Projektowanie Efektywnych Algorytmow
Projekt
22/12/2020

248820 Przemysław Rychter

(4) Simulated annealing

1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu symulowanego wyżarzania w problemie Komiwojażera. Problem komiwojażera (eng. *Travelling salesman problem, TSP)* to zagadnienie polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym.Algorytm symulowanego wyżarzania posiada wiele parametrów, ich dobór ma kluczowe znaczenie, dla działania algorytmu.

2. Metoda

Metoda wywodzi się z metody symulacji inaczej zwanej algorytmem Metropolis – algorytm statycznego symulowania (Monte Carlo) zmian ciała w gorącej kąpieli aż do stanu równowagi termicznej.

1. Losowo generowanie kolejnego stanu ciała ***j*** i określenie jego energii E_j
2. Jeżeli $E_j - E_i < 0$ to ***j*** jest nowym stanem bieżącym, w przeciwnym razie stan j jest akceptowany z pewnym prawdopodobieństwem wynoszącym:

$$\exp\left(\frac{E_i - E_j}{k_B T}\right) \qquad (1)$$

T – temperatura kąpieli
k_B – stała Boltzmana

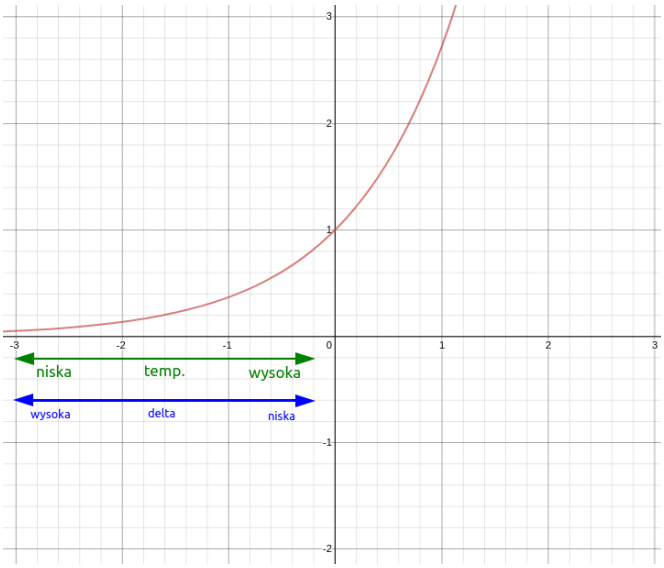
Algorytm symulowanego wyżarzania jest analogiczny do powyższej metody symulacji bazującej na zjawisku fizycznym zwanym wyżarzaniem – jest to podgrzanie materiału a następne jego kontrolowane chłodzenie, podczas takiego procesu cząsteczki mają dużo czasu aby ustawić się w ustrukturyzowany sposób, dzięki temu osiąga się bardziej krystaliczną, symetryczną strukturę.

Prawdopodobieństwo wyboru gorszego rozwiązania w algorytmie symulowengo wyżarzania jest dane wzorem:

$$\exp\left(\frac{\Delta}{T}\right) \qquad (2)$$

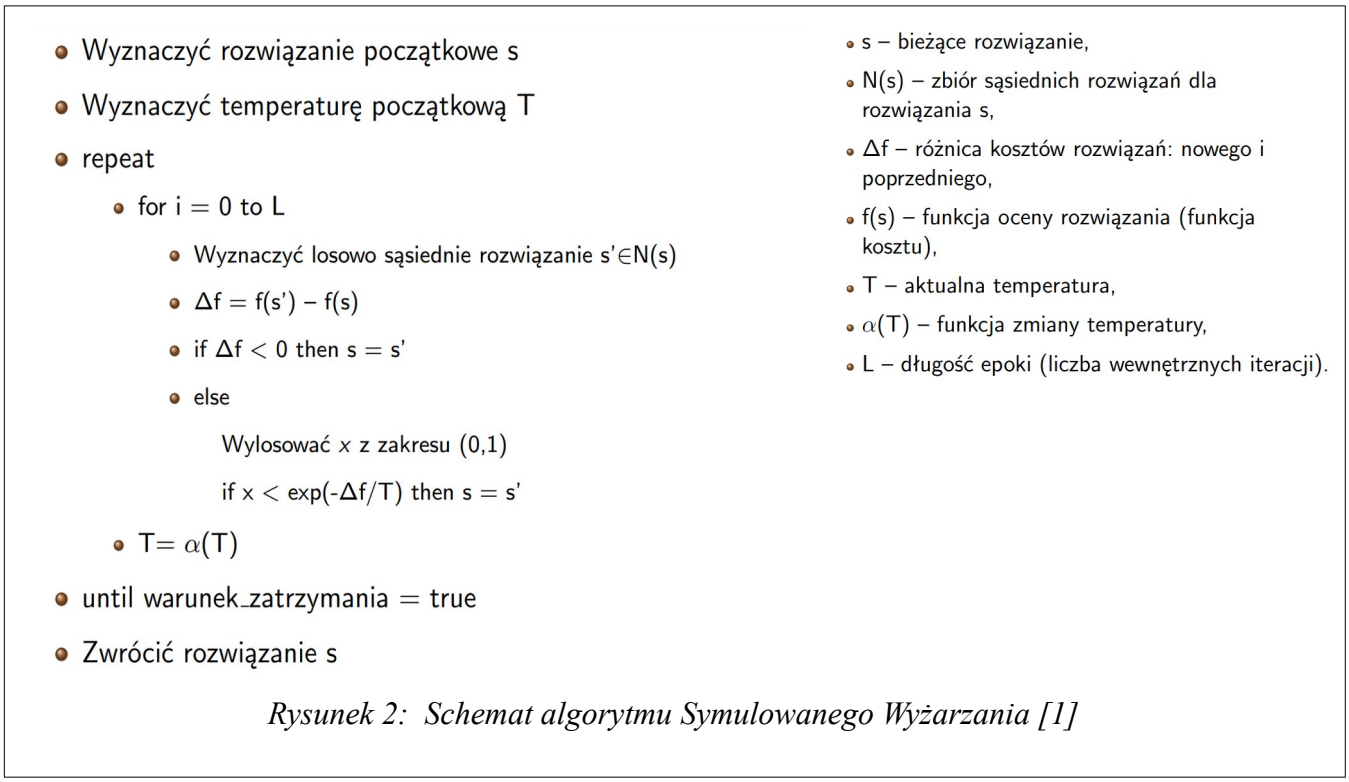
T – temperatura parametr, obiżany z czasem działania algorytmu
Δ – różnica kosztu pomiędzy aktualnym rozwiązaniem a znaleziononym sąsiednim

Można powiedzieć, że algorytm jest rozwinięciem algorytmu przeszukiwania lokalnego z akceptacją (ograniczoną) pogorszenia wartości fukcji celu. Algorytm szuka sąsiedniego rozwiązania do aktualnego, jeżeli sąsiednie rozwiązanie okaże się lepsze, jest ono wybierane jako aktualne, jeżeli jest gorsze to jest może być wybrane jako aktualne z pewnym (malejącym w czasie) prawdopodobieństwem zależnym od temperatury, która w trakcie działania algorytmu maleje. Na początku działania algorytm jest mniej wymagający, więc z większym prawdopodobieństwem akceptuje, gorsze rozwiązania, co pozwala wychodzić z znalezionego minimum lokalnego. W miarę obniżania temperatury, algorytm staje się bardziej wymagający, staję się podobny do przeszukiwania lokalnego, ponieważ prawdopodobieństwo akceptacji gorszego rozwiązania maleje.



Rysunek 1: Wykres przedstawiający wpływ temperatury oraz róznicy w kosztach na prawdopodobieństwo akceptacji gorszego rozwiązania

3. Algorytm



Rysunek 2: Schemat algorytmu Symulowanego Wyżarzania [1]

Parametry algorytmu:

T_o – temperatura początkowa
f(T) – funkcja zmiany temperatury
L – długość epoki, określa jak długo szukane są nowe rozwiązania w danej temperaturze

Inne stotne części algorytmu:

Warunek zatrzymania – algorytm powinien kończyć pracę kiedy nie ma możliwości poprawy rozwiązania a więc w sytuacji w której temperatura jest tak niska, że prawdopodobieństwo wyjścia z minimum lokalnego jest bardzo małe.

Sposób przeglądania sąsiedztwa – określa w jaki sposób przeglądamy przestrzeń rozwiązań (greedy, steepest)

Sposób wboru rozwiązania w sąsiedztwie – w jaki sposób zdefiniowane jest sąsiedztwo (swap, insert, invert)

Sposób wyznaczania rozwiązania początkowego – np. losowo wyznaczony cykl Hamiltona

Sposób wyznacznia temperatury początkowej – np. wybranie stałej wartość przykładowo 100000 jednostek

n – ilość węzłów
T0 – temp początkowa
x0 – cykl początkowy

Szukaj rozwiązań dla kolejnej temperatury, dopóki w 100 kolejnych epokach(temp) aktualne rozwiązanie (x) nie zmieniło się

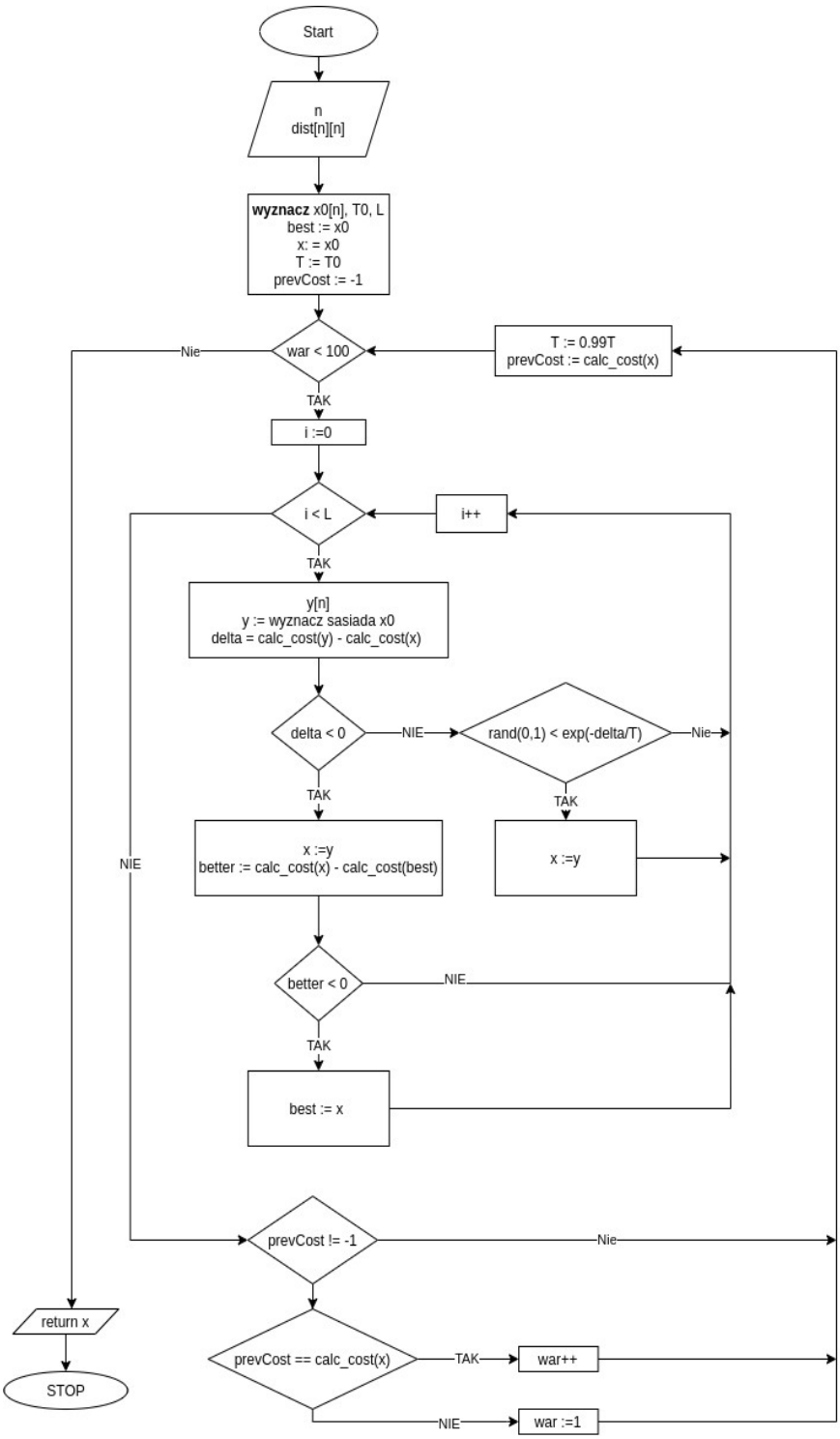
Szukaj kolejnego sąsiada dopóki nie osiągnięto rozmiaru epoki L

wyznacz sąsiada

Jeśli kosz sąsiada lepszy to ustaw sąsiada y jako aktualne rozwiązanie x, jeśli gorszy to ustaw pewnymprawdopodobieństwem

Jeśli rozwiąznie lepsze niż najlepsze znalezione ustaw je jako najlepsze znalezione

Jeśli aktualne rozwiązanie takie samo jak dla wcześniejszej epoki, to zwiększ licznik odpowiadający warunkowi zatrzymania, jeśli nie to ustaw go na 1



Rysunek 3: Schemat blokowy algorytmu symulowanego wyżarzania

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:

- tsp_6_1.txt
- tsp_10.txt
- tsp_12.txt
- tsp_13.txt
- tsp_14.txt
- tsp_15.txt
- tsp_17.txt

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

Algorytm dla powyższych instancji, zwrócił scieżki o koszcie optymalnym

Do badań w tym „dostrojenia" parametrów algorytmu wybrano następujący zestaw instancji:

- m9.atsp
- p43.atsp
- ft53.atsp
- ftv64.atsp
- ftv70.atsp
- gr17.tsp
- gr21.tsp
- bayg29.tsp
- dantzig42.tsp
- berlin52.tsp
- brazil58.tsp
- st70.tsp
- eil76.tsp

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php#instances

Do wykonania ostatecznych badań wybrano następujący zestaw instancji:

- Instancje symetryczne

- gr17.tsp
- gr21.tsp
- bayg29.tsp 2
- dantzig42.tsp
- berlin52.tsp
- brazil58.tsp
- st70.tsp
- eil76.tsp
- gr96.tsp
- kroB100.tsp
- pr107.tsp
- gr120.tsp
- bier127.tsp
- pr136.tsp
- pr144.tsp
- pr152.tsp
- brg180.tsp
- rat195.tsp
- gr202.tsp
- gr229.tsp
- gil262.tsp
- a280.tsp
- pr299.tsp
- linhp318.tsp
- rd400.tsp
- fl417.tsp

- Instancje asymetryczne

- m9.atsp
- br17.atsp
- ftv33.atsp
- ftv38.atsp
- p43.atsp
- ft53.atsp
- ftv64.atsp
- ftv70.atsp
- ftv170.atsp
- rbg323.atsp
- rbg358.atsp
- rbg403.atsp
- rbg443.atsp

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php#instances

5. Procedura badawcza

W celu dostrojenia algorytmu zbadałem wpływ sposobu wyznaczania parametrów, tempa chłodzenia, sposobu wyboru rozwiązania w sąsiedztwie, rozmiaru epoki na czas wykonania oraz błąd względem rozwiązania optymalnego algorytmu.

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

Procedura badawcza polegała na uruchomieniu programu sterowanego plikiem inicjującym .ini format pliku:

nazwa_instancji liczba_wykonań rozwiązanie_optymalne
...
nazwa_instancji liczba_wykonań rozwiązanie_optymalne
nazwa_pliku_wyjściowego

Instancje testowe pochodziły ze stron:

- http://elib.zib.de/pub/mp-testdata/tsp/tsplib/atsp/index.html (**ATSP**)
- http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html (**TSP)**
- http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Instancje z dwóch pierwszych powyższych adresów zostały pobrane ze strony:

- http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php

Poniżej przykładowa treść plik „conf.ini” (dla badań mających na celu dostrojenie algorytmu).

```
m9.atsp 10 215
br17.atsp 10 39
ftv33.atsp 10 1286
ftv38.atsp 10 1530
p43.atsp 10 5620
ft53.atsp 10 6905
ftv64.atsp 10 1839
ftv70.atsp 10 1950
gr17.tsp 10 2085
gr21.tsp 10 2707
bayg29.tsp 20 1610
dantzig42.tsp 15 699
berlin52.tsp 10 7542
brazil58.tsp 10 25395
st70.tsp 10 675
eil76.tsp 10 538
Opis_jak_dostrojone.csv
```

Każda instancji rozwiązywana była zgodnie z liczbą jej wykonań, np. ftv64.atsp wykonana została 10 razy. Do pliku wyjściowego Opis_jak_dostrojone.csv zapisywane były informacje o instancji: jej nazwa, liczba wykonań algorytmu oraz ilość wierzchołków. Następnie zapisywane były czasy wykonań algorytmu dla tej instancji. Plik wyjściowy zapisywany był w formacie csv. Poniżej przedstawiono fragment zawartości przykładowego pliku wyjściowego.

```
ft53.atsp Reps: 10 Nodes:  53
6604922
5205713
6375854
6400286
7179492
5108533
6247084
5229969
7029841
6305316
ftv64.atsp Reps: 10 Nodes:  65
```

Na standardowe wyjście dla każdego powtórzenia algorytmu zwracana była wartość błędu, znalezionego rozwiązania względem optymalnego podanego w pliku „conf.ini” oraz średnia wartość tego błedu dla każdej z instancji. Na koniec wyjścia zwracane było średnie wartości błedu w koleności odpowiadającej badanym instancją. Poniżej fragment pliku do którego stdandardowe wyjście było przekierowywane w celu zapisania danych o błędach i ich analizie.

```
_____________________________________________
eil76.tsp               nodes: 76
blad: 9.47955 %
blad: 9.29368 %
blad: 9.8513 %
blad: 10.5948 %
blad: 9.8513 %
blad: 7.0632 %
blad: 14.4981 %
blad: 9.8513 %
blad: 11.1524 %
blad: 6.3197 %
sredni blad (w %) dla aktualnej instancji ponizej
9.79554

_____________________________________________
SREDNIE BLEDY w % DLA KOLEJNO WSZYSTKICH INSTACJI Z conf.ini
0
0.339858
17.5091
19.3692
21.3436
5.45652
11.9014
9.79554
```

Pomiary zostały wykonane na platofrmie sprzętowej:

- procesor: Intel® Core™ i5-8250U CPU 1.60GHz × 8
- pamięć operacyjna: 31,2 GB
- system operacyjny: z rodziny Linux - Ubuntu 20.04.1 LTS 64-bit

Pomiary czasu zostały wykonane za pomocą biblioteki std::chrono [6].

```
Sa sa = Sa();
auto start = std::chrono::high_resolution_clock::now();

solution_cost = sa.calculate(vertices, distances);

auto stop = std::chrono::high_resolution_clock::now();
sa.~Sa();

auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop - start);
file_out << duration.count() << std::endl;
```

Rysunek 4: Fragment kodu przedstawiający sposób pomiaru czasu wykonania algorytmu

Wyniki zostały opracowane w programie LibreOffice Calc.

6. Parametry początkowe oraz dostrojenie algorytmu

Sposób przegląda sąsiedztwa – greedy
Sposób wyboru rozwiązania w sąsiedztwie – 2-zamiana (swap)

Rozwiązania początkowe

Początkowa ścieżka jest wyznaczana losowo. Nie wydaje się, aby stosowanie innej metody miało znaczący wpływ na osiągi algorytmu.

Długość epoki

Długość epoki (ilość iteracji algorytmu dla jednej temperatury) według [2] powinna być proporcjonalna do rozmiaru sąsiedztwa aktualne rozwiązania. W mojej implementacji zastosowałem sąsiedztwo typu swap, w tego typu sąsiedztwie sąsiada można wybrać na n po 2 sposobów (wybieramy dwa wiedzchołki do zamiany):

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

Temperatura początkowa

Ideą algorytmu jest schładzanie temperatury dla kolejnych epok. Temperatura wpływa na prawdopodobieństwo akceptacji gorszego rozwiązania. Zbyt wysoka temperatura prowadzi tylko do straty czasu, ponieważ jej początkowe schładzanie nie będzie zmieniało prawd. akceptacji. Zbyt niska temperatura początkowa może spowodować, że zbyt szybko algorytm utknie w minimum lokalnym, jest to odejście od idei algorytmu i utracenie korzyści powolnego schładzania. Zatem temperatura powinna być wystarczająco wysoka, aby na początku prawdopodobieństwo akceptacji gorszego rozwiązania było bliskie jeden. Zatem możemy wstawić liczbę bliską jeden do wzoru na prawd. akceptacji:

$$\exp\left(\frac{\Delta_{obliczone}}{T_{początkowa}}\right)=0.99 \qquad (3)$$

Po określeniu delty dla równania (3) jedyną niewiadomą zostanie temperatura, więc poprzez przekształcenia otrzymamy wzór na temp. początkową. Delta to różnica kosztów pomiędzy aktualnym a sąsiednim rozwiązaniem. Na początku temperatura powinna pozwalać na przejścia nawet do o wiele gorszego rozwiązania. Dlatego interesują nas rożnice pomiędzy sąsiadami, aby znaleźć adekwatną róznice można spróbkować (wylosować) kilka tysięcy cykli z przestrznii rozwiązań, wylosować do każdej próbki sąsiada a następnie policzyć rożnicę w kosztach, a ostatecznie za delte przyjąć największą obliczoną rożnice. Można też policzyć średnią ze wszystkich rożnic – właśnie tak wyznaczyłem delte do wzoru (3) w mojej implementacji.

Warunek zatrzymania

Algorytm powinien się zatrzymać kiedy temperatura jest tak niska, że prawdopodobieństwo przejścia do rozwiązania gorszego jest bardzo małe, wtedy szansa na wyjście z minimum lokalnego jest mała i w konsekwencji szansa znalezienia lepszego rozwiązania również jest mała i ciągle maleje. Trudno określić dokładną wartość temp. końcowej, dlatego w mojej implementacji algorytm zakończy działanie jeżeli dla kolejnych 100 epok, koszt aktualnego rozwiązania nie poprawi się.

Funkcja chłodzenia

Zaimplementowany został geometryczny schemat chłodzenia więc funkcja zmiany temp. wygląda następująco:

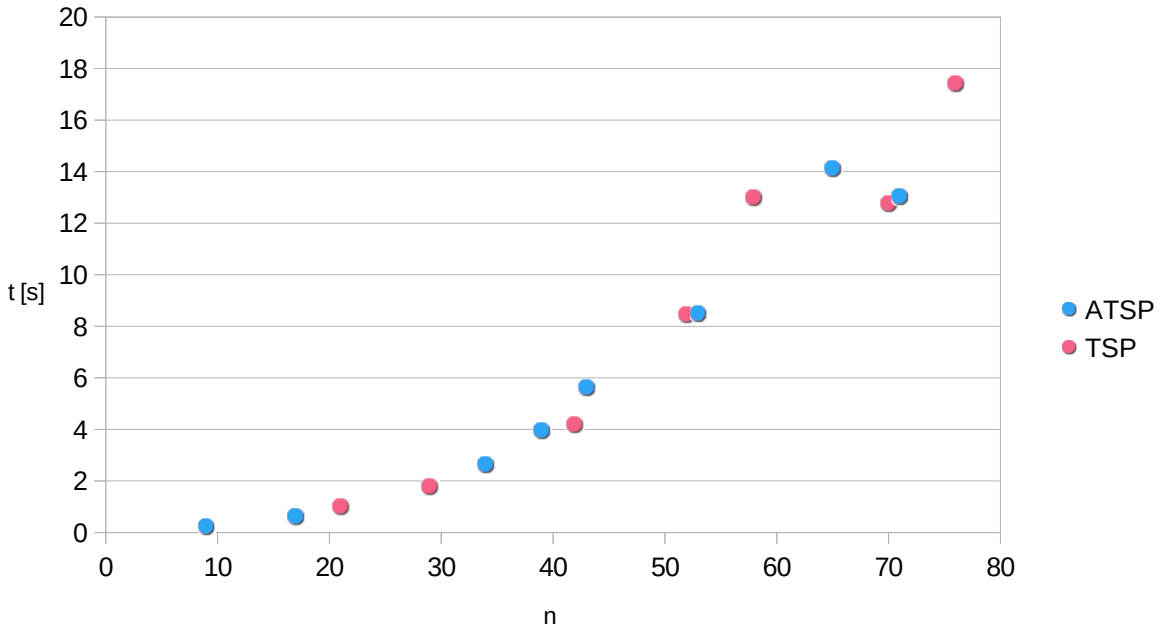
$$f(k)=\alpha^{(k)}T_{(początkowa)} \qquad (4)$$

k – *numer epoki* $k \in N$

α – *paramter informujący o szybkości schładzania* $\alpha \in (0,1)$

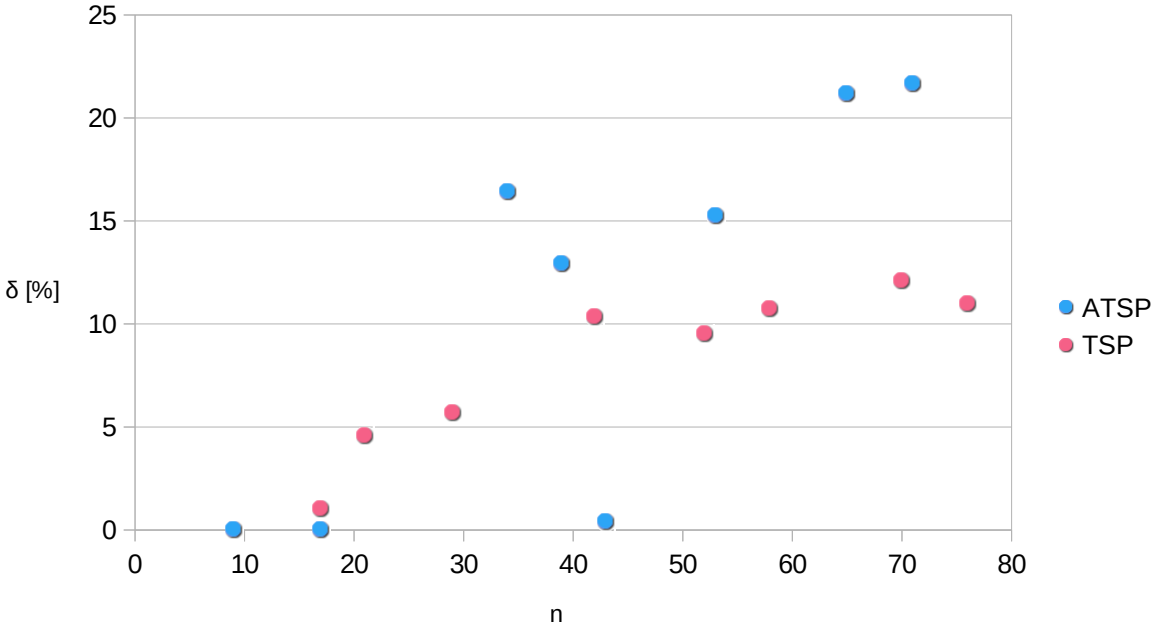
Im paramter jest bliżej 1 schłodzenie będzie przebiegać wolniej. Według [1] powinien on należeć do przedziału (0.82, 0.98). Ideą algorytmu jest powolne schładzanie, w mojej implementacji przyjąłem $\alpha = 0.99$

Wszystkie badania przeprowadzone w tej sekcji używają wymienionych instancji „do dostrojenia". Dla przyjętych wyżej parametrów i sposobów ich wyznaczania przeprowadzono podstawowe badanie czasu wykonywania algorytmu oraz błedu, względem wielkości instancji - rysunek 5 oraz 6.



Rysunek 5: Wpływ wielkości instancji n na czas rozwiązania (A)TSP algorytmem SA z podstawowymi parametrami

Jak widać na rysunku 5 czas wykonywania algorytmu, jest podobny dla instancji symetrycznych i asymetrycznych, czas wzrasta wraz z wzrostem wielkości instancji tak samo dla instancji TSP oraz ATSP.

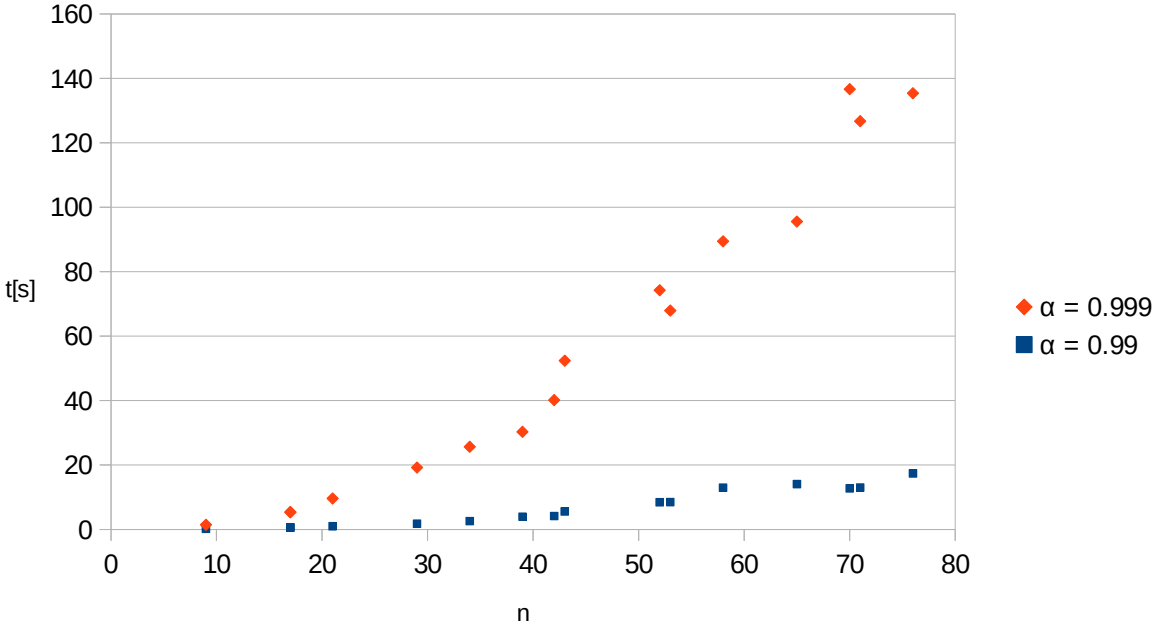


Rysunek 6: Wpływ wielkości instancji n na błąd względny δ uzyskanego rozwiązania algorytmem SA z podstawowymi parametrami

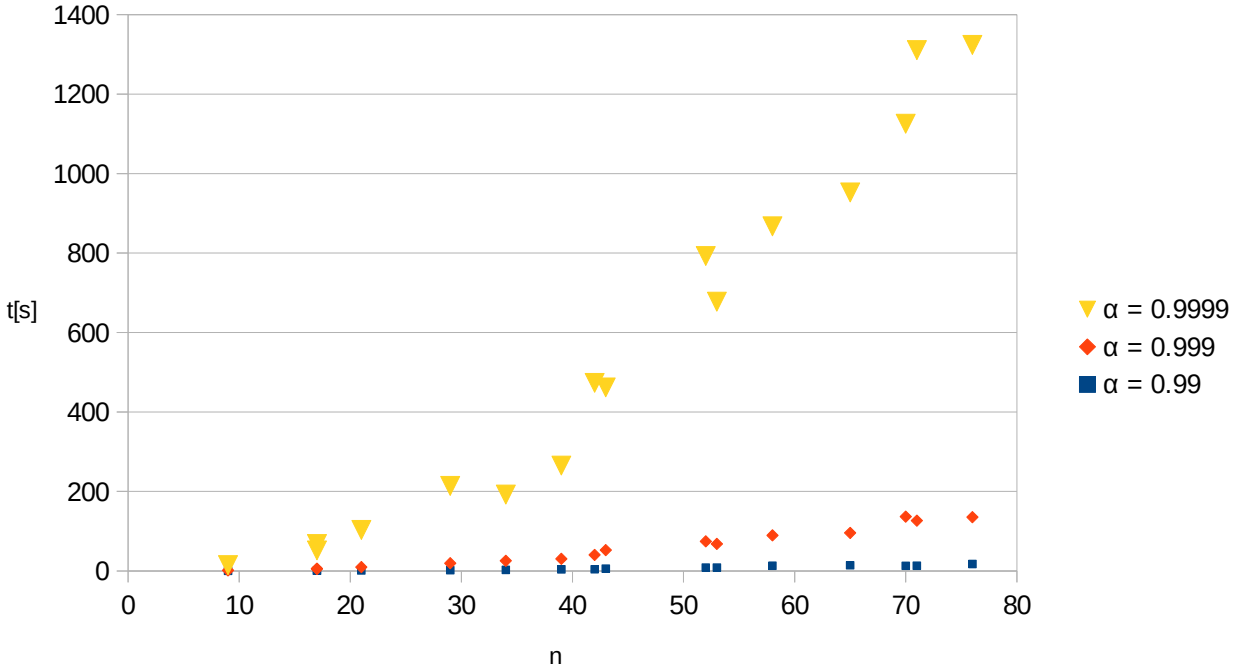
Wykres błędu względnego pokazuje, że dla większych instancji asymetrycznych błąd jest o wiele większy niż dla takiej samej wielkości instacji symetrycznych np. dla instancji o 70 wierzchłkach błąd jest prawie 2 większy.

6.1 Zmiejszenie tempa chłodzenia

Szybkość chłodzenia metodą geometryczną możemy modyfikować podany wcześniej parametrem α. Im jest on bliższy 1 tym chłodzenie będzie przebiegać wolniej. W celu uzyskaniania mniejszych błedów zmieniłem wartość α na 0.999 oraz 0.9999.



Rysunek 7: Porównanie wpływu szybkości chłodzenia na czas uzyskanego rozwiązania (A)TSP algorytmem SA



Rysunek 8: Porównanie wpływu szybkości chłodzenia na czas uzyskanego rozwiązania (A)TSP algorytmem SA

Czas wykonania algorytmu wzrasta około 10-krotnie wraz z „dodawaniem” 9 (rysunek 8).

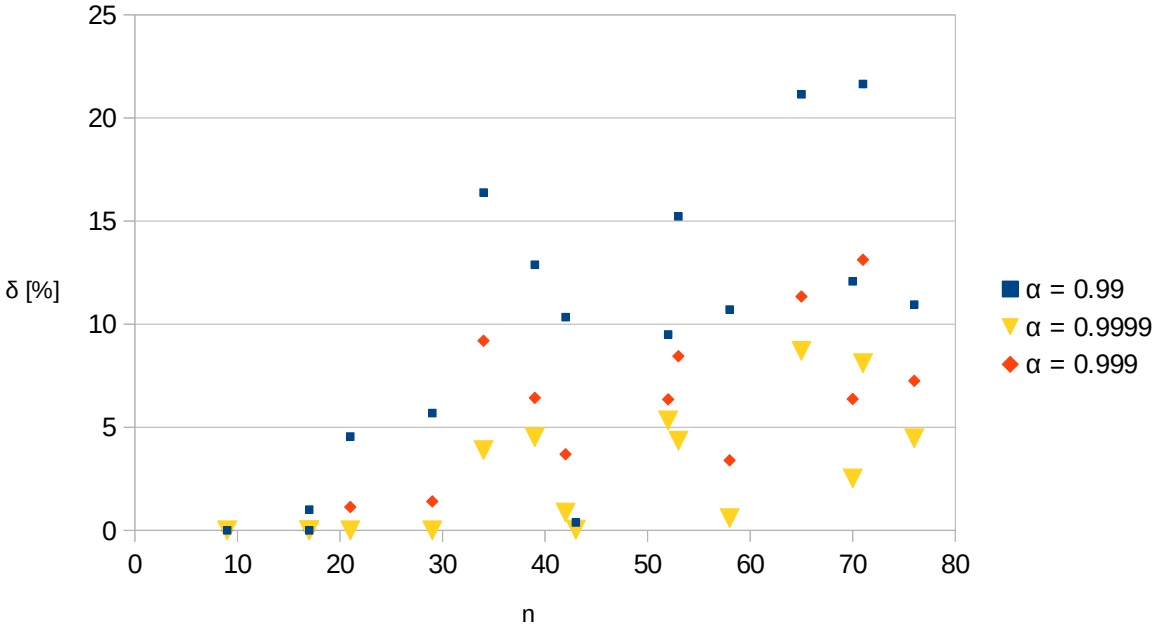


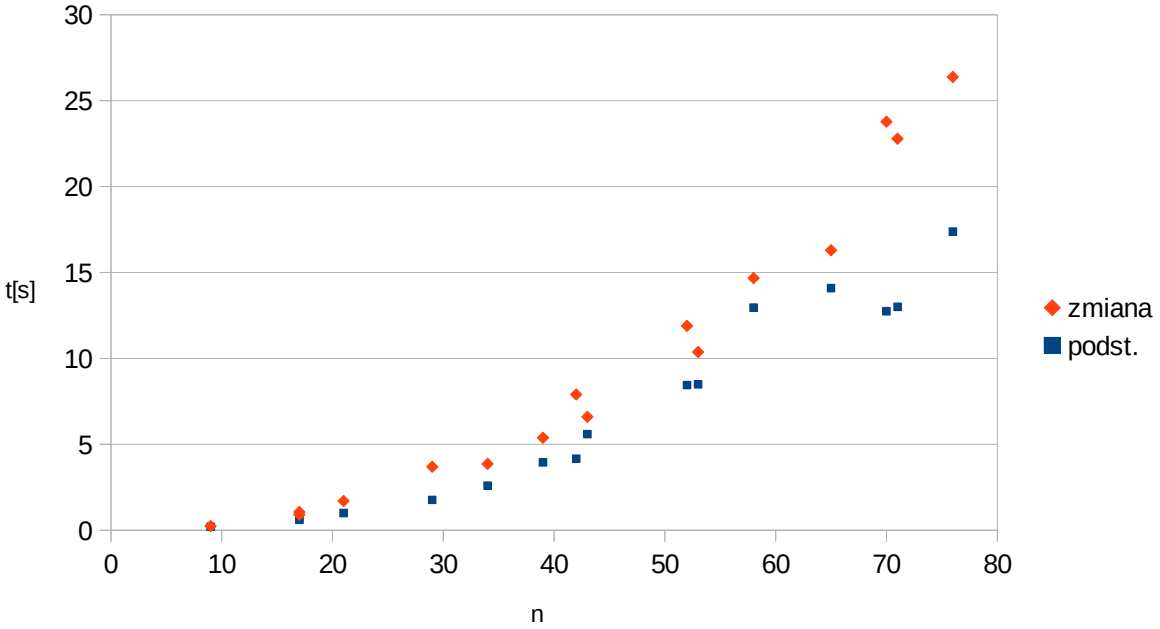
Figure 9: Porównanie wpływu szybkości chłodzenia na błąd względny δ uzyskanego rozwiązania (A)TSP algorytmem SA

Natomiast błądy w zależności od instancji maleją około 2-krotnie (rysunek 9).

Jak wynika z wykresu z rysunku 9. Dla każdej instancji udało się uzyskać błąd około 2 razy mniejszy wraz z zmiejszeniem tempa chłodzenia, jest to znacząca poprawa, aczkolwiek czas wykonywania (rysunek 8) wzrasta aż około 10 krotnie, nie jest to zadowalające, ponieważ wyniki chciałbym otrzymywać w krótszym czasie. Finalnie dopóki nie zmodyfikujemy innych paramterów, zmiejszenie szybkości chłodzenia (zwiększające liczbę epok) nie jest akceptowalne ze względu na drastyczne wydłużenie czasu działania.

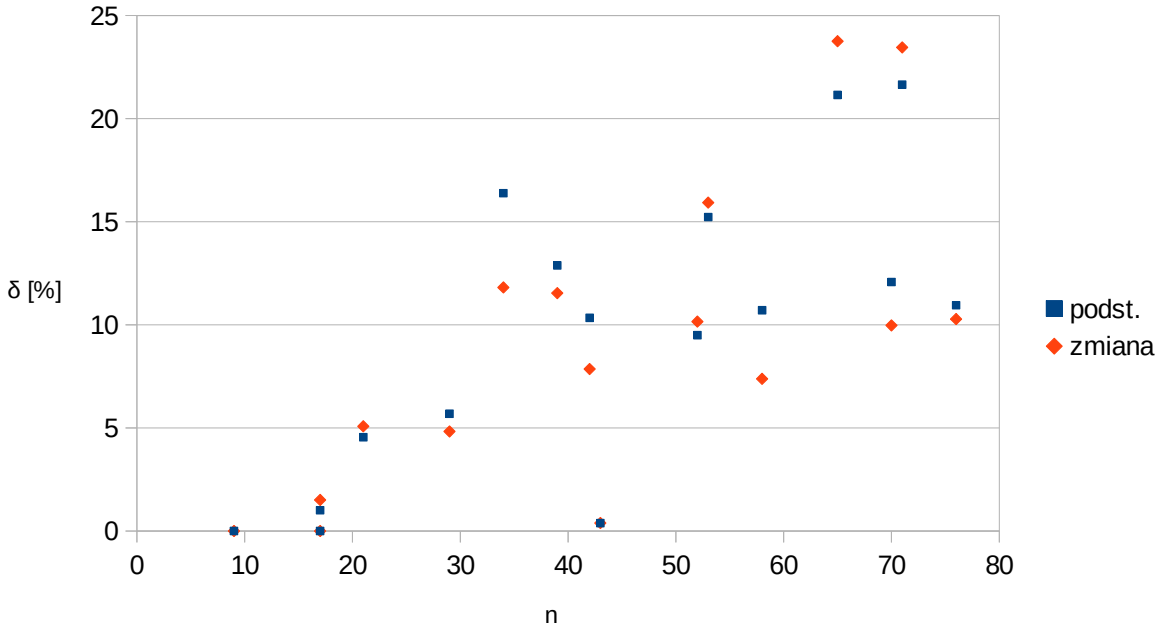
6.2 Zmiana doboru temperatury początkowej i warunku końcowego

W początkowym algorytmie temperatura początkowa jest wybierana na podstawie przestrzeni rozwiazań instancji. Warunek zatrzymania jest spełniony jeśli dla 100 epok bieżące rozwiązanie nie zmieni się - temperatura jest tak niska, że nie pozwala uciec z minimum lokalnego. Podglądając działanie algorytmu wyznacza on wartość temp. początkową zależnie od instancji na kilka do kilkadziesiąt tysięcy jednostek. Natomiast temperatura końcowa waha się między wartościami kilkanaście a setne części jednostek. W tym badaniu ustaliłem temperaturę początkową na 100000 jednostek, a warunek zatrzymania zmieniłem na spadek temperatury poniżej 0.01 jednostki.



Rysunek 10: Porównanie wpływu sposobu wyboru T0 oraz war. zatrzymania na czas uzyskanego rozwiązania (A)TSP algorytmem SA

Wykres na rysunku 10 pokazuje że ustalenie stałej temperatury początkowej oraz zmiana warunku zatrzymania spowodowały wzrost czasu wykonania algorytmu, wzrost ten jest tym większy im większy rozmiar instancji.

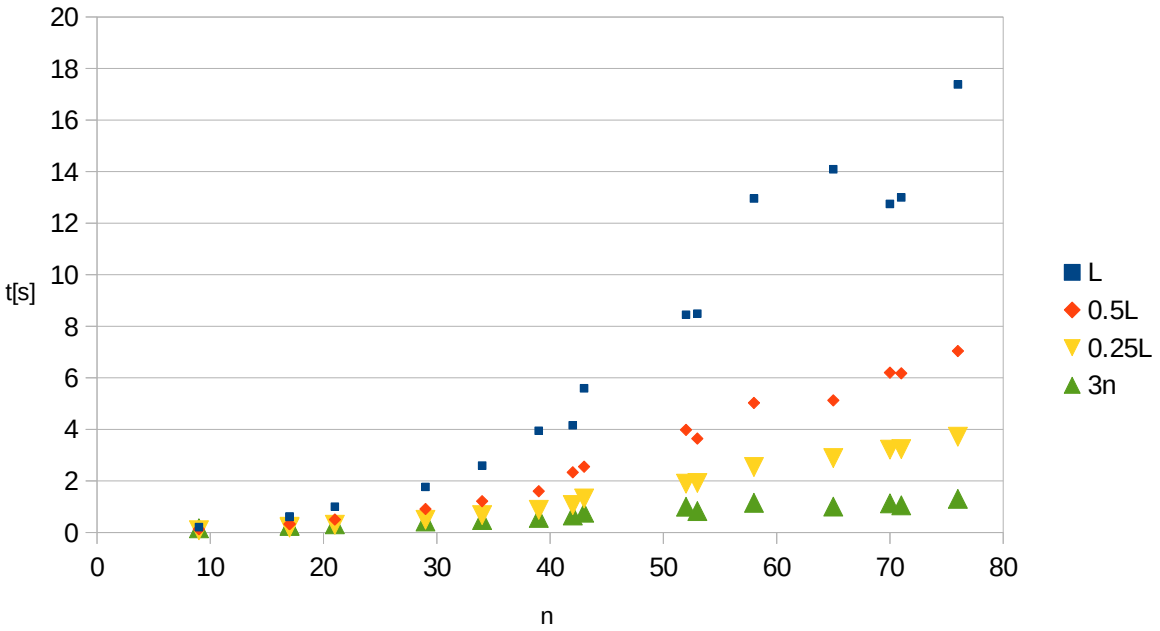


Rysunek 11: Porównanie wpływu sposobu wyboru T0 oraz war. Zatrzymania na błąd względny δ uzyskanego rozwiązania (A)TSP algorytmem SA

Zmiana parametrów znacząco nie zmieniła rozmiarów błędów, w zależności od instacji, wzrosły one lub zmałały. Badanie pokazuje, że wyznaczanie temp. początkowej w wersji podstawowej algorytmu jest lepsze (czas wykonania algorytmu), i adekwatne do rozmiaru instancji. Zmieniony warunek końcowy prawdopodobnie tylko wydłużył czas działania algorytmu kiedy już jest on „zablokowany” w minimum lokalnym.

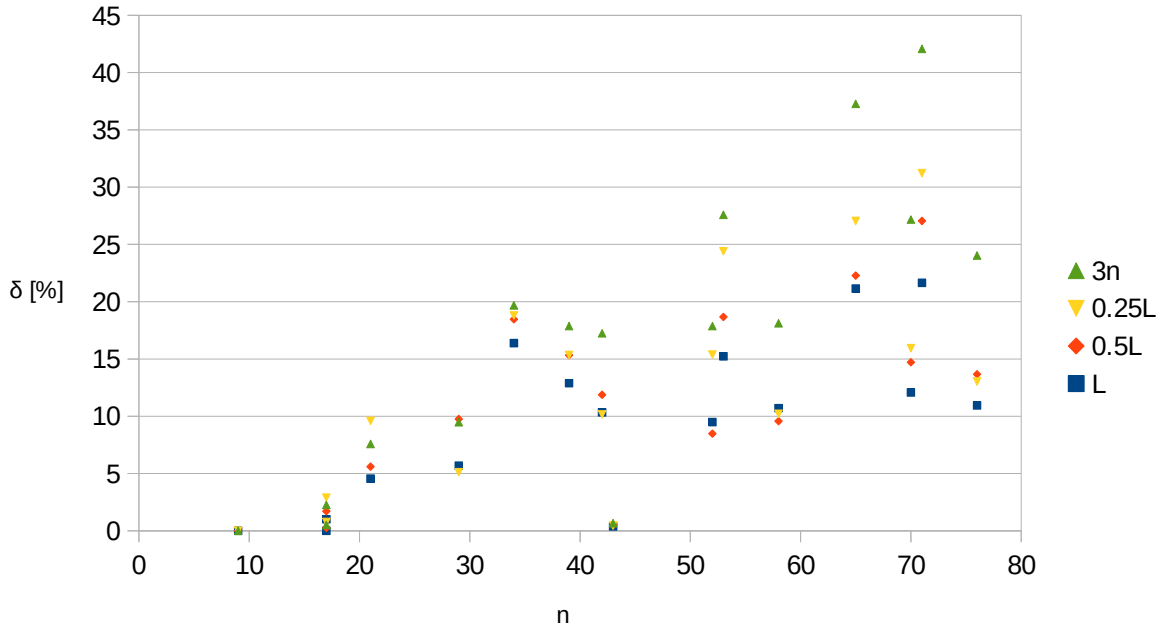
6.1 Zmiana długości epoki

W celu przyspieszenia działania algorytmu można zmiejszyć dlugość epoki np. przyjąć inny wzór jej obliczania, w tej implementacji długość epoki L określona została jako rozmiar sąsiedztwa typu swap. Zmodyfikuje długość epoki podaną wcześniej we wzorze (2) mnożać ją przez 2, 0.5, 0,25, oraz przujmując długość epoki równą 3n.



Rysunek 12: Porównanie wpływu długości epoki na czas rozwiązania (A)TSP algorytmem SA

Jak wynika z powyszego wykresu (rysunek 12) zmiejszanie długości epoki daje proporcjonalne zmiejszenie czasu wykonania algorytmu



Rysunek 13: Porównanie wpływu długości epoki na błąd względny δ uzyskanego rozwiązania (A)TSP algorytmem SA

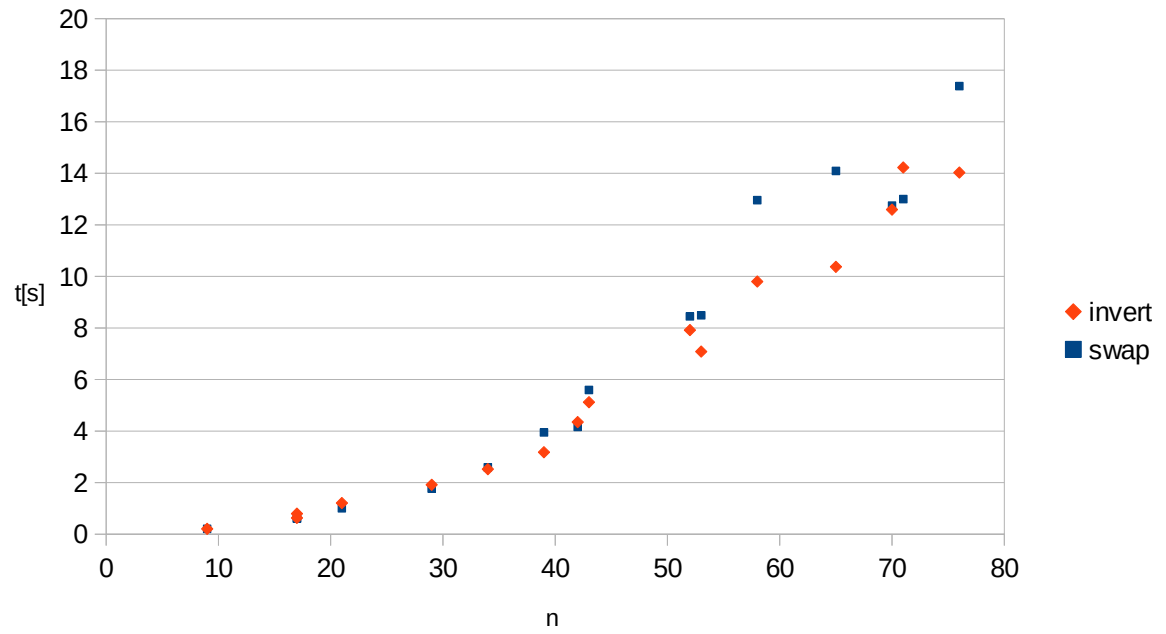
Na powyższym wykresie (rysunek 13) widać że im mniejsza długość epoki tym większy bład, aczkolwiek wartość błedu nie rośnie proporcjonalnie.

Wniosek z 2 powyższych wykresow jest taki że zmniejszenie dlugosci epoki do np. 0.25 początkowej lub 3n powoduje znaczne (proporcjonalne) przyspieszenie dzialania algorytmu, jednocześnie powodując niewielki wzrost błędu.

Bazujac na dotychczacowych wynika wydaje się ze dobrym pomyslem w późniejszych badniach będzie ustalenie wielkości epoki na wartosc mniejszą, natomiast parametru alfa odpowiedzialnego za szybkosc schladzania zbliczenie do 1.

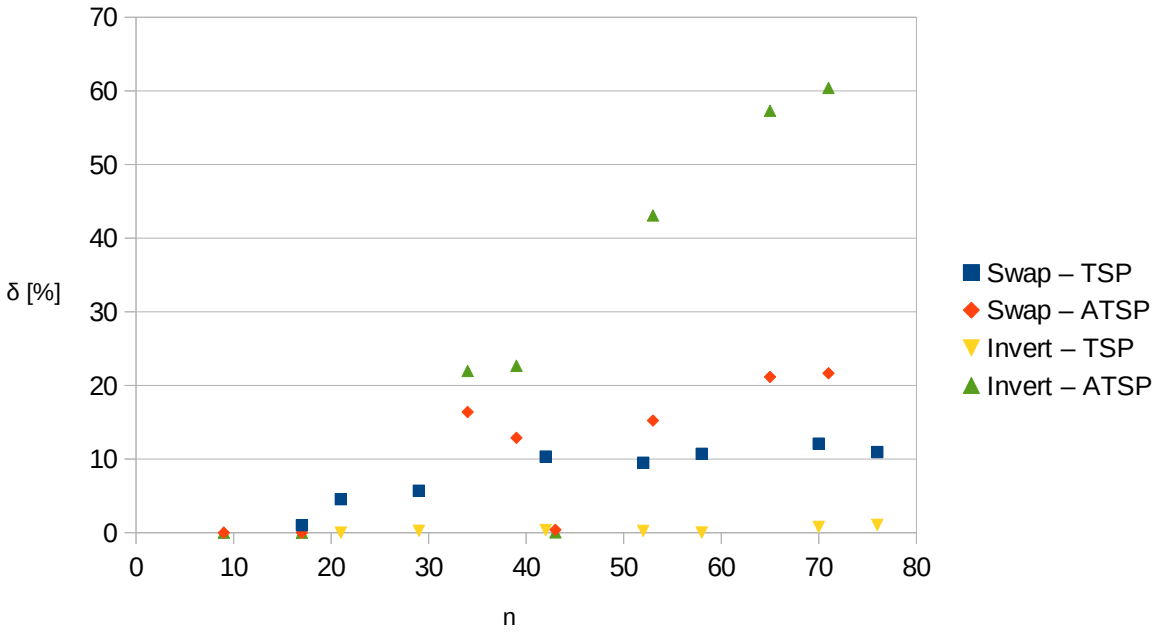
6.3 Zmiana wyboru rozwiązania w sąsiedztwie

W tym badaniu zmianiłem sposób wyboru rozwiązania w sąsiedztwie z typu *swap* na *invert.*



Rysunek 14: Porównanie wpływu typu sąsiedztwa w jakim szukane jest następne rozwiązanie na czas rozwiązania (A)TSP algorytmem SA

Jak widać z wykresu na rysuku 14 sposób wyboru sąsiedniego rozwiązania (typ sąsiedztwa) nie wpływa znacząco na czas wykonywania algorytmu.

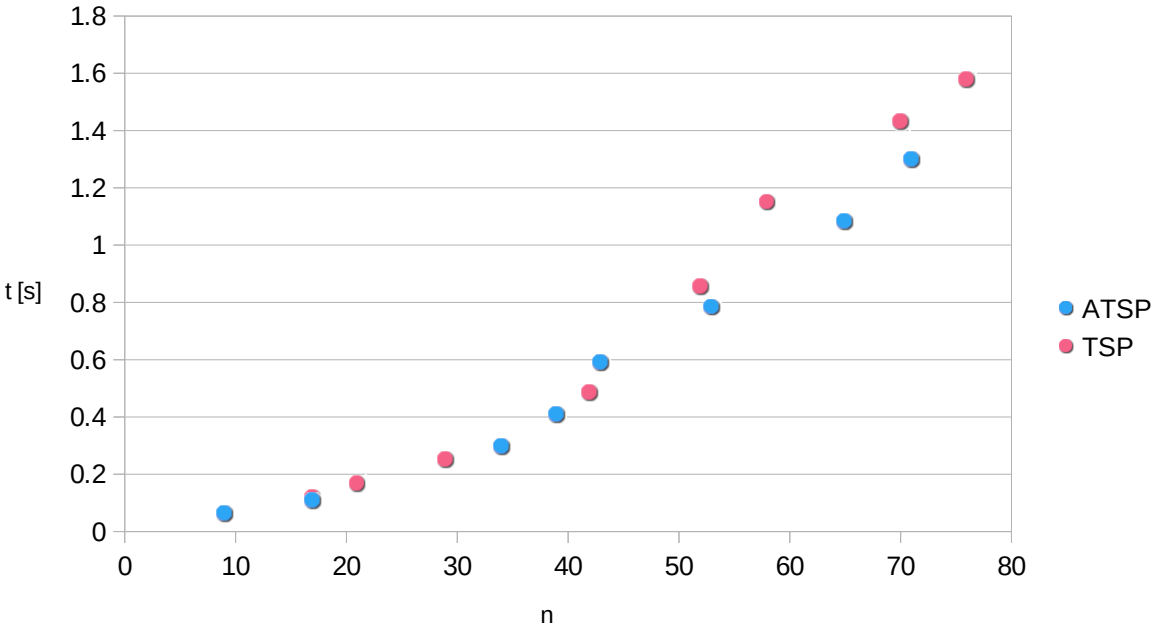


Rysunek 15: Porównanie wpływu typu sąsiedztwa w jakim szukane jest następne rozwiązanie na błąd względny δ uzyskanego rozwiązania (A)TSP algorytmem SA

Na rysunku 15 widzimy jak zastosowanie sasiedztwa typu *invert* wplynęło na błędy. Dla instancji ATSP znacząco wzrosły w porównianiu do sąsiedztwa typu *swap*. Natomiast dla instancji TSP błedy drastycznie zmalały. Jeżeli naszym celem jest rozwiązywanie instancji symetrycznych, powinniśmy przeszukiwać sąsiedztwo typu *invert*. Jeżeli chcemy rozwiązywać instancje asymetryczne lepiej zastosować sąsiedztwo typu *swap*.

6.5 Najlepsza uzyskana konfigurajca (TSP)

Bazując na wcześnieszych badaniach, wykonałem następne, manipulując wcześniej zbadanymi opcjami, skupiając się na instancja symetrycznych, „najlepsze” wyniki - bardzo krótki czas, bardzo mały błąd udało mi się uzyskać dla podstawowej wersji algorytmu ze zmienionym typem wyboru rozwiązania w sąsiedztwie – zamiast swap ***invert***, oraz ze zmienioną długością epoki z *n(n-1)/2 na* ***n(n-1)/2 * 0.1.*** Poniżej uzyskane wyniki, dla instancji do „dostrojenia”.



Rysunek 16: Wpływ wielkości instancji n na czas rozwiązania (A)TSP algorytmem SA z „najlepszymi” dla TSP parametrami

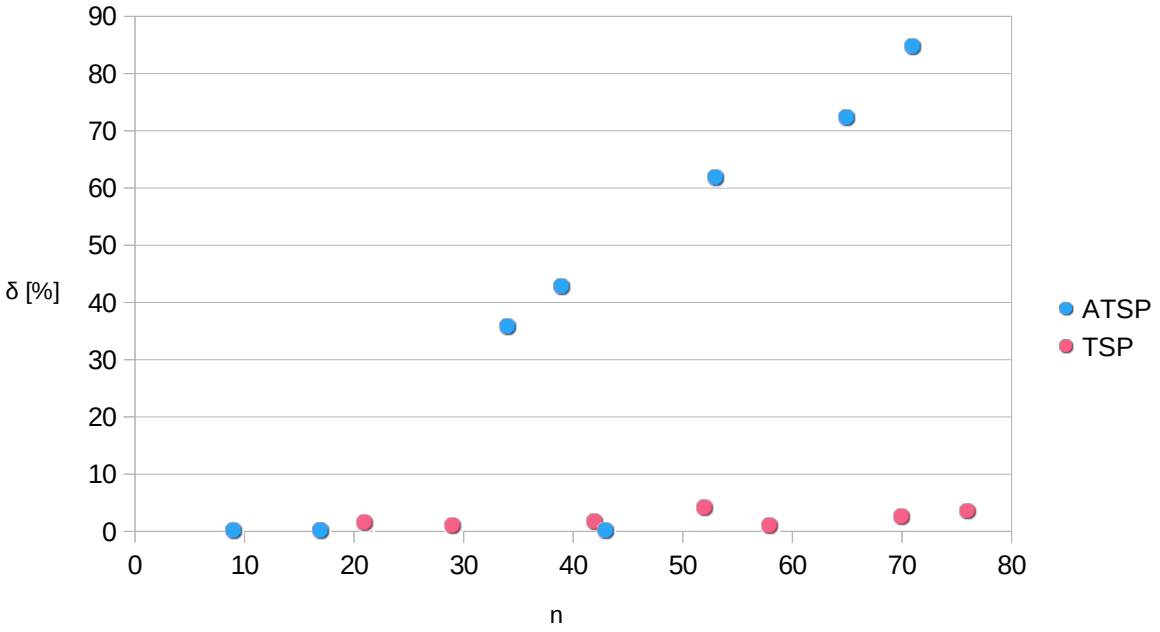


Figure 17: Wpływ wielkości instancji n na błąd względny δ uzyskanego rozwiązania algorytmem SA z „najlepszymi” dla TSP parametrami

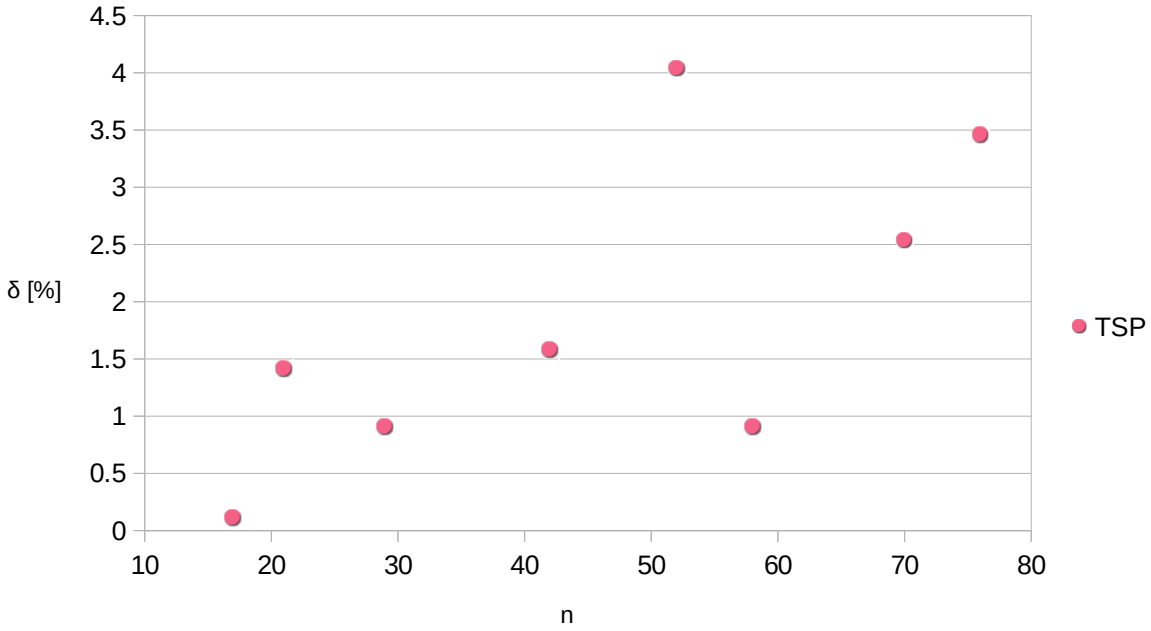


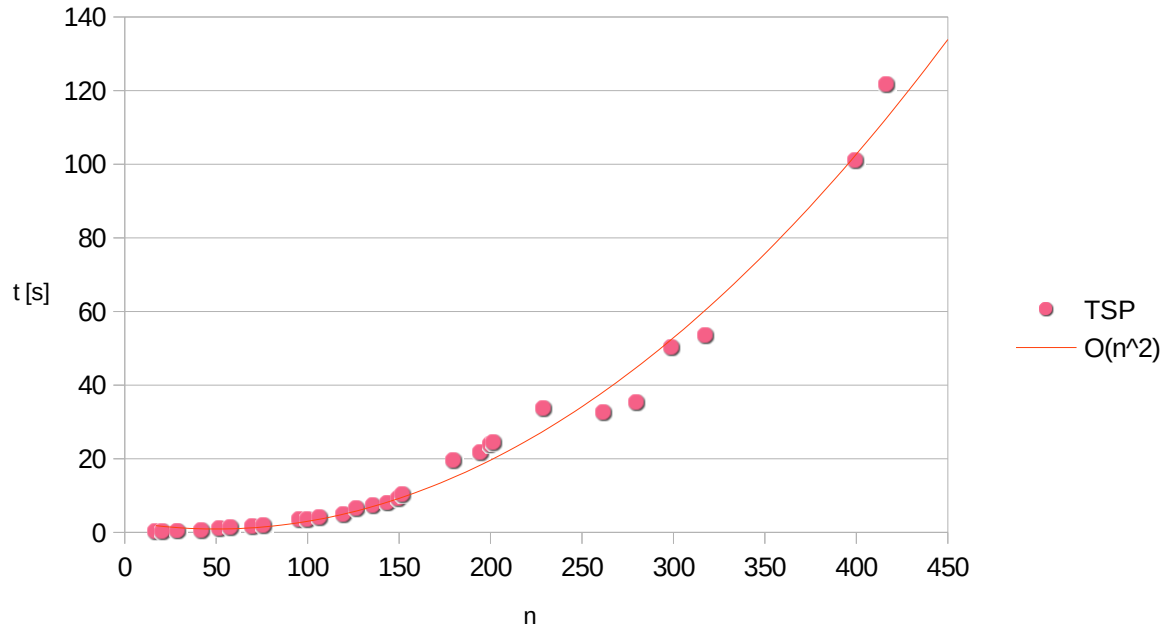
Figure 18: Wpływ wielkości instancji n na błąd względny δ uzyskanego rozwiązania algorytmem SA z „najlepszymi” dla TSP parametrami (wykres bez ATSP)

6. Wyniki

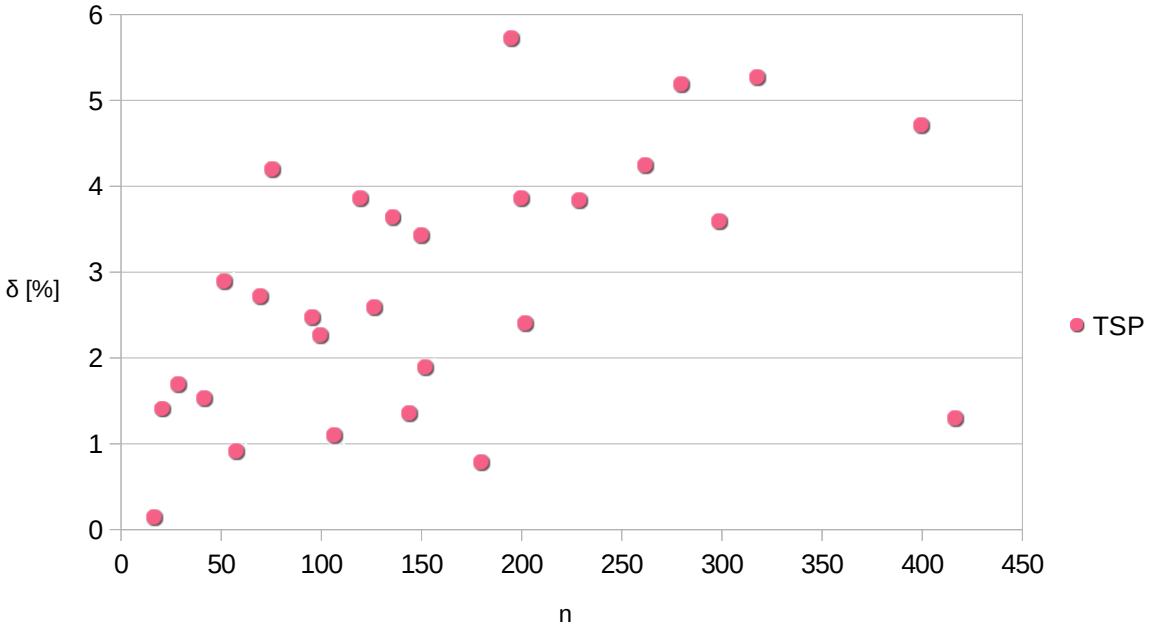
Wyniki zgromadzone zostały w plikach csv :

Pliki został dołączone do raportu i znajdują się na dysku Google pod adresem https://drive.google.com/drive/folders/1yHS-PS9DVc7rIv4o933_UdkAuBjO8zVU.

Ostateczne wyniki dla wcześniej podanej „najlepszej" konfiguracji dla TSP. W ostatecznym badaniu pominięte zostały instancję asymetryczne, tak jak pokazały wcześniejsze badania: sposób wyboru sąsiada typu swap, mniejsze tempo chłodzenia, oraz większa długość epoki, prowadzą do niższych błedów kosztem czasu, i takie ustawienia należało by przyjąć w celu uzyskania niskiego błędu dla instancji ATSP. Poniżej rezultaty dla TSP.



Rysunek 19: Wpływ wielkości instancji n na czas rozwiązania TSP algorytmem SA w ostatecznej konfiguracji

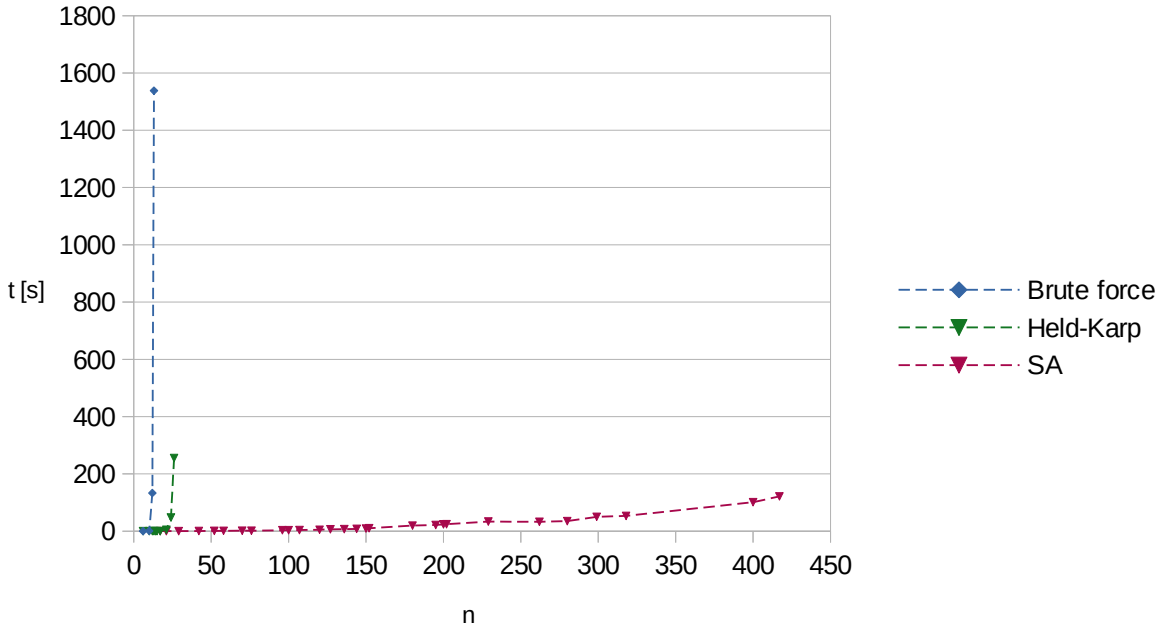


Rysunek 20: Wpływ wielkości instancji n na błąd względny δ uzyskanego rozwiązania algorytmem SA w ostatecznej konfiguracji

Na wykresie zależności czasu wykonania algorytmu w ostatecznej konfiguracji od wielkości instancji (rysunek 19) nałożono krzywą $O(n^2)$, ponieważ pasuje ona do pomiarów uznałem, że algorytm posiada taką właśnie złożoność czasową.

Jak widać na wykresie błedów w zależności od wielkości instancji (rysunek 20) nie wydaje się aby punkty tworzyły funkcję liniową. Do dokładniejszego stwierdzenia jak zachowuje się bład względny, należałoby przeprowadzić badania z instacjami o większych rozmiarach. Być może błedy względne tworzą na wykresie funkcję logarytmiczną, albo nawet nie ma wzrostu błedu, i pozostaje on na podobny poziomie wraz ze wzrostem wielkości instancji, co sugeruje prawa połowa wykresu.

6.1 Porównanie z innymi algorytmami



Rysunek 21: Porównianie czasu działania badanych algorytmów w zależności od wielkości instancji

Powyższy wykres (rysunek 21) ukazuje jak algorytm symulowanego wyżarzania góruje nad wcześniej zbadanymi pod względem czasu wykonywania. Jest to bardzo duże przyspieszenie działania, które pozwala rozwiazywać problem w rozsądnym czasie nawet dla instancji o 400 wierzchołkach. Należy jednak pamiętać, że algorytm symulowanego wyżarzania nie zawsze zwraca cykl optymalny, przeważnie jego koszt rożni się od kosztu optymalnego. W mojej implementacji udało się uzyskać błąd względny do 6% do 400 wierzchołków dla instancji symetrycznych. Dla instancji asymetrycznych ostateczna wersja algorytmu (nie podana w raporcie) uzyskiwała rozwiązania obarczone dużym błędem względnym (do 200%), ale jak wcześniej napisałem dla instancji ATSP należałoby przyjąć inny typ sąsiedztwa w celu zredukowania błędu.

7. Analiza wyników i wnioski

Udało się z powodzeniem zaimplementować algorytm SA o złożoności $O(n^2)$. Algorym ten sprawdza się dla instancji o większym rozmiarze, jednak jest mała szansa, że zwróci rozwiązanie optymalne. Dla intancji symetrycznych do 400 wierzchołków udało się zwracać rozwiązania z błedem do 6% w stosunku do rozwiązania optymalnego. Bardziej dogłębna analiza wyników i wnioski umieszczone są również prawie pod każdym wykresem.

Źródła

[1] http://155.158.112.25/~algorytmyewolucyjne/materialy/algorytm_symulowanego_wyzarzania.pdf?fbclid=IwAR2ZmpuqvMo5FvYJIWfin4WOTtwzp4Jds8_pHIGgOljy2aqUT_xCG0fSMlw
[2] https://www.youtube.com/watch?v=gX-X85dCib0&t=693s

spis rysunków