Projektowanie Efektywnych Algorytmow

Projekt

21/12/2021

252690 Karolina Nogacka

(4) Symulowane wyzarzanie

spis treści	***strona***

1. **Sformułowanie zadania:**

   Zadanie polega na stworzeniu algorytmu znajdującego rozwiązanie problemu komiwojażera opartego o metodę symulowanego wyżarzania.

   Problem komiwojażera (TSP - Traveling salesman problem):

   Zakładając, że ktoś (np. kurier, sprzedawca, listonosz) ma zbiór miast/domów do odwiedzenia szukamy drogi, która zawierać będzie wszystkie wyżej wymienione miejsca i będzie drogą najkrótszą (aby listonosz nie musiał się nachodzić).

   Innymi słowy problem ten jest zagadnieniem optymalizacyjnym , polegającym na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Wszystkich cykli w takim grafie będzie *n*! (w związku z czym złożoność dla większych grafów będzie szybko rosła), pewne z nich będą w rzeczywistości tymi samymi cyklami, zaczynającymi się jednak w różnych węzłach grafu.

2. Metoda:

Metoda symulowanego wyżarzania zainspirowana jest zjawiskami zachodzącymi podczas obróbki metali. Kluczowym elementem metody jest parametr sterujący „temperatura". W metalurgii im wyższa temperatura, tym cząsteczki zachowują się bardziej chaotycznie. Podczas procesu wyżarzania przyjmuje się pewną temperaturę początkową, a następnie obniża się ją.

Przebieg:

1. Losowy wybór punktu startowego *S* (w przypadku TSP losowy wybór rozwiązania początkowego). Przyjęcie pewnej temperatury początkowej $T = T_{max}$,
2. Wyznaczyć rozwiązanie w sąsiedztwie rozwiązania *S*, czyli *S'*,
3. Obliczyć $\delta = \mathrm{f}(S') - \mathrm{f}(S)$, jeśli delta <0, to przyjąć *S* = *S'*,
4. W p.p. wylosować p z $\mathrm{zakresu}(0,1)$, jeśli $p < e^{(-\delta/T)}$, to *S* = *S'*, czyli nawet jeśli rozwiązanie nie jest lepsze od aktualnego, przyjąć należy z pewnym prawdopodobieństwem *S'* jako aktualnie najlepsze rozwiązanie. Zapobiega to utknięciom w minimach lokalnych,
5. $T = \alpha(T)$**,**
6. Dopóki niespełniony warunek końca, powtarzane są kroki [2, …) (Wewnętrznych iteracji tyle ile wynosi długość epoki *L*),
7. Zwracane jest rozwiązanie *S*.

S – bieżące rozwiązanie,

S' – rozwiązanie z sąsiedztwa *S*,

δ – różnica wartości funkcji celu *S* i *S'*,

f(*n*) – funkcja celu,

T – aktualna temperatura,

$\alpha(T)$ – funkcja zmiany temperatury, schemat chłodzenia (Boltzmanna, geometryczny)

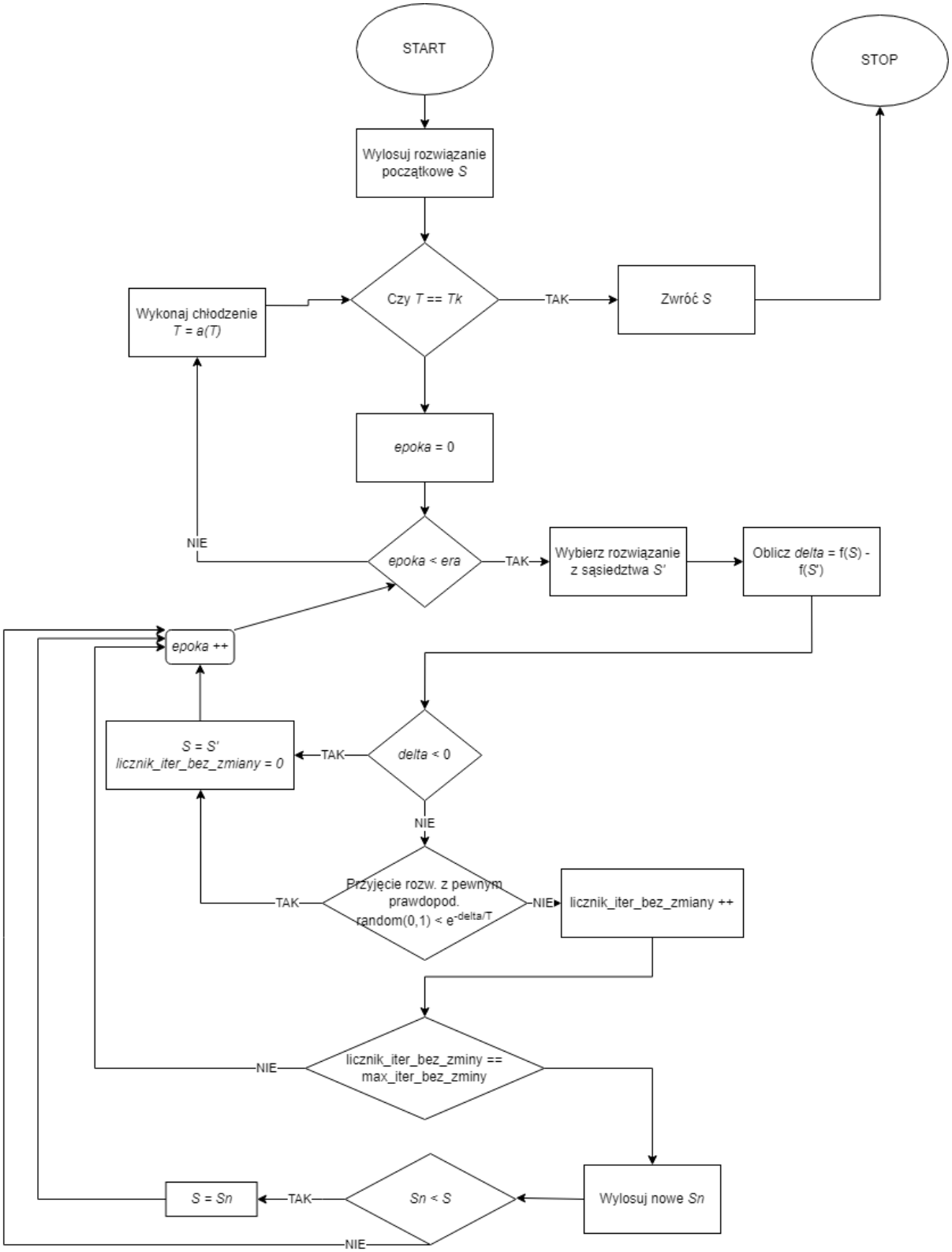
L – długość epoki – liczba wewnętrznych iteracji (prób znalezienie rozwiązania w sąsiedztwie).

Metoda nie zapewnia znalezienia zawsze rozwiązania dokładnego.

Spodziewanym jest uzyskanie wyników dla obszerniejszych instancji niż w przypadku algorytmów opartych o: brute force, programowanie dynamiczne, czy bxb, lecz niestety wyników niedokładnych, obarczonych pewnym błędem.

3. Algorytmy:

Algorytm dla problemu komiwojażera oparty na metodzie symulowanego wyżarzania stara się iteracyjnie poprawiać aktualne rozwiązanie.



Rysunek 1. Schemat blokowy algorytmu znajdującego rozwiązanie problemu komiwojażera, opartego na metodzie symulowanego wyżarzania.

4. Dane wejściowe i wyjściowe

Co zawiera program badawczy:

- Plik wykonywalny tsp_sa.exe
  Aby działał poprawnie musi być w jednym katalogu z plikiem config.ini i pliki tekstowe z grafami.

- Pliki tsp_6_1.txt, tsp_6_2.txt, tsp_10.txt, tsp_12.txt, itd.
  Zawierają grafy w postaci macierzy sąsiedztwa, które będziemy badać. Źródło: http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/ [data dostępu: 13.11.2021, 17:48]

- Pliki gr21.tsp, ulysses22.tsp, br17.atsp, itd., wszystkie wypisane w wymaganiach projektu.
  Zawierają grafy w postaci macierzy sąsiedztwa, które będziemy badać. Źródło: http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php [data dostępu: 16.12.2021, 17:54]

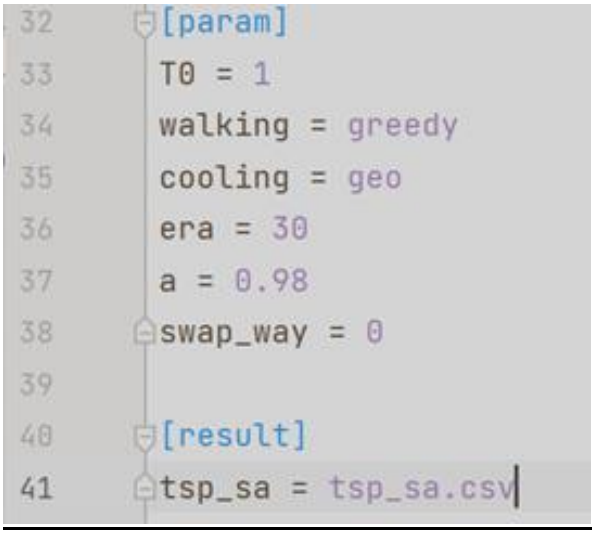
- Plik inicjujący config.ini

```
[data]
tsp_6_1 = tsp_6_1.txt; 132; [0, 1, 2, 3, 4, 5, 0]
tsp_6_2 = tsp_6_2.txt; 80; [0, 5, 1, 2, 3, 4, 0]
tsp_10 = tsp_10.txt; 212; [0, 3, 4, 2, 8, 7, 6, 9, 1, 5, 0]
tsp_12 = tsp_12.txt; 264; [0, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10, 0]
tsp_13 = tsp_13.txt; 269; [0, 10, 3, 5, 7, 9, 11, 2, 6, 4, 8, 1, 12, 0]
tsp_14 = tsp_14.txt; 282; [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 1, 12, 0]
tsp_15 = tsp_15.txt; 291; [0, 12, 1, 14, 8, 4, 6, 2, 11, 13, 9, 7, 5, 3, 10, 0]
tsp_17 = tsp_17.txt; 39; [0, 11, 13, 2, 9, 10, 1, 12, 15, 14, 5, 6, 3, 4, 7, 8, 16, 0]

gr21 = gr21.tsp; 2707; []
ulysses22 = ulysses22.tsp; 7013; []
gr24 = gr24.tsp; 1272; []
fri26 = fri26.tsp; 937; []
bays29 = bays29.tsp; 2020; []
att48 = att48.tsp; 10628; []
eil51 = eil51.tsp; 426; []
berlin52 = berlin52.tsp; 7542; []
br17 = br17.atsp; 39; []
ftv33 = ftv33.atsp; 1286; []
ftv35 = ftv35.atsp; 1473; []
ftv38 = ftv38.atsp; 1530; []
p43 = p43.atsp; 5620; []

gr96 = gr96.tsp; 55209; []
kroA100 = kroA100.tsp; 21282; []
kroB150 = kroB150.tsp; 26130; []
pr152 = pr152.tsp; 73682; []
ftv170 = ftv170.atsp; 2755; []
rbg323 = rbg323.atsp; 1326; []
```

Rysunek 2. Zawartość pliku config.ini.



Rysunek 3. Zawartość pliku config.ini c.d.

Pierwsze zdjęcie zawiera nazwy plików ze strony http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/, http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/pea.php oraz optymalne rozwiązania dla grafów w nich zawartych. Drugie zdjęcie zawiera parametry algorytmu oraz plik nazwę pliku wynikowego.

Sekcja [data] zawiera nazwę plików wejściowych z grafami.
Sekcja [param] zawiera parametry algorytmu.
Sekcja [result] zawiera nazwę pliku wyjściowego.

- Skrypty tsp_sa.py, my_writer.py:

Skrypt tsp_sa.py jest głównym plikiem programu, zawiera on wywołania metod z innych plików. my_writer.py zapewnia poprawny zapis do plików wynikowych.

- Plik tsp_sa_out-analiza.csv:

Zawiera zbiorcze dane i wykresy.

5. Procedura badawcza

- **Kolejność wykonywanych badań:**
    - Pobieranie danych z pliku inicjującego,
    - Wczytanie grafów z plików wejściowych,
    - Uruchomienie badań dla grafów.

Wynikiem działania programu (zapisywanym do pliku) jest czas znalezienia rozwiązania dla każdej instancji, długość najkrótszego cyklu, oraz sam cykl (w postaci listy węzłów).

Wyniki zostały opracowane po 20-krotnym uruchomieniu programu (algorytm wykonał się 20 razy dla każdej instancji). Jako górną granicę czasową dla algorytmu przyjęto godzinę działania całego programu oraz maksymalnie godzina pracy nad jedną instancją.

Algorytm był w stanie policzyć wyniki dla każdej zadanej instancji w zadanym czasie. Wyniki czasowe jednak znacznie różnią się w zależności od zadanych parametrów.

Specyfikacja sprzętu:

a. Procesor Intel i7-10510U, 1.80GHz – 2.30 GHz
b. 16,0 GB pamięci ram
c. System Windows 10 Home Edition

- **Metoda badania zużycia czasu:**

```
187    def work(matrix: instance, tsp_sa_params):
188        print("Liczę dla: " + matrix.name)
189
190        start_time = time.time()
191        path, cost, tsp_sa_params = simul_annealing(matrix.matrix, tsp_sa_params)
192        end_time = time.time() - start_time
193
194        tsp_result = [str(matrix.name), int(matrix.length), end_time, cost, path]
195
196        return tsp_result, tsp_sa_params
```

Rysunek 4. Metoda pomiaru czasu.

W funkcji *work()* przed i po uruchomieniu funkcji *simul_annealing()* mierzony jest czas za pomocą funkcji z biblioteki python'a *time. Różnica zapisywana jest do pliku wyjściowego jako czas wykonania algorytmu dla zadanej instancji.*

Wybór parametrów:

5.1. Wybór rozwiązania początkowego (dwa sposoby):

- Wybór pierwszego rozwiązanie bazujący na losowym doborze odwiedzanych miast,
- Wybór pierwszego rozwiązania jako wylosowanie pierwszego miasta, dobór kolejnych miast bazujący na „najbliższym sąsiedzie”.

5.2. Wybór temperatury początkowej (dwa sposoby):

- Temperatura taka sama dla każdej instancji $T = 1$ (zadana w pliku .ini),
- Temperatura początkowa zależna od rozmiaru instancji.

5.3. Wybór temperatury końcowej:

- Taka sama dla każdej instancji 0.001,

5.4. Wybór sposobu chłodzenia (dwa sposoby):

- Logarytmiczny (zadany w pliku .ini),
- Geometryczny (zadana w pliku .ini).

5.5. Wybór parametru a:

- Taka sama dla wszystkich instancji, 0.98, 0.997, 0.999 (zadany w pliku .ini),

5.6. Wybór długości ery (dwa sposoby):

- Taka sama dla każdej instancji (zadana w pliku .ini),
- Obliczana jako 3 krotność wielkości instancji.

5.7. Wybór rozwiązania w sąsiedztwie (trzy sposoby):

- 2-zamiana, zamiana dwóch miast miejscami w rozwiązaniu (zadana w pliku .ini, jako *swap_way* = 0),
- Zamiana miejsc dwóch części rozwiązania (zadana w pliku .ini, jako *swap_way* = 1),
- Odwrócenie części rozwiązania (zadana w pliku .ini, jako *swap_way* = 2).

Przeglądanie sąsiedztwa metodą „greedy”.

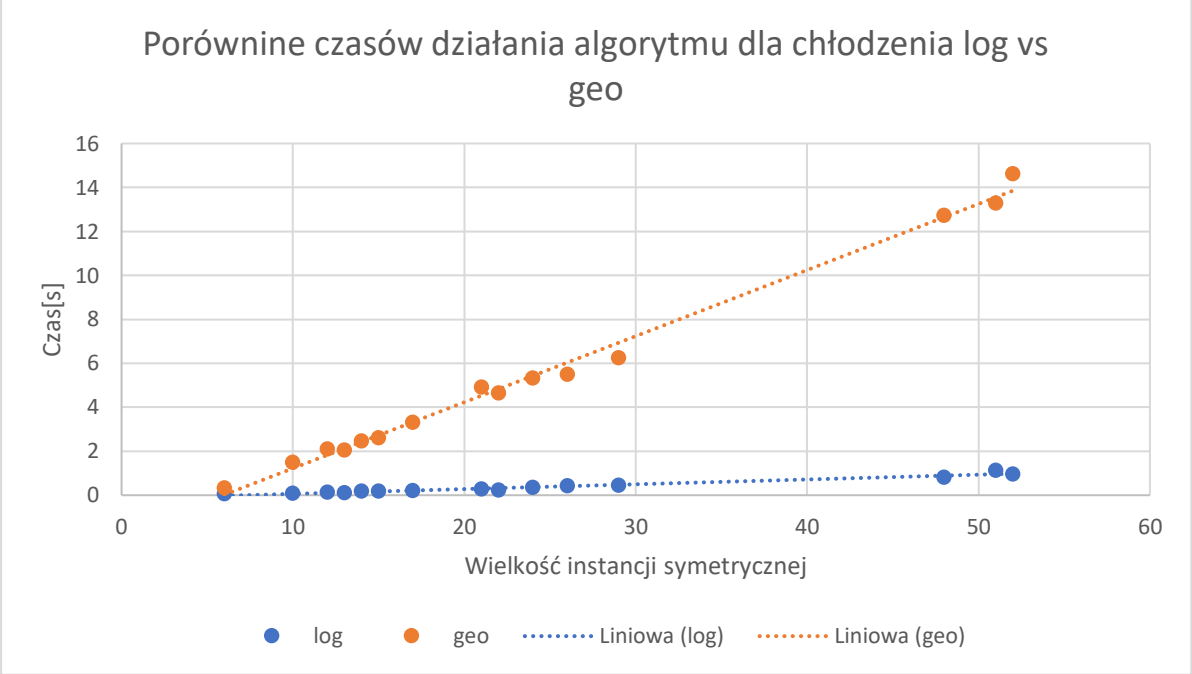
5.8. Warunki końca:

- Temperatura osiągnęła wartość końcową (pętla zewnętrzna),
- Liczba kroków przeglądania sąsiedztwa osiągnęła wartość licznika ery (pętla wewnętrzna),
- Liczba kroków bez poprawy rozwiązania osiągnęła zadane maksimum (pętla wewnętrzna).

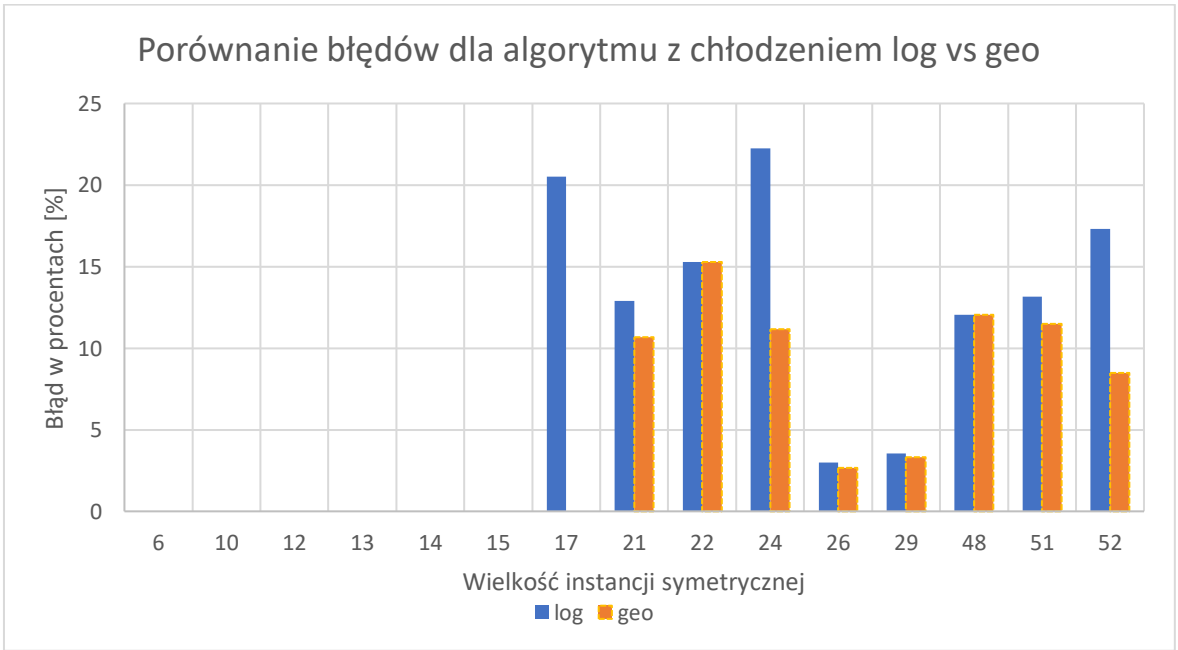
6. Wyniki

Porównanie sposobów chłodzenia:

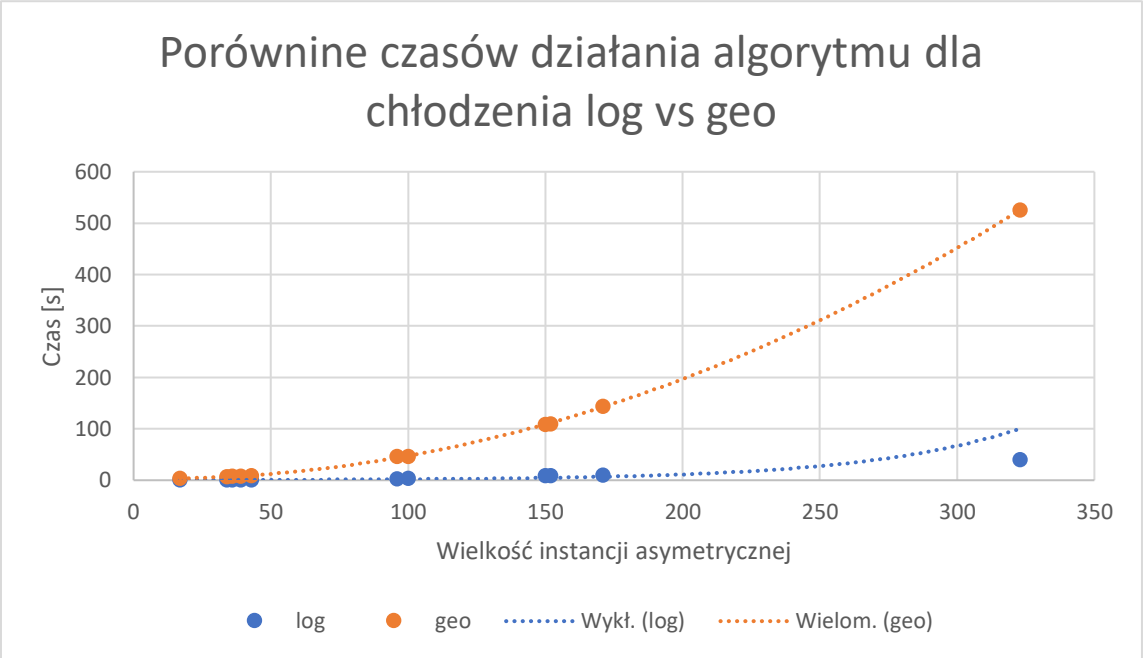
Jeśli niesprecyzowane, generowanie sąsiednich rozwiązań domyślnie: 2-zamiana.



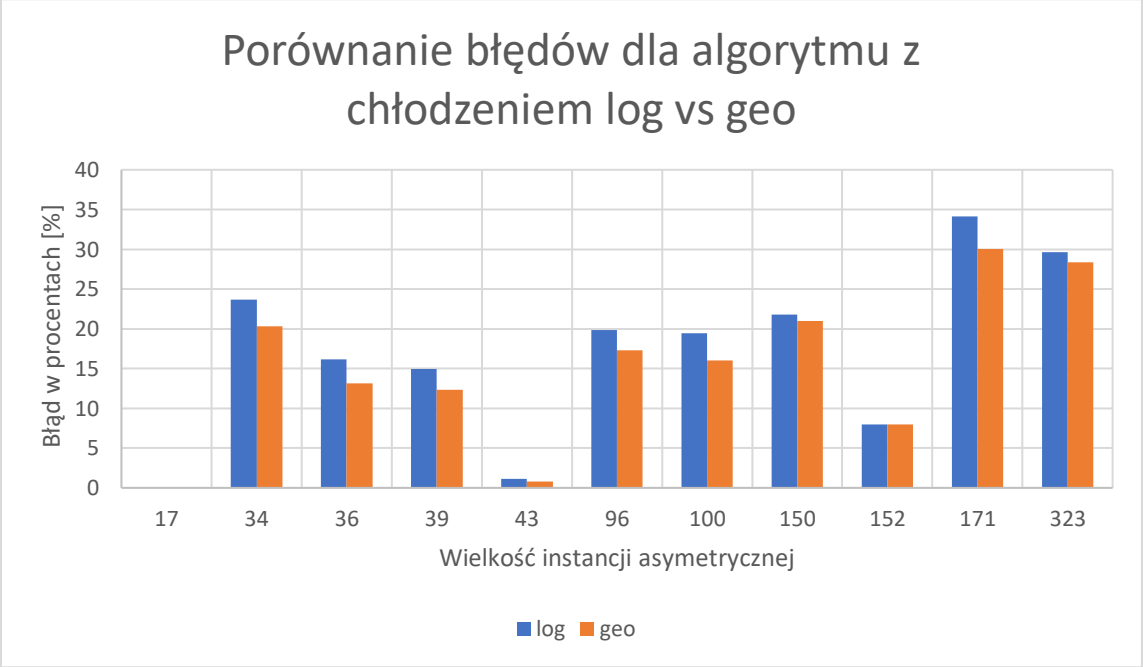
Rysunek 5. Porównanie czasu działania algorytmu dla chłodzenia logarytmicznego i geometrycznego dla instancji symetrycznych. Długość ery = 20, a = 0.98, T = 1, Tk =0.001.



Rysunek 6. Porównanie błędów działania algorytmu dla chłodzenia logarytmicznego i geometrycznego dla instancji symetrycznych. Długość ery = 20, a = 0.98, T = 1, Tk =0.001

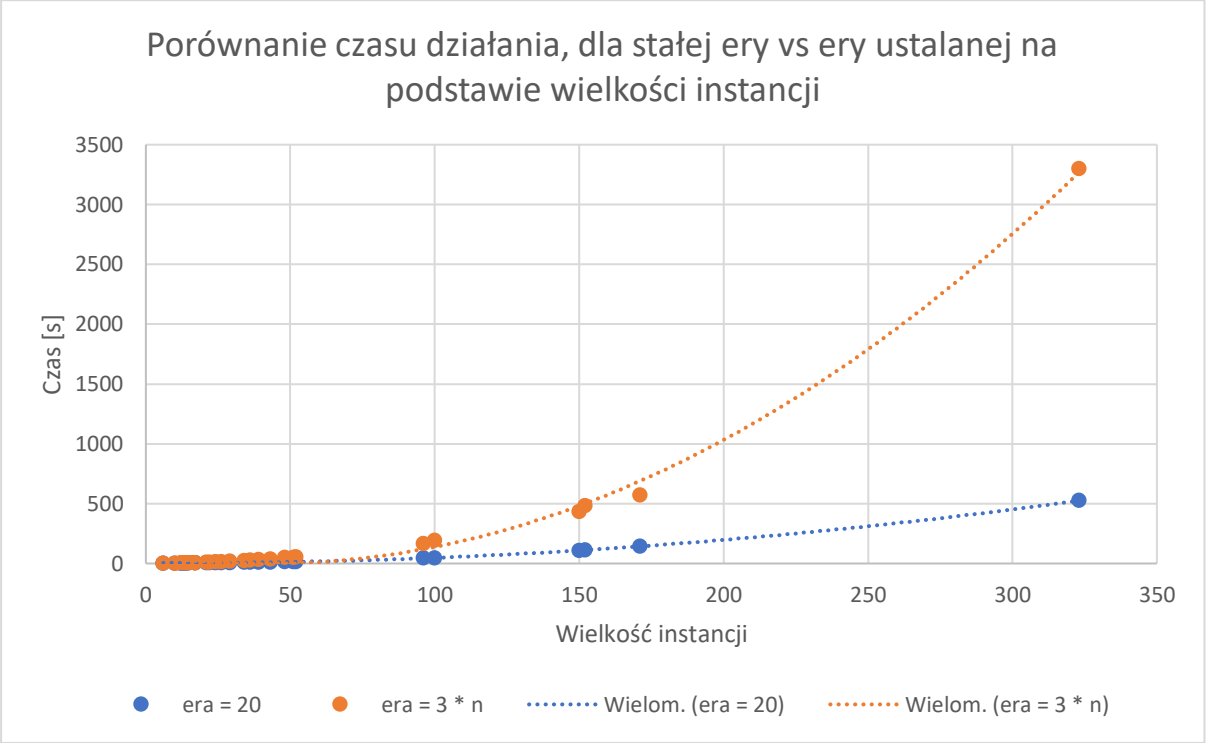


Rysunek 7. Porównanie czasu działania algorytmu dla chłodzenia logarytmicznego i geometrycznego dla instancji asymetrycznych. Długość ery = 20, a = 0.98, T = 1, Tk =0.001.

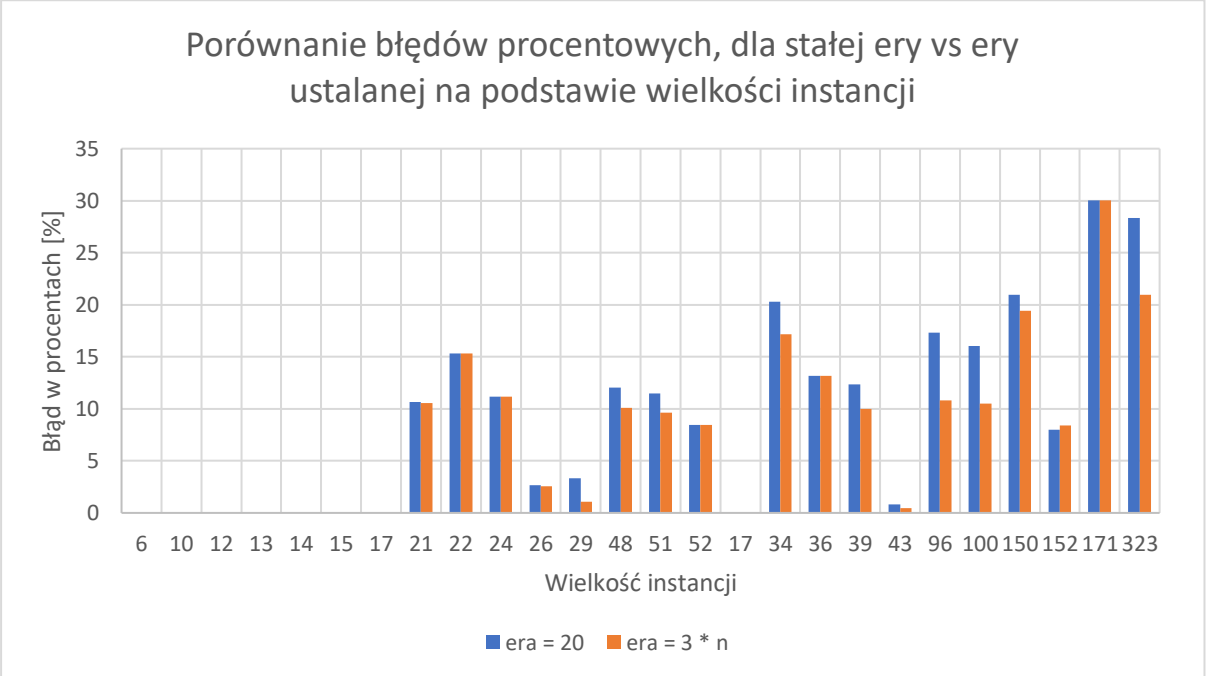


Rysunek 8. Porównanie błędów działania algorytmu dla chłodzenia logarytmicznego i geometrycznego dla instancji asymetrycznych. Długość ery = 20, a = 0.98, T = 1, Tk =0.001

Porównanie długości ery:

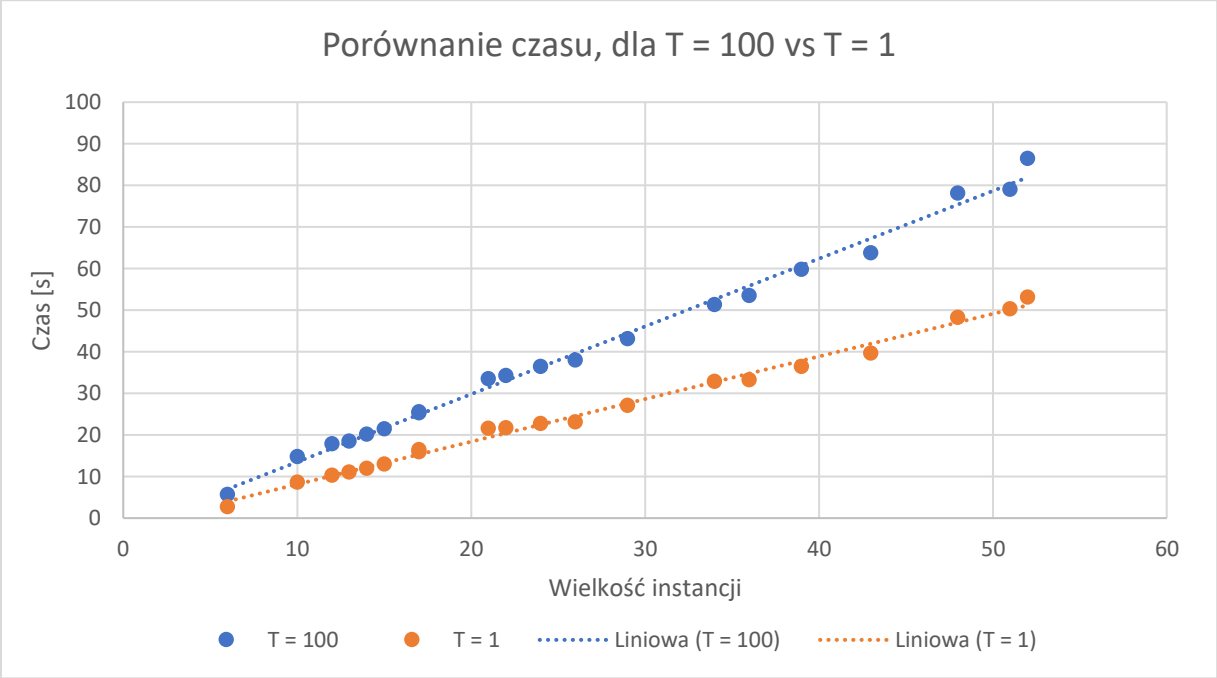


Rysunek 9. Porównanie czasu działania dla różnych sposobów wyznaczanie długości ery. Chłodzenie geo, a = 0.98, T = 1, Tk =0.001.

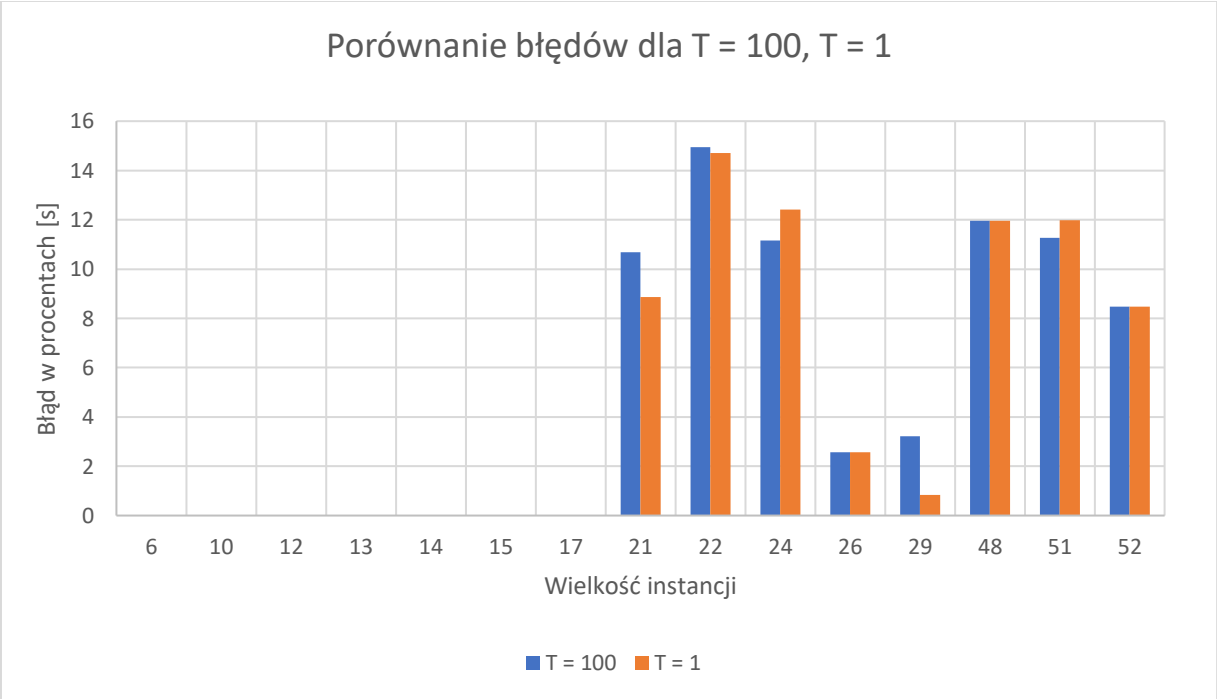


Rysunek 10. Porównanie błędów dla różnych sposobów wyznaczanie długości ery. Chłodzenie geo, a = 0.98, T = 1, Tk =0.001.

Porównanie dla temperatury początkowej:

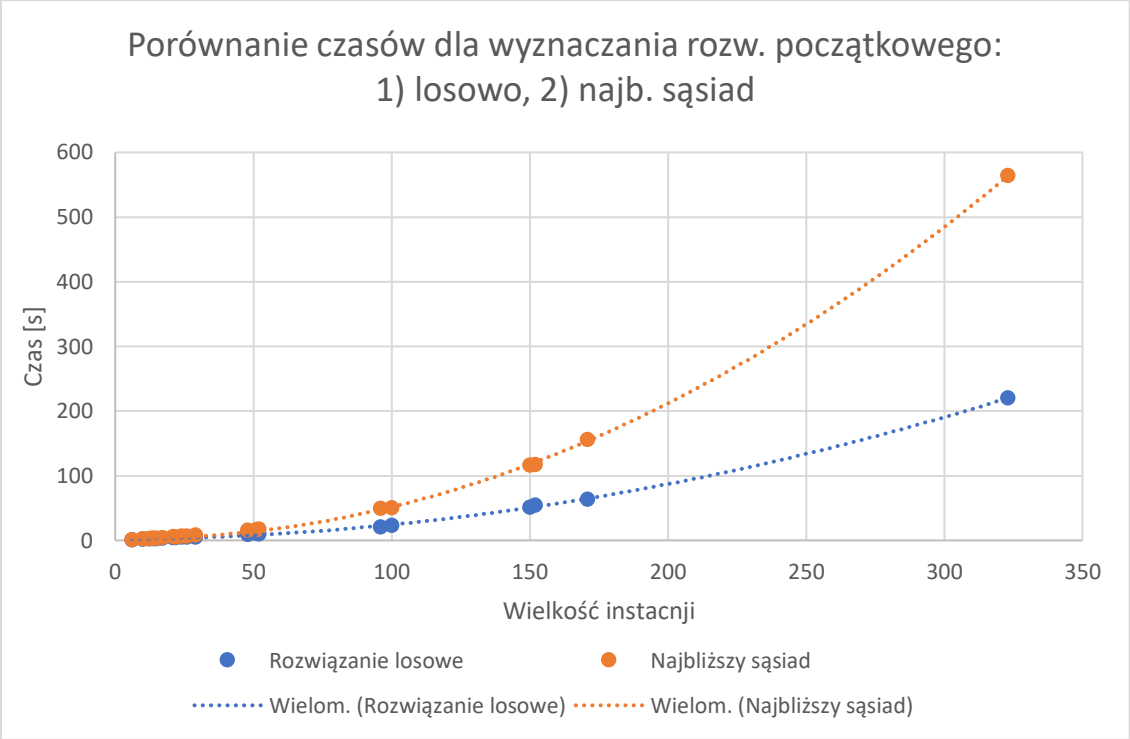


Rysunek 11. Porównanie czasu, dla T = 100 vs T = 1. Chłodzenie geo, a = 0.99, Tk =0.001, era = 60.

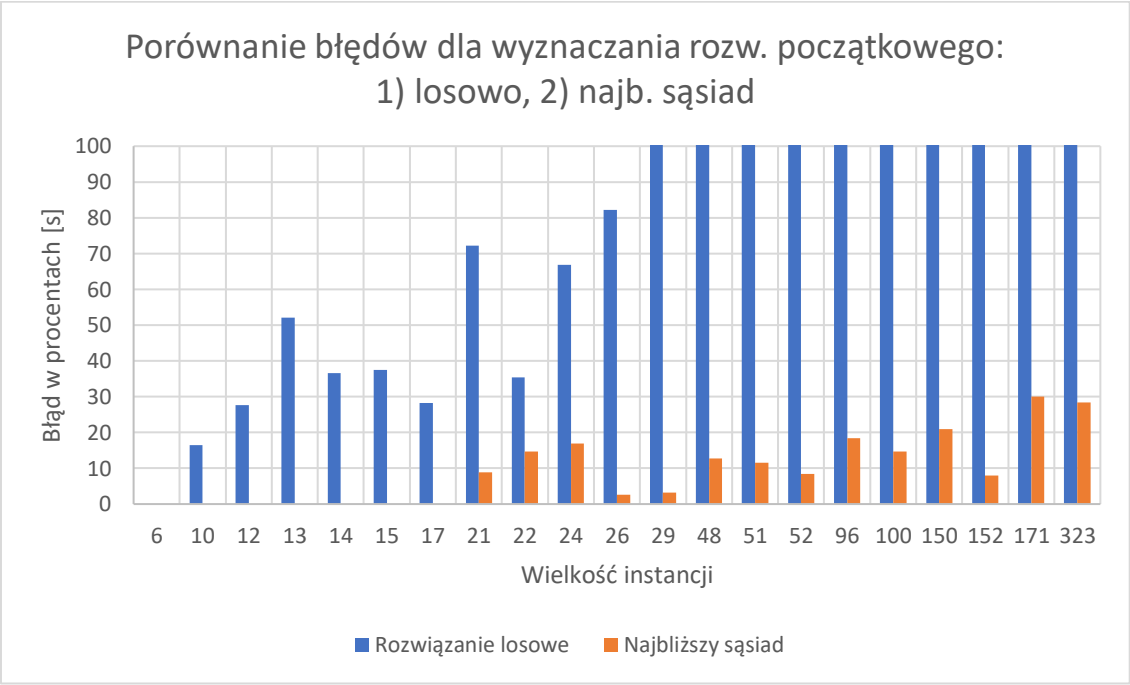


Rysunek 12. Porównanie błędów dla T = 100, T = 1. Chłodzenie geo, a = 0.99, Tk =0.001, era = 60.

Porównanie dla sposobu wyboru rozwiązania początkowego:

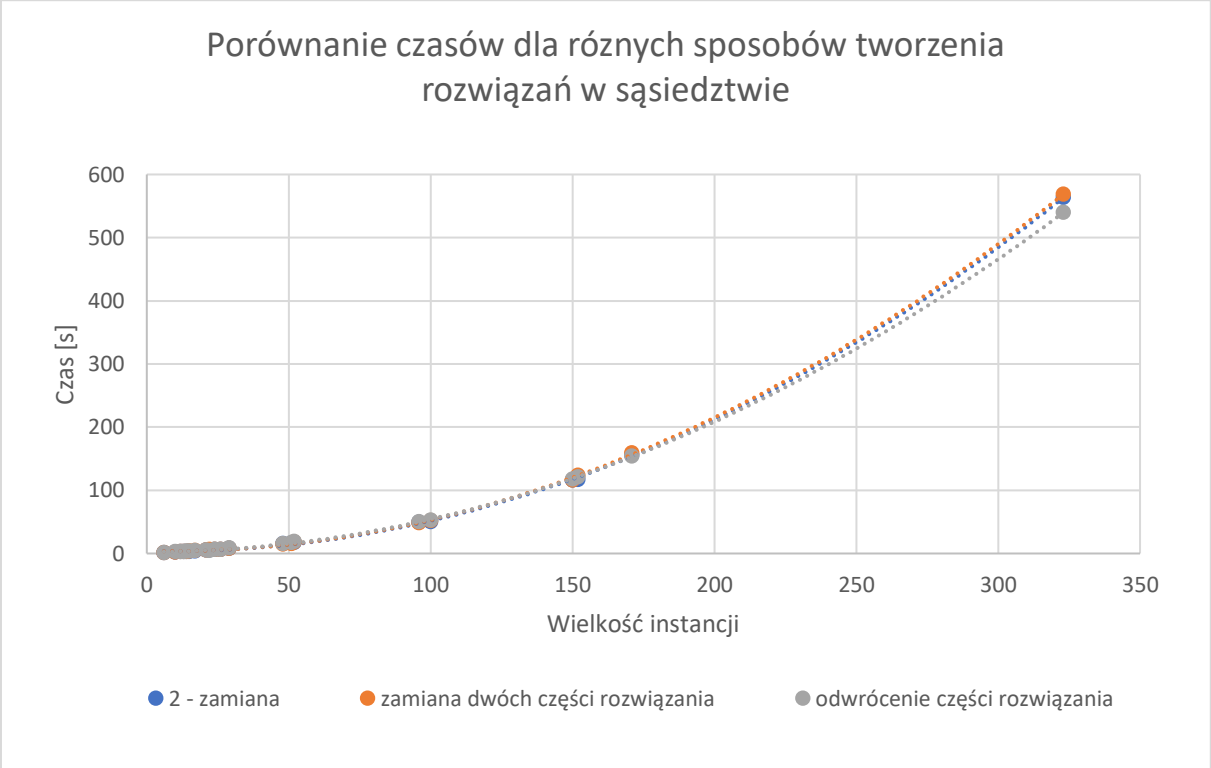


Rysunek 13. Porównanie czasów dla różnych sposobów wyboru rozwiązania początkowego. Chłodzenie geo, a = 0.99, Tk =0.001, era = 30.

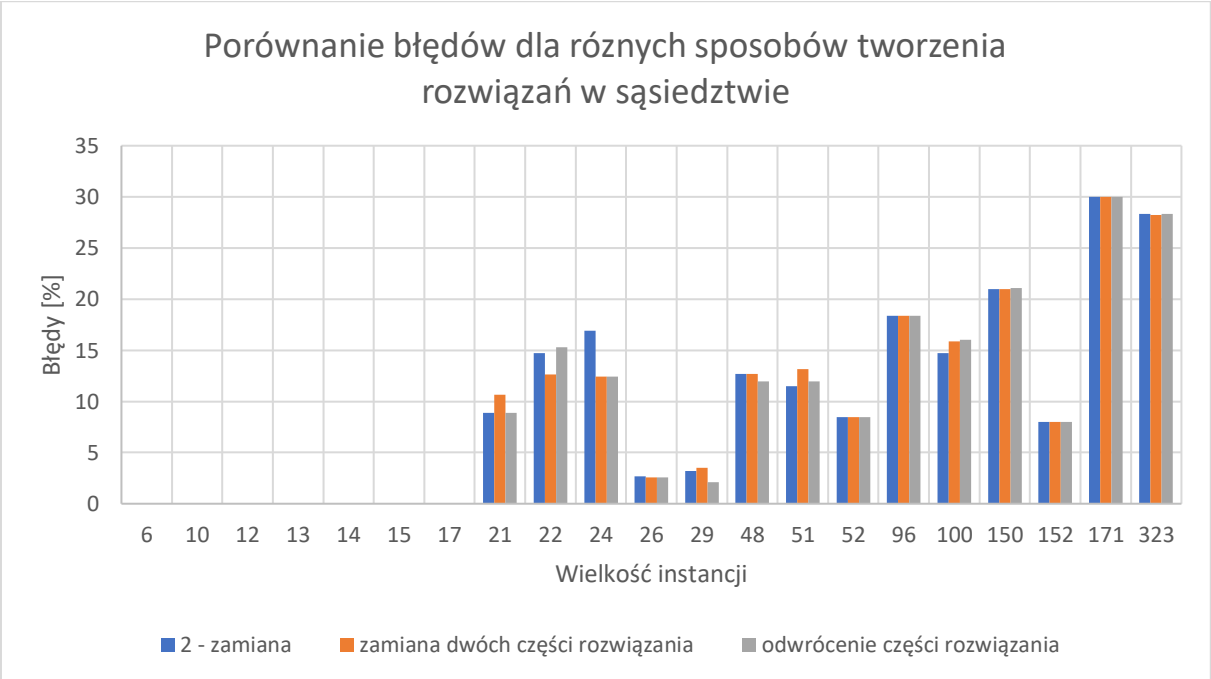


Rysunek 14. Porównanie błędów dla różnych sposobów wyboru rozwiązania początkowego. Chłodzenie geo, a = 0.99, Tk =0.001, era = 30.

Porównanie dla różnych sposobów generowanie rozwiązań w sąsiedztwie:

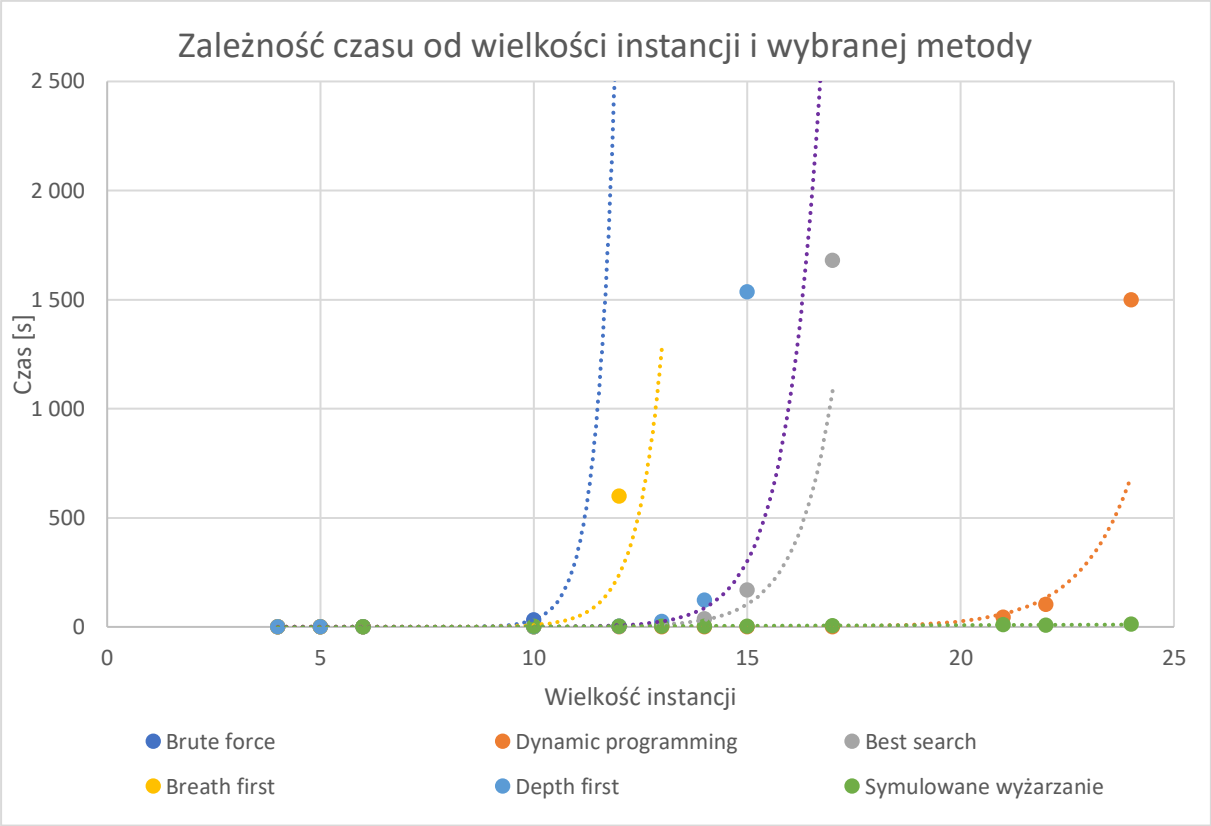


Rysunek 15. Porównanie czasów dla różnych sposobów generowania rozwiązań w sąsiedztwie. Chłodzenie geo, a = 0.98, Tk =0.001, era = 30.



Rysunek 16. Porównanie błędów dla różnych sposobów generowania rozwiązań w sąsiedztwie. Chłodzenie geo, a = 0.98, Tk =0.001, era = 30.

Najbardziej dokładne wyniki otrzymano dla: Chłodzenia = geo, era = 3*n, a = 0.98, dlatego one właśnie zostaną porównane do wyników poprzednich algorytmów.



Rysunek 17. Porównanie algorytmu symulowanego wyżarzania do pozostałych.

7. Analiza wyników i wnioski

Algorytm skonstruowany w oparciu o metodę symulowanego wyżarzania, zgodnie z oczekiwaniami okazał się szybszy od poprzednio analizowanych metod, jest on w stanie w podobnym do nich czasie (ok. 1h) wygenerować wyniki dla instancji do 323 wierzchołków (*Rysunek 17*).

Błąd wyniku (otrzymany wynik w odniesieniu do optymalnego) dla poszczególnych instancji nie przekraczał 30% (*Rysunki 6, 8, 10, 12, 16*).

Porównano dwa sposoby chłodzenia (*Strona 9, 10*) (*Rysunek 5, 6, 7, 8*). Chłodzenie logarytmiczne okazało się być szybsze (mniej więcej dwukrotnie), jednak błędy okazały się większe niż dla chłodzenia geometrycznego, lub równe. Powinno się więc wybrać ten sposób, który spełnia aktualne potrzeby użytkownika (lepszy czas/lepsze wyniki).

Wykonano porównanie dla dwóch sposobów wyznaczania długości *ery (Strona 11) (Rysunek 9, 10).* Pierwszy z nich to przypisanie dla każdej instancji *era* = 20. Czas dla takiego sposobu rósł liniowo. Drugi sposób to *era* = 3*n, tak jak się spodziewano, w tym przypadku czas rósł szybciej (mniej więcej wykładniczo). Jeśli wziąć pod uwagę otrzymane błędy w procentach to drugi sposób okazał się bardziej precyzyjny dla niektórych instancji.

Wyższa temperatura początkowa (*Strona 12*) (*Rysunek 11, 12*) powoduje dłuższy czas wykonani a algorytmu, ale nie powoduje spadku współczynnika błędu, w niektórych przypadkach wręcz go podwyższa (nieznacznie). Można jednak wyciągnąć wniosek, że skuteczniej będzie użyć czym niższej temperatury początkowej (ponieważ jej zwiększanie nie robi różnicy dla lepszego wyniku, a zabiera czas).

Porównanie dla różnych sposobów wyboru losowego rozwiązania (w szczególności początkowego) (*Strona 13*) (*Rysunek 11, 12*), wybór rozwiązania w sposób zupełnie losowy (każde kolejne miast jest losowane) okazał się zupełnie nieskuteczny (wartość błędów znacznie przekraczają próg akceptowalności tj. 50%). Bardziej odpowiednim sposobem wydaje się być wybór losowy pierwszego wierzchołka rozwiązania, a następnie dobieranie kolejnych szukając najbliższych sąsiadów.

Trzy wybrane sposoby generowanie rozwiązań z sąsiedztwa okazały się generować bardzo podobne wyniki czasowe, nakładają się na siebie na wykresie. (*Strona 14*) (*Rysunek 13, 14*). Również różnice w otrzymanych błędach są podobne. Powyżej instancji wielkości 50, są bliskie identyczność. Zapewne można było wybrać znacznie bardziej odpowiednie do testowania i porównywania sposoby generowania rozwiązań z sąsiedztwa.