Systemy Inteligentne

Sprawozdanie 3

Algorytmy mrowiskowe

Laboratorium czwartek godzina 8:00

Paweł Lurka
Paweł Chmielarski
Informatyka IV rok 2019/2020, IO1

1. Wstęp teoretyczny

Algorytmy mrówkowe to probabilistyczne techniki znajdywania rozwiązań w problemach kombinatorycznych i optymalizacji problemów które można przedstawić w reprezentacji grafowej. Ten stosunkowo nowy gatunek algorytmów został stworzony przez Marco Dorigo w 1992 na podstawie obserwacji zachowań mrówek jako jednostek i działań mrowiska jako całości. Podstawowym problemem mrowiska jest znalezienie źródła pożywienia, a następnie przetransportowanie go do koloni. Mimo iż pojedyncze jednostki są bardzo ograniczone w percepcji i zachowaniach to efektywność działania koloni wynika ze współpracy jednostek i komunikowania się za pomocą pozostawianego śladu feromonowego.

Algorytmy mrówkowe w swoim uogólnieniu posiadają następujące cechy:

- Optymalizacja przeprowadzana jest iteracyjnie przez stałą, określoną na początku, liczbę agentów
- Agenci komunikują się poprzez informacje zostawianą w otoczeniu, która reprezentowana jest przez ślad feromonowy. Ilość pozostawianego przez mrówkę feromonu zależy od jakości rozwiązania uzyskanego przez mrówkę.
- Agenci losowo przeszukują przestrzeń, faworyzując rozwiązania wskazywane im przez informację z otoczenia.
- Poszczególne mrówki nie posiadają własnej inteligencji a jedynie pamięć która umożliwia im zapamiętanie trasy.

Opisywany algorytm doskonale nadaje się do rozwiązywania **problemu komiwojażera,** który polega na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Oznacza to że należy znaleźć trasę która przebiega przez każdy wierzchołek grafu (miasto) dokładnie raz i wraca do punktu początkowego. Punkt początkowy jest tu dowolnym wierzchołkiem grafu.

W niniejszej pracy analizowane będą tylko symetryczne problemy komiwojażera (sTSP), które polegają na tym że dla dowolnych punktów A i B odległość/koszt przejścia z A do B jest taka sama jak z B do A.
sTSP posiada liczbę możliwych rozwiązań która wynosi $\frac{(n-1)!}{2}$ gdzie n to liczba wierzchołków grafu.

Adaptacja algorytmu **ACO** (ant colony optimization) do rozwiązania problemu aTSP prezentuje się następująco:

- inicjuje się połączenia między punktami niewielką ilością feromonu.
- Wszystkie mrówki umieszczane są w losowo wybranych wierzchołkach grafu. Mrówka przemieszcza się do kolejnego miasta z prawdopodobieństwem P które jest funkcją odległości do celu oraz wielkości śladu feromonowego pozostawionego na danej krawędzi. Prawdopodobieństwo $P_{ij}(t)$ w danej chwili można opisać wzorem:

$$P_{ij}(t) = \begin{cases} \frac{\left[\tau_{ij}\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum_{j \in \Omega} \left[\tau_{ij}\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}, & \text{dla } j \in \Omega \end{cases}$$

gdzie:

τ_{ij} - natężenie śladu feromonowego na krawędzi,
η_{ij} - wartość funkcji kryterium związana z widocznością punktu j z punktu i,
α - parametr sterujący ważnością intensywności śladu feromonowego τ_{ij},
β - parametr sterujący ważnością widoczności następnego miasta,
Ω - zbiór miast nie odwiedzonych dotychczas przez mrówkę.

- Przy przejściu z miasta i do j mrówka pozostawia za sobą ślad feromonowy odwrotnie proporcjonalny do długości trasy którą przebyła

$$\eta_{ij} = \frac{1}{d_{ij}^2}$$

gdzie d_{ij} to długość krawędzi pomiędzy miastami i oraz j.

- Pojedyncza mrówka wyposażona jest w pamięć, reprezentowaną przez **tablicę tabu** w której zapisywana jest lista miast odwiedzonych w danej iteracji. Pamięć ta jest czyszczona po każdym cyklu.
- Aby uniknąć zjawiska przedwczesnej zbieżności algorytmu wprowadza się odparowywanie po każdej iteracji śladu feromonowego, który jest pomniejszany o określoną wartość. Wartość tą można wyrazić wzorem:

$$\tau_{rs}(t, t+n) = (1-\rho) \cdot \tau_{rs}(t) + \alpha \cdot \sum_{k=1}^{m} \Delta\tau_{rs}^{k}(t, t+n),$$

gdzie:

ρ — współczynnik zwany **wyparowywaniem feromonu** taki, że $(1-\rho)$ reprezentuje poświatę feromonu; jest to liczba z przedziału $\langle 0, 1\rangle$,

$\tau_{rs}(t)$ — wielkość śladu feromonowego,

(r, s) — krawędź najlepszej znalezionej trasy w ramach jednostkowej iteracji,

m — liczba mrówek, które przeszły przez (r, s) pomiędzy chwilami (t) a $(t+n)$.

- **Współczynniki** determinujące działanie algorytmu ACO

➔ Liczba mrówek
➔ Liczba iteracji
➔ ρ - współczynnik parowania

- ➔ q - ilość feromonu do rozdysponowania przez pojedynczą mrówkę
- ➔ α - waga feromonu w wyborze
- ➔ β waga heurystyki w wyborze

Powyższe współczynniki dobierane są eksperymentalnie.

- **Obliczanie odległości**

Standardowo odległość między wierzchołkami i oraz j dla odległości euklidesowej oblicza się ze wzoru:

$$\delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Często jednak w zbiorach przykładowych problemów TSPLIB podawane są **współrzędne geograficzne** i w tym przypadku obliczanie odległości prezentuje się następująco:

Mając dane 2 punkty o określonych współrzędnych geograficznych, obliczamy ich koordynaty sferyczne zapisane w radianach:

$\varphi_1 = (90^o - x_{1s}) * \frac{\pi}{180^o}$ gdzie x_{1s} to szerokość geograficzna pierwszego punktu

$\varphi_2 = (90^o - x_{2s}) * \frac{\pi}{180^o}$ gdzie x_{2s} to szerokość geograficzna drugiego punktu

$\lambda_1 = (90^o - y_{1w}) * \frac{\pi}{180^o}$ gdzie y_{1w} to wysokość geograficzna pierwszego punktu

$\lambda_2 = (90^o - y_{2w}) * \frac{\pi}{180^o}$ gdzie y_{2w} to wysokość geograficzna drugiego punktu

$$\Delta\lambda = \lambda_2 - \lambda_1$$

$R = 6371\,km$ - promień kuli ziemskiej

Odległość d pomiędzy dwoma punktami obliczamy:

$$d = \text{acos}(\sin\varphi_1 \cdot \sin\varphi_2 + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \cos\Delta\lambda) \cdot R$$

1. Implementacja

W niniejszej implementacji wykorzystane zostały biblioteki „tsplib95” służąca do odczytu danych zapisanych w formacie *.tsp oraz ich rozwiązań *.opt.toura także biblioteka „acopy” służąca do przeprowadzania badań algorytmem mrówkowym, strojenia parametrów i wyznaczająca koszt przejścia według zasad opisanych w poprzednim punkcie.

Korzystając z tych bibliotek pierwszym krokiem było utworzenie obiektu klasy Solver szukającego najlepszego rozwiązania przy pomocy kolonii mrówek oraz klasy Colony opisującego kolonię i określenie ich parametrów.

```
solver = acopy.solvers.Solver(rho=.03, q=1)
colony = acopy.ant.Colony(alpha=1, beta=3)
```

Parametry, które podajemy przy tworzeniu powyższych obiektów to kolejno:

- rho (float) – procent feromonu, który wyparowuje z każdą iteracją
- q (float) – ilość feromonu, którą może pozostawić pojedyncza mrówka
- alpha (float) – waga feromonu podczas wyboru drogi
- beta (float) – waga dystansu do pokonania podczas wyboru drogi

Kolejnym krokiem było wczytanie danych z plików .tsp za pomocą biblioteki tsplib95 do obiektów klasy Problem.

```
problem = tsplib95.load_problem('ulysses16.tsp', distance_on_unit_sphere)
problem.special=distance_on_unit_sphere

problem2 = tsplib95.load_problem('st70.tsp')
```

Jak widać na powyższym zrzucie ekranu podczas wczytywania pliku można określić funkcję, która będzie odpowiadała za przeliczanie odległości między dwoma punktami. O ile punkty na płaszczyźnie wyliczane są bez najmniejszego problemu to już współrzędne geograficzne, a więc umieszczone na kuli wyliczane będą w inny sposób. Aby otrzymać poprawne wyniki należało napisać funkcję dokonującą takiego przeliczenia.

```python
import math

def distance_on_unit_sphere(a, b):
    lat1 = a[0]
    long1 = a[1]
    lat2 = b[0]
    long2 = b[1]
    # Przelicz szerokość i wysokość geograficzną na
    # współrzędne na kuli w radianach.
    degrees_to_radians = math.pi / 180.0

    # phi = 90 - szerokość
    phi1 = (90.0 - lat1) * degrees_to_radians
    phi2 = (90.0 - lat2) * degrees_to_radians

    # theta = wysokość
    theta1 = long1 * degrees_to_radians
    theta2 = long2 * degrees_to_radians

    # Oblicz odległość na kuli ze współrzędnych na kuli.
    cos = (math.sin(phi1) * math.sin(phi2) * math.cos(theta1 - theta2) +
           math.cos(phi1) * math.cos(phi2))
    arc = math.acos(cos) * 6371

    # arc został przemnożony przez współrzędne ziemi w kilometrach (6371)
    return arc
```

Po załadowaniu problemu dane z biblioteki zapisywane są w następujący sposób

```python
    # specification
    self.edge_weight_type = kwargs.get('EDGE_WEIGHT_TYPE')
    self.edge_weight_format = kwargs.get('EDGE_WEIGHT_FORMAT')
    self.edge_data_format = kwargs.get('EDGE_DATA_FORMAT')
    self.node_coord_type = kwargs.get('NODE_COORD_TYPE')
    self.display_data_type = kwargs.get('DISPLAY_DATA_TYPE')

    # data
    self.depots = kwargs.get('DEPOT_SECTION')
    self.demands = kwargs.get('DEMAND_SECTION')
    self.node_coords = kwargs.get('NODE_COORD_SECTION')
    self.edge_weights = kwargs.get('EDGE_WEIGHT_SECTION')
    self.display_data = kwargs.get('DISPLAY_DATA_SECTION')
    self.edge_data = kwargs.get('EDGE_DATA_SECTION')
    self.fixed_edges = kwargs.get('FIXED_EDGES_SECTION', set())

    self.wfunc = None
    self.special = special
```

Po drodze można wczytać także plik z rozwiązaniem problemu *.opt.tour by móc porównać wyniki po wykonanych obliczeniach.

```
solution = tsplib95.load_solution('ulysses16.opt.tour')
solution2 = tsplib95.load_solution('st70.opt.tour')
```

Następnie przekazujemy dane do obiektu klasy Graph metodą get_graph() klasy Problem. Graph ten zawiera informacje o wierzchołkach i krawędziach potrzebne do znalezienia optymalnej trasy. Wybieramy ilość mrówek, które będą szukać rozwiązania oraz ilość iteracji po których algorytm zakończy swoje działanie i przekazujemy te parametry razem z grafem i kolonią do funkcji solve a wynik klasy Solution zapisywany jest do zmiennej.

```
tour = solver.solve(G, colony, ants, iterations)
tour2 = solver.solve (G2, colony, ants, iterations)
```

Na koniec pozostaje wyświetlić dane i porównać je z oficjalnymi wynikami.

```
print("ulysses16 external solution: ", problem.trace_tours(solution))
print("ulysses16 cost: ", tour.cost)
print("ulysses16 nodes: ", tour.nodes)
print("st70 external solution: ", problem2.trace_tours(solution2))
print("st70 cost: ", tour2.cost)
print("st70 nodes: ", tour2.nodes)
```

Badania (Paweł Chmielarski)

Badania przeprowadzono na przykładowych zbiorach danych pobranych ze strony: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

Wybrano 3 zbiory danych:

1. ulyssess16.tsp – współrzędne geograficzne
2. att48.tsp – współrzędne typ ATT
3. eil51.tsp – współrzędne euklidesowe 2d

Badania wykonano w następującej konwencji:

- Wybrano kilka zestawów parametrów
- Dla każdego zestawu parametrów wykonano algorytm 9 razy i obliczono średnie odchylenie wyników od rozwiązania optymalnego dla danego problemu. Sposób wykonywania obliczeń przedstawia się następująco:

```
ants = 25
iterations = 100
deviation1 = []
solver = acopy.solvers.Solver(rho=0.03, q=1)
optimal_solution = 6859

for i in range(0, 9):
    G = problem.get_graph()
    colony = acopy.ant.Colony(alpha=1, beta=3)
    tour = solver.solve(G, colony, ants, iterations)
    print(tour.cost)
    deviation1.append(abs(optimal_solution-tour.cost))

result1 = sum(deviation1) / len(deviation1)
print("Ants: ", ants, "Iterations: ", iterations)
print("Averave deviation: ", result1)
print("nodes: ", tour.nodes)
print("\n")
```

- Po wykonaniu obliczeń na pierwszym zbiorze danych wyciągnięto wnioski i dopasowano zestawy parametrów dla kolejnego zbioru danych.

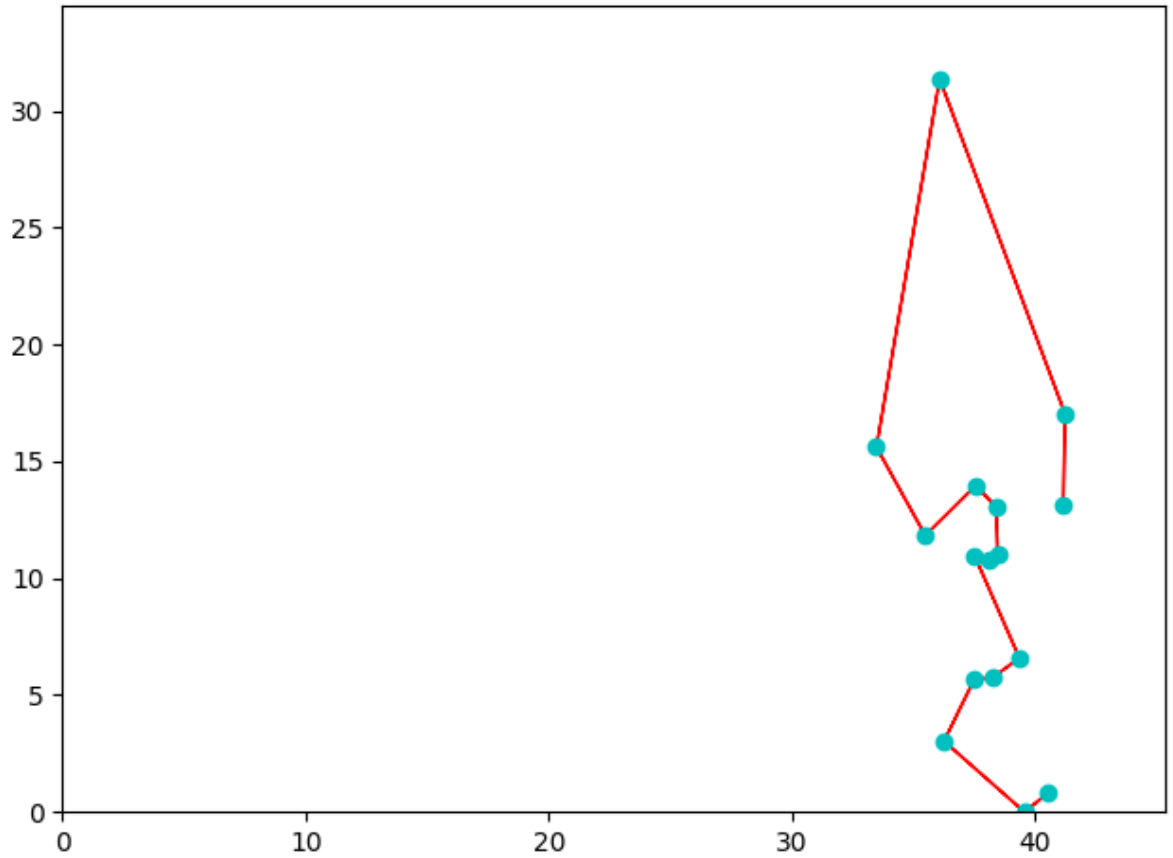
1. ulyssess16.tsp

```
NAME: ulysses16.tsp
TYPE: TSP
COMMENT: Odyssey of Ulysses (Groetschel/Padberg)
DIMENSION: 16
EDGE_WEIGHT_TYPE: GEO
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
 1 38.24 20.42
 2 39.57 26.15
 3 40.56 25.32
 4 36.26 23.12
 5 33.48 10.54
 6 37.56 12.19
 7 38.42 13.11
 8 37.52 20.44
 9 41.23 9.10
 10 41.17 13.05
 11 36.08 -5.21
 12 38.47 15.13
 13 38.15 15.35
 14 37.51 15.17
 15 35.49 14.32
 16 39.36 19.56
 EOF
```

Rozwiązanie optymalne wynosi: 6859
Im mniejsze odchylenie, tym lepsze rozwiązanie.

Lp.	L. mrówek	iteracje	ρ	q	α	β	Odchylenie
1	25	100	0.03	1	1	3	96.265
2	25	100	0.2	1	1	2	27.805
3	25	300	0.2	1	1	2	42.802
4	25	100	0.2	1	2	1	1900.159
5	25	100	0.4	1	1	2	24.458
6	100	100	0.2	1	1	2	48.164
7	50	100	0.2	1	1	1	21.901
8	30	100	0.1	1	1	2	35.318
9	40	100	0.5	1	1	2	32.160
10	100	300	0.2	1	3	1	1664.807
11	100	300	0.2	1	1	3	66.358
12	25	100	0.05	1	1	2	73.631
13	25	100	0.2	5	1	2	28.387

Wizualizacja rozwiązania nr 7:



Numery kolejno odwiedzanych węzłów:
path = [3, 2, 4, 8, 1, 16, 14, 13, 12, 7, 6, 15, 5, 11, 9, 10]

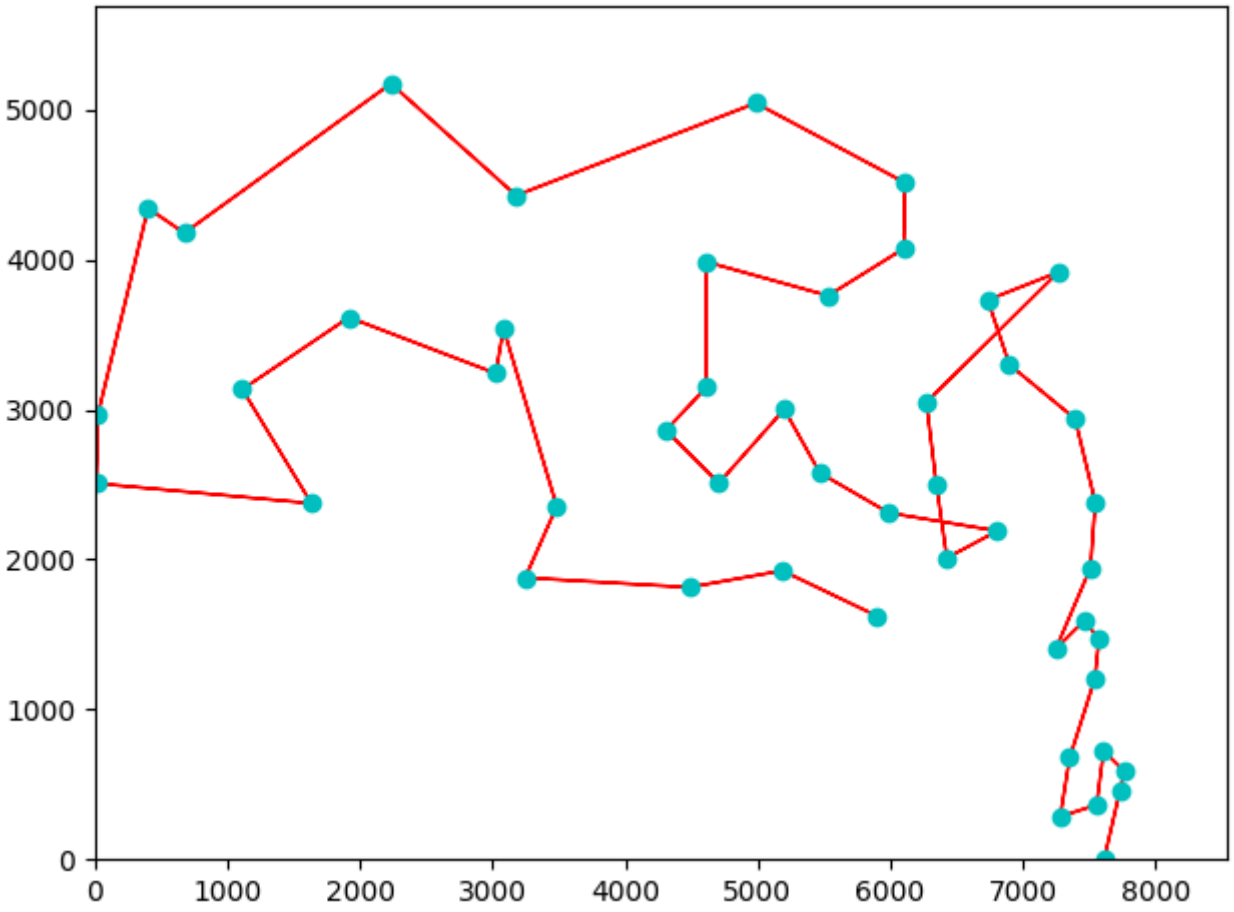
Następnie przeanalizowano powyższe wyniki, odrzucono najgorsze zestawy parametrów i wybrano nowe zestawy parametrów dla kolejnego pliku tsp.

2. att48.tsp

Rozwiązanie optymalne wynosi: 33522

Lp.	L. mrówek	iteracje	ρ	q	α	β	Odchylenie
1	25	100	0.2	1	1	2	3583.77
2	25	100	0.2	5	1	3	3298.55
3	25	100	0.4	1	1	2	3514.11
4	100	100	0.2	1	1	2	2187.88
5	50	100	0.2	1	1	1	5717.44
6	30	100	0.1	1	1	2	3440
7	40	100	0.5	1	1	2	2757.77
8	25	100	0.2	5	1	2	3821.0

Wizualizacja rozwiązania nr 4:



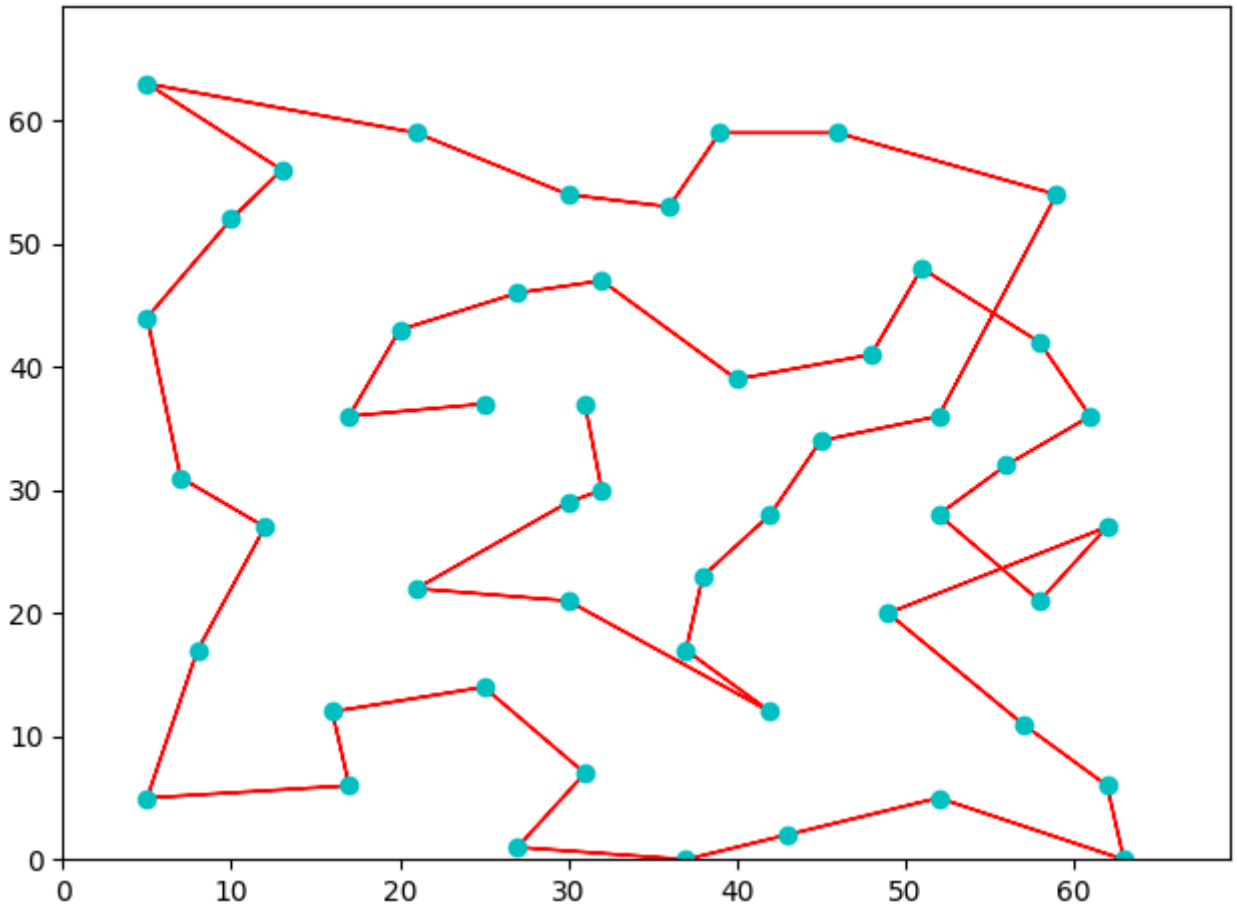
`path` = [17, 19, 37, 6, 27, 43, 30, 28, 7, 18, 36, 44, 31, 38, 9, 1, 8, 40, 15, 33, 46, 12, 11, 23, 13, 25, 14, 34, 3, 22, 16, 41, 29, 2, 26, 4, 35, 45, 24, 10, 42, 48, 5, 39, 32, 21, 47, 20]

3. eil51.tsp

Rozwiązanie optymalne wynosi: 426

Lp.	L. mrówek	iteracje	ρ	q	α	β	Odchylenie
1	200	100	0.2	1	1	2	32
2	25	100	0.2	5	1	3	50.444
3	100	100	0.4	1	1	2	33.777
4	200	100	0.2	3	1	2	33.333
5	200	100	0.2	1	1	1	60.444
6	200	200	0.2	1	1	2	25.222
7	100	100	0.3	1	2	2	136.777
8	200	100	0.2	5	1	5	27.222

Wizualizacja rozwiązania nr 6:



path = [47, 18, 4, 17, 37, 5, 49, 10, 30, 34, 50, 16, 29, 21, 2, 20, 35, 36, 3, 28, 31, 26, 8, 48, 23, 7, 43, 24, 14, 25, 13, 41, 19, 40, 42, 44, 15, 45, 33, 39, 9, 38, 11, 32, 1, 22, 27, 6, 51, 46, 12]

Badania (Paweł Lurka)

Badania przeprowadzono na przykładowych zbiorach danych pobranych ze strony: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

Wybrano 2 zbiory danych:

1. gr96.tsp – współrzędne geograficzne
2. berlin52.tsp – współrzędne typ ATT

Badania wykonano w następującej konwencji:

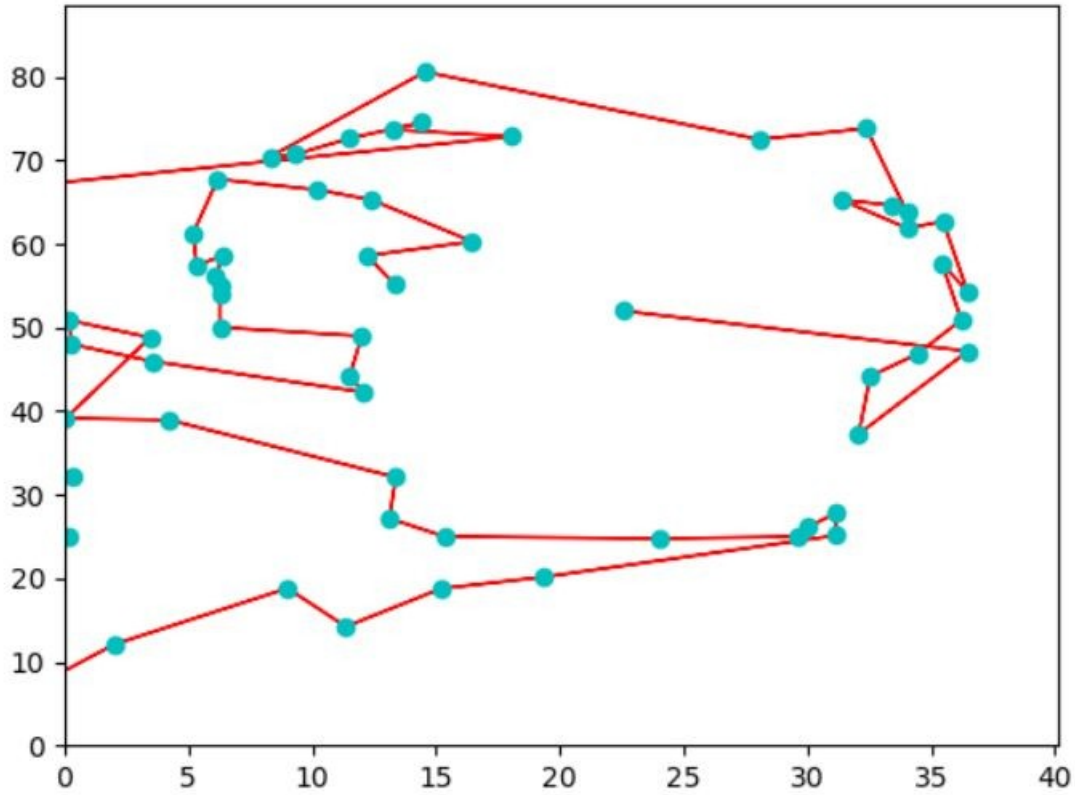
- z książki Thomasa Stutzle dobrano wartości parametrów odpowiednio
  - ρ = 0.2
  - $q=1$
  - $\alpha=1$
  - $\beta=2$
  - Liczba mrówek = 25
  - Iteracje = 150
- Parametry były zwiększane w kolejnych iteracjach odpowiednio
  - ρ było zwiększane o 0.05 co 2 iteracje
  - α było równe 1 w pierwszych 3ch iteracjach, 2 w kolejnych 2ch i 3 w 2ch ostatnich
  - β zmieniała się w zależności od α między wartościami 1,2 i 3
  - Liczba była zwiększana o 5 co iterację
  - Liczba iteracji zwiększała się o 25 co iterację

1. gr96.tsp

Rozwiązanie optymalne wynosi: 514

Lp.	L. mrówek	iteracje	ρ	q	α	β	Odchylenie
1	25	150	0.2	1	1	2	187.2
2	30	175	0.25	1	1	2	142.2
3	35	200	0.25	1	1	3	104.2
4	40	225	0.3	1	2	1	223.0
5	45	250	0.3	1	2	3	189.6
6	50	275	0.35	1	2	1	225.6
7	55	300	0.35	1	2	2	221.0

Wizualizacja rozwiązania nr 3:



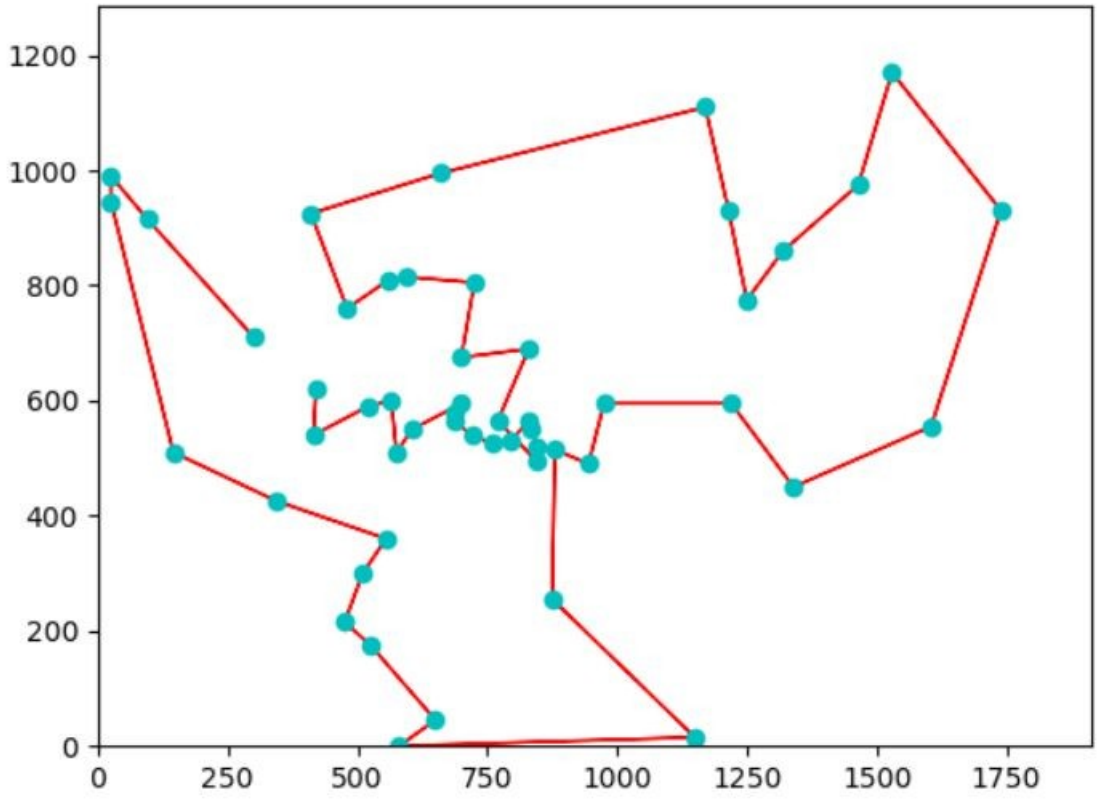
path = [45, 44, 33, 34, 35, 38, 39, 41, 40, 42, 43, 49, 50, 46, 47, 48, 53, 55, 51, 52, 58, 54, 25, 24, 23, 21, 19, 20, 17, 18, 22, 26, 28, 27, 65, 96, 94, 95, 93, 92, 76, 77, 78, 88, 89, 90, 91, 83, 82, 81, 70, 69, 57, 56, 59, 61, 60, 62, 63, 64, 66, 68, 67, 72, 71, 73, 75, 74, 84, 85, 86, 87, 80, 79, 29, 31, 30, 32, 36, 37, 1, 2, 3, 6, 5, 4, 7, 8, 10, 9, 12, 14, 15, 16, 13, 11]

2. berlin52.tsp

Rozwiązanie optymalne wynosi: 7542

Lp.	L. mrówek	iteracje	ρ	q	α	β	Odchylenie
1	25	150	0.2	1	1	2	726.0
2	30	175	0.25	1	1	2	683.8
3	35	200	0.25	1	1	3	306.4
4	40	225	0.3	1	2	1	5664.2
5	45	250	0.3	1	2	3	1893.8
6	50	275	0.35	1	2	1	10989.8
7	55	300	0.35	1	2	2	6669.6

Wizualizacja rozwiązania nr 3:



path = [31, 18, 22, 1, 32, 49, 34, 35, 36, 39, 40, 38, 48, 24, 5, 15, 37, 46, 44, 16, 50, 20, 23, 30, 29, 47, 26, 28, 27, 13, 14, 52, 11, 51, 12, 25, 4, 6, 43, 33, 9, 10, 8, 41, 19, 45, 3, 17, 7, 2, 42, 21]

Wnioski

Patrząc na powyższe wyniki można dojść do następujących wniosków

- zwiększenie wartości q ma pozytywny wpływ na wynik, czyli im więcej feromonu mrówka może zostawić tym lepiej wpływa to na znalezienie optymalnej trasy
- gdy mamy dostępną dużą ilość węzłów dobrze jest posłużyć się większą ilością mrówek
- jeśli $\alpha > \beta$ to wynik znacząco odbiega od optimum, ponadto uzyskane wyniki są tym lepsze im mniejszy stosunek α do β, czyli im ważniejsza przy podejmowaniu decyzji była długość odcinka do przejścia w porównaniu do wagi feromonu tym lepszy wyniki były uzyskiwane
- znaczące zmniejszenie q czyli szybkości parowania feromonu negatywnie wpływa na wyniki
- jeśli nie oddalać się zbytnio od wartości podanych w książce Thomasa Stutzle to największy wpływ na wyniki mają wagi feromonu i dystansu