

Asymmetric Travelling Salesman Problem - Projekt optymalizacyjny

Kacper Wnęk, Krzysztof Sawicki, Natalia Safiejko, Piotr Kosakowski, Wojciech Grabias

4 czerwca 2023

Spis treści

| | | |
|----------|--|-----------|
| 1 | Streszczenie | 3 |
| 2 | Wstęp | 3 |
| 3 | Opis implementowanej metody | 3 |
| 4 | Główne wyniki | 6 |
| 4.1 | Plik ftv33 | 6 |
| 4.2 | Plik ftv170 | 9 |
| 4.3 | Plik kro124p | 12 |
| 4.4 | Najlepsze zestawy parametrów | 15 |
| 4.5 | Wyniki wyścigu | 15 |
| 5 | Podsumowanie i wnioski | 16 |
| 6 | Porównanie z innym zespołem | 17 |
| 7 | Dalsze możliwości rozwoju | 17 |
| 8 | Bibliografia | 18 |

1 Streszczenie

Przedstawiony raport opisuje implementację i badania metody symulowanego wyżarzania w rozwiązaniu asymetrycznego problemu komiwojażera (ATSP). Metoda symulowanego wyżarzania jest metaheurystycznym algorytmem optymalizacji inspirowanym procesem chłodzenia stopionego metalu. Raport zawiera opis implementacji algorytmu oraz wyniki przeprowadzonych badań, których celem było znalezienie optymalnych parametrów dla różnych instancji problemu. Badania obejmowały zmianę funkcji prawdopodobieństwa, parametrów chłodzenia, początkowej temperatury i liczby iteracji. Na podstawie wyników z plików ftv33, ftv170, kro124p oraz rgb323 stwierdzono, że najlepsze wyniki zostały osiągnięte dla temperatury początkowej 15000 i parametru chłodzenia równego 0.99999. Wyniki te były zgodne dla różnych funkcji prawdopodobieństwa. Zauważono również, że najlepsze wyniki zostały osiągnięte po około 800000 iteracjach. Porównano algorytmy symulowanego wyżarzania (SA) i symulowanego wyżarzania z parallel tempering (TP_SA) na różnych problemach komiwojażera. TP_SA osiągał niewielką przewagę w niektórych przypadkach, ale dla większych problemów SA wypadł znacznie lepiej i charakteryzował się stabilnością wyników. Jednym z głównych problemów implementowanego algorytmu symulowanego wyżarzania było wpadanie w minima lokalne, co utrudniało znalezienie optymalnego rozwiązania. Możliwymi modyfikacjami do rozwiązania tego problemu mogą być zmiana warunku stopu, zastosowanie równoległego wykonania wyżarzania oraz dynamiczny dobór parametrów w zależności od aktualnej temperatury i pozostałego czasu. Wprowadzenie tych zmian powinno być poprzedzone odpowiednimi testami, aby dostosować je do dostępnych zasobów czasowych i sprzętowych.

2 Wstęp

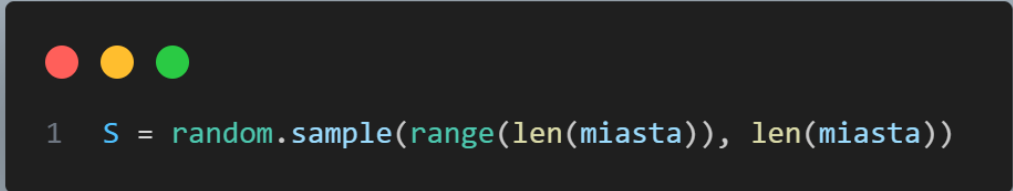
W niniejszym raporcie przedstawiamy wyniki naszego projektu z przedmiotu "Warsztaty Badawcze". Celem projektu było napisanie algorytmu rozwiązującego asymetryczny problem komiwojażera (Asymmetric Traveling Salesman Problem; ATSP) jak najlepiej w rozsądnym czasie. W tym celu nasz zespół zaimplementował metodę optymalizacji symulowanego wyżarzania (simulated annealing), opisanym w [1], oraz przeprowadził badania mających na celu znalezienie optymalnych parametrów dla różnych instancji problemu.

3 Opis implementowanej metody

Implementowana przez nas metoda to symulowane wyżarzanie (simulated annealing) - metaheurystyczny algorytm optymalizacji. Jest on inspirowany procesem chłodzenia stopionego metalu, gdzie kontrolowane schładzanie prowadzi do minimalizacji energii systemu. Algorytm symulowanego wyżarzania próbuje znaleźć optymalne rozwiązanie poprzez losowe eksplorowanie przestrzeni rozwiązań, a także akceptowanie gorszych rozwiązań w początkowej fazie działania, aby uniknąć utknięcia w lokalnym minimum. Implementacja inspirowana jest poruszoną w [3] ujęciu stosowania symulowanego wyżarzania w ogólnym problemie komiwojażera.


Algorytm polega na:

1. Wybraniu losowego rozwiązania początkowego.




```
1 S = random.sample(range(len(miasta)), len(miasta))
```

2. Wybraniu temperatury początkowej.
3. Powtarzaniu następującej sekwencji aż do spełnienia warunku zakończenia:
 - Wygenerowanie losowych sąsiednich rozwiązań.




```
1 S_p = S.copy()
2     a = random.randint(0, len(S_p) - 1)
3     b = random.randint(0, len(S_p) - 1)
4     S_p[a], S_p[b] = S_p[b], S_p[a]
```

- Obliczenie różnicy kosztu (funkcji celu) między aktualnym rozwiązaniem a sąsiednim.



```
1 C1 = oblicz_koszt(S, odleglosc)
2 C2 = oblicz_koszt(S_p, odleglosc)
```

- Jeśli różnica kosztu jest mniejsza niż zero, przyjęcie sąsiedniego rozwiązania jako aktualne. Jeśli różnica kosztu jest większa lub równa zero, obliczenie prawdopodobieństwa akceptacji na podstawie funkcji prawdopodobieństwa i temperatury. Porównanie wylosowanej wartości z prawdopodobieństwem akceptacji, jeśli jest mniejsza, przyjęcie sąsiedniego rozwiązania jako aktualne.




```

1  if C2 < C1:
2      S = S_p
3  else:
4      delta_e = abs(C1-C2)
5      p = Prob_function(delta_e, T, prob_parameter)
6      if random.random() < p:
7          S = S_p

```

- Zmniejszenie temperatury według określonej funkcji zmiany temperatury.




```

1  T = T_function(T_init, T, i, cool_parameter)
2      if T < break_point:
3          break

```

4. Zwrócenie najlepszego znalezionej rozwiązania.



```

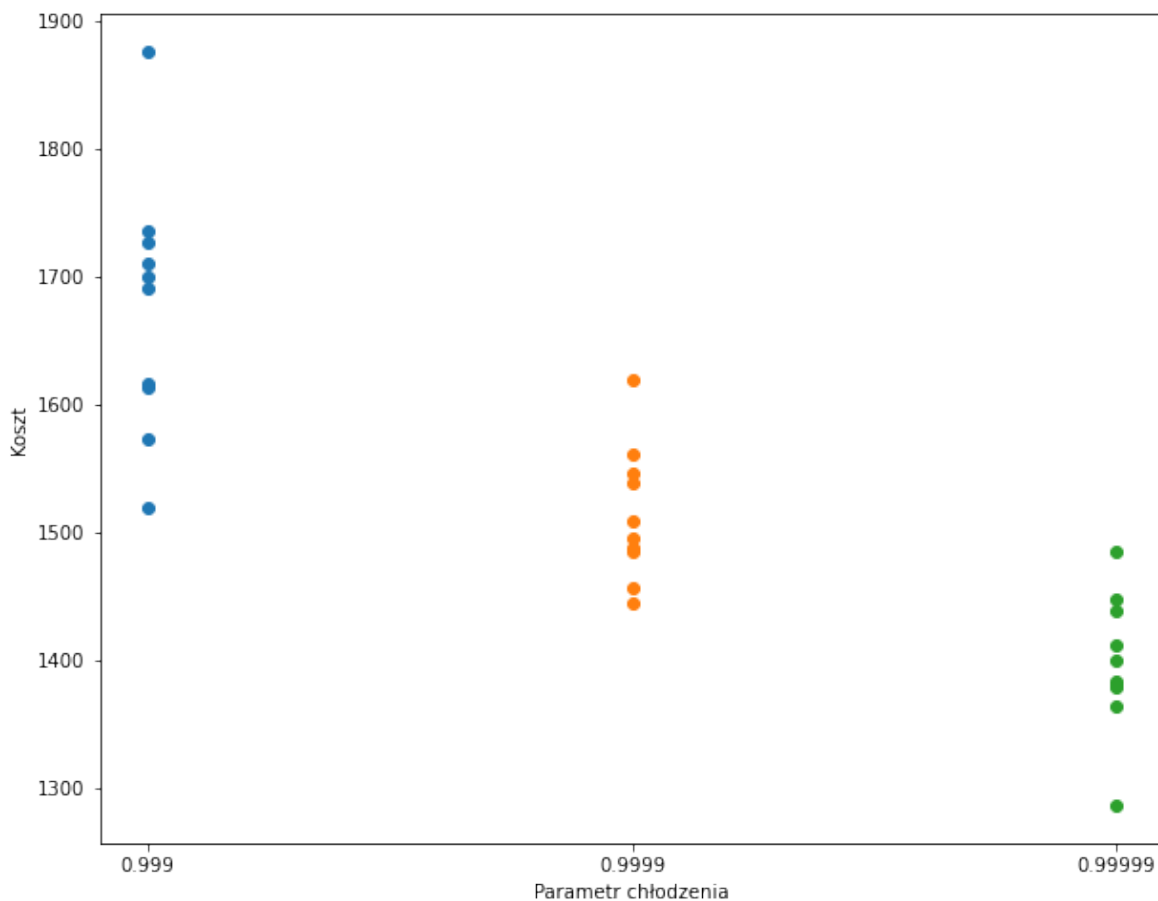
1  najlepsze_rozwiazanie = [miasta[i] for i in S]
2      end = time.time()
3      return najlepsze_rozwiazanie, oblicz_koszt(S, odleglosc), end - start

```

4 Główne wyniki

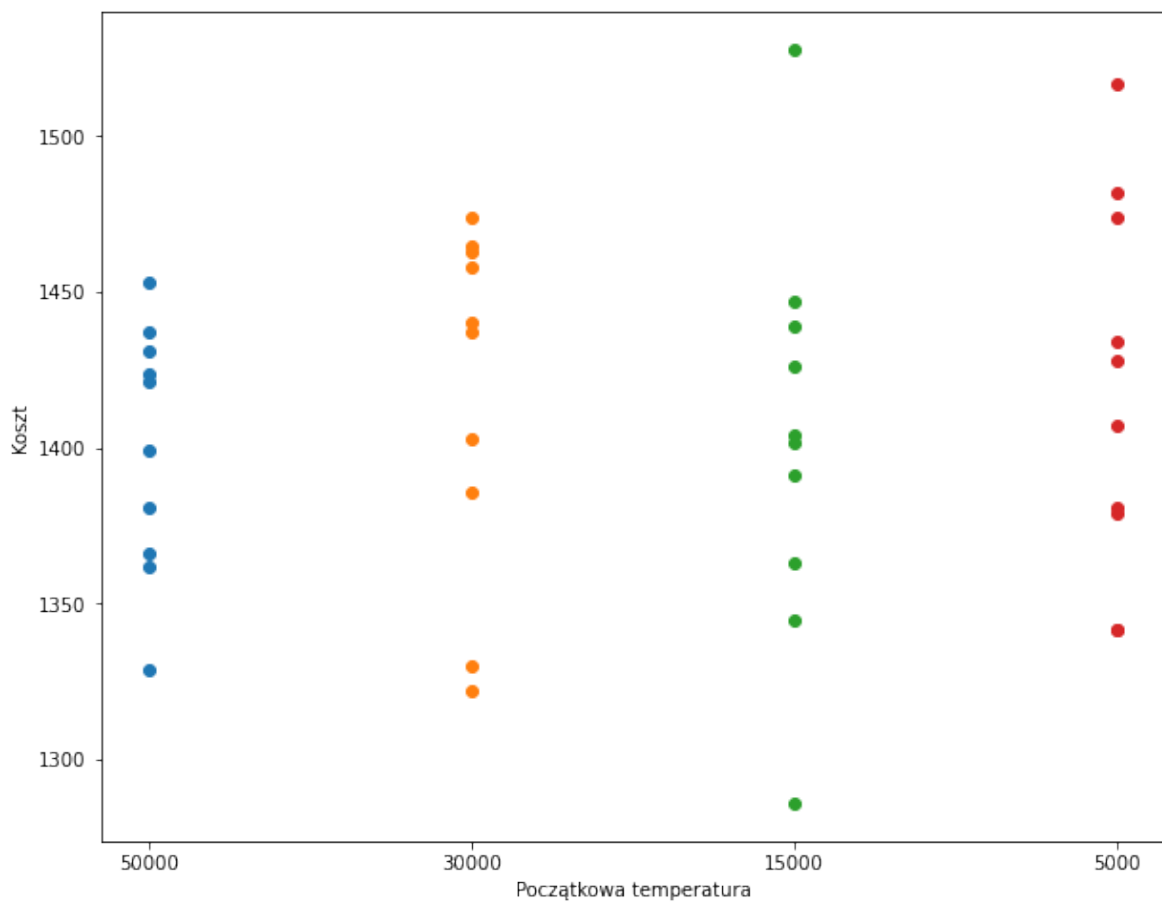
W celu znalezienia najlepszych parametrów przeprowadziliśmy testy na różnych plikach, które pomogły wybrać nam te najbardziej efektywne. Zmienialiśmy funkcję prawdopodobieństwa, jej parametry, parametr chłodzenia oraz początkową temperaturę. Wybraliśmy przykładowe pliki i wykresy przedstawiające wyniki naszych eksperymentów.

4.1 Plik ftv33



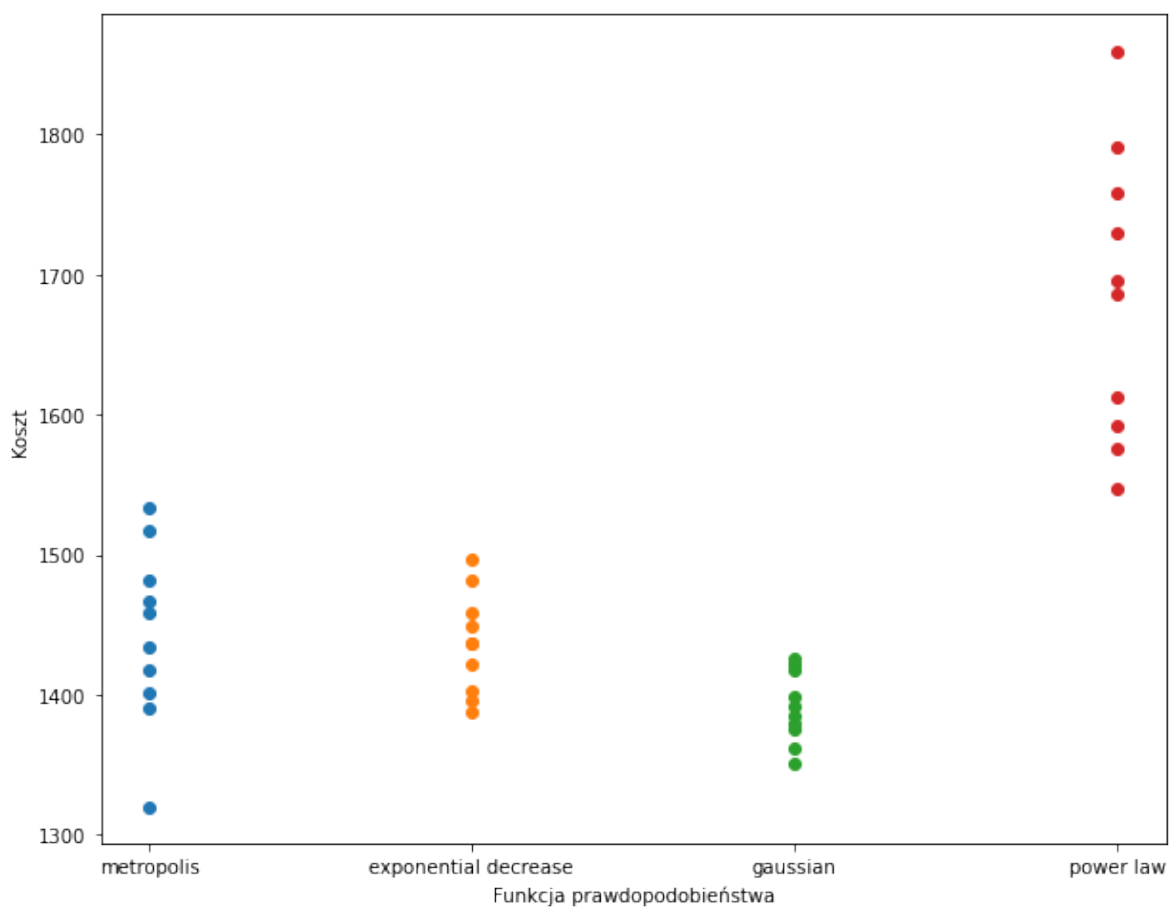
Rysunek 1: Zależność kosztu od parametru chłodzenia dla pliku ftv33

Na podstawie wykresu można wywnioskować, że im wolniej maleje temperatura, tym lepsze wyniki.



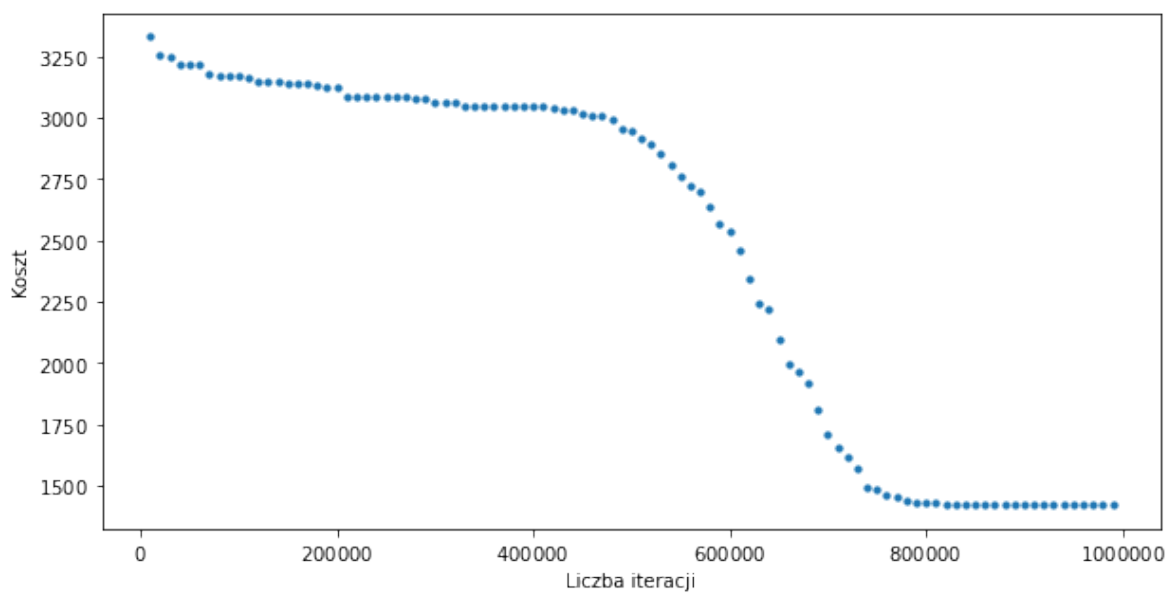
Rysunek 2: Zależność kosztu od początkowej temperatury dla pliku ftv33

Nieoczywistym wnioskiem jest, że nie zawsze im wyższa temperatura początkowa, tym niższy koszt. W tym przypadku najbardziej optymalna jest 15000.



Rysunek 3: Zależność kosztu od funkcji prawdopodobieństwa dla pliku ftv33

Najlepszy wynik został osiągnięty dla funkcji metropolis.

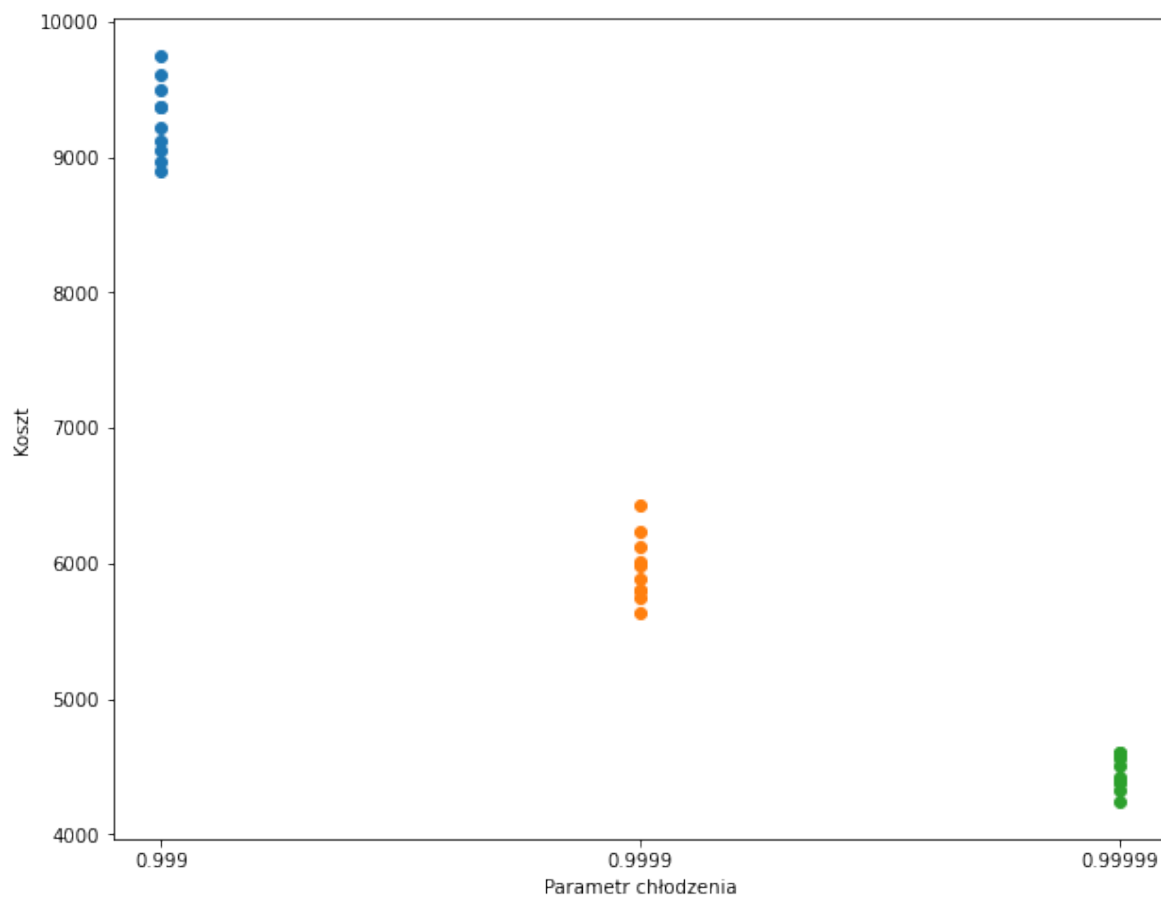


Rysunek 4: Zależność kosztu od liczby iteracji dla pliku ftv33

Na podstawie tych wykresów stwierdziliśmy, że najlepsze parametry dla pliku ftv33 to funkcja

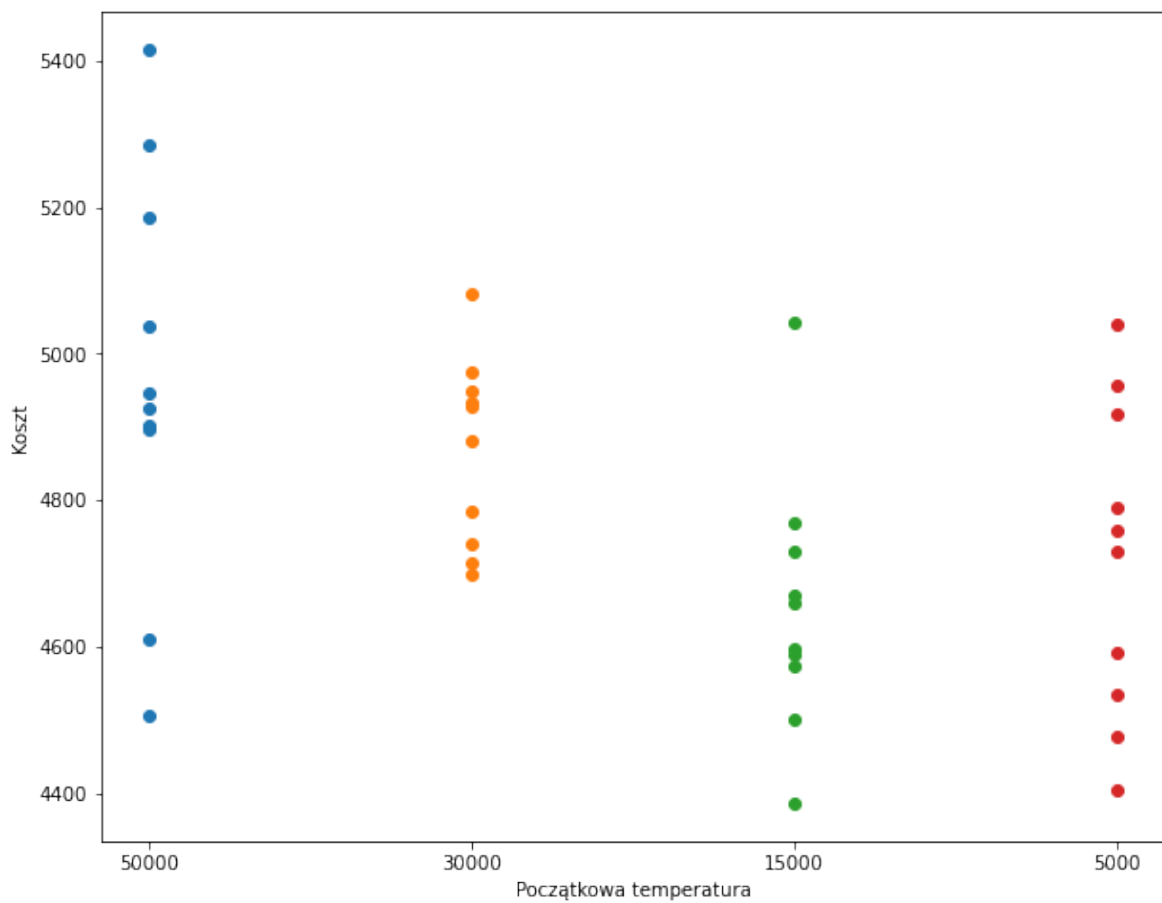
metropolis probability, z parametrem chłodzenia równym 0.99999 i początkową temperaturą równą 15000.

4.2 Plik ftv170



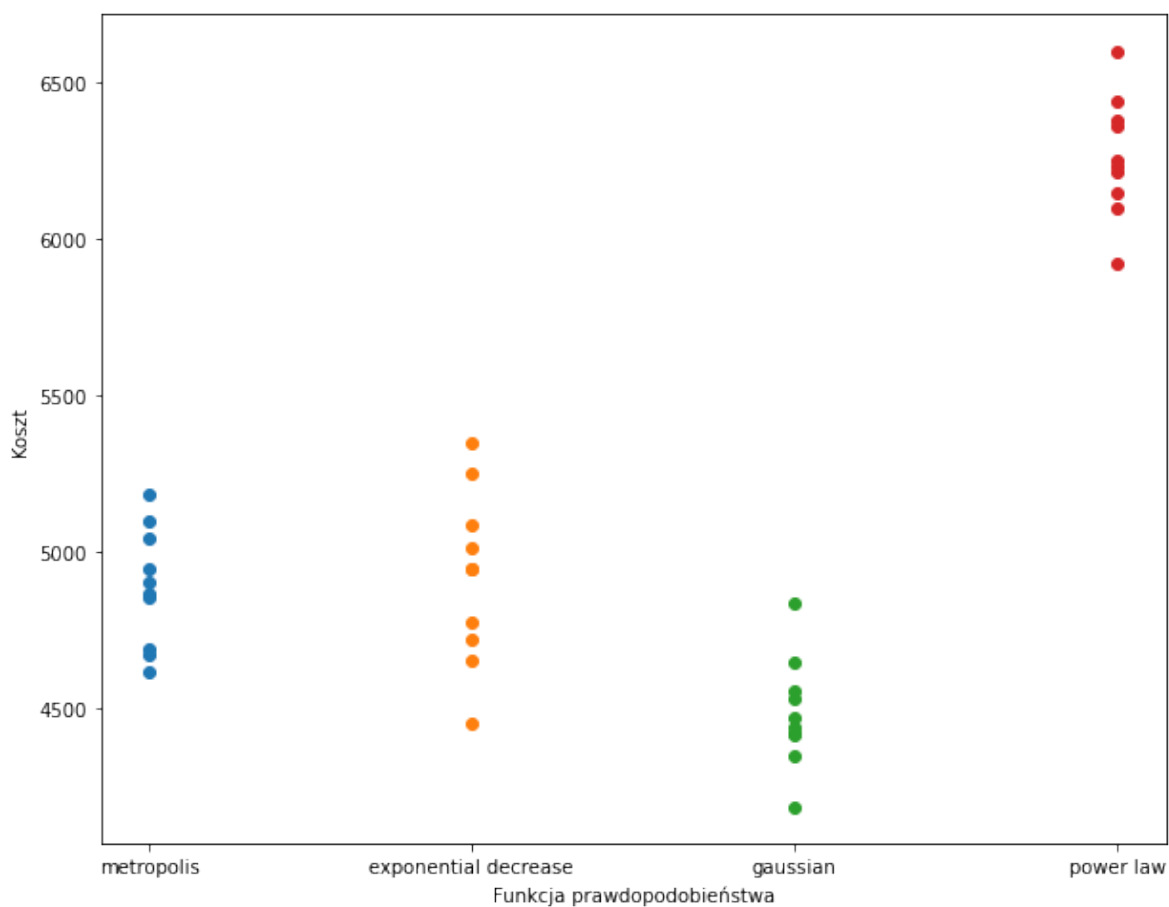
Rysunek 5: Zależność kosztu od parametru chłodzenia dla pliku ftv170

Zdecydowanie najlepszym wyborem parametru chłodzenia w tym przypadku jest 0.99999.



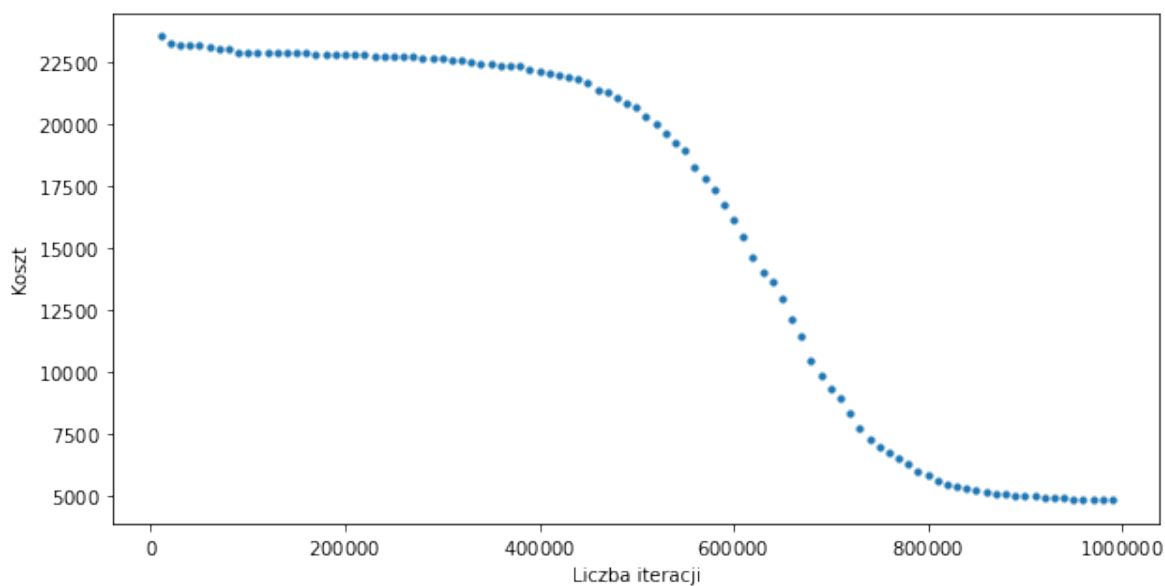
Rysunek 6: Zależność kosztu od początkowej temperatury dla pliku ftv170

W przypadku temperatury ponownie lepsze okazały się te niższe- 15000 i 5000.



Rysunek 7: Zależność kosztu od funkcji prawdopodobieństwa dla pliku ftv170

Najlepsze wyniki były osiągane korzystając z funkcji gaussian.

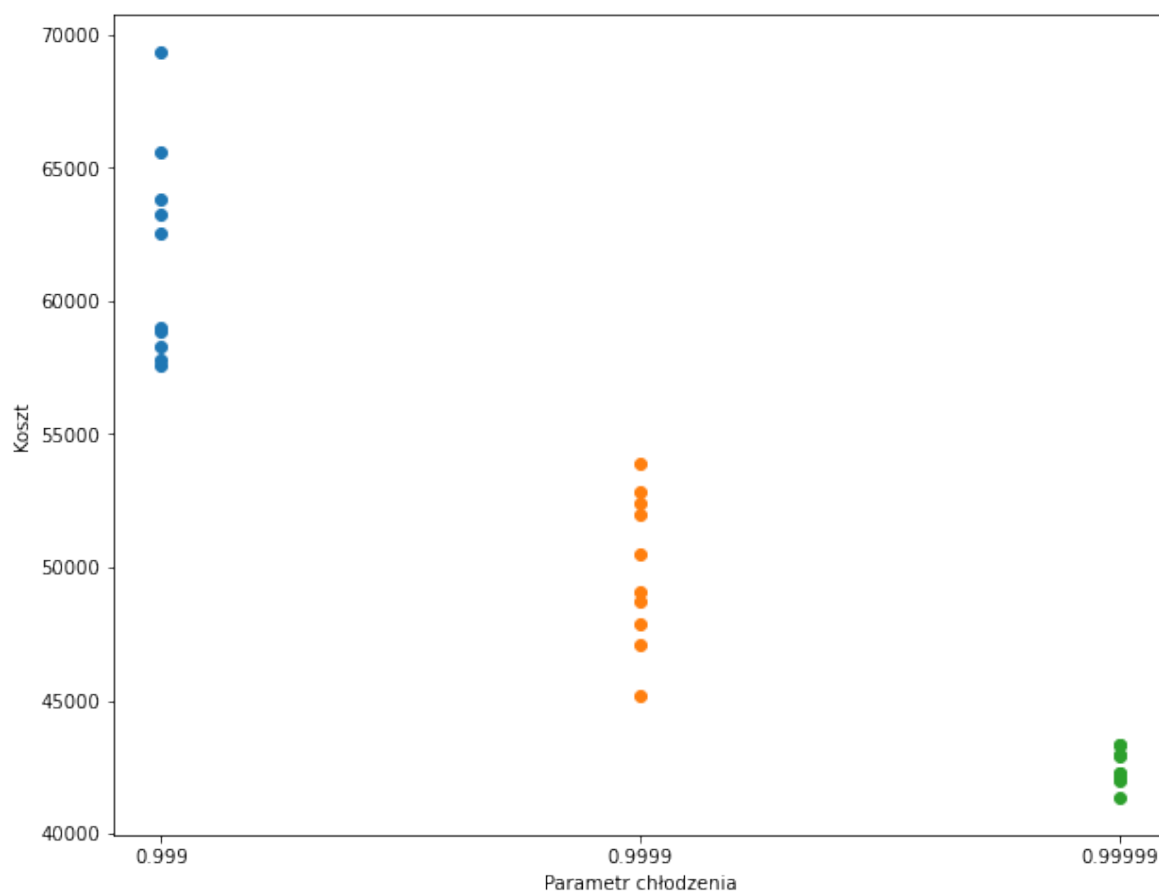


Rysunek 8: Zależność kosztu od liczby iteracji dla pliku ftv170

Najlepsze parametrami dla pliku ftv33 okazały się być: funkcja gaussian probability, z parametrem

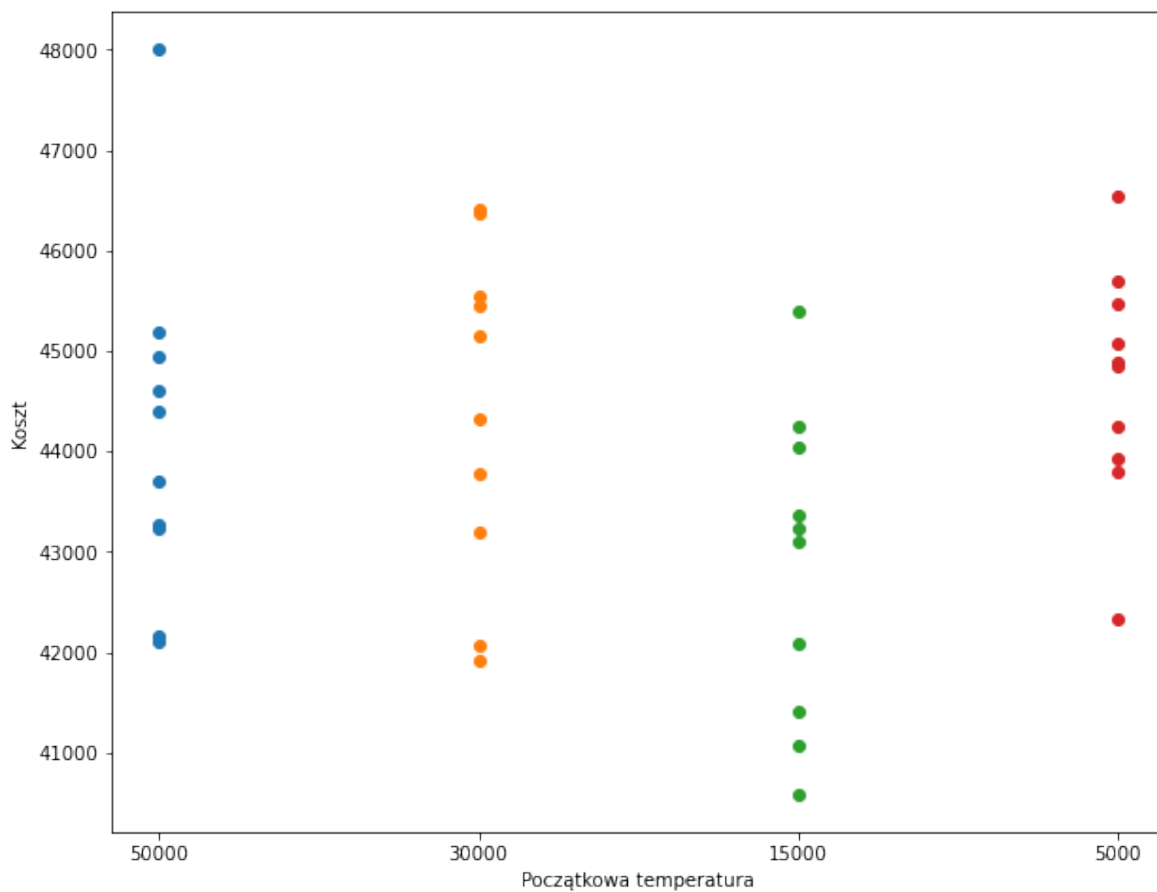
chłodzenia równym 0.99999 i początkową temperaturą równą 15000.

4.3 Plik kro124p



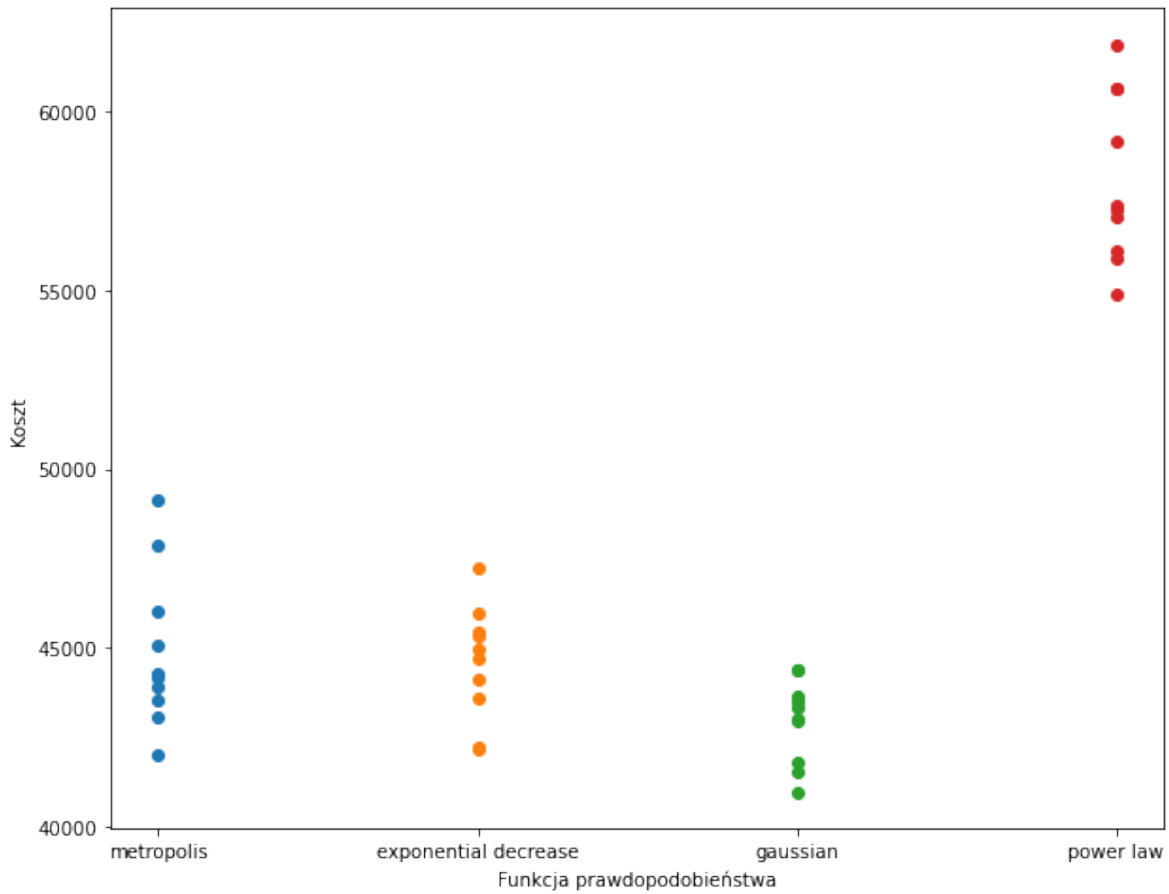
Rysunek 9: Zależność kosztu od parametru chłodzenia dla pliku kro124p

Ponownie najlepszy parametr chłodzenia to 0.99999.



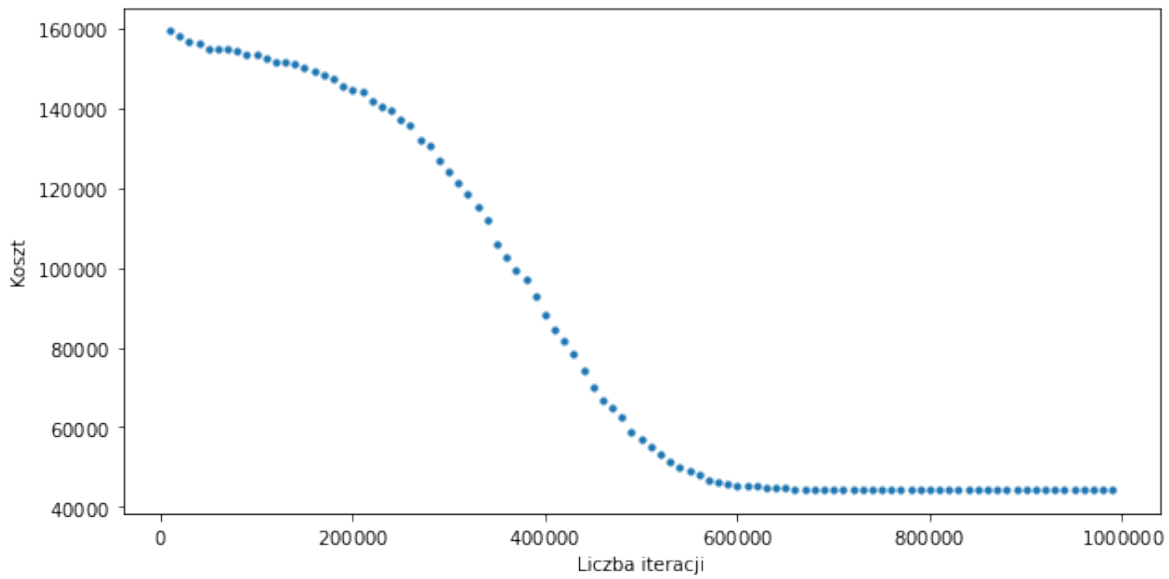
Rysunek 10: Zależność kosztu od początkowej temperatury dla pliku kro124p

Także temperatura 15000 kolejny raz okazała się być najbardziej optymalna.



Rysunek 11: Zależność kosztu od funkcji prawdopodobieństwa dla pliku kro124p

Korzystanie z funkcji gaussian w przypadku tego pliku daje najlepsze wyniki.



Rysunek 12: Zależność kosztu od liczby iteracji dla pliku kro124p

W tym przypadku najlepsze parametry to funkcja gaussian probability, z parametrem chłodzenia równym 0.99999 i początkową temperaturą równą 15000.

4.4 Najlepsze zestawy parametrów

Korzystaliśmy z funkcji, która zwracała najlepsze parametry biorąc pod uwagę końcowy koszt oraz czas.

Wyniki dla poszczególnych plików:

| | funkcja prawdopodobieństwa | parametr funkcji | parametr chłodzenia | temperatura początkowa |
|----|----------------------------|------------------|---------------------|------------------------|
| 1. | gaussian | 3.0 | 0.999999 | 15000 |
| 2. | exponential decrease | 3.0 | 0.999999 | 15000 |
| 3. | gaussian | 2.0 | 0.999999 | 15000 |
| 4. | exponential decrease | 0.3 | 0.999999 | 15000 |
| 5. | boltzman | - | 0.999999 | 15000 |

Rysunek 13: Parametry i wyniki dla pliku ft70

| | funkcja prawdopodobieństwa | parametr funkcji | parametr chłodzenia | temperatura początkowa |
|----|----------------------------|------------------|---------------------|------------------------|
| 1. | gaussian | 3.0 | 0.99999 | 15000 |
| 2. | exponential decrease | 3.0 | 0.99999 | 15000 |
| 3. | gaussian | 2.5 | 0.99999 | 15000 |
| 4. | gaussian | 1.5 | 0.99999 | 15000 |
| 5. | exponential decrease | 0.1 | 0.99999 | 15000 |

Rysunek 14: Parametry i wyniki dla pliku ftv53

| | funkcja prawdopodobieństwa | parametr funkcji | parametr chłodzenia | temperatura początkowa |
|----|----------------------------|------------------|---------------------|------------------------|
| 1. | gaussian | 0.4 | 0.999999 | 15000 |
| 2. | gaussian | 0.1 | 0.999999 | 15000 |
| 3. | gaussian | 1.0 | 0.999999 | 15000 |
| 4. | gaussian | 3.0 | 0.999999 | 15000 |
| 5. | gaussian | 2.0 | 0.999999 | 15000 |

Rysunek 15: Parametry i wyniki dla pliku kro124p

| | funkcja prawdopodobieństwa | parametr funkcji | parametr chłodzenia | temperatura początkowa |
|----|----------------------------|------------------|---------------------|------------------------|
| 1. | gaussian | 0.4 | 0.999999 | 15000 |
| 2. | gaussian | 2.0 | 0.999999 | 15000 |
| 3. | gaussian | 0.6 | 0.999999 | 15000 |
| 4. | gaussian | 0.1 | 0.999999 | 15000 |
| 5. | exponential decrease | 1.5 | 0.999999 | 15000 |

Rysunek 16: Parametry i wyniki dla pliku rbg323

Ostatecznie stwierdziliśmy, że dwa najlepsze zestawy to funkcja gaussian z parametrem 0.4 i gaussian z parametrem 3.0. Wyboru, z którego będziemy korzystać dokonywaliśmy losowo.

4.5 Wyniki wyścigu

Otrzymane wnioski wykorzystaliśmy podczas wyścigów algorytmów.

- plik ftv38
koszt: 1652
ścieżka: [3, 38, 2, 4, 35, 33, 30, 26, 23, 24, 36, 21, 22, 25, 31, 32, 34, 29, 28, 27, 20, 13, 12, 11, 15, 6, 7, 5, 8, 37, 9, 10, 16, 14, 17, 18, 19, 0, 1]

- plik ft70
koszt: 40286
ścieżka: [29, 23, 56, 54, 52, 51, 50, 17, 16, 15, 6, 38, 35, 42, 41, 40, 36, 39, 37, 69, 65, 68, 67, 63, 66, 64, 21, 27, 24, 28, 30, 34, 32, 12, 14, 10, 9, 11, 7, 13, 8, 20, 18, 19, 46, 45, 49, 47, 55, 53, 2, 4, 22, 25, 3, 5, 1, 0, 26, 33, 31, 62, 61, 60, 59, 58, 57, 44, 48, 43]
- plik ftv38
koszt: 7019
ścieżka: [131, 130, 66, 269, 147, 366, 306, 242, 262, 224, 77, 88, 334, 122, 272, 344, 193, 348, 212, 367, 328, 358, 58, 97, 208, 266, 163, 183, 113, 368, 332, 53, 341, 149, 85, 91, 71, 346, 137, 21, 168, 32, 124, 55, 63, 398, 110, 285, 196, 292, 288, 268, 199, 261, 0, 233, 222, 86, 310, 382, 302, 127, 148, 372, 307, 390, 161, 24, 335, 352, 246, 11, 396, 65, 343, 74, 384, 392, 376, 289, 40, 134, 205, 362, 339, 29, 151, 329, 60, 14, 379, 119, 181, 207, 185, 120, 68, 78, 259, 225, 312, 175, 106, 95, 2, 273, 172, 173, 235, 117, 381, 43, 234, 64, 182, 126, 287, 18, 248, 349, 359, 264, 115, 178, 31, 282, 342, 209, 146, 156, 143, 94, 249, 23, 394, 371, 364, 41, 313, 36, 93, 247, 291, 378, 320, 16, 338, 81, 57, 123, 166, 9, 164, 1, 283, 258, 238, 221, 318, 304, 388, 290, 128, 211, 165, 135, 322, 192, 237, 319, 391, 252, 360, 125, 70, 333, 385, 69, 49, 239, 72, 99, 229, 136, 256, 355, 103, 80, 167, 356, 274, 253, 317, 204, 47, 96, 220, 201, 142, 255, 155, 383, 297, 174, 38, 397, 203, 218, 6, 354, 59, 210, 375, 13, 98, 401, 278, 325, 299, 215, 267, 92, 144, 159, 244, 198, 160, 111, 271, 154, 350, 284, 377, 50, 270, 12, 214, 217, 20, 337, 386, 157, 108, 109, 33, 105, 275, 380, 277, 361, 82, 369, 186, 389, 4, 25, 374, 56, 46, 351, 293, 227, 150, 303, 301, 340, 8, 61, 311, 232, 114, 133, 353, 177, 73, 300, 162, 190, 243, 400, 326, 226, 314, 279, 327, 286, 195, 363, 52, 223, 263, 79, 280, 104, 34, 197, 112, 202, 373, 170, 260, 228, 51, 324, 45, 365, 236, 121, 44, 189, 37, 141, 116, 179, 138, 22, 308, 102, 67, 305, 316, 39, 230, 27, 17, 100, 153, 26, 7, 35, 330, 158, 323, 296, 118, 101, 219, 281, 321, 188, 240, 140, 76, 42, 15, 345, 231, 357, 184, 295, 241, 145, 48, 90, 5, 331, 75, 107, 28, 54, 265, 402, 10, 276, 132, 294, 171, 191, 187, 84, 370, 395, 216, 89, 336, 87, 315, 3, 19, 251, 206, 399, 194, 129, 347, 309, 213, 250, 298, 30, 200, 393, 169, 180, 387, 62, 254, 83, 139, 245, 176, 257, 152]

5 Podsumowanie i wnioski

Najczęściej powtarzającym się wnioskiem w naszych testach było to, że najlepsze wyniki są osiągane dla temperatury 15000 i parametrem chłodzenia 0.999999- czyli im wolniej spada temperatura tym lepiej. Najbardziej optymalne wyniki otrzymywaliśmy korzystając z funkcji Gaussian Probability z parametrem 0.3 oraz z 4.0. Wyboru pomiędzy tymi dwoma zestawami dokonywaliśmy losowo.

W niektórych przypadkach- na przykład przy pliku br17 wybór parametrów nie był aż tak istotny- najbardziej optymalny wynik był osiągany bez względu na funkcję czy szybkość chłodzenia. Jedyna różnica to czas obliczeń- również niewielka.

Algorytm w krótkim czasie znajdował bardzo dobre rozwiązania- o koszcie zbliżonym do tego najbardziej optymalnego. Zazwyczaj w przeciągu 5 minut otrzymywaliśmy wynik różniący się od najlepszego o maksymalnie 10%

Implementowany algorytm działa w taki sposób, że kandydat na minimum rzadko kiedy jest zmieniany na inny (widoczna pozioma linia na wykresie). Dlatego też czasami korzystniej byłoby w ciągu 5min przeliczyć kilka razy dany zestaw przy szybszym chłodzeniu niż czekać, aż algorytm zwróci jedno rozwiązanie. Zaobserwowaliśmy, że zazwyczaj najlepszy wynik jest osiągany po około 800 000 iteracjach, więc jeśli zależałoby nam na jak najszybszym znalezieniu rozwiązania, moglibyśmy ograniczyć ich liczbę.

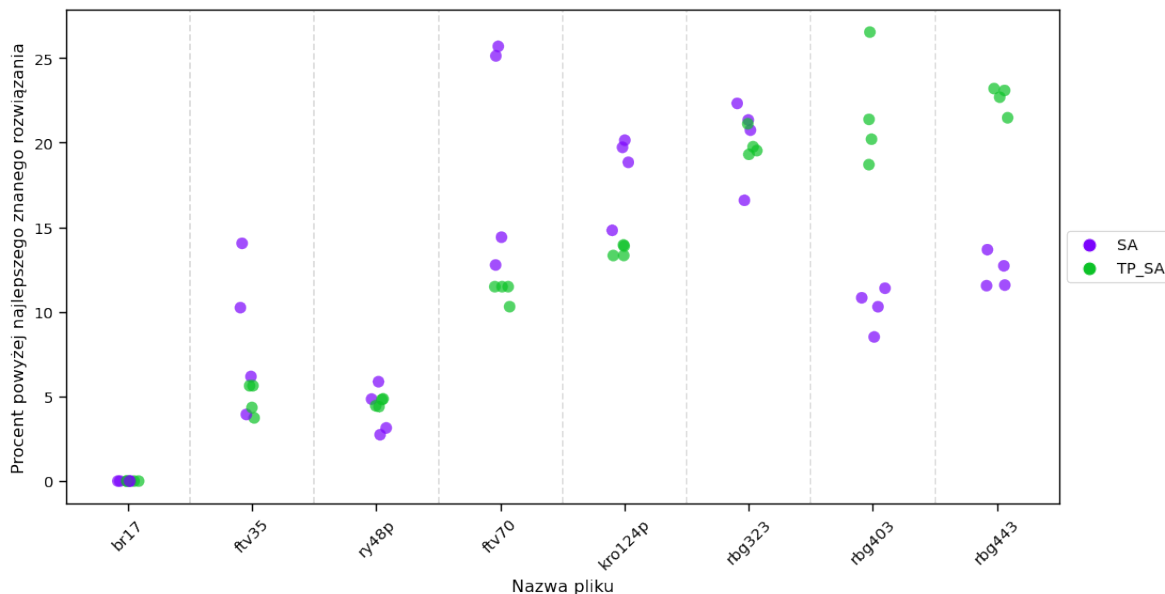
Wybierając losowo parametry nie osiągaliliśmy tak dobrych wyników. Bardzo pomocna okazała się funkcja, która wskazała nam te najbardziej optymalne wielkości. Mimo, że nie dla wszystkich plików otrzymaliśmy te same rezultaty, udało nam się wybrać dwa najbardziej uniwersalne.

Działanie algorytmu silnie zależy od początkowego wyboru ścieżki. Wybory są dokonywane losowo- między innymi to, czy zaakceptowany zostanie gorszy wynik czy też nie. Dlatego też można zauważyć duże różnice pomiędzy rozwiązaniami.

W związku z tym każdy zestaw parametrów testowaliśmy wiele razy, aby uniknąć wyciągania wniosków z losowych wyników (co można zauważyć na wykresach w Głównych wynikach).

6 Porównanie z innym zespołem

Zaimplementowany algorytm symulowanego wyżarzania porównany zostanie do algorytmu łączącego symulowane wyżarzanie z parallel tempering, który działał na zasadzie wielowątkowego wywoływania parallel tempering, używającego Metropolis Transition, następnie Replica Transition, a następnie wywoływane było symulowane wyżarzanie. Cały proces powtarzał się, w zależności od rozmiaru problemu, dwa lub sześć razy. Działanie algorytmów porównywane było na podstawie, ze względu na ograniczone zasoby czasowe, czterech wywołań każdego z nich na ośmiu różnych problemach. Dokładne nazwy problemów znajdują się na wykresie poniżej. Sprawdzane jest jak w danym wywołaniu algorytm blisko był od najlepszego znanego rozwiązania. TP_SA oznacza symulowane wyżarzanie z parallel tempering, SA symulowane wyżarzanie.



Rysunek 17: Procent ponad najlepsze znane rozwiązanie dla każdego pliku i algorytmu

W przypadku 17 miast obydwa algorytmy poradziły sobie równie dobrze, osiągając najlepsze znane rozwiązanie za każdym razem. W przypadku 35, 70 i 124 miast algorytm TP_SA charakteryzował się niewiele lepszymi wynikami. W przypadku dużej ilości miast, tutaj 403 i 443, algorytm SA konsekwentnie wypadał lepiej, nie przekraczając wyniku piętnastu procent ponad najlepsze znane rozwiązanie. Ponadto, każdy wynik dla SA był zbliżony, co świadczy również o dobrej stabilności przy takich problemach. W pozostałych przypadkach algorytmy osiągają zbliżone wyniki. Zaletą TP_SA jest stabilność wyników na niemalże każdym problemie.

Warto jednak pamiętać, iż TP_SA wymaga większych zasobów obliczeniowych, ze względu na wykorzystywanie wielowątkowości.

7 Dalsze możliwości rozwoju

Jednym z głównych problemów implementowanego algorytmu było wpadanie w minima lokalne. Z tego powodu dalszy rozwój powinien zostać rozpoczęty od rozwiązywania tego problemu. Drugim problemem było dysponowanie czasem, przez który program mógł działać. W pierwotnej wersji algorytm działał do momentu aż temperatura nie osiągnęła wcześniej zadanego poziomu, lub czas nie przekroczył pięciu minut. Przez taki wybór, podczas porównywania z rozwiązaniami innych zespołów, gdy każdy z nich miał pięć minut na znalezienie jak najlepszego rozwiązania, raz skończył on działać po około minucie, zaś raz działał na tyle wolno, że temperatura nie zdążyła zejść do niskiego poziomu, a znalezione rozwiązanie było dalekie od optymalnego.

Tak więc najprostszą modyfikacją może być zmniejszenie warunku stopu. Mianowicie dopóki czas nie przekroczy żądanej wartości w przypadku osiągnięcia granicznej temperatury program zapamiętuje

najlepsze rozwiązanie i wykonuje wyżarzanie na nowo, możliwie z mniejszą temperaturą początkową, lub szybszym chłodzeniem.

Podobnie, można by zaimplementować algorytm wykonujący wyżarzanie na tyle szybko, by wykonał on w zadanym czasie na pewno parę iteracji, lecz z różnych miejsc startowych.

Zamiast wykonywać wyżarzanie kilka razy z rzędu można by wykonywać je, używając wielowątkowości lub wieloprocusowości, kilka razy jednocześnie. To rozwiązanie jednak gorzej jest bardziej wymagające sprzętowo, więc nie sprawdzi się najlepiej na słabszych komputerach. Oczywiście, takie równoległe wywołania również można wykonać kilkukrotnie, gdy czas na to pozwala.

Ciekawym pomysłem na unikanie ekstremów lokalnych wydaje się być stosowanie w końcowych etapach wyżarzania algorytmów jak 2-opt. Polega on na przeiterowaniu przez wszystkie możliwe pary i, j , takie że $0 \leq i < j \leq n$, gdzie n jest ilością miast, i odwróceniu kolejności miast między i -tą, a j -tą pozycją. Takie podejście pomaga w wyjściu z minima lokalnego, lecz warto zauważyć, że złożoność tego algorytmu jest rzędu $O(n^2)$. Rozwiązaniem tu może być wybranie tylko pewnego podzbioru par i, j .

Ostatnim pomysłem jest dynamiczny dobór parametrów. W zależności od aktualnej temperatury lub pozostałego czasu do końca parametry jak funkcja chłodzenia, jej parametry, czy funkcja prawdopodobieństwa. W przypadku kolejnego wywołania wyżarzania także temperatury początkowej. Ponadto, aktualna wersja w każdej iteracji zamienia dwa losowo wybrane miasta miejscami szukając lepszego rozwiązania. Tutaj również początkowo, na przykład, mogła by być zamieniana większa ilość miast dla szybszej eksploracji, zaś później zmiany były by mniejsze aby szukać w obszarze podejrzanym bycie okolicą minimum. Możliwe było również użycie Sequentially Modified Cost Function, poruszone w [2]. Powyższe propozycje mogły by również być łączone, jednakże niektóre z nich mogą mieć istotny wpływ na szybkość obliczeń lub dokładność w szukaniu rozwiązań, tak więc wprowadzenie zmian musi być poprzedzone testami, a także dobierane pod zasoby czasowe i sprzętowe potencjalnych użytkowników.

8 Bibliografia

Literatura

- [1] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.
- [2] Jiayin Chen and Hendra I Nurdin. Generalized simulated annealing with sequentially modified cost function for combinatorial optimization problems. In *2019 Australian & New Zealand Control Conference (ANZCC)*, pages 48–53. IEEE, 2019.
- [3] Shi-hua Zhan, Juan Lin, Ze-jun Zhang, and Yi-wen Zhong. List-based simulated annealing algorithm for traveling salesman problem. *Computational intelligence and neuroscience*, 2016, 2016.