|  |
| --- |
| Kajetan Parzyszek |
| Projektowanie Efektywnych Algorytmów |
| Zadanie projektowe nr 3: Symetryczny i Asymetryczny Problem Komiwojażera. Populacyjne algorytmy metaheurystyczne. Algorytm Genetyczny. |

|  |
| --- |
| K P  1-21-2019 |

Spis treści

[1. Wstęp teoretyczny 2](#_Toc535838342)

[1.1. Symetryczny i Asymetryczny Problem Komiwojażera 2](#_Toc535838343)

[1.2. Algorytm Genetyczny 3](#_Toc535838344)

[2. Plan eksperymentu 4](#_Toc535838345)

[3. Wyniki/Wnioski 5](#_Toc535838346)

[3.1. Algorytm Tabu Search 5](#_Toc535838347)

[3.2. Algorytm Simulated Annealing 7](#_Toc535838348)

[3.3. Tabu Search a Simulated Annealing 9](#_Toc535838349)

[4. Podsumowanie 10](#_Toc535838350)

# 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

## 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 .

## Algorytm Genetyczny

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 , 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 , a więc teoretyczne asymptotyczne tempo wzrostu zapisane za pomocą notacji dużego O będzie wynosić O(n2).

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.*

# Plan eksperymentu

* Językiem programowania użytym w eksperymencie był C++ w standardzie ISO C++11
* Badanie poszczególnych algorytmów wykonane zostało dla *17, 47, 70, 170 miast w przypadku ATSP oraz dla 17, 48, 96, 136 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* zpierwszą wartością w pliku określającą rozmiar macierzy
* Dla każdej instancji i zestawu parametrów pomiar został powtórzony dwudziestokrotnie
* Otrzymanym wynikiem jest % wartość błędu stosunku otrzymanego do optymalnego rozwiązania problemu zależnie od zestawu parametrów oraz zależnie od liczby iteracji dla testowanego optymalnego zestawu parametrów
* Dla algorytmu *Genetycznego* możliwe zestawy parametrów to:
  + Rozmiar populacji: {10, 30, 50, 70}
  + Prawdopodobieństwo krzyżowania: {0.5, 0.6, 0.7, 0.8, 0.9, 1}
  + Prawdopodobieństwo mutacji: {0.01, 0.05, 0.1, 0.15}
* Dla każdej instancji problemu na podstawie pięciu zestawów parametrów dających najmniejszy % błędu wyliczono średnio ważoną (wagi: 1. Miejsce 5, 2. Miejsce 4, … 5. Miejsce 1) dla każdego z parametrów, którą następnie wykorzystano w pomiarach % błędu zależnego od liczby iteracji
* Wartości losowe generowane były przy użyciu biblioteki *random*, przy wyborze miast oraz punkcie krzyżowania korzystano z *uniform\_int\_distribution,* a przy testowaniu występienia krzyżowania oraz mutacji z *uniform\_real\_distribution*
* Początkowa populacja generowana była poprzez wybór kolejnych losowych miast z puli możliwych

# Wyniki/Wnioski

Czasy wykonywania poszczególnych algorytmów podane zostały w mikrosekundach.

## Algorytm Tabu Search

 

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(n2), 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.

## Algorytm Simulated Annealing



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.

## 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.

# 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