Mateusz Krupa 256280

# Projektowanie efektywnych algorytmów
# Asymetryczny Problem Komiwojażera
# Algorytm Symulowanego Wyżarzania

# Spis treści

# 1 Wstęp teoretyczny

Celem zadania projektowego była implementacja oraz analiza efektywności algorytmu Symulowanego Wyżarzania. Należało zbadać jak dobre rozwiązania algorytm ten otrzymuje w określonym czasie i przedstawić je na wykresie jako zależność błędu względnego (podanego w procentach) w funkcji czasu wykonywania algorytmu. Błąd względny wyrażono jako:

$$\frac{|f - f_{opt}|}{f_{opt}}$$

gdzie:
$f$ - wartość obliczona przez testowany algorytm
$f_{opt}$ - wartośc optymalna - najlepsze znane rozwiązanie

Problem komiwojażera (eng. TSP) to zagadnienie polegające na znalezieniu najkrótszej drogi w grafie nieskierowanym, w którym znamy wszystkie wierzchołki oraz krawędzie. Dzięki tym informacjom jesteśmy w stanie wyznaczyć różne drogi w zależności o interesujące nas kryteria. W projekcie skupimy się na znalezieniu najkrótszej drogi od miasta początkowego (czyli 0) poprzez wszystkie pozostałe miasta i powrót do miasta początkowego.

## 1.1 Algorytm Symulowanego Wyżarzania

Symulowane wyżarzanie (eng. Simulated annealing) jest odniesieniem do zjawiska stygnięcia metali kiedy to wraz z powolnym spadkiem temperatury cząsteczki układają się i tworzą regularną strukturę. Jako algorytm jest metodą iteracyjną, w kolejnych krokach modyfikujemy rozwiązanie, które początkowo daje lepsze i gorsze rozwiązania jednak wraz z upływem czasu algorytm odrzuca wszystkie gorsze rozwiązania licząc, że uzyskał globalnie optymalne rozwiązanie.

# 2 Opis działania algorytmu

W algorytmie symulowanego wyżarzania kolejne stany były otrzymywane za pomocą losowej zamiany dwóch miast. W sytuacji gdy otrzymany stan był lepszy od poprzedniego zawsze jest przyjmowany on jako rozwiązanie. Ponadto, możliwe jest przyjęcie gorszego rozwiązania, ma to na celu umożliwienie wyjścia z maksimum lokalnego. Gorsze rozwiązanie jest przyjmowane z pewnym prawdopodobieństwem wyrażonym wzorem:

$$exp(\frac{f_i - f_j)}{T})$$

gdzie:
$f_i$ - możliwe rozwiązanie
$f_j$ - najlepsze znane rozwiązanie
$T$ - temperatura

Wynik porównujemy z losową wartością od 0 do 1 jeżeli wartość wylosowana jest większa przyjmujemy gorsze rozwiązanie. Wraz z upływem czasu temperatura maleje co oznacza, że prawdopodobieństwo przyjęcia gorszego rozwiązania również maleje. Spadek temperatury odbywa się poprzez pomnożenie aktualnej wartości o na przykład 0.999, aby uzyskać lepszy wynik spadek nie odbywa się z każda iteracją pętli, odbywa się co którąś iterację i jeśli wyobrazimy sobie wykres zmiany temperatury w czasie to przybrał by on wygląd malejącego wykresu schodkowego. Algorytm kończy się w przypadku osiągnięcia warunku końcowego, warunek końcowy w mojej implementacji dzieli się na dwa rodzaje: po osiągnięciu jego maksymalnego trwania lub po osiągnięciu tej samej drogi bardzo dużą ilość razy z rzędu.

# 3 Wyniki eksperymentu i wnioski

## 3.1 Testy

Przy testach aby uzyskać jak najlepsze wyniki zmieniano wartość mnożnika oraz długości epoki.
$f$ - mnożnik
$c$ - długość epoki

### 3.1.1 br17

Dla tego przykładu użyłem tych samych parametrów wejściowych dla wszystkich pomiarów czasu ponieważ jest to dość prosty przykład i wyniki były bardzo podobne dla różnych czasów i te parametry wystarczyły by osiągnąć zadowalające wyniki.
f = 0.99
c = 20

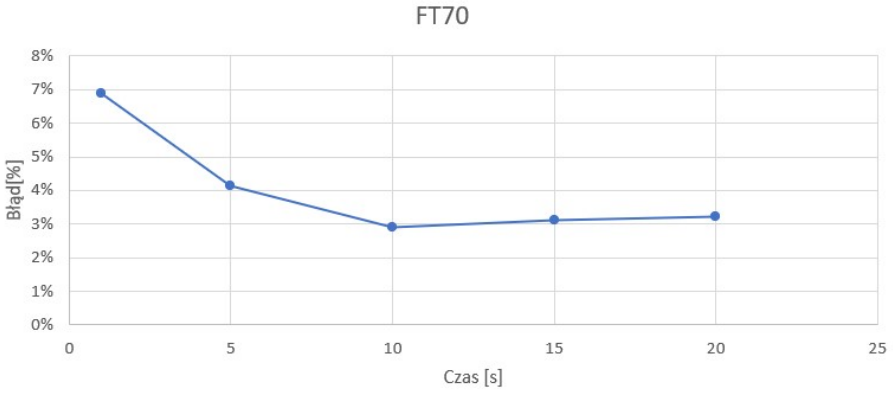
| **Czas [ms]** | **Błąd[%]** |
|---|---|
| 5 | 4% |
| 25 | 2% |
| 100 | 2% |
| 500 | 0% |
| 1000 | 0% |


Rysunek 1: Błąd względny od funkcji czasu

### 3.1.2 ft70

Dla tego przykładu zmieniałem parametry wejściowe tak, aby udało się osiągnąć jak najlepszy wynik dla poszczególnego czasu.

| **Czas [ms]** | **Błąd[%]** | **f** | **c** |
|---|---|---|---|
| **1** | 7% | 0.999 | 20 |
| **5** | 4% | 0.9999 | 40 |
| **10** | 3% | 0.9999 | 900 |
| **15** | 3% | 0.99999 | 100 |
| **20** | 3% | 0.99999 | 200 |

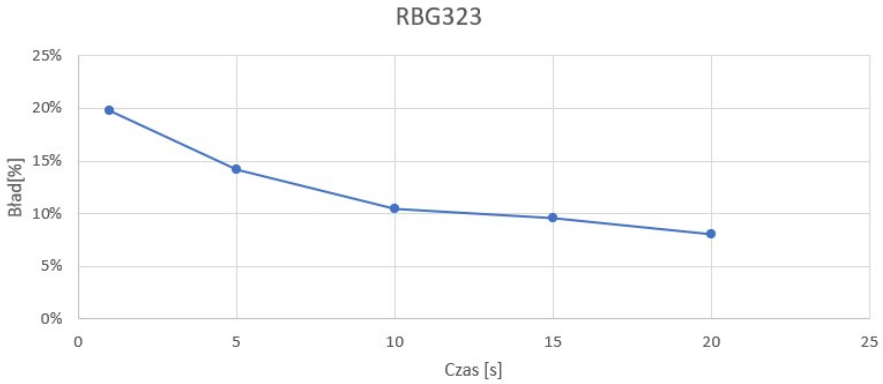



Rysunek 2: Błąd względny od funkcji czasu

### 3.1.3 rbg323

Dla tego przykładu zmieniałem parametry wejściowe tak aby udało się osiągnąć jak najlepszy wynik dla poszczególnego czasu.

| Czas [ms] | Błąd[%] | f | c |
|---|---|---|---|
| **1** | 20% | 0.99 | 4 |
| **5** | 14% | 0.999 | 100 |
| **10** | 10% | 0.9999 | 50 |
| **15** | 10% | 0.9999 | 80 |
| **20** | 8% | 0.99999 | 12 |




Rysunek 3: Błąd względny od funkcji czasu

## 3.2 Wnioski

Podczas testowania algorytmu zauważyłem, że nie była dla mnie konieczna zmiana temperatury początkowej. W sytuacji kiedy chciałem by algorytm był bardziej chaotyczny wystarczyło tylko zmniejszyć mnożnik albo zwiększyć długość epoki. Taki pomysł dawał dobre rezultaty. Dla małych instancji błąd względny był niewielki, jednak dla większych instancji potrzeba było więcej czasu. Zauważyłem, że dla instancji br17 pomimo długiego działania algorytmu i tak możliwe było nie znalezienie optymalnej drogi, co wynika z tego, że jest to algorytm heurystyczny. Algorytm symulowanego wyżarzania daje bardzo dobre wyniki w porównaniu z czasem potrzebnym, aby znaleźć rozwiązanie, które jest bliskie optymalnemu przy niedużym zapotrzebowaniu na pamięć.

# 4 Literatura

[1] *http://cs.pwr.edu.pl/zielinski/lectures/om/localsearch.pdf*
[2] *https://youtu.be/gX-X85dCib0*
[3] *A. Debudaj-Grabysz, S. Deorowicz, J. Widuch, Algorytmy i struktury danych. Wybór zaawansowanych metod, Wydawnictwo Politechniki Śląskiej, Gliwice, 2012*
[4] *https://pl.wikipedia.org/wiki/Symulowane_wy%C5%BCarzanie*