Projektowanie efektywnych algorytmów
Projekt

263934 Michał Pawlus

20.12.2023

Simulated annealing

Spis treści	strona

Sformułowanie zadania
Zadanie polega na opracowaniu i implementacji algorytmu symulowanego wyżarzania, który rozwiązywać będzie problem znalezienia minimalnego cyklu Hamiltona w grafie ważonym. Problem ten nazywa się problemem komiwojażera i jest problemem NP-trudnym.

Metoda

Metoda symulowanego wyżarzania jest heurystycznym algorytmem optymalizacji, który naśladuje proces wyżarzania metalu. Proces ten polega na stopniowym schładzaniu metalu, co prowadzi do ułożenia jego cząsteczek w bardziej stabilnej strukturze. Bazując na danych z internetu przewiduje, że powinno udać mi się w rozsądnym czasie(do 10minut) wykonywać instancje o liczbie wierzchołków ok. 200-300 z błędem nie przekraczającym 50%.
Główne kroki tego algorytmu:

Inicjalizacja:
Wygeneruj losowe rozwiązanie początkowe dla problemu optymalizacyjnego.

Temperatura początkowa:
Ustal początkową temperaturę, która reprezentuje poziom "gorączki". Na początku procesu temperatura jest wysoka.

Generowanie sąsiada:
Wygeneruj losowego sąsiada aktualnego rozwiązania. To może być perturbacja, czyli niewielka zmiana w aktualnym rozwiązaniu np. 2-zmiany lub invert

Funkcja oceny:
Oceni różnicę w jakości między aktualnym rozwiązaniem a nowo wygenerowanym sąsiadem. W przypadku problemu komiwojażera, funkcją oceny jest długość trasy.

Funkcja akceptacji:
Zdefiniuj funkcję określającą, czy przyjąć nowe rozwiązanie. Jeśli nowe rozwiązanie jest lepsze, zawsze je akceptujemy. Jeśli jest gorsze, akceptujemy je z pewnym prawdopodobieństwem. Prawdopodobieństwo wyboru gorszego rozwiązania w algorytmie symulowanego wyżarzania jest dane wzorem:

$$\exp(\frac{\Delta}{T})$$

T – temperatura parametr, obiżany z czasem działania algorytmu
Δ – różnica kosztu pomiędzy aktualnym rozwiązaniem a znaleziononym sąsiedznim

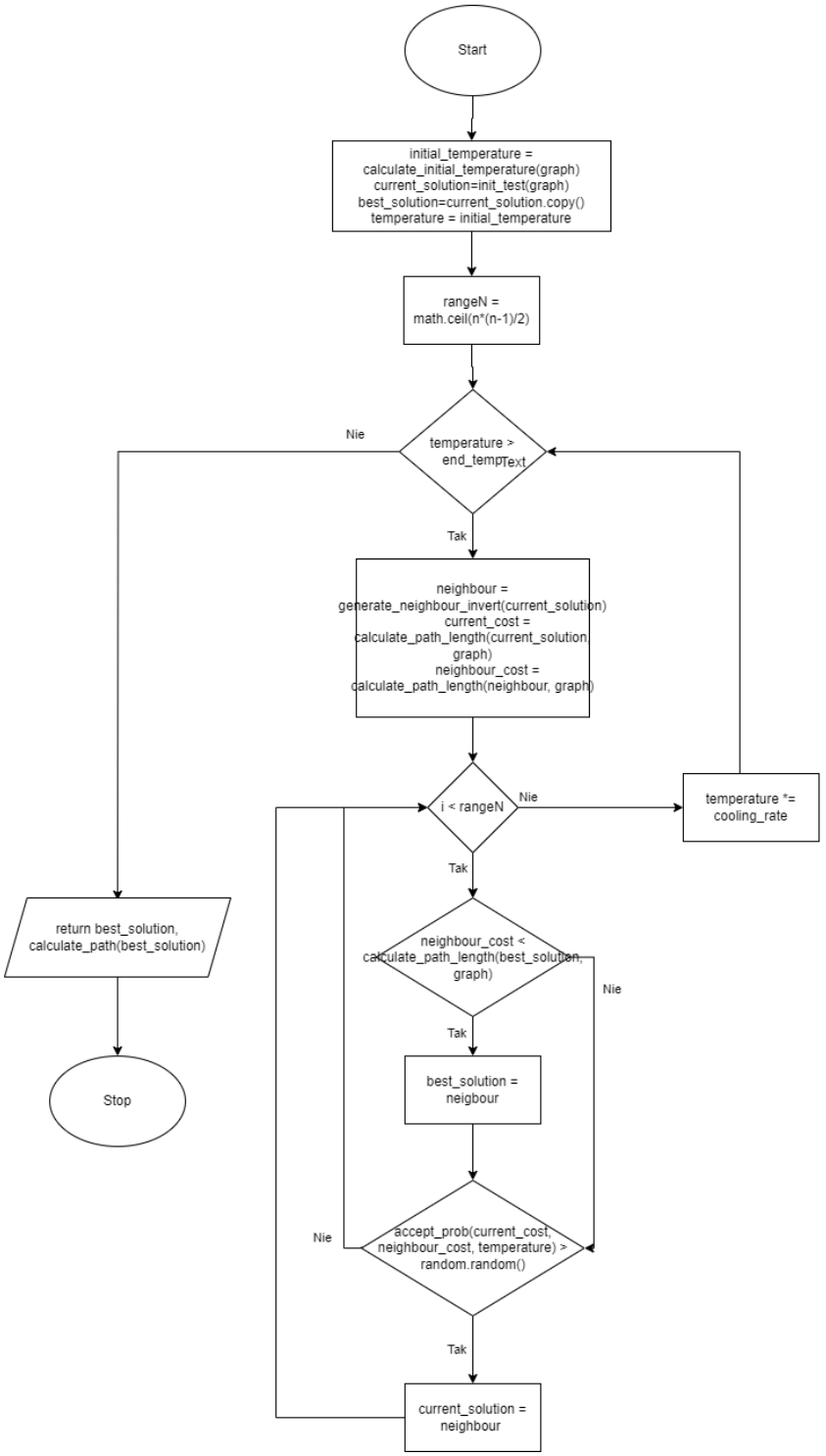
Schemat chłodzenia:
Zastosuj schemat chłodzenia, który stopniowo obniża temperaturę w czasie. Powszechnie stosowanym schematem jest geometryczne chłodzenie, które zmniejsza temperaturę w sposób wykładniczy

Iteracje:
Powtarzaj kroki generowania sąsiada, oceny, akceptacji i aktualizacji temperatury przez pewną liczbę iteracji lub do momentu spełnienia pewnego warunku zatrzymania.

Warunek zatrzymania:
Algorytm kończy działanie gdy temperatura spadnie poniżej pewnego poziomu.

Metoda symulowanego wyżarzania jest przykładem metaheurystyki, która pomaga unikać minimum lokalnych poprzez przyjęcie czasami gorszych rozwiązań, co pozwala na bardziej globalne poszukiwanie optymalnych rozwiązań.

Algorytm



n - ilosć wierzchołków
cooling_rate - współczynnik chłodzący

Obliczenie początkowej temperatury, początkowej ścieżki oraz obliczenie długości epoki(rangeN)

Główna pętla algorytmu sprawdza czy temperatura osiągnęła wartość końcową

Wygenerowania sąsiada. Obliczenie długości sąsiada i aktualnej ścieżki

Pętla epoki, w której przyjmowane są nowe ścieżki zgodnie z obliczonym prawdopodobieństwem

Rysunek 1: Schemat blokowy algorytmu

Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:
(Nazwa pliku)(optymalna ścieżka)

- test_6_1.csv 132
- test_10.csv 212
- tsp_12.csv 264
- tsp_13.csv 269
- tsp_14.csv 282

Dla powyższych instancji używając algorytmu SM otrzymałem rozwiązania optymalne.

Do badań w tym optymalizacji parametrów algorytmu wybrano następujący zestaw instancji:

- bayg29.txt_converted.csv 1610
- gr17.txt_converted.csv 2085
- gr21.txt_converted.csv 2707
- gr24.txt_converted.csv 1272
- gr48..txt_converted.csv 5046
- ftv33.txt_converted.csv 1286
- ftv38.txt_converted.csv 1530
- ftv44.txt_converted.csv 1613
- ftv47.txt_converted.csv 1776

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php#instances
Do wykonania ostatecznych badań wybrano następujący zestaw instancji:
Instancje symetryczne

- gr17.txt_converted.csv 2085
- bayg29.txt_converted.csv 1610
- gr48..txt_converted.csv 5046
- brazil58.txt_converted.csv 25395
- eil76.txt_converted.csv 538
- gr96.txt_converted.csv 55209
- kroA100.txt_converted.csv 21282
- ch130.txt_converted.csv 6110
- kroA150.txt_converted.csv 26524

Instancje asymetryczne

- ftv170.txt_converted.csv 2755
- ftv33.txt_converted.csv 1286
- ftv44.txt_converted.csv 1613

Procedura badawcza
W celu znalezienia optymalnych parametrów dla algorytmu badałem ich wpływ na czas trwania algorytmu i błąd względem rozwiązania optymalnego. Badane parametry to np. współczynnik chłodzenia, sposób wyboru rozwiązania w sąsiedztwie, rozmiar epoki.

Dla ostatecznej wersji algorytmu zbadałem czas rozwiązania problemu oraz błędu względem rozwiązania optymalnego w zależności od wielkości instancji.

Procedura badawcza polegała na podaniu pliku inicjalizującego z listą grafów, ręcznym zmienianiu wartości parametrów i wykonaniu badania. Wyniki(czas i błąd) zapisywane były w pliku output.csv, które były zbierane do jednego folderu.

Przykładowy plik inicjalizujący:

Dostrajanie/bayg29.txt_converted.csv 5 1610
Dostrajanie/gr17.txt_converted.csv 5 2085
Dostrajanie/gr21.txt_converted.csv 5 2707
Dostrajanie/gr24.txt_converted.csv 5 1272
Dostrajanie/gr48..txt_converted.csv 5 5046
Dostrajanie/ftv33.txt_converted.csv 5 1286
Dostrajanie/ftv38.txt_converted.csv 5 1530
Dostrajanie/ftv44.txt_converted.csv 5 1613
Dostrajanie/ftv47.txt_converted.csv 5 1776

nazwa_instancji liczba_wykonań rozwiązanie_optymalne
...

Na standardowe wyjście dla każdej instancji problemu zwracana była średnia wartość błędu oraz ostatnia wybrana ścieżka.

Platforma sprzętowa:
• procesor: AMD Ryzen 2500U 4 x 2.00GHz
• pamięć operacyjna: 14.9 GB
• system operacyjny: Windows 10

Sposób pomiaru czasu:
Do pomiaru czasu używam biblioteki tiemit i znajdującego się tam timera.

```
timer_start = timer()
sol, sol_len = anneling(matrix, n[0])
timer_end = timer()
time = time_end - time_start
```

Rysunek 2: pomiar czasu

Optymalizacja parametrów

Metoda przeglądania sąsiedztwa - zachłanna
Sposób wyboru rozwiązania w sąsiedztwie – 2-zamiana (swap)

Rozwiązanie początkowe - jest generowane przez przejście przez wszystkie wierzchołki po kolei, wybór innej metody generowania nie wydaje się istotnie wpływać na efektywność algorytmu.

Długość epoki(Liczba iteracji przy stałej temperaturze), W mojej implementacji, korzystam z sąsiedztwa typu swap. Sąsiada więc można wybrać na n po 2 sposoby.

$$L=\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

Temperatura początkowa
W algorytmie symulowanego wyżarzania temperatura początkowa jest używana do kontrolowania prawdopodobieństwa akceptacji gorszych rozwiązań w celu uniknięcia utknięcia w lokalnym minimum. Gdy temperatura jest wysoka, algorytm jest bardziej skłonny do akceptowania gorszych rozwiązań, co pomaga w "przeskakiwaniu" lokalnych minimów i poszukiwaniu globalnego minimum.

Temperature początkową można wybrać obliczając różnice kosztów między sąsiadami dla np. 100 iteracji bez zmiany temperatury. Następnie należy obliczyć średnią z tych różnic i podzielić przez logarytm naturalny o liczbie logarytmowanej wartości 0.9.

Warunek zatrzymania
Algorytm powinien się zatrzymać kiedy temperatura jest tak niska, że prawdopodobieństwo przejścia do rozwiązania gorszego jest bardzo małe. W mojej implementacji warunkiem zatrzymania jest przekroczenie wartości minimalnej 0.1. Ustaliłem to poprzez obserwację działania algorytmu dla instancji początkowych.

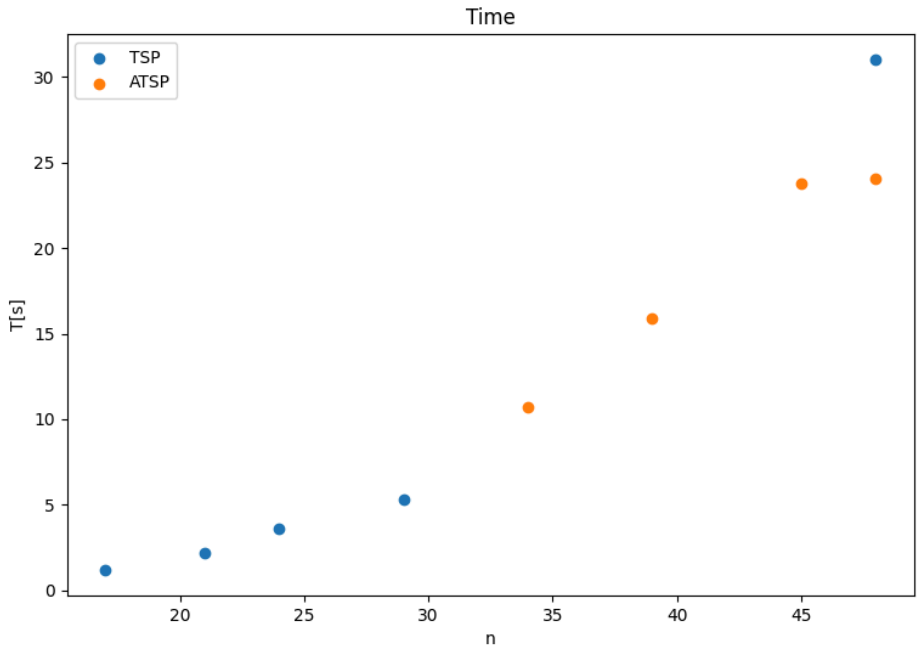
Funkcja chłodzenia
Zaimplementowany został geometryczny schemat chłodzenia. Funkcja zmiany temp:

$$f(k)=\alpha^{(k)}T_{(początkowa)}$$

$k-$ *numer epoki* $\quad k\in N$
α – *paramter informujący o szybkości schładzania* $\quad \alpha\in(0,1)$

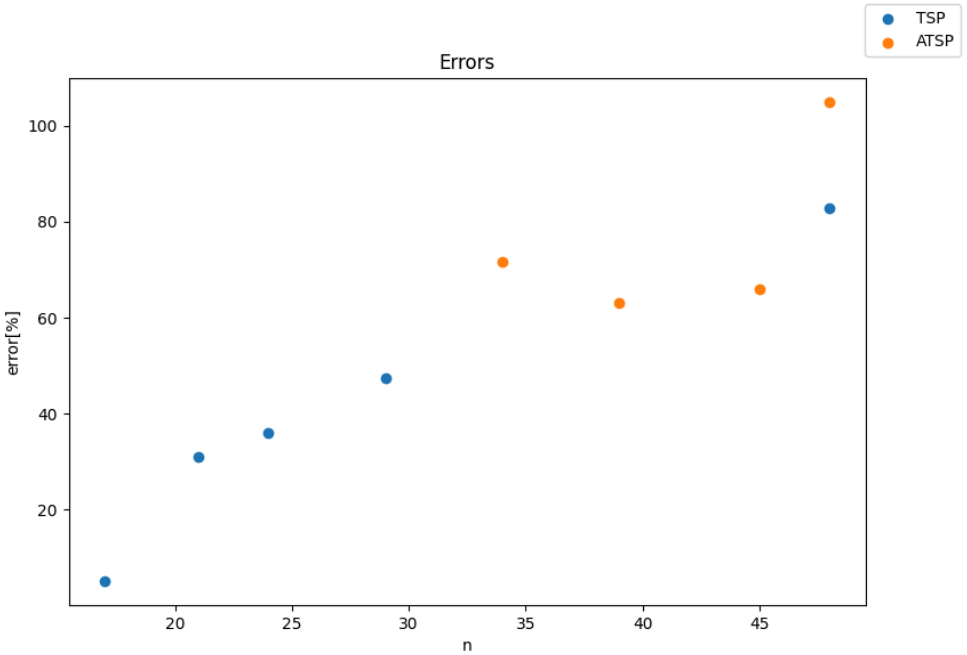
Ideą algorytmu jest powolne schładzanie, w mojej implementacji na początku przyjąłem α = 0.99

Konfiguracja podstawowa



Rysunek 3: Wpływ wielkości instancji na czas dla podstawowych parametrów

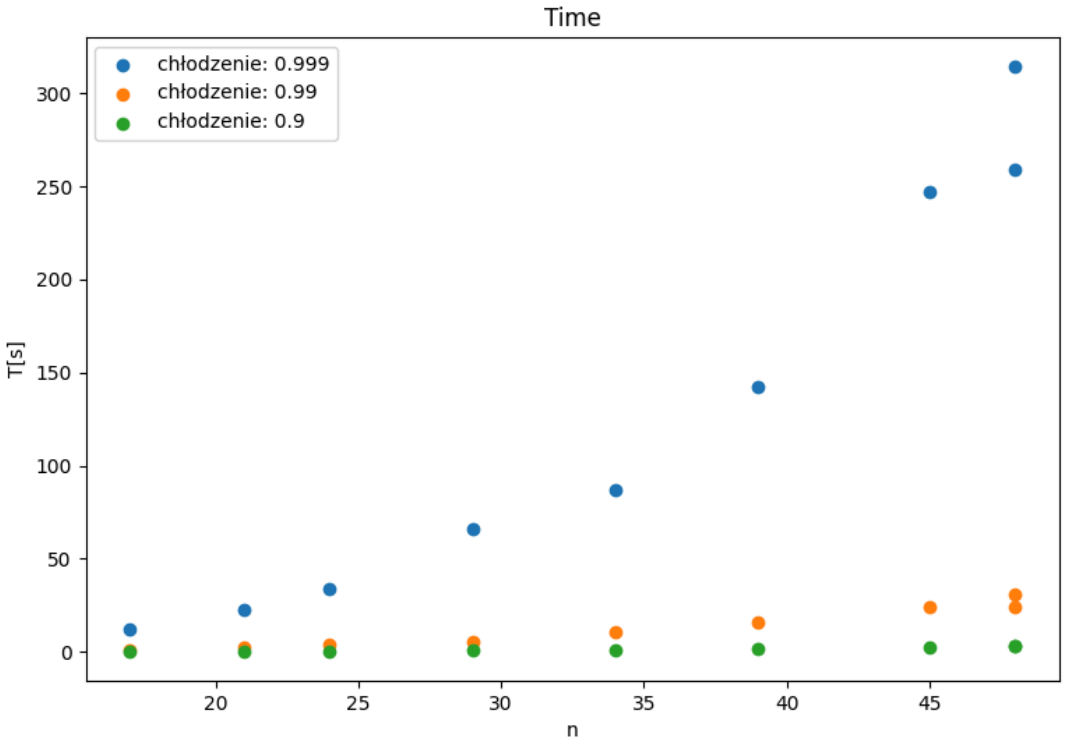
Jak widać na rysunku czas czas wzrasta wraz z wzrostem wielkości instancji tak samo dla instancji TSP oraz ATSP



Rysunek 4: Wpływ wielkości instancji na błąd dla podstawowych parametrów

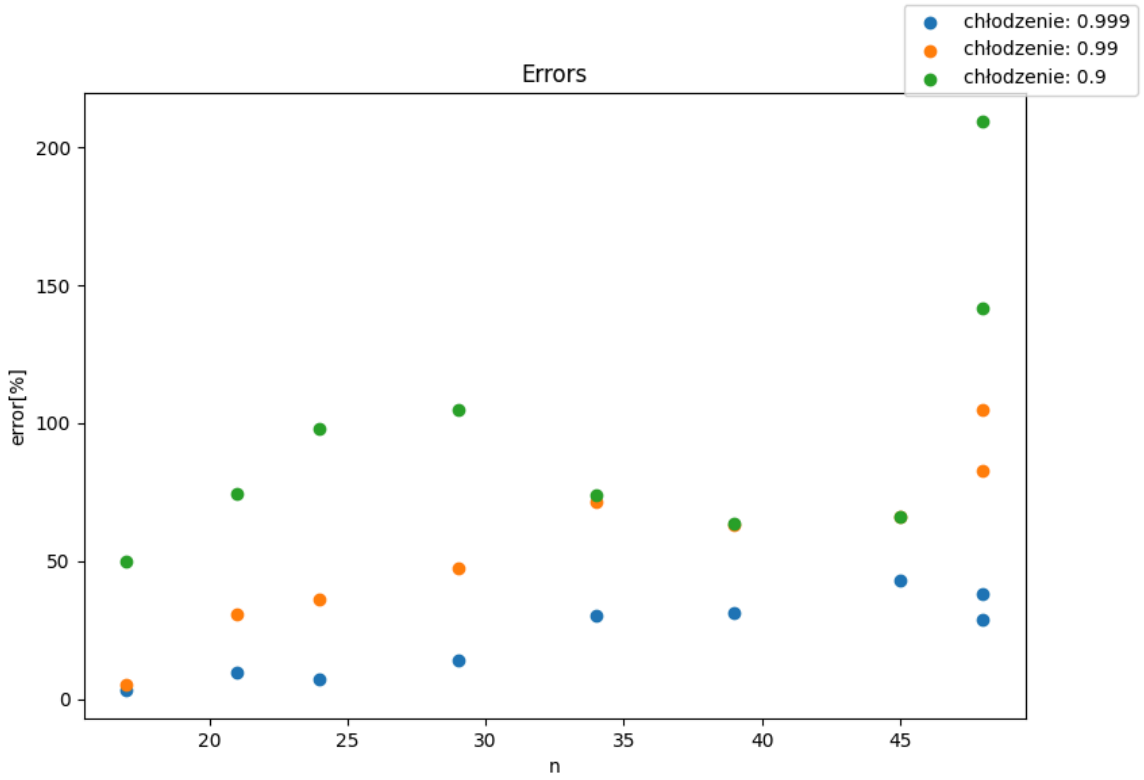
Patrząc na rysunek można zauważyć, że błąd jest większy dla instancji ATSP.

Zmniejszenie tempa chłodzenia:
Współczynnik szybkości chłodzenia jest jednym z parametrów, które znacząco wpływają zarówno na szybkość jak i dokładność algorytmu. Wartości bliżej 1 pozwalają na bardziej dokładne przeszukiwanie.



Rysunek 5: Wpływ współczynnika chłodzącego na czas

Jak widać na rysunku zwiększanie współczynnika chłodzenia ma olbrzymi wpływ na czas wykonywania algorytmu. Z dodaniem każdej 9 czas zwiększa się ok 10 razy.

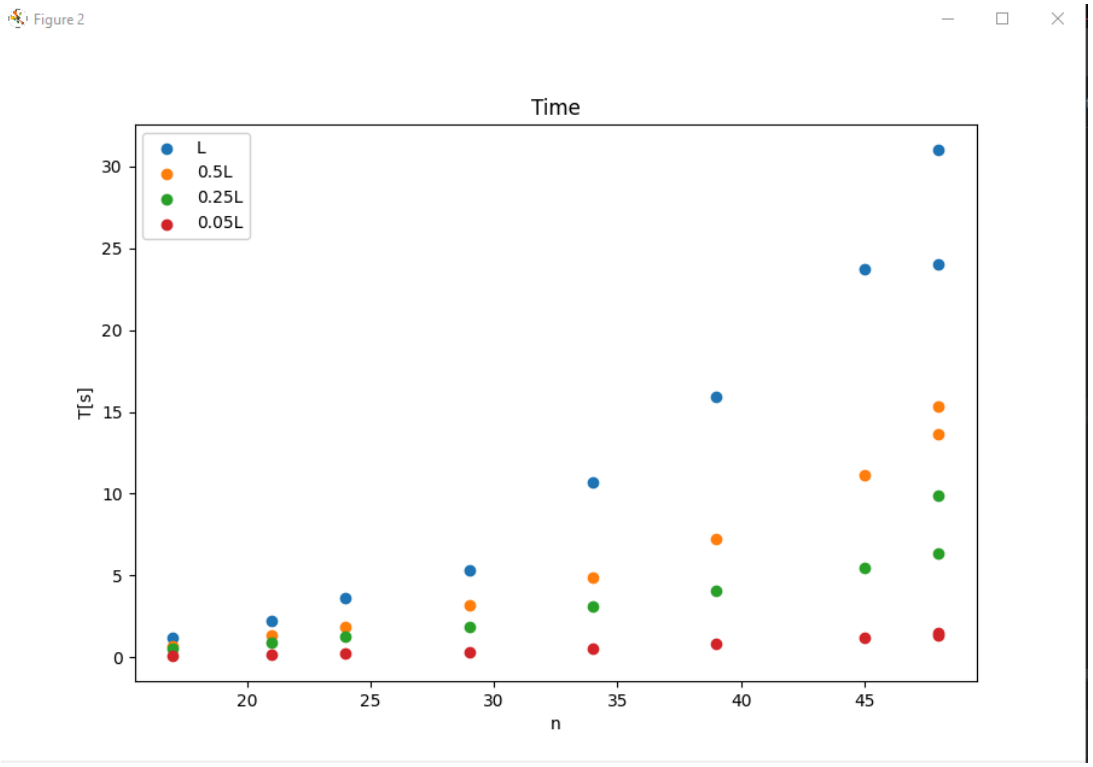


Rysunek 6: Wpływ współczynnika chłodzącego na błąd

Na rysunku widać, że zwiększenie współczynnika pozytywnie wpływa na dokładność algorytmu. Dla 0.999 udaje nam się otrzymać dokładność do której dążymy (błąd 50%). Wraz ze wzrostem współczynnika błędy zmniejszają się ok 2 razy.

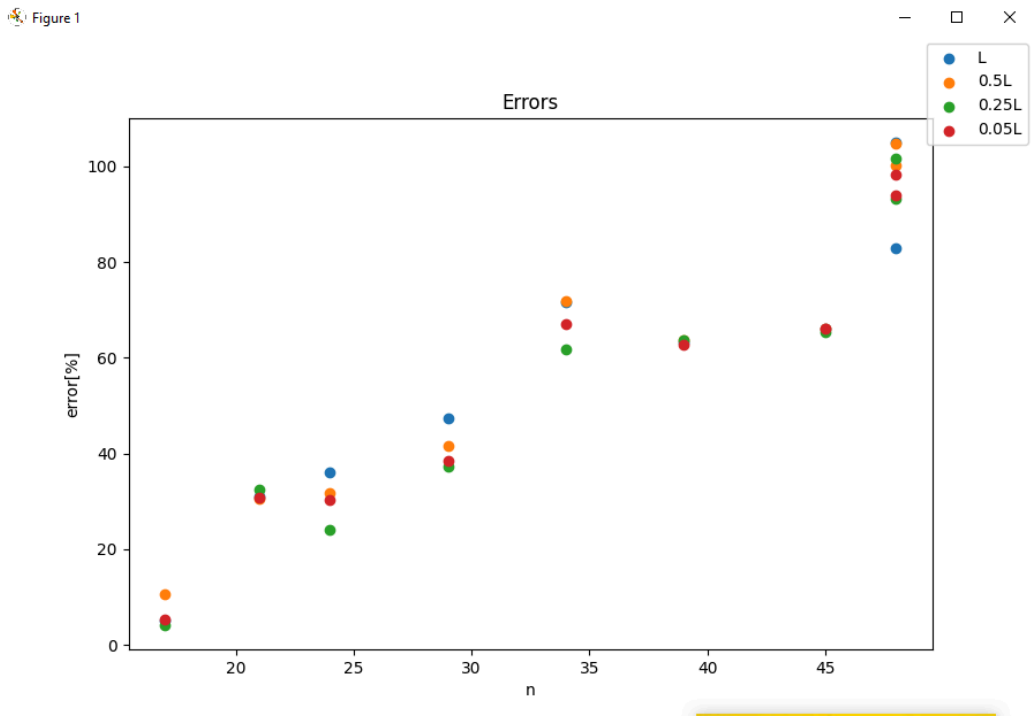
Długość epoki:

Działanie algorytmu można przyspieszyć zmniejszając długość epoki. Sprawdzę działanie algorytmu dla długości epoki 0.5L 0.25L 0.05L gdzie L to wcześniej obliczony rozmiar sąsiedztwa typu swap



Rysunek 7: Wpływ długości epoki na czas

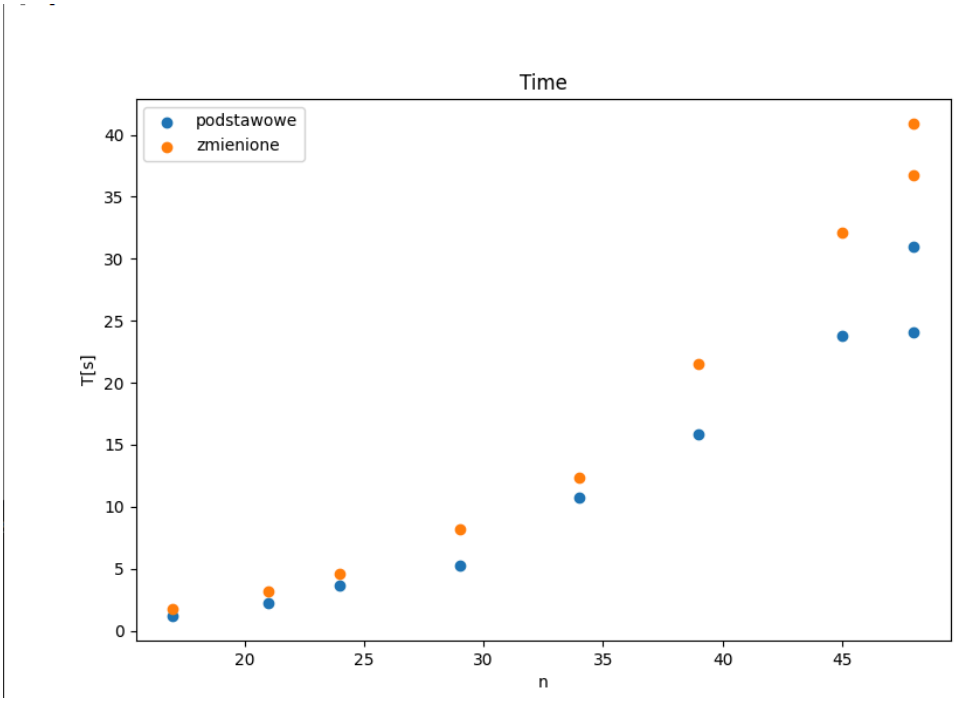
Jak wynika z powyższego wykresu zmniejszanie długości epoki daje wprost proporcjonalne zmniejszenie czasu wykonywania algorytmu.



Rysunek 8: Wpływ długości epoki na błąd

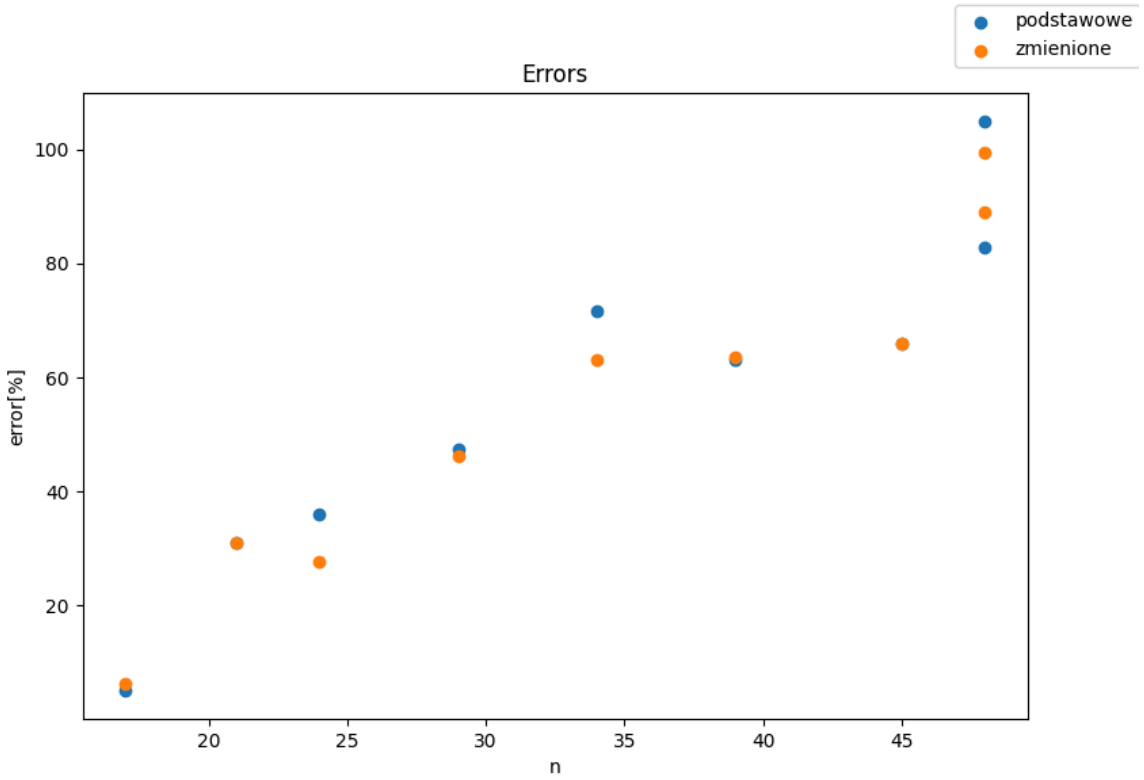
Z rysunku wynika, że długość epoki ma mały wpływ na błąd. Zmniejszenie tego parametru może znacznie obniżyć czas wykonywania, przy stosunkowo niewielkiej utracie dokładności.

Zmiana temperatury początkowej i końcowej:
W implementacji początkowej temperatura wybierana jest na podstawie przestrzeni rozwiązań instancji, a warunkiem zatrzymania jest uzyskanie temperatury mniejszej niż 0.1. W tym badaniu sprawdzę jak ustawienie temperatury początkowej na temperaturę stałą o wartości 100000 i końcową 0.05. Temperatura początkowa o takiej wartości jest większa od tych otrzymywanych ze wzoru w konfiguracji startowej, a temp końcowa niższa 2-krotnie.



Rysunek 9: Wpływ temperatury końcowej i początkowej na czas

Jak widać na rysunku zwiększenie temp początkowej i zmniejszenie końcowe wpływa negatywnie na czas wykonywania algorytmu.

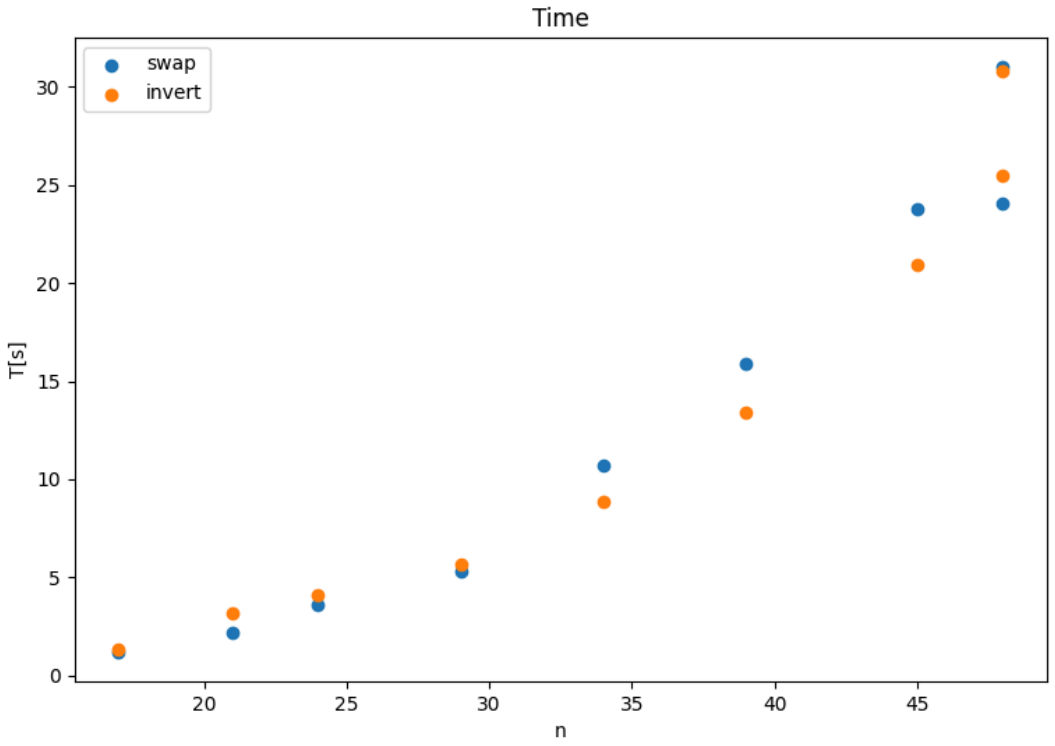


Rysunek 10: Wpływ temperatury końcowej i początkowej na błąd

Na powyższym wykresie widać, że taka zmiana wpłynęła na zmniejszenie błędu.

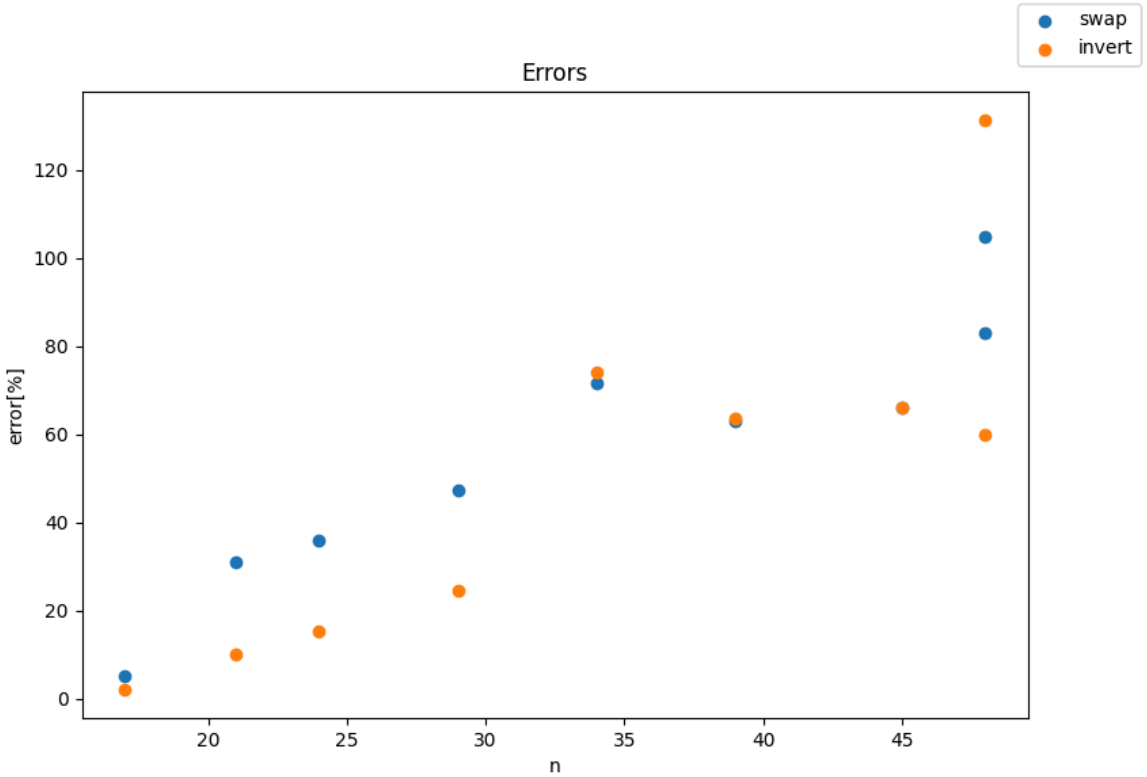
Zmiana wyboru rozwiązania w sąsiedztwie

Sprawdzenie jak zmiania sposobu rozwiązania(ze swap na invert) może wpływać na działanie algorytmu.



Rysunek 11: Wpływ typu sąsiedztwa na czas

Jak widać na powyższym wykresie zmiana ta nie wpływa na czas wykonywania algorytmu.



Rysunek 12: Wpływ typu sąsiedztwa na błąd

Porównując ten wykres z rysunkiem 4 można zauważyć, że dla symetrycznej instancji problemu TSP błędy znacznie się zmniejszyły. Natomiast dla instancji asymetrycznych powiększyły się. Wynika z tego, że w jeśli chcemy rozwiązywać instancje symetryczne problemu komiwojażera powinniśmy używać invert, a w przypadku instancji asymetrycznych powinniśmy używać swap.

Zmiana chłodzenia

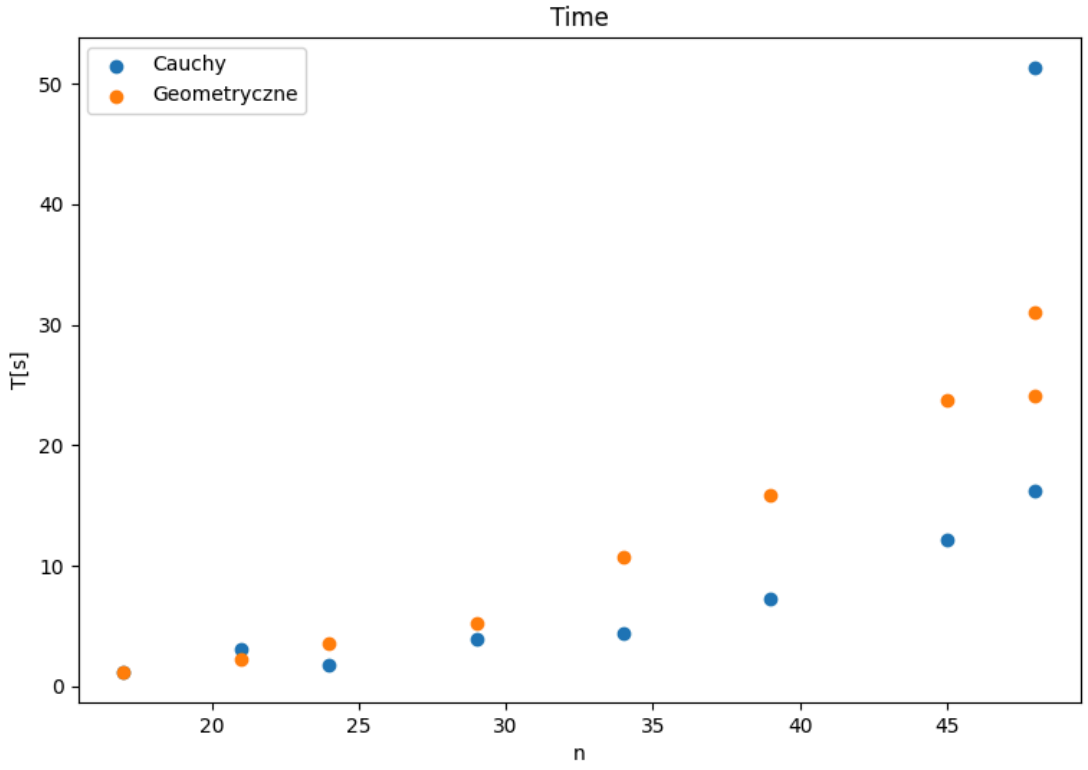
Schemat Cauchy'ego dane jest wzorem:

$$T_k = \frac{T_{\text{init}}}{1+\alpha \cdot k}$$

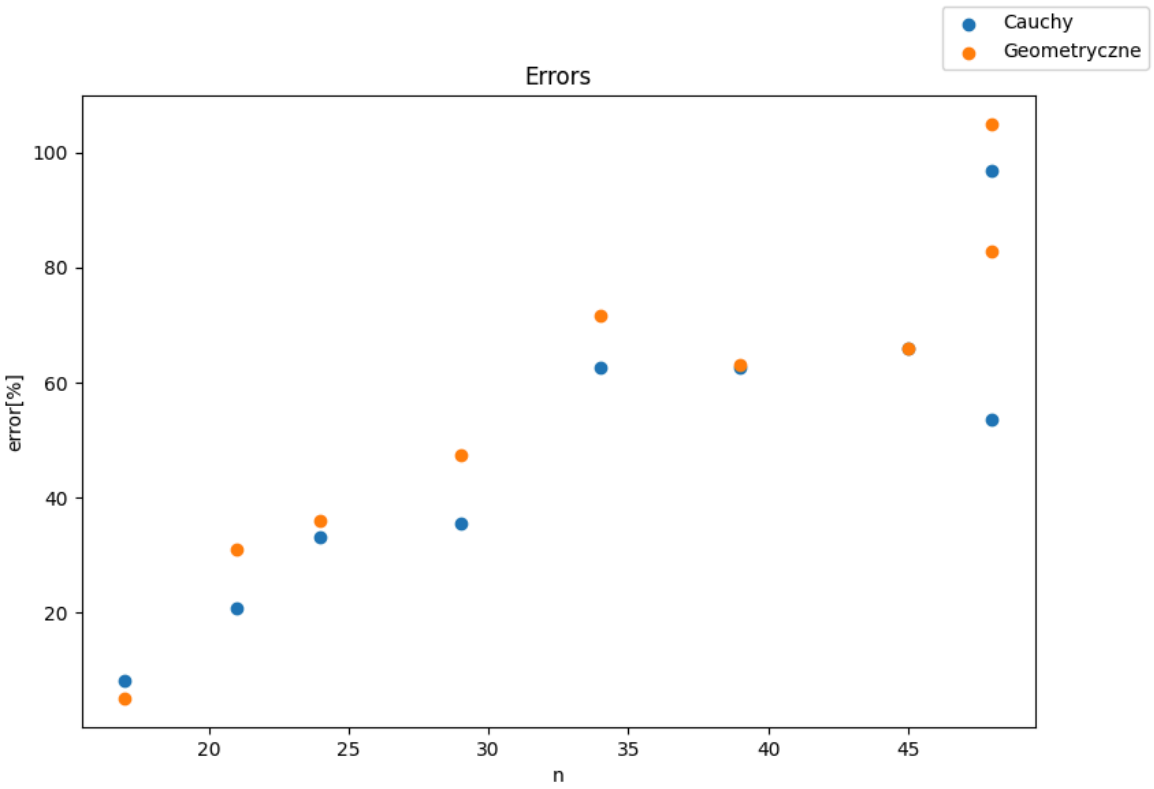
gdzie:

- T_k to temperatura po k-tej iteracji,
- T_{init} to temperatura początkowa,
- α to pewna stała zwykle rzędu 10^{-3} do 10^{-1},
- k to numer iteracji.

Do implementacji tego schematu użyłem konfiguracji z parametrami podstawowymi. Zmieniłem w niej wartość współczynnika chłodzenia na 2 i temperatury końcowej na 1.



Rysunek 13: Wpływ schematu chłodzenia na czas

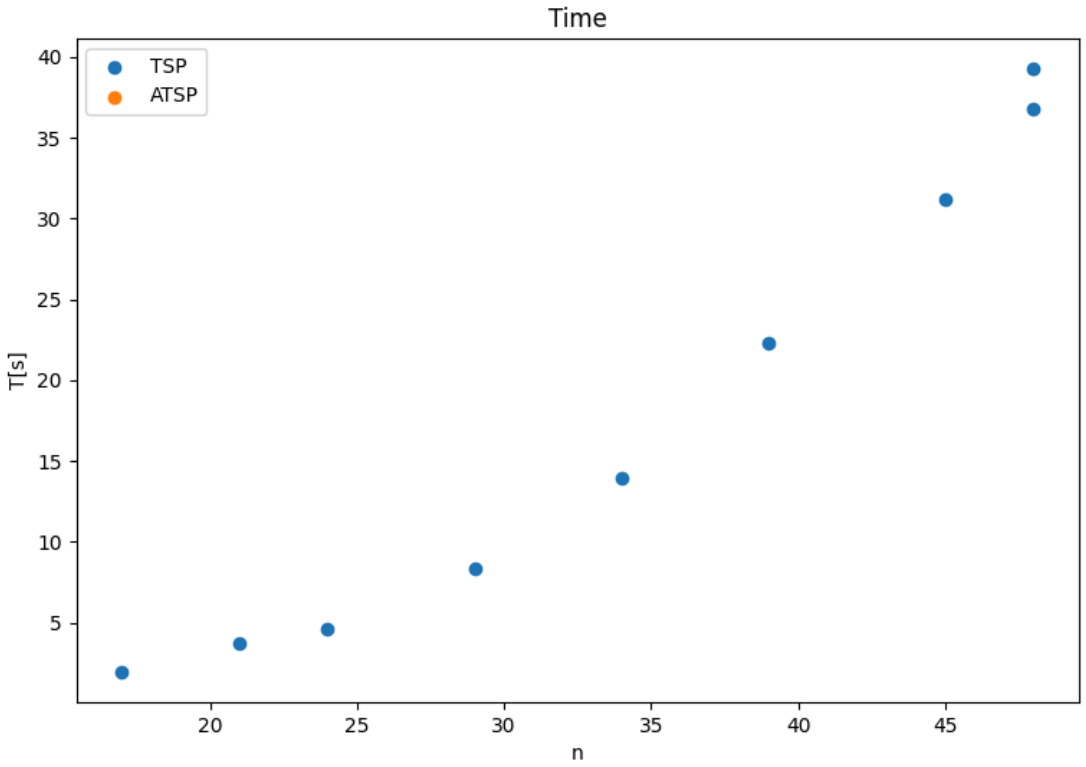


Rysunek 14: Wpływ schematu chłodzenia na błąd

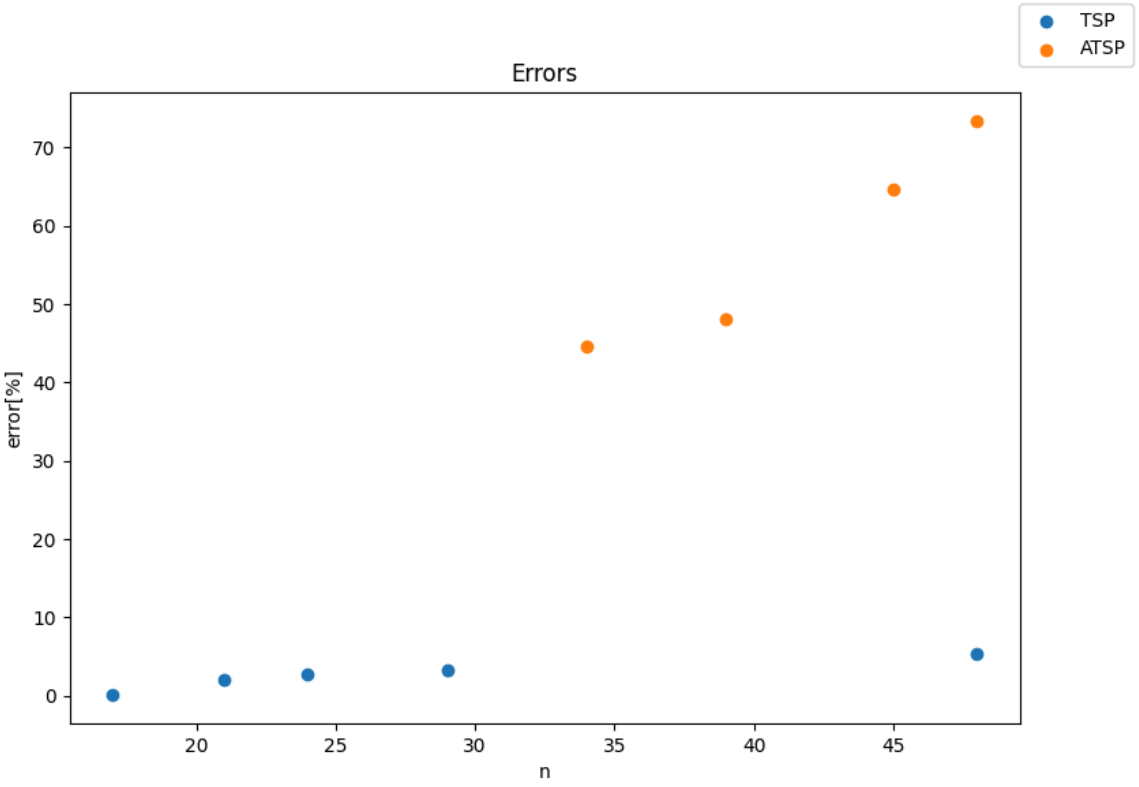
Jak widać na powyższych wykresach schemat chłodzenia ma wpływ na czas działania algorytmu. Dla schematu Cauchy'ego jest on, dla prawie wszystkich instancji,mniejszy. Błąd też wydaje się być mniejszy dla tego schematu chłodzenia. Mądrym posunięciem byłaby próba zoptymalizowania parametrów dla tego schematu chłodzenia, ponieważ jest szansa, że dawałby on wyniki lepsze od schematu geometrycznego. Jednak ze względu na brak czasu tego nie wykonam.

Najlepsza uzyskana konfiguracja

Biorąc pod uwagę wcześniej wykonane badania ustaliłem konfiguracje parametrów, która będzie miała wystarczającą precyzje do zachowania błędu 50% przy jak najmniejszym czasie trwania. Konfiugracja: chłodzenie geometryczne, współczynnik chłodzenia 0.999, długość epoki L/10, wybór rozwiązania w sąsiedztwie typu invert.



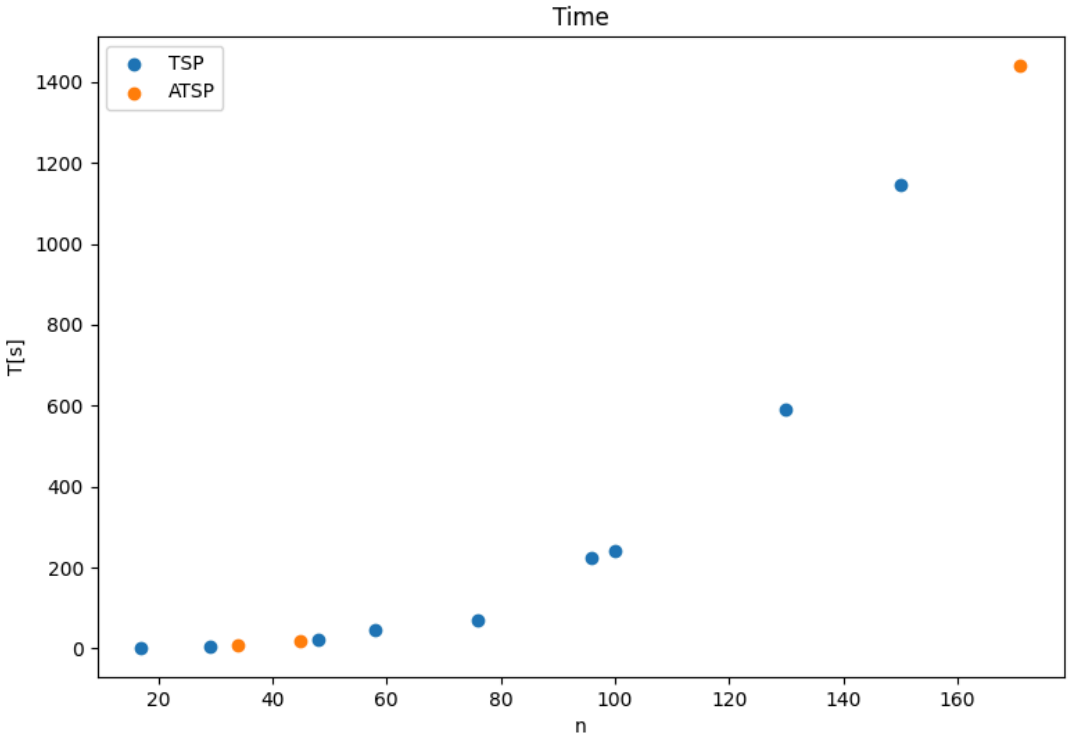
Rysunek 15: Wpływ wielkości instancji na czas dla zoptymalizowanych parametrów



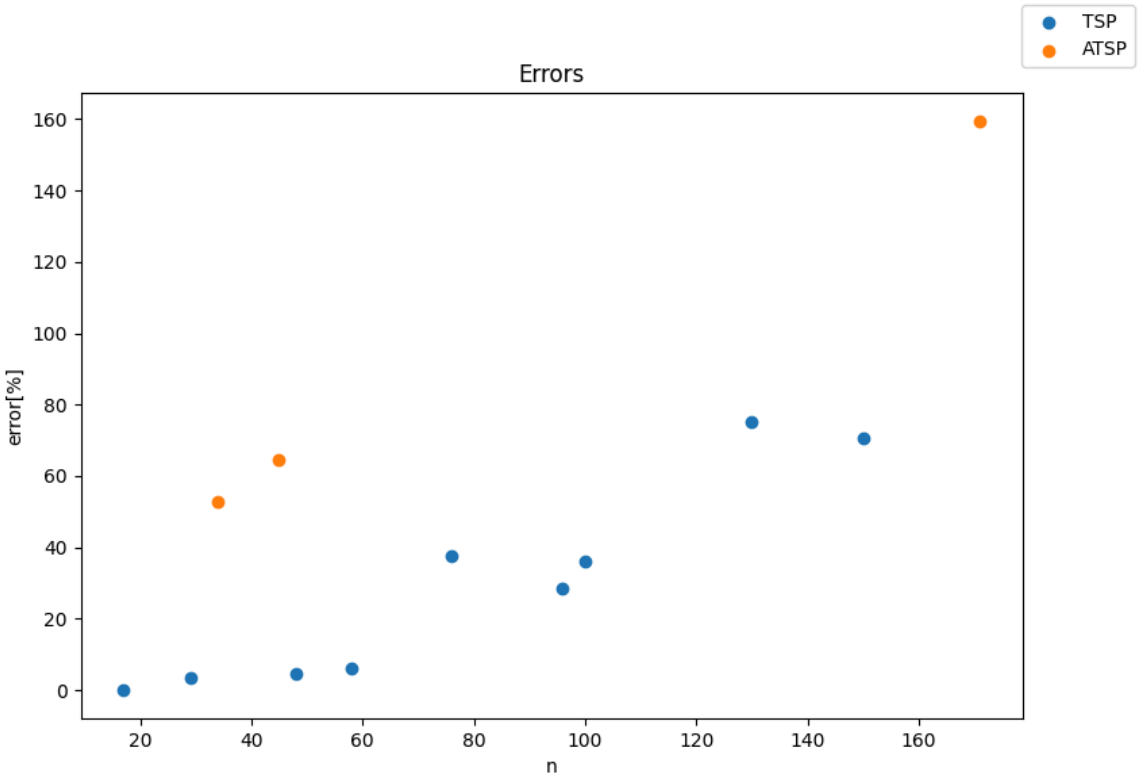
Rysunek 16: Wpływ wielkości instancji na błąd dla zoptymalizowanych parametrów

Wyniki
Wyniki zostały zgromadzone w plikach csv. Pliki zostały dołączone do sprawozdania.

Konfiguracja przez mnie stworzona była zoptymalizowana pod względem instancji symetrycznych. Na wykresach zostało pokazane kilka instancji asymetrycznych w celu pokazania różnicy w działaniu dla tych instancji. Nie są one jednak wykorzystywane do analizowania uzyskanych błędów.



Rysunek 17: Wpływ wielkości instancji na czas

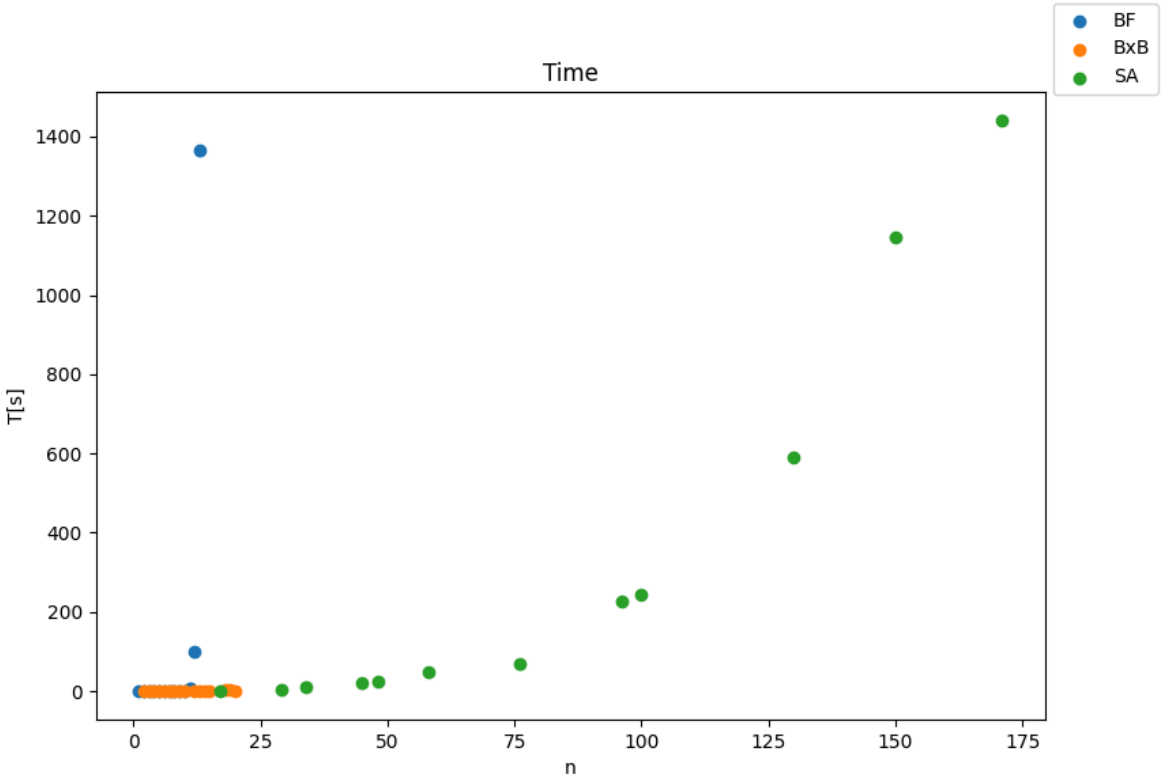


Rysunek 18: Wpływ wielkości instancji na błąd

Wykres zależności czasowej przypomina wykres funkcji n^2, ponieważ pasuje ona do pomiarów uznałem, że algorytm posiada taką właśnie złożoność czasową.

Jak widać na rysunku błąd względny wydaje się rosnąć w wraz ze wzrostem wielkości instancji w sposób podobny do liniowego.

Porównanie z innymi algorytmami



Rysunek 18: Porównanie czasu działania algorytmów

Jak widać na powyższym wykresie algorytm symulowanego wyżarzania pozwala na obliczenie ścieżki dla o wiele większych instancji. Jednak należy pamiętać, że nie zwraca on rozwiązania optymalnego. do 100 wierzchołków w algorytmie SA błąd nie przekracza 50%. Jest on zależny od wielkości instancji.

Wnioski

Udało się zaimplementować algorytm SA, który w rozsądnym czasie(do 10 min) rozwiązywał instancje o wielkości do 100 wierzchołków z założonym błędem poniżej 50% dla symetrycznej instancji problemu Komiwojażera. Został zbadany również wpływ różnych parametrów zarówno pod względem czasu jak i błędu. 100 wierzchołków wydaje się dosyć mała instancją dla tego algorytmu (przypuszczałem że będzie udawało się rozwiązywać problemy do ok. 200-300 wierzchołków) może być to spowodowne suboptymalną optymalizacją algorytmu. Częściowo również może być to spowodowane przez używanie języka python.