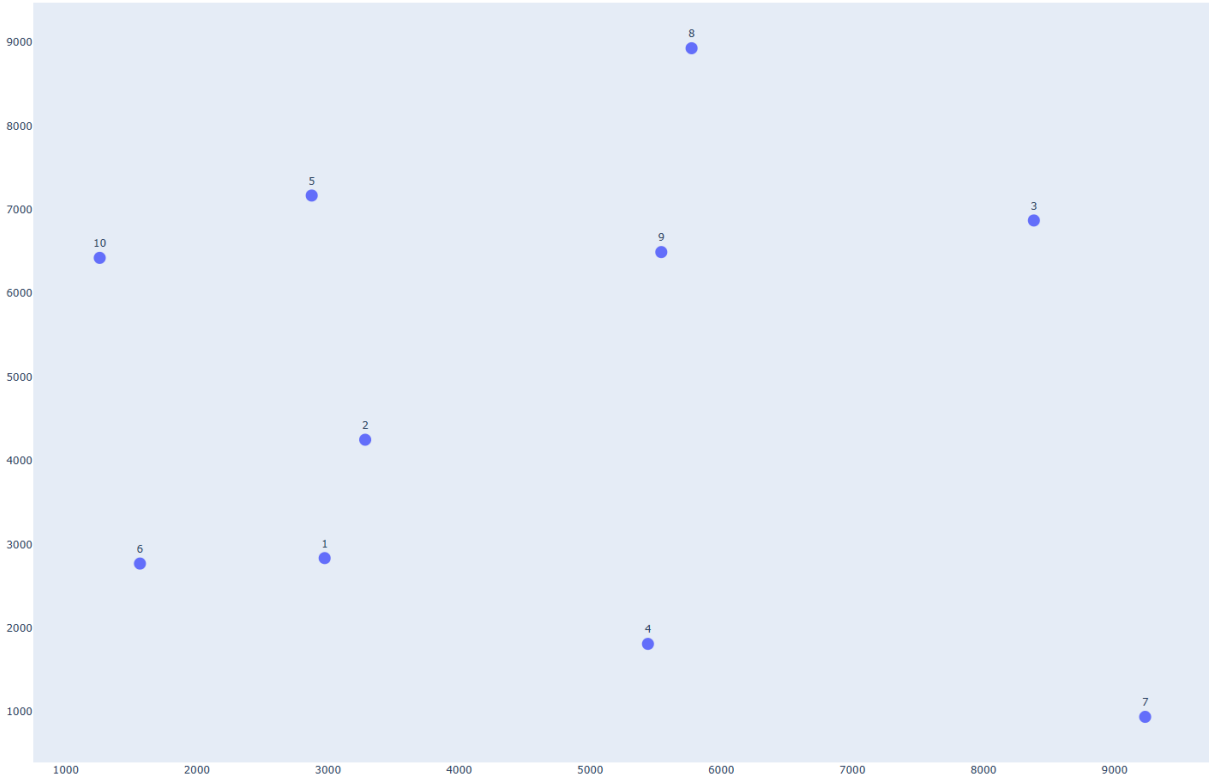
Problem komiwojażera
Sprawozdanie
Algorytm mrówkowy + 2-opt

Filip Pieprzyk 162267
Wiktor Janowski 162444

1. Inicjalizacja



2. Opis Algorytmu

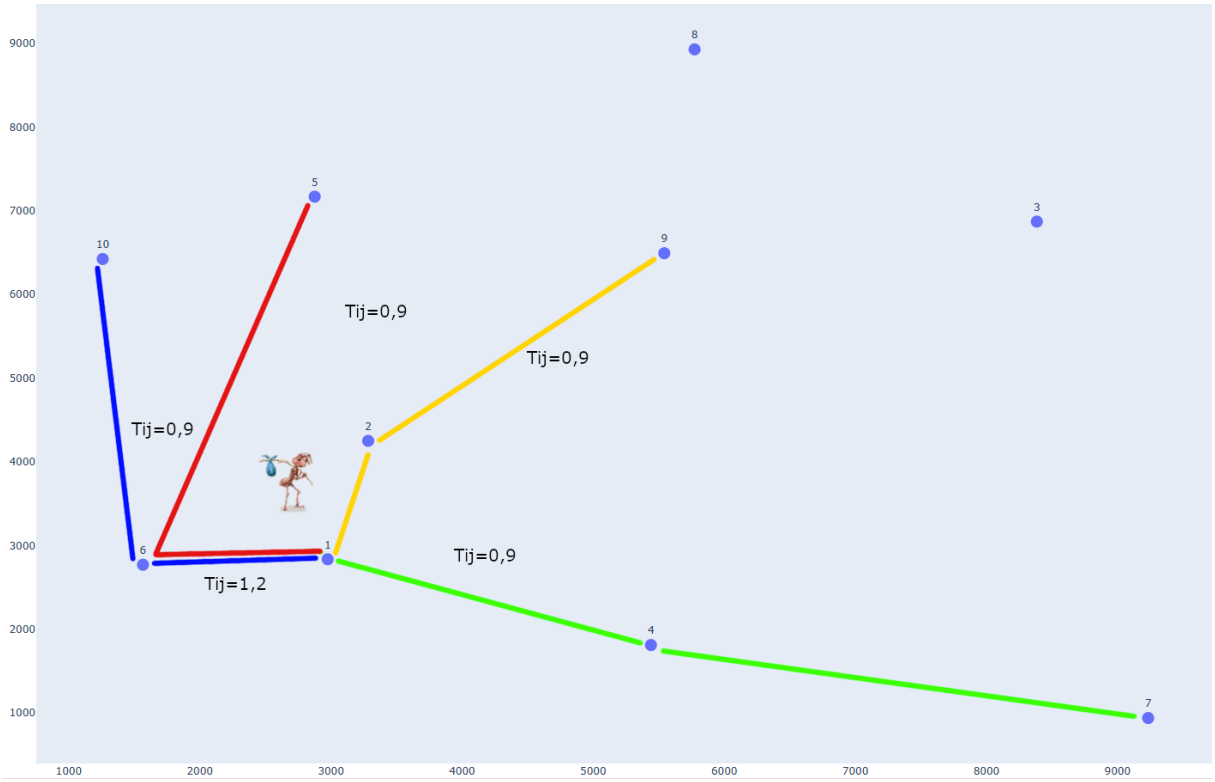
Algorytm mrówkowy jest metaheurystyką inspirowaną zachowaniem rzeczywistych mrówek, które poszukują najkrótszych ścieżek między gniazdem a źródłem pożywienia. Sztuczne mrówki przemieszczają się pomiędzy punktami, reprezentującymi miasta, tworząc permutacje tras na podstawie poziomu feromonu oraz atrakcyjności heurystycznej, będącej odwrotnością odległości. Im wyższy poziom feromonu i im krótsza odległość, tym większe prawdopodobieństwo wyboru danej krawędzi. Po każdej iteracji poziomy feromonu są aktualizowane - trasy krótsze wzmacniają feromony bardziej, a dodatkowo następuje ich parowanie, co zapobiega zbytniemu skupieniu się algorytmu na jednej ścieżce. W celu poprawy lokalnej jakości rozwiązania, na końcu każdej konstrukcji trasy wykonywana jest optymalizacja metodą 2-opt, polegająca na zamienianiu fragmentów ścieżki w taki sposób, by skrócić jej długość. Cały proces powtarza się przez zadaną liczbę iteracji, a jako wynik zwracana jest najlepsza znaleziona trasa oraz jej długość.

3. Pseudokod

```
dla i = 0 do n - 1:
  dla j = 0 do n - 1:
    feromony[i][j] = 1

dla iteracja = 1 do ilość_iteracji:
  dla m = 0 do liczba_mrówek - 1:

    start = wybierz_losowe_miasto()
    odwiedzone = tablica[n] wypełniona False
    ścieżka = tablica[n + 1] wypełniona -1

    ścieżka[0] = start
    odwiedzone[start] = True

    dla krok = 1 do n - 1:
      aktualne = ścieżka[krok - 1]
      suma_atrakcyjności = 0
      atrakcyjność = tablica[n]

      dla u = 0 do n - 1:
        jeżeli odwiedzone[u]:
          atrakcyjność[u] = 0
        inaczej:
          tau = feromony[aktualne][u] ^ alpha
          eta = (1 / odległość[aktualne][u]) ^ beta
          atrakcyjność[u] = tau * eta
          suma_atrakcyjności += atrakcyjność[u]

      los = wartość losowa z przedziału [0, suma_atrakcyjności]

      jeżeli losowa() <= q0:
        u = indeks_maksymalnej_atrakcyjności(atrakcyjność)
      inaczej:
        u = wybierz_proporcjonalnie(atrakcyjność, suma_atrakcyjności)

      ścieżka[krok] = u
      odwiedzone[u] = True

    ścieżka[n] = ścieżka[0]

    zastosuj_2opt_na(ścieżka, macierz_odległości)

    długość_ścieżki = suma odległości między kolejnymi miastami w ścieżce

    jeżeli długość_ścieżki < długość(global_best):
      global_best = ścieżka
```

```
  dla i = 0 do n - 1:
    a = ścieżka[i]
    b = ścieżka[i + 1]
    feromony[a][b] += Q / długość_ścieżki
    feromony[b][a] = feromony[a][b]

dla i = 0 do n - 1:
  dla j = 0 do n - 1:
    feromony[i][j] *= (1 - parowanie)
```

4. Przykład obrazujący działanie

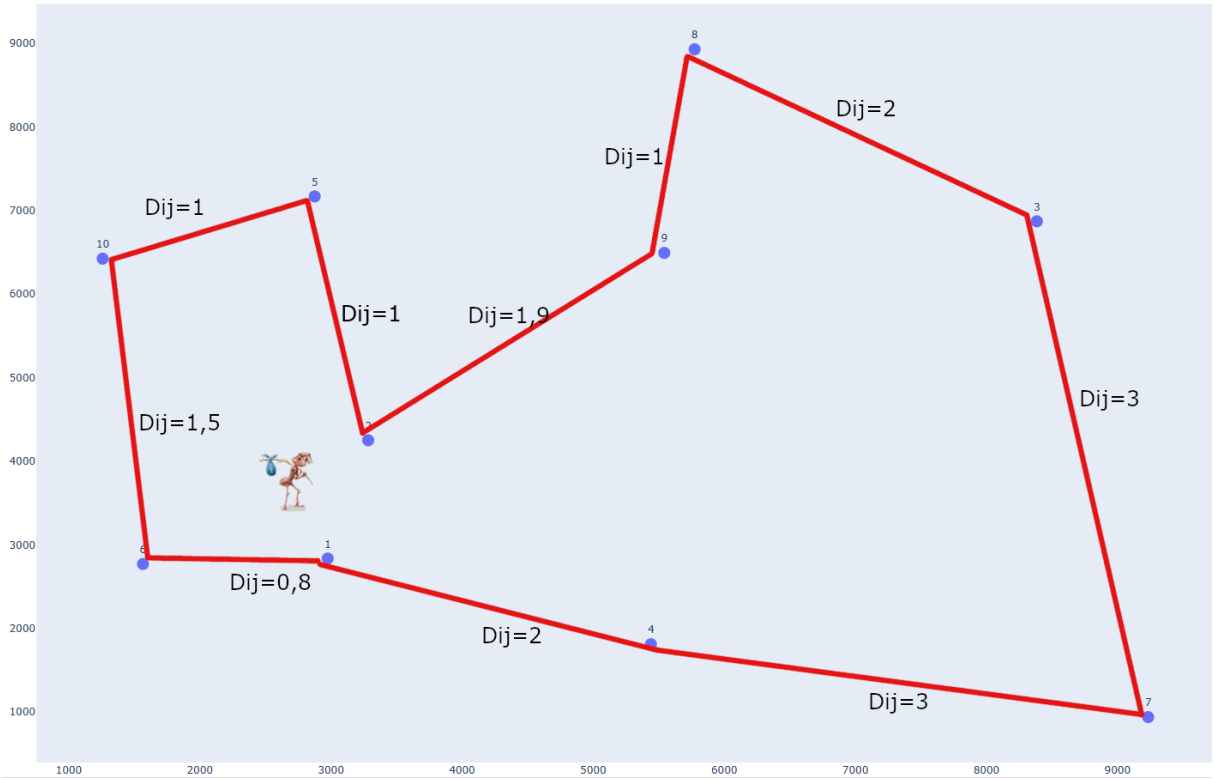


Każda mrówka rozpoczyna trasę od losowego wierzchołka i buduje ją krok po kroku, wybierając kolejne wierzchołki spośród nieodwiedzonych. Mrówka oblicza prawdopodobieństwo wyboru każdego z sąsiadów, używając feromonów i heurystyki. Początkowo wszystkie punkty mają jednakowe poziom fermonów równy 1, a więc wybór jest losowy. W przypadku dalszych iteracji działa pseudoruletka która wybiera wierchołek na podstawie prawdopodobieństwa wyboru. Wzór na prawdopodobieństwo:

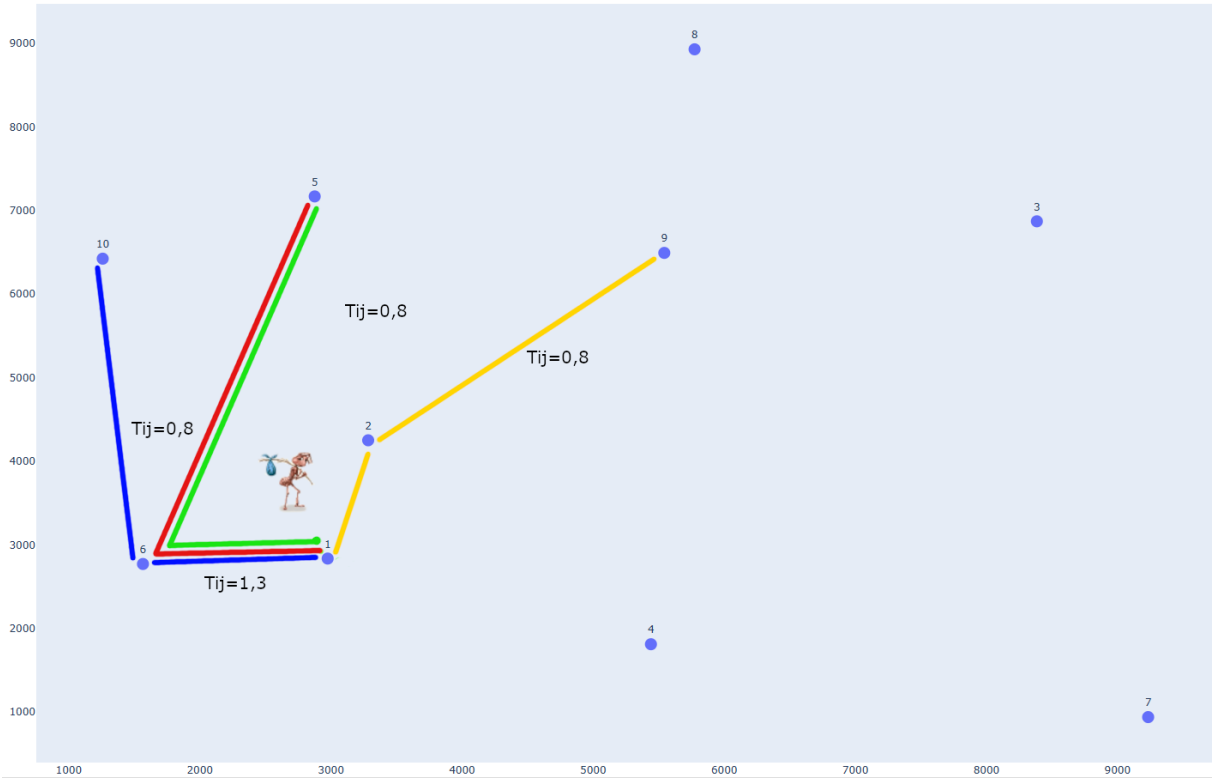
$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in N_i} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}}$$

gdzie:

- P_{ij}: prawdopodobieństwo wyboru wierzchołka j przez mrówkę znajdującą się w i,
- τ_{ij}: ilość feromonów na krawędzi (i, j),
- $\eta_{ij} = \frac{1}{d_{ij}}$: odwrotność odległości,
- α: wpływ feromonów na wybór,
- β: wpływ heurystyki na wybór,
- N_i: zbiór sąsiadów możliwych do odwiedzenia z i.

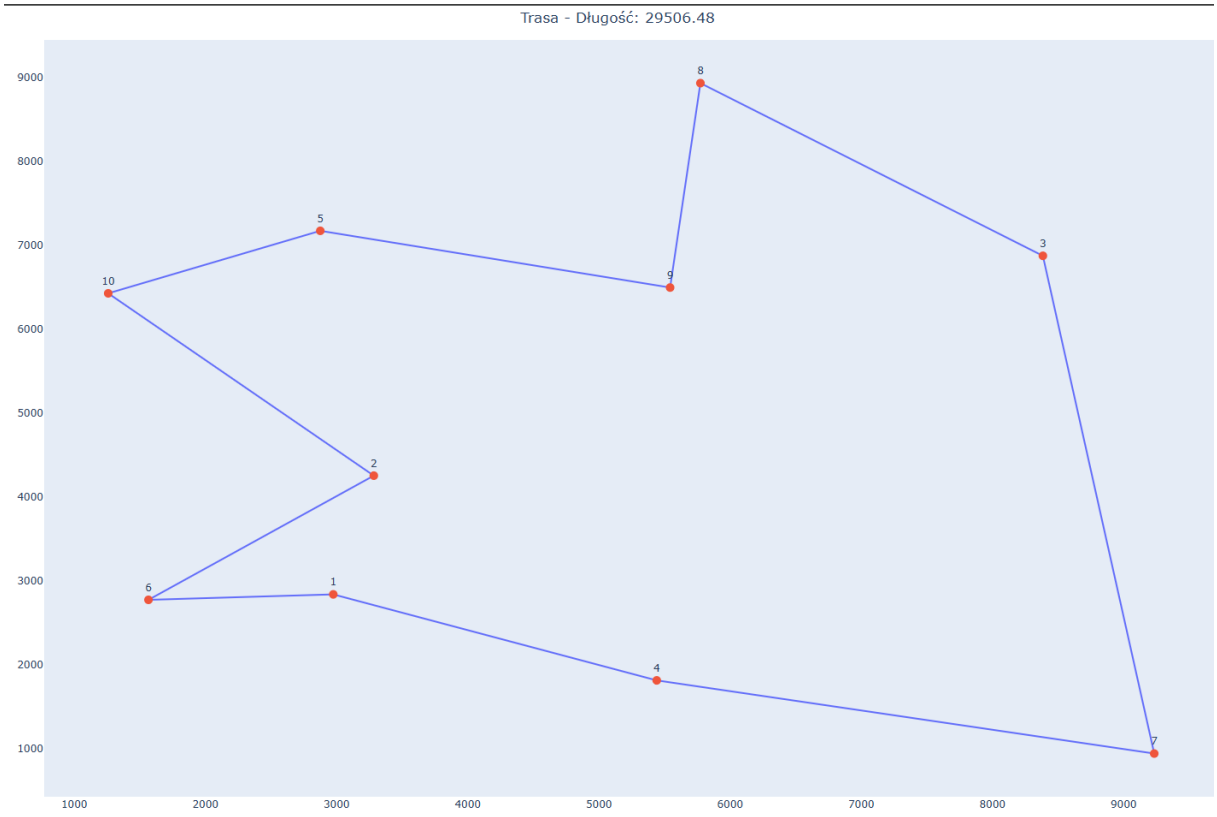


Po odwiedzeniu wszystkich wierzchołków, mrówka wraca do punktu startowego. Obliczana jest odległość wszystkich tras i wybierana ta najlepsza. Do czasu znalezienia w kolejnych iteracjach krótszej ścieżki, obecna najlepsza trasa jest uważana za optymalną. W tym momencie działa też algorytm 2opt, który bada czy występują przecięte ścieki. Jeżeli znajdzie takową, to zamienia dwie sąsiednie i sprawdza czy trasa nie jest krótsza.



Po iteracji następuje parowanie wszystkich fermonów o współczynnik parowania. Następuje też wzrost poziomu fermonów o iloraz $\frac{Q}{L}$ gdzie L to długość. W ten sposób ścieżki częściej odwiedane są bardzie atrakcyjne w przyszłych iteracjach.

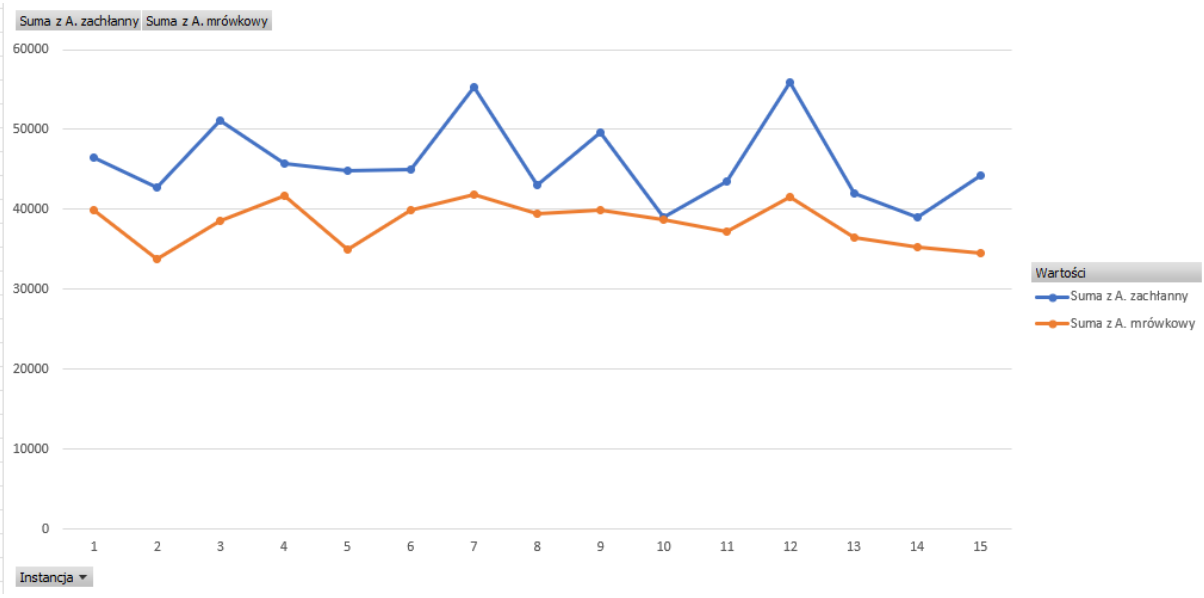
5. Finalizacja



Po wyznaczonej liczbie iteracji otrzymujemy cykl. Przy odpowiednim doborze parametrów będzie on o wiele krótszy od standardowego algorytmu zachłannego.

Wyniki i analiza

1. Porównanie wyników A. zachłannego i A. mrówkowego dla 15 losowych instancji.



Widać że w każdym scenariuszu algorytm mrówkowy jest lepszy od algorytmu zachłannego. Świadczy to o dobrym dostrojeniu parametrów. Średnio trasy wyznaczone przy pomocy a. mrówkowego są lepsze o $19,88\%$.

2. Błąd względny względem wartości optymalnych

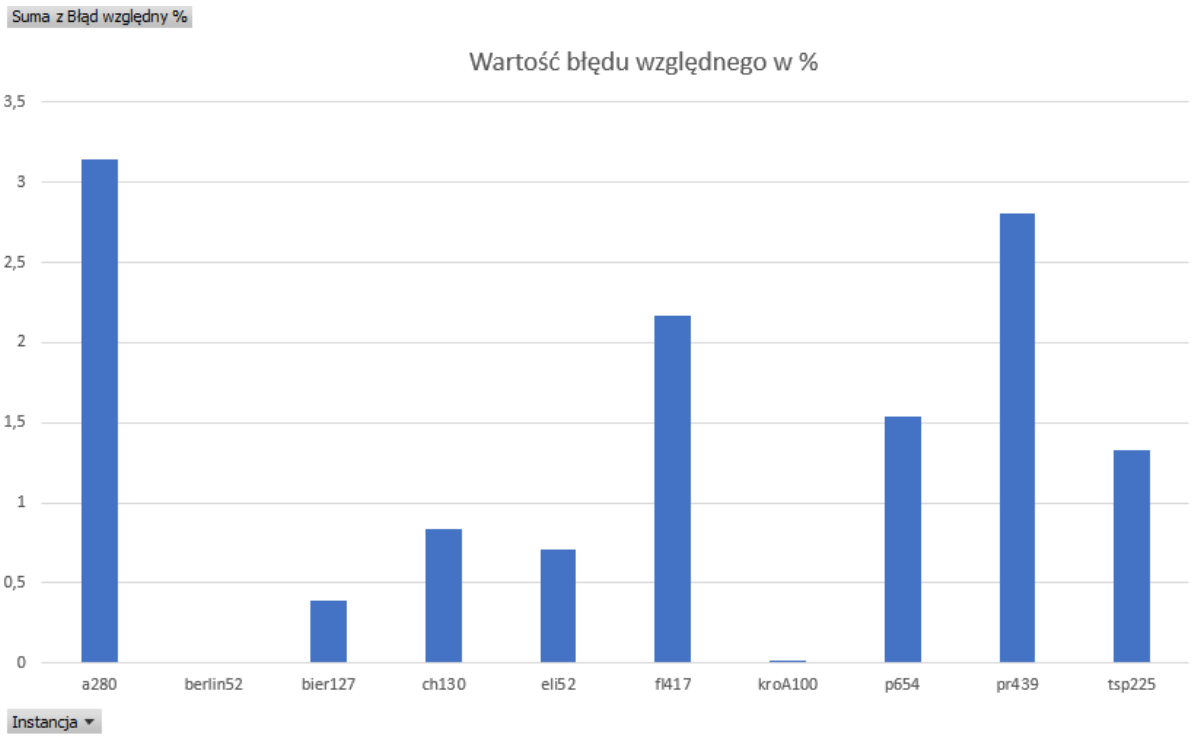


Figure 1: Wartość błędu względnego algorytmu mrówkowego w stosunku do wartości optymalnej

3. Tabela rankingowa

Instancja	**Wynik**
berlin52	7544
bier127	118747
tsp250	12693
tsp500	86611
tsp1000	23373