Kajetan Parzyszek

Projektowanie Efektywnych Algorytmów

Zadanie projektowe nr 2: Symetryczny i Asymetryczny Problem Komiwojażera. Algorytmy metaheurystyczne poszukiwania lokalnego. Algorytm Tabu Search oraz Simulated Annealing.

K P
12-10-2018

Spis treści

1. Wstęp teoretyczny

Celem przeprowadzonego eksperymentu było zbadanie efektywności algorytmów *metaheurystycznych poszukiwania lokalnego* dla *symetrycznego* i *asymetrycznego problemu komiwojażera*(dalej nazywane odpowiednio *STSP* i *ATSP)*. Aby tego dokonać najpierw trzeba było zaimplementować odpowiednią strukturę przechowującą listę miast, wybrana została *macierz sąsiedztwa*, oraz algorytmy, a mianowicie, *algorytm Tabu Search* oraz *algorytm Simulated Annealing*. Język programowania jaki został użyty przy programowaniu struktur, algorytmów jak i przy przeprowadzaniu testów to C++.

Badaną wielkością było asymptotyczne tempo wzrostu konkretnych algorytmów, zapisywane za pomocą notacji dużego O, w celu późniejszego opisu złożoności obliczeniowej tychże algorytmów. Drugą z badanych wartości była dokładność algorytmów wyrażana za pomocą procentowego stosunku otrzymanych wyników do optymalnych

1.1. Symetryczny i Asymetryczny Problem Komiwojażera

Problem komiwojażera jest to zagadnienie optymalizacyjne skupiające się na znalezieniu minimalnego cyklu Hamiltona(czyli cyklu w którym każdy wierzchołek, oprócz pierwszego, w grafie odwiedzany jest dokładnie raz) w pełnym grafie ważonym(czyli grafie który posiada wszystkie możliwe krawędzie pomiędzy wierzchołkami, a krawędzie mają przyporządkowany koszt przejścia krawędzi).

Nazwa problemu podchodzi od jego typowej ilustracji, jako sprzedawcy(komiwojażer), który ma odwiedzić konkretną liczbę miast i wrócić do punktu wyjścia. Znane są odległości między miastami, a celem jest znalezienie drogi o najmniejszym koszcie(czy to czasowym, kosztowym, odległościowym).

Problem Komiwojażera dzieli się na symetryczny i asymetryczny, w problemie symetrycznym koszt ścieżki między dwoma miastami A i B jest taki sam dla podróży z A -> B jak i B -> A, natomiast dla problemu asymetrycznego koszt ścieżki z A -> B może być różny od kosztu ścieżki z B ->A.

Największą trudnością problemu komiwojażera jest ilość danych wymagających analizy. Dla *n* miast liczba możliwych kombinacji wynosi $\frac{(n-1)!}{2}$.

1.2. Algorytm Tabu Search

Algorytm *Tabu Search*, inaczej *poszukiwanie z zakazami,* należy do rodziny algorytmów typu *local search,* inaczej *wyszukiwanie lokalne.* Metoda ta polega na analizie sąsiednich rozwiązań dla obecnie posiadanego i wyborze w każdej iteracji rozwiązania najlepszego z możliwych wygenerowanych.

Algorytm ten jest prosty w implementacji, jednakże nie gwarantuje znalezienia optymalnego rozwiązania, a w swojej najprostszej postaci generuje spory % błędu. Czas wykonania algorytmu zależny jest przede wszystkim od ustalonej liczby iteracji jakie ma wykonać oraz sposobu przeszukiwania sąsiadów obecnego rozwiązania. Zgodnie z tym w przypadku problemu komiwojażera liczba przeszukiwanych sąsiadów danego rozwiązania wynosić będzie $\frac{(n-1)*(n-2)}{2}$, gdzie *n* to liczba miast. Sąsiedzi dla danego rozwiązania generowani są za pomocą metody typu *swap*, która podmienia kolejno ze sobą wybrane miasta w drodze.

Schemat działania algorytmu przedstawia się następująco:

1. Wybieramy drogę początkową na sposób dowolny
2. Wyliczamy jej koszt, tworzymy wyzerowaną listę tabu
3. Przeszukujemy wszystkich sąsiadów, wybieramy najlepszego(chyba że wcześniej natkniemy się na poprawę obecnej drogi względem sąsiada większą od naszego kryterium aspiracji – wtedy przerywamy poszukiwania po założonej liczbie iteracji)
4. Modyfikujemy listę tabu, tak aby dana zamiana nie mogła wystąpić ponownie przez żądaną liczbę iteracji, kadencję poprzednich zmian zmniejszamy o 1
5. Obecnie znalezioną drogę ustawiamy jako początkową oraz jako optymalną jeżeli jest lepsza od optymalnej.
6. Powtarzamy kroki 3-5 dopóki nie wykonamy zadanej liczby iteracji

Zgodnie z powyższym, z łatwością możemy określić teoretyczną złożoność obliczeniową algorytmu Simulated Annealing dla problemu komiwojażera. Główny wpływ na złożoność algorytmu ma liczba możliwych sąsiadów danego rozwiązania, która równa jest $\frac{(n-1)*(n-2)}{2}$, a więc teoretyczne asymptotyczne tempo wzrostu zapisane za pomocą notacji dużego O będzie wynosić $O(n^2)$.

Czynnikami jakie mogą wpływać na jakość rozwiązania jest *ilość iteracji algorytmu*, *długość kadencji zamiany na liście tabu*, *wartość aspiracji(*dla której ignorujemy zapis na liście tabu) oraz *ilość iteracji po skorzystaniu z kryterium aspiracji.*

1.3. Algorytm Simulated Annealing

Algorytm *Simulated Annealing*, inaczej *symulowane wyżarzanie,* jest *techniką probabilistyczną*. Metoda ta opiera się na wyżarzaniu w metalurgii, gdzie podgrzewa się i schładza materiał aby zwiększyć rozmiar budujących kryształów oraz zmniejszyć jego defekty. W kolejnych iteracjach algorytmu generowane jest losowo nowe rozwiązania, które jeżeli jest lepsze od obecnego ustawiane jest na obecne, a jeżeli nie jest to nadal możliwy jest wybór tego rozwiązania na podstawie prawdopodobieństwa wyliczanego ze wzoru $e^{\frac{długość\ drogi\ poprzedniej - długość\ drogi\ obecnej}{temperatura}}$. Temperatura ustawiana jest na początku pracy algorytmu, a następnie z każdym obiegiem jest zmniejszana.

Algorytm ten jest prosty w implementacji, jednakże nie gwarantuje znalezienia optymalnego rozwiązania, a w swojej najprostszej postaci generuje spory % błędu. Czas wykonania algorytmu zależny jest przede wszystkim od ustalonej liczby iteracji jakie ma wykonać oraz innych wybieranych przez programistę parametrów.

Schemat działania algorytmu przedstawia się następująco:

1. Wybieramy drogę początkową na sposób dowolny
2. Wyliczamy jej koszt, ustawiamy parametry algorytmu
3. Wybieramy losowo nowe rozwiązanie(np. metodą *invert* dla losowo wybranych wierzchołków)
4. Jeżeli nowe rozwiązanie jest lepsze od poprzedniego lub przejdzie test prawdopodobieństwa ustawiamy je jako obecne
5. Zmniejszamy temperaturę o ustaloną wartość
6. Powtarzamy punkty 3-5 dopóki nie wykonamy zadanej liczby iteracji lub temperatura spadła poniżej określonego minimum lub od określonej liczby iteracji nie nastąpiła żadna zmiana

W przeciwieństwie do algorytmu *Tabu Search*, gdzie określenie złożoności obliczeniowej nie sprawiało problemu, tutaj nie jest to do końca możliwe. Niezależnie od wybranego kryterium końcowego górna granicę stanowiła będzie wybrana maksymalna liczba iteracji.

Czynnikami jakie mogą wpływać na jakość rozwiązania jak i czas operacji jest *maksymalna ilość iteracji algorytmu*, *temperatura, tempo schładzania, minimalna temperatura, maksymalna ilość iteracji w których nie nastąpiła żadna zmiana.*

2. Plan eksperymentu

- Językiem programowania użytym w eksperymencie był C++ w standardzie ISO C++11
- Badanie poszczególnych algorytmów wykonane zostało dla *17, 33, 47, 64, 70, 170, 323 miast w przypadku ATSP oraz dla 17, 48, 52, 76, 96, 136, 150 miast w przypadku STSP*
- Dane na temat odległości między miastami jak i optymalne rozwiązania dla wybranych danych testowych zaczerpnięte zostały z *TSPLIB*
- Miasta i odległości między nimi były przechowywane w pamięci komputera z wykorzystaniem *macierzy sąsiedztwa*
- Dane w plikach z testowymi instancjami były przechowywane w formie *grafu pełnego* zapisanym w postaci *macierzy sąsiedztwa* z pierwszą wartością w pliku określającą rozmiar macierzy
- Dla każdej instancji i algorytmu pomiar został powtórzony pięćdziesięciokrotnie
- Otrzymanym wynikiem jest zmierzony czas podzielony przez ilość wykonanych operacji oraz % wartość błędu stosunku otrzymanego do optymalnego rozwiązania problemu
- Czas wykonania operacji mierzony był przy użyciu biblioteki *chrono* przy pomocy funkcji *std::chrono::high_resolution_clock*
- Dla algorytmu *Tabu Search* parametry przedstawiają się następująco: ilość iteracji – 200, długość kadencji – 10, wartość aspiracji – 10, ilość iteracji po aspiracji – 10
- Dla algorytmu *Simulated Annealing* parametry przedstawiają się następująco: maksymalna ilość iteracji – l.miast^4, temperatura - $\frac{dł.początkowej\ drogi}{l.miast^2}$, zmiana temperatury – 0,999*temperatura, maksymalna ilość iteracji bez zmiany - $\frac{l.miast^4}{1000}$
- Wartości losowe generowane były przy użyciu biblioteki *random*, przy wyborze miast korzystano z *uniform_int_distribution,* a przy prawdopodobieństwie zależnym od temperatury *uniform_real_distribution*
- Startowa droga w przypadku obu algorytmów generowana była przy użyciu *algorytmu zachłannego*

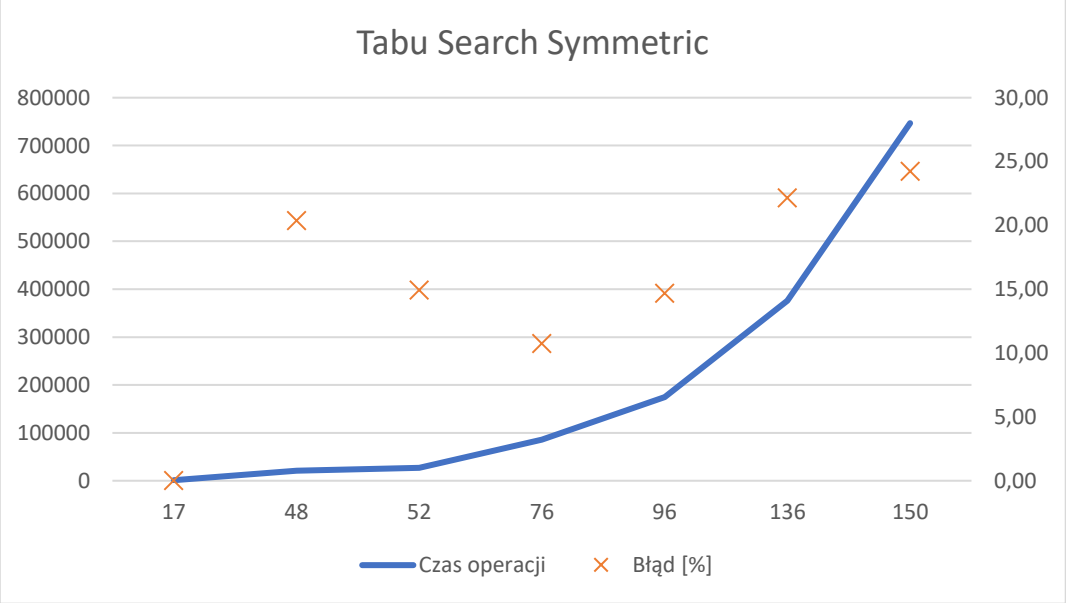
3. Wyniki/Wnioski

Czasy wykonywania poszczególnych algorytmów podane zostały w mikrosekundach.

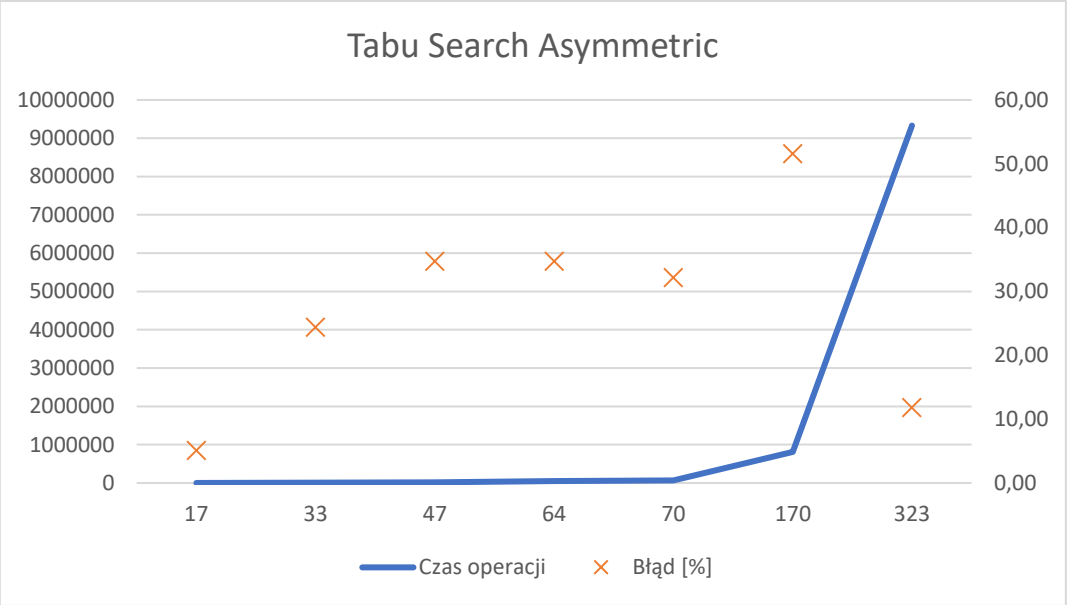
3.1. Algorytm Tabu Search

Tabu Search Symmetric				Tabu Search Asymmetric			
L.miast	Czas operacji	Rozwiazanie	Błąd [%]	L.miast	Czas operacji	Rozwiazanie	Błąd [%]
17	1110	2085	0,00	17	1222	41	5,13
48	20951	6074	20,37	33	6295	1600	24,42
52	27263	8667	14,92	47	18380	2393	34,74
76	85753	119784	10,75	64	50378	2478	34,75
96	174673	63314	14,68	70	64367	2578	32,21
136	375950	118212	22,16	170	811798	4176	51,58
150	746829	8110	24,23	323	9330453	1483	11,84

Tabela 1 Czas[µs] wykonywania algorytmu Tabu Search oraz % błędu rozwiązania w zależności od l. miast dla symetrycznego i asymetrycznego problemu komiwojażera



Wykres 1 Zależność między liczbą miast a czasem wykonywania oraz błędem rozwiązania algorytmu Tabu Search dla symetrycznego problemu komiwojażera



Wykres 2 Zależność między liczbą miast a czasem wykonywania oraz błędem rozwiązania algorytmu Tabu Search dla asymetrycznego problemu komiwojażera

Otrzymane wyniki dla algorytmu *Tabu Search* zgadzają się z teoretyczną złożonością obliczeniową założoną we wstępie $O(n^2)$, gdzie n to liczba miast.

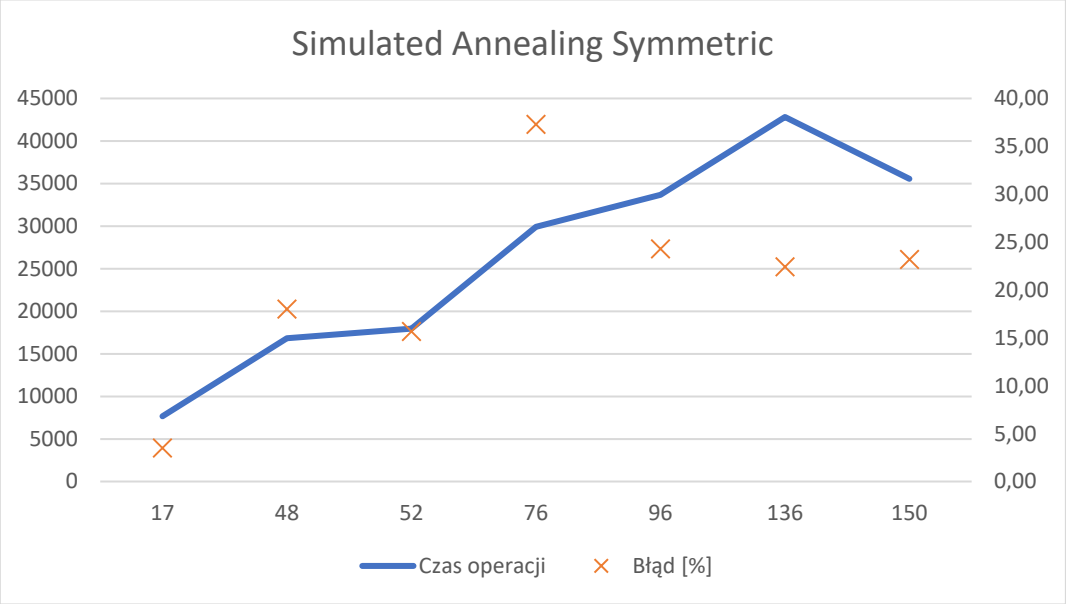
Powyższy wniosek o zgodności teorii z otrzymanymi danymi wynika z przeprowadzonej analizy w wyniku której otrzymano stały współczynnik, stosunku wyliczonej teoretycznej złożoności na podstawie liczby miast do czasu wykonywania algorytmu. Różnic w czasie wykonywania algorytmu zależnie od symetryczności problemu nie stwierdzono.

Rozwiązanie optymalne udało otrzymać się jedynie dla liczby miast równej 17 i to jedynie dla problemu symetrycznego. Dla przypadku STSP % błędu wacha się w graniach od 0% - 25%, a dla przypadku ATSP od 5% - 52%, w obu przypadkach z tendencją wzrostową wraz ze wzrostem liczby miast. W przypadku STSP i ATSP korzystano z różnych zestawów danych, a więc nie można ich bezpośrednio porównać. W obu przypadkach zauważyć możemy lokalne minima % błędu, co może wskazywać na to, że dobrane parametry algorytmu były odpowiednie dla niektórych przypadków, a dla innych ich zmiana mogła by doprowadzić do zmniejszenia % błędu.

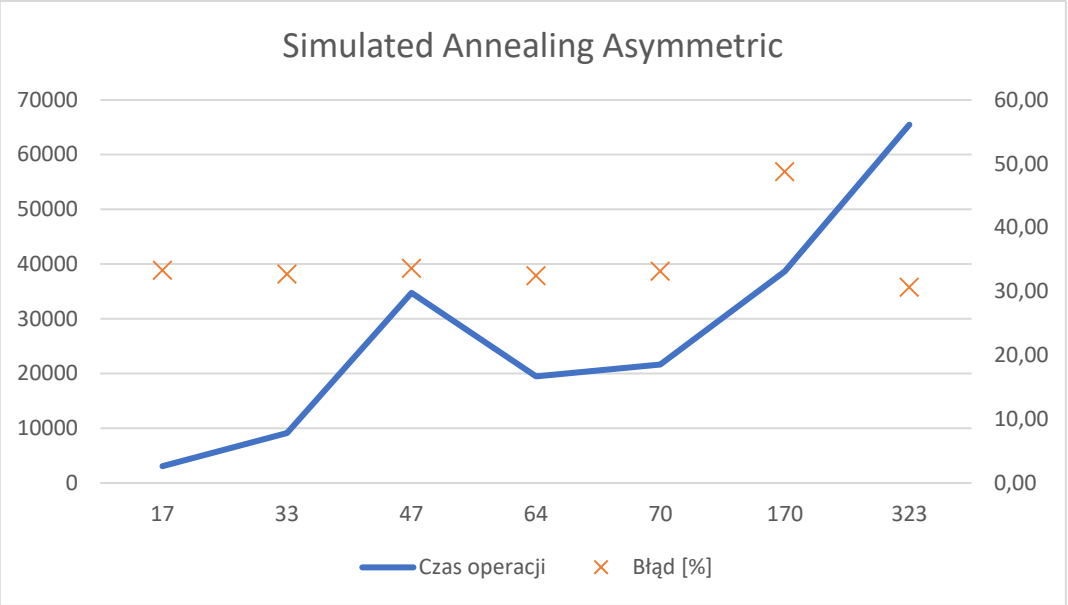
3.2. Algorytm Simulated Annealing

Simulated Annealing Symmetric				Simulated Annealing Asymmetric			
L.miast	Czas operacji	Rozwiazanie	Błąd [%]	L.miast	Czas operacji	Rozwiazanie	Błąd [%]
17	7667	2158	3,50	17	3075	52	33,33
48	16828	5955	18,01	33	9139	1707	32,74
52	17972	8723	15,66	47	34734	2374	33,67
76	29918	148507	37,30	64	19462	2437	32,52
96	33697	68628	24,31	70	21650	2597	33,18
136	42824	118452	22,40	170	38682	4099	48,78
150	35569	8042	23,19	323	65485	1733	30,69

Tabela 2 Czas[µs] wykonywania algorytmu Simulated Annealing oraz % błędu rozwiązania w zależności od l. miast dla symetrycznego i asymetrycznego problemu komiwojażera



Wykres 3 Zależność między liczbą miast a czasem wykonywania oraz błędem rozwiązania algorytmu Simulated Annealing dla symetrycznego problemu komiwojażera

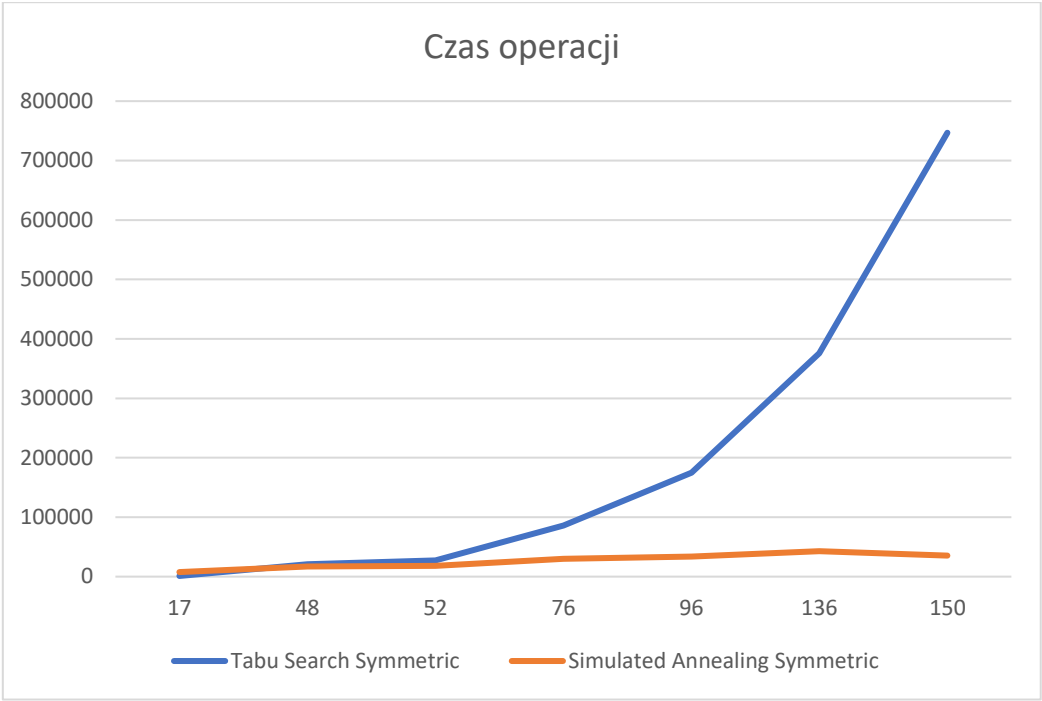


Wykres 4 Zależność między liczbą miast a czasem wykonywania oraz błędem rozwiązania algorytmu Simulated Annealing dla symetrycznego problemu komiwojażera

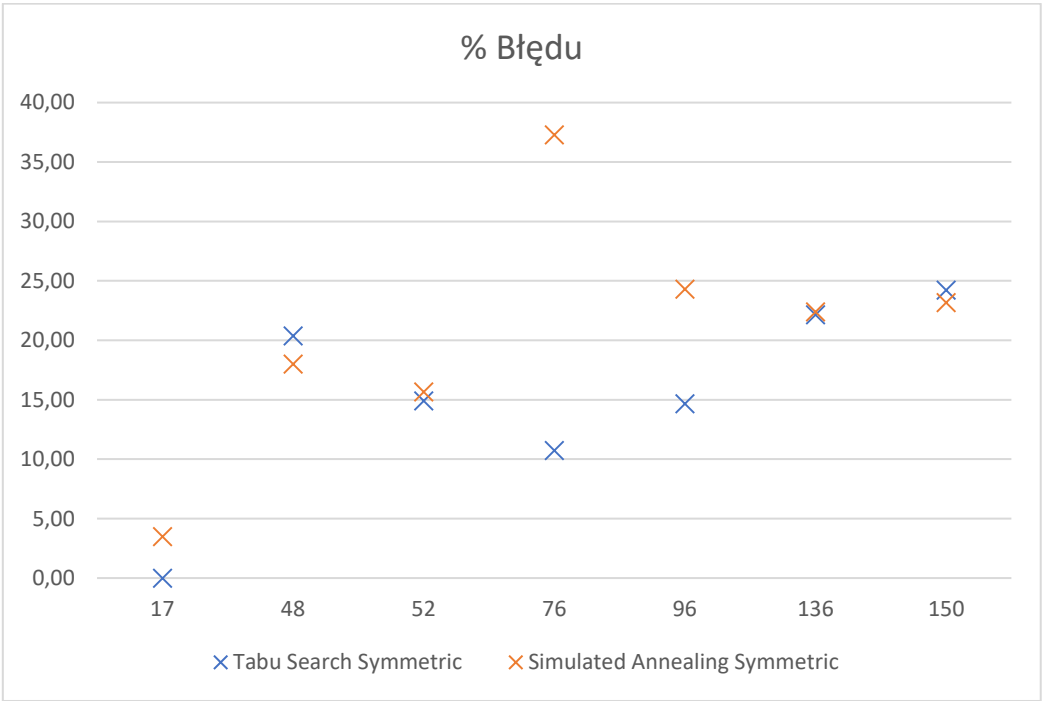
Niemożliwe jest porównanie otrzymanych wyników z teoretyczną złożonością obliczeniową gdyż nie została ona jednoznacznie zdefiniowana.

Nie udało się otrzymać rozwiązania optymalnego dla żadnego z zestawu danych. Dla przypadku STSP % błędu wacha się w graniach od 4% - 37%, a dla przypadku ATSP od 32% - 49%, w obu przypadkach utrzymuje się na podobnym poziomie niezależnie od liczby miast. W przypadku STSP i ATSP korzystano z różnych zestawów danych, a więc nie można ich bezpośrednio porównać. W obu przypadkach otrzymano stosunkowo duże % błędów, co spowodowane jest tym że zaimplementowano jedynie podstawową wersję algorytmu oraz może być spowodowane nieodpowiednim doborem parametrów algorytmu.

3.3. Tabu Search a Simulated Annealing



Wykres 3 Zależność między liczbą miast a czasem wykonywania algorytmu dla konkretnych algorytmów dla symetrycznego problemu komiwojażera



Wykres 4 Zależność między liczbą miast a % błędu otrzymane rozwiązania dla konkretnych algorytmów dla symetrycznego problemu komiwojażera

Pod względem czasu wykonania algorytmu znaczącą przewagę posiada symulowane wyważanie, gdyż jego czas wykonywania nie jest bezpośrednio zależny od liczb miast jak w przypadku przeszukiwania z zakazami, a od dobranych parametrów i warunku zakończenia pracy algorytmu.

Natomiast pod względem poprawności otrzymywanych wyników przewagę posiada przeszukiwanie z zakazami, gdyż lepsze wyniki udało się uzyskać w 5 przypadkach z 7 testowanych oraz w jednym z tych przypadków, choć nieznaczącym, udało się uzyskać optymalny rezultat.

Na wykresach porównano oba algorytmy jedynie dla symetrycznego problemu komiwojażera, gdyż nie różnią się one znacząco od wyników otrzymanych dla asymetrycznej wersji problemu.

4. Podsumowanie

- Otrzymane wyniki dla algorytmu Tabu Search zgadzają się z teorią, dla algorytmu Simulated Annealing natomiast z powodu braku możliwości porównania z teoria słusznym stwierdzeniem będzie jedynie to że jego szybkość wykonania zależy od ustawionych parametrów i wybranych kryteriów końcowych

- Gdyby decydować nad wyborem algorytmu, który będzie implementowany, na podstawie czasu wykonywania zdecydowanie lepszym wyborem jest algorytm Simulated Annealing, generuje on jednak dalsze od poprawnych rozwiązania, przez co w przypadku gdy potrzebujemy dokładniejszego algorytmu lepszym wyborem może być Tabu Search

- Oba algorytmy w swojej najprostszej postaci nie stwarzają problemów przy implementacji, przez co nie ma to wpływu na wybór któregokolwiek z nich. Prostszy natomiast wydaje się wybór parametrów dla Tabu Search, dla którego nawet dla niedokładnie dobranych argumentów otrzymujemy stosunkowo dobre wyniki, w przeciwieństwie do Simulated Annealing