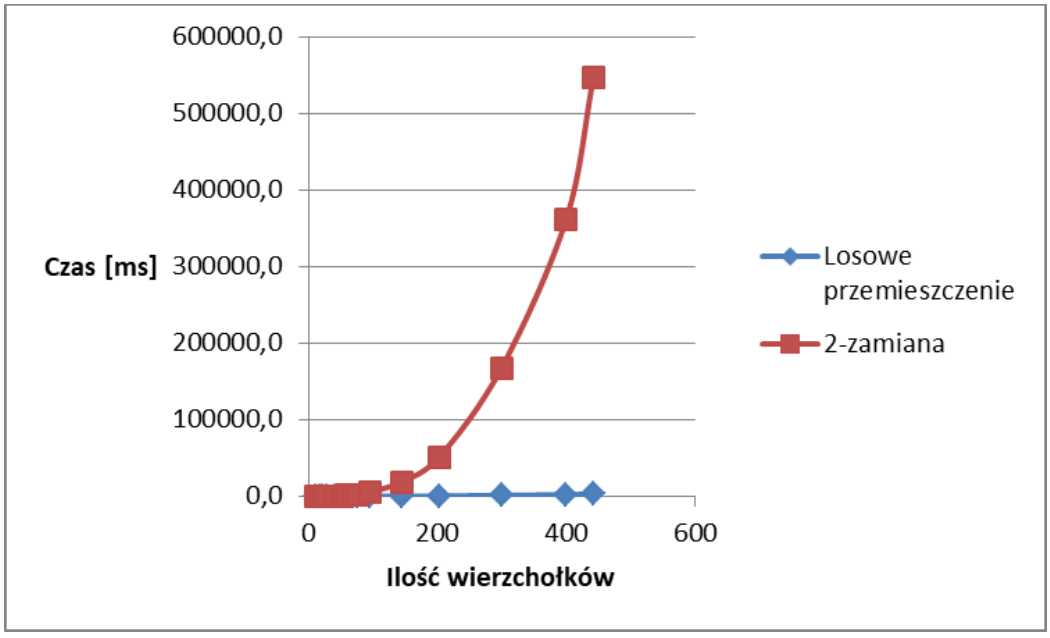
Rysunek 7: Wpływ długości epoki na jakość rozwiązania dla obu sposobów wyboru rozwiązania w sąsiedztwie dla uśrednionego błędu dla wszystkich instancji dla danej długości epoki

Tabela 4: Wpływ długości epoki na zużycie pamięci dla losowego przemieszczenia

Pomiar pamięci: losowe przemieszczenie (pomiary przedstawiono w MB)					
nr pomiaru/dl. Epoki, gdzie L to rozmiar sąsiedztwa	0,25L	0,5L	0,75L	1L	1,25L
1	0,7	0,7	0,7	0,7	0,7
2	0,7	0,7	0,7	0,7	0,7
3	0,7	0,7	0,7	0,7	0,7
4	0,7	0,7	0,7	0,7	0,7
5	0,7	0,7	0,7	0,7	0,7
srednia [MB]	0,7	0,7	0,7	0,7	0,7

6.2 Wpływ sposobu wyboru rozwiązania w sąsiedztwie na jakość rozwiązania, czas oraz zużycie pamięci

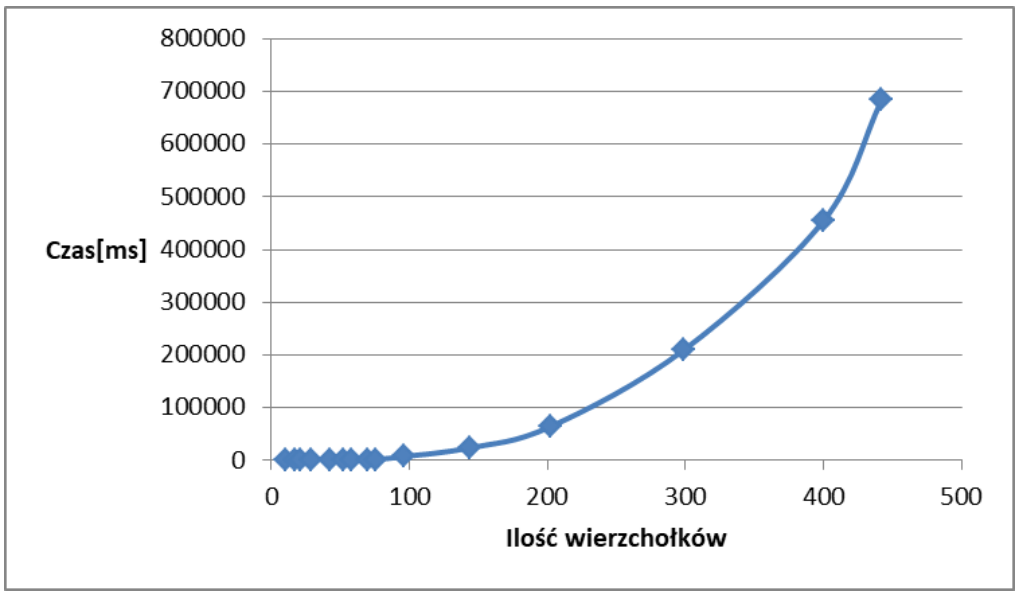
Następnie przeprowadzone zostały badania wpływu sposobu wyboru rozwiązania w sąsiedztwie na czas rozwiązania (rysunek 8) dla długości epoki. Wpływ sposobu wyboru rozwiązania w sąsiedztwie na jakość rozwiązania można zaobserwować na rysunku 7, stąd też nie było potrzeby sporządzania osobnego wykresu. Co więcej, na rysunku 7 wpływ ten można zaobserwować dla wszystkich długości epok. Z uwagi na to, że wszystkie potrzebne dane zostały już zgromadzone w podpunkcie 6.1, nie ponawiano pomiarów uznając to za bezcelowe. Z uwagi na brak wpływu sposobu wyboru rozwiązania w sąsiedztwie na zużycie pamięci nie sporządzono wykresu dla tych danych. Dla każdej długości epoki oraz każdego sposobu wyboru rozwiązania w sąsiedztwie było to 0,7MB (tabela 2, tabela 4).



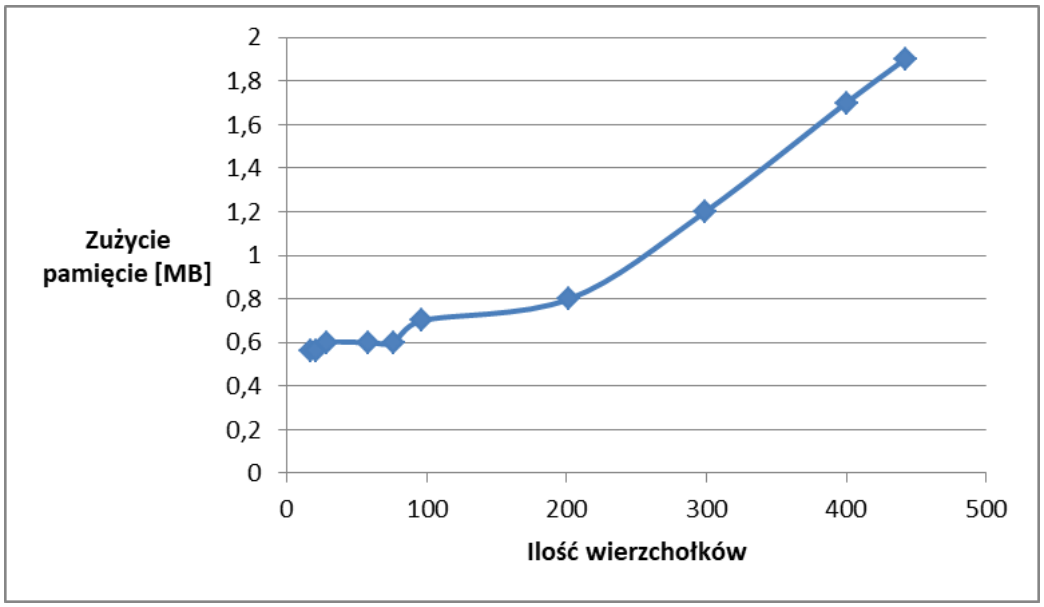
Rysunek 8: Wpływ sposobu wyboru rozwiązania w sąsiedztwie na czas wykonania algorytmu SA

6.3 Wpływ ilości wierzchołków w grafie na czas wykonania, jakość wyniku oraz zużycie pamięci.

Przeprowadzone zostały również badania wpływu ilości wierzchołków na czas wykonania (rysunek 9), oraz zużycie pamięci (rysunek 10). Oba wykresy sporządzone zostały dla danych otrzymanych dla długości epoki 1L oraz 2-zamiany, jako że był to najbardziej czasochłonna testowana konfiguracja i została później użyta do porównań z badanymi wcześniej algorytmami. Na rysunkach 4 i 6 można zaobserwować, że ilość wierzchołków ma marginalny wpływ na jakość rozwiązania i wydaje się to zależeć przede wszystkim od wartości w grafie, a nie stricte ilości wierzchołków. Stąd też rysunki 4 i 6 uznano za wystarczające w tej kwestii i nie sporządzono dodatkowych.



Rysunek 9: Wpływ ilości wierzchołków na czas wykonania algorytmu SA



Rysunek 10: Wpływ ilości wierzchołków na zużycie pamięci algorytmu SA

6.4 Platforma sprzętowa

Pomiary zostały przeprowadzone na sprzęcie o następujących parametrach:

- Procesor Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz
- RAM 8GB
- System operacyjny Windows 10 64-bit, procesor x64

Czas został zmierzony za pomocą biblioteki std::chrono, natomiast zużycie pamięci za pomocą menedżera zadań systemu Windows 10.

7.Analiza wyników i wnioski dot. algorytmu SA

Wszystkie otrzymane wyniki nie wykazują cech mogących wskazywać na błędy w algorytmie lub instancjach. Nie stwierdzono podstaw aby wykluczyć jakiekolwiek z pomiarów i wszystkie wykonane przedstawiono na odpowiednich wykresach.

Złożoność czasowa algorytmu SA dla badanych instancji jest zgodna z oczekiwaną i nawet dla najbardziej niekorzystnych z badanych parametrów (długość epoki 1,25L, gdzie L to rozmiar sąsiedztwa typu swap) umożliwia rozwiązywanie instancji o setkach wierzchołków w rozsądnym czasie. Dla największej badanej instancji (442 wierzchołki) było to niespełna 12 minut. Jednocześnie błąd nie przekraczał 25% (rysunek 4), co przerosło nawet początkowe oczekiwania. Można z tego wyciągnąć wnioski, że program udało się napisać bez niepotrzebnych operacji, takich jak bezcelowe alokowanie zmiennych zbyt dużo razy w pętlach, co w połączeniu z rodzajem implementowanego algorytmu pozwoliło na osiągnięcie tak dobrych rezultatów. Można zauważyć, że długość epoki ma istotny wpływ na czas wykonania, im dłuższa epoka tym wyższy ten czas jest (rysunek 3, rysunek 5). Zależność ta ma miejscu zarówno przy użyciu 2-zamiany jak i losowego przemieszczenia. Ma to związek z tym, że długość epoki ma bezpośredni wpływ na ilość iteracji które algorytm wykonuje.

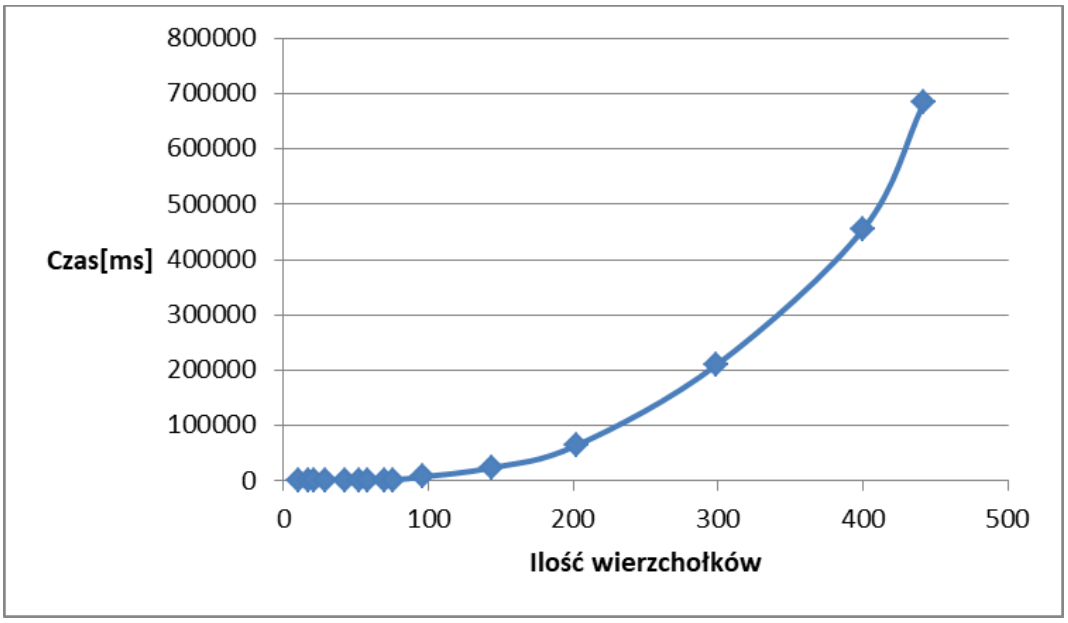
Jakość otrzymanych rozwiązań, tak jak oczekiwano wykazuje korelację zarówno ze sposobem wyboru rozwiązania w sąsiedztwie, jak i długością epoki, natomiast nie wydaje się mieć związku z ilością wierzchołku w grafie. Warto w tym momencie podkreślić, że mój algorytm długość epoki ma nierozerwalnie związaną z rozmiarem sąsiedztwa (jest podawana jako jego krotność). W przypadku podania długości epoki jako liczbę iteracji dla jednej temperatury, dla poprawnie zaimplementowanego algorytmu oczywiście ilość wierzchołków miałaby wpływ na wielkość błędu. Jednak z uwagi na fakt, że takie podejście zafałszowałoby wynik pomiarów i utrudniło jego opracowanie powodując konieczność dodatkowych skomplikowanych obliczeń, długość epoki jest wielokrotnością rozmiaru sąsiedztwa. Ma ona istotny wpływ na wielkość błędu obliczonego cyklu Hamiltona w taki sam sposób dla 2-zamiany i losowego przemieszczenia i w obydwu przypadkach wraz ze wzrostem długości epoki błąd maleje (rysunek 7). Jakość rozwiązania ma też związek ze sposobem wyboru rozwiązania w sąsiedztwie i zgodnie z oczekiwaniami jest większa dla losowego przemieszczenia (rysunek 7). Co więcej, losowe przemieszczenie tak jak oczekiwano jest też znacznie mniej czasochłonne niż 2-zamiana (rysunek 8). Używając losowego przemieszczenia można znacznie zwiększyć długość epoki, przypuszczalnie daleko powyżej badanego przedziału zwiększając jakość i z powodzeniem rozwiązywać w ten sposób instancje nieosiągalne dla przeglądu zupełnego w bardzo szybkim czasie przy jednoczesnej niewielkiej ilości dokładności.

Warto też na koniec wspomnieć o przeprowadzonych pomiarach zużycia pamięci. Tak jak przewidywano, okazało się że jedynie ilość wierzchołków w grafie ma zauważalny wpływ na wyniki tych pomiarów. Co więcej, zużycie to nadal jest pomijalne w porównaniu z choćby zwykłą przeglądarką. Wszystkie inne badane parametry nie wykazują dla badanych instancji korelacji ze zużyciem pamięci. Algorytm SA nie wykorzystuje żadnych złożonych struktur które mogłyby znacząco wpływać na zużycie pamięci i to można uznać za przyczynę takiego stanu rzeczy. W szczególności nie wykorzystuje się takich struktur ani w sposobie wyboru rozwiązania w sąsiedztwie ani przy ustalaniu długości epoki, największą (choć i tak niewielką w porównaniu z dostępnymi zasobami) strukturą jest sam graf.

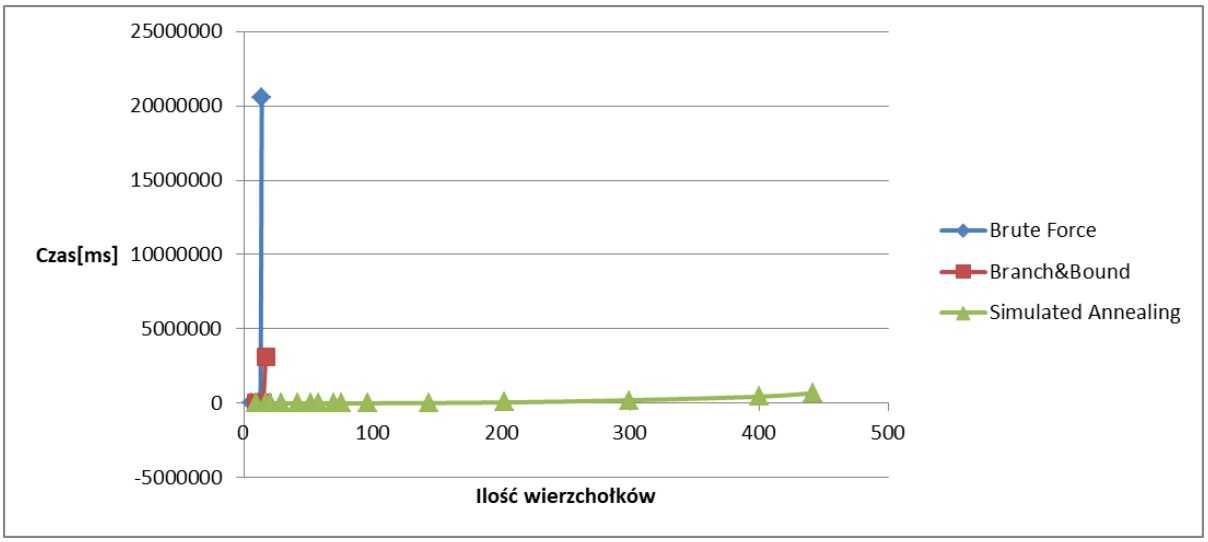
8.Porównanie algorytmów BF, B&B i SA

8.1 Pomiary

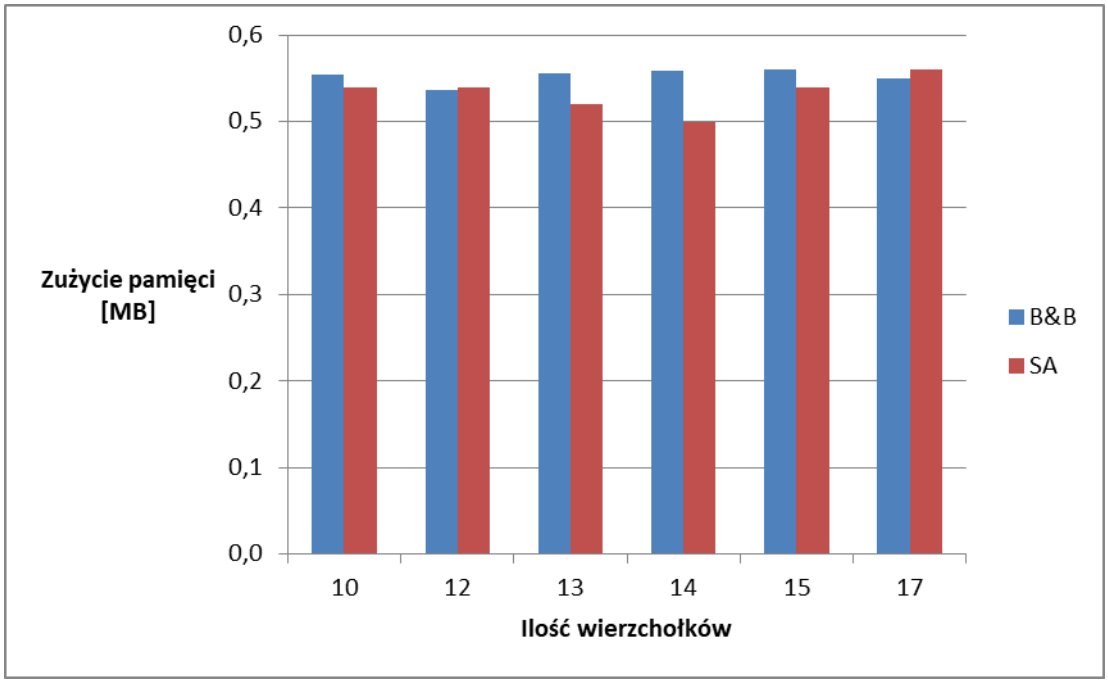
W celu porównania algorytmu SA z dotychczas badanymi algorytmami BF i B&B przeprowadzono pomiary porównawcze zużycia czasu w zależności od ilości wierzchołków (rysunek 11) oraz zużycia pamięci (rysunek 12). W celu poprawienia czytelności wyżej wymienionych rysunków przedstawiono ponownie także rysunek 9.



Rysunek 9: Wpływ ilości wierzchołków na czas wykonania algorytmu SA



Rysunek 11: Porównanie czasu wykonania w zależności od ilości wierzchołków dla algorytmów SA, brute force i branch and bound



Rysunek 12: Porównanie zużycia pamięci dla algorytmów S i B&B dla takich samych instancji

8.2 Wnioski dotyczące porównania algorytmów B&B, BF i SA

Zgodnie z przypuszczeniami, algorytm SA okazał się kolosalnie bardziej wydajny czasowo niż algorytmy B&B i BF. Wynika to z faktu, że jego zasada działania jest zupełnie odmienna (przestawiono ją w sekcji 2) i nie przeprowadza on przeglądu zupełnego (nawet zredukowanego o nierokujące jako optymalne rozwiązania dla algorytmu B&B). Jego wydajność pamięciowa jest zgodnie z przewidywaniami niewielka, zaledwie około 0,5 MB, podobnie jak dla algorytmu B&B, gdyż oba algorytmy nie wykorzystują dużych struktur mogących istotnie wpłynąć na zużycie pamięci.