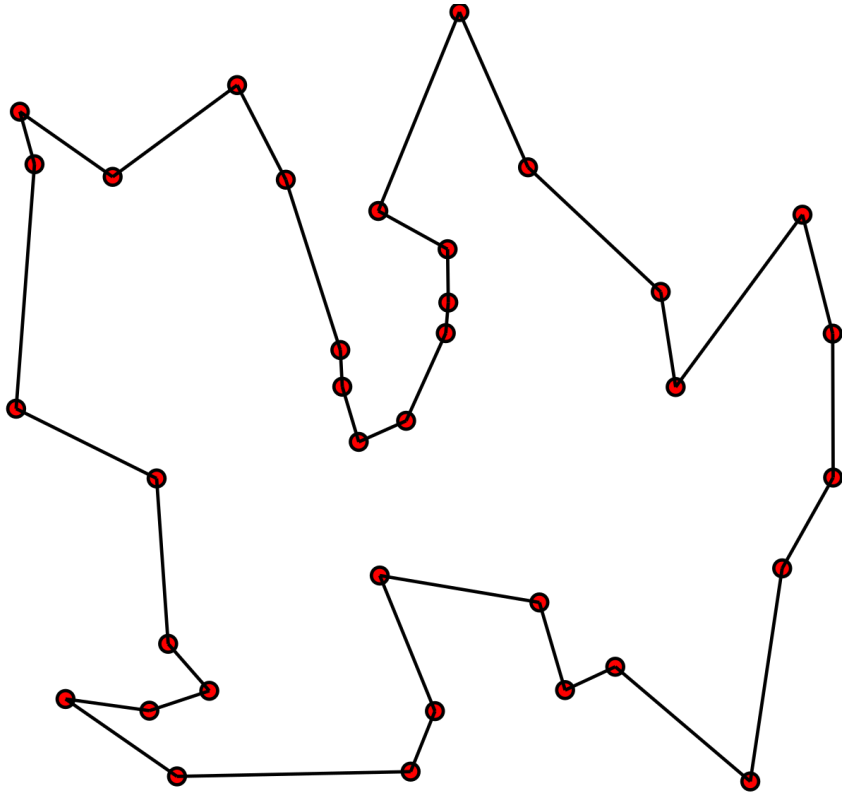
Algorytmy mrowiskowe (ang. ACO)

Adam Lewiński

21 marca 2023

Wstęp

Pierwszy algorytm mrowiskowy został zaproponowany przez Marco Dorigo w 1992 roku, zaś jego celem było rozwiązanie problemu komiwojażera (ang. TSP) tj. znalezienia trasy o najmniejszym koszcie, która przechodzi przez wszystkie miasta i kończy się w punkcie początkowym.



Rysunek: Przykład rozwiązania TSP.

Inspiracja naturą

Cechy mrówek użyte w algorytmach mrówkowych:

- Każda mrówka zostawia za sobą ślad feromonowy.
- Ślad feromonowy paruje wraz z upływem czasu.
- Każdy osobnik przy podejmowaniu decyzji kieruje się śladem feromonowym.
- Mrówki są w stanie znaleźć najkrótszą drogę do pożywienia.
- W przypadku pojawienia się przeszkody, mrówki na początku obierają różne drogi, a finalnie wszystkie maszerują najkrótszą.

Postępowanie mrówek

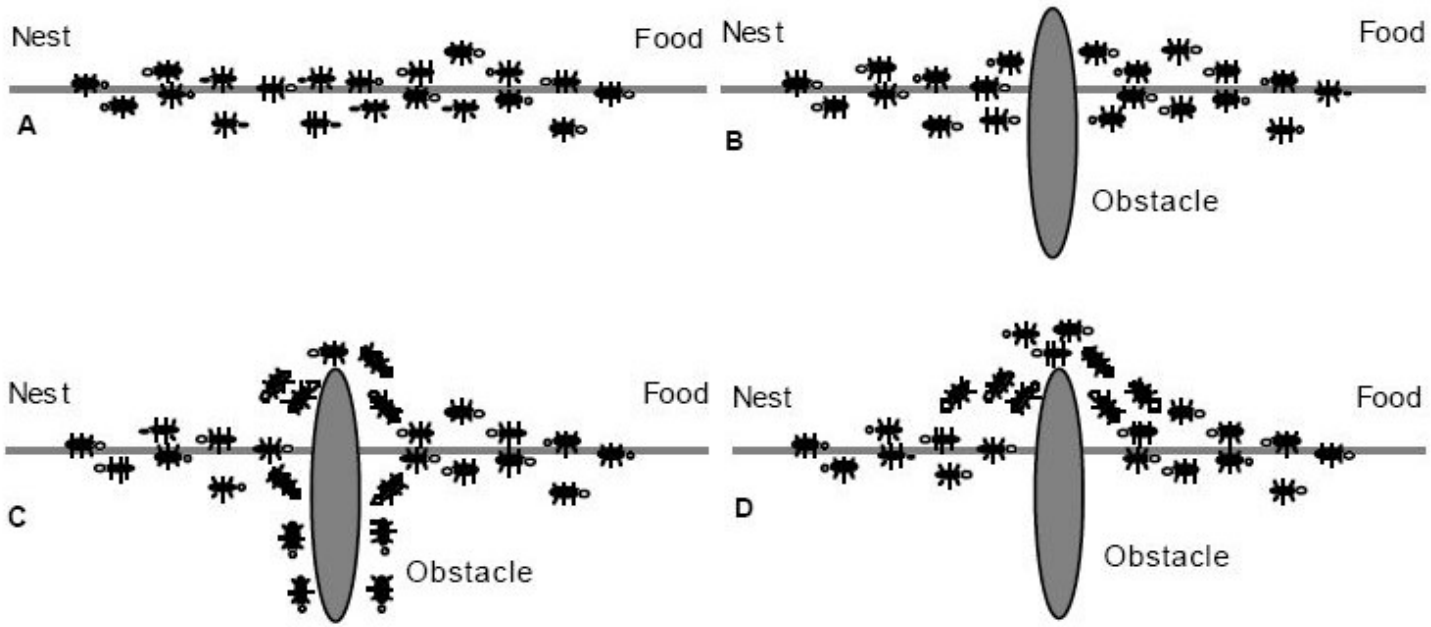


Fig.1. (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

Rysunek: Postępowanie mrówek.

Działanie algorytmu - TSP

Działanie algorytmu dla TSP:

1. Rozstaw **m** mrówek w losowych miastach.
2. Każda mrówka **k** wykonuje krok do miasta w którym jeszcze nie była.
3. Zaktualizuj ślad feromonowy.
4. Jeżeli istnieją nieodwiedzone miasta przejdź do kroku drugiego.
5. Oblicz długości tras mrówek. Zapisz najlepszą trasę.
6. Jeżeli został spełniony warunek stopu (wszystkie mrówki poszły tą samą trasą albo wykonano zadaną ilość cykli) to zakończ działanie, w przeciwnym wypadku przejdź do kroku 1.

Wybór miasta

Wybór miasta odbywa się na podstawie prawdopodobieństwa, które wyliczane jest na podstawie wzoru (1):

$$p_{ij}^{k} = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{j \in J_i^k} [\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}} \tag{1}$$

gdzie:
τ_{ij} - feromon między miastem i i j,
η_{ij} - widoczność miasta j z miasta i, $\eta_{ij} = \frac{1}{d_{ij}}$,
d_{ij} - odległość między miastem i i j,
$J_i^k = \{j : j \notin tabu_k\}$ - zbiór nieodwiedzonych do tej pory miast przez mrówkę k,
α, β - parametry pozwalające użytkownikowi sterować względną ważnością intensywności śladu i widocznością miast.

Aktualizacja feromonu - stały i średni

Możemy rozróżnić kilka sposobów aktualizowania śladu feromonowego. Dwa z nich - feromon stały i feromon średni - charakteryzuje następujący wzór (2):

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) \tag{2}$$

gdzie:
τ_{ij} - feromon między miastem i i j,
ρ - współczynnik wyparowania feromonu.

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t, t+1)$$

Feromon stały

Dla feromonu stałego:

$$\Delta\tau_{ij}^{k}(t, t+1) = \begin{cases} Q & \text{jeżeli mrówka } k \text{ przechodzi z } i \text{ do } j \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

gdzie:
Q - pewna stała wartość.

Feromon średni

Dla feromonu średniego:

$$\Delta\tau_{ij}^{k}(t, t+1) = \begin{cases} \frac{Q}{d_{ij}} & \text{jeżeli mrówka } k \text{ przechodzi z } i \text{ do } j \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

gdzie:
Q - pewna stała wartość,
d_{ij} - odległość pomiędzy miastem i i j.

Feromon cykliczny

Trzecim sposobem jest feromon cykliczny (3). Istotne jest to, że aktualizujemy go dopiero, kiedy każda mrówka pokona całą trasę.

$$\tau_{ij}(t+n) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t, t+n) \tag{3}$$

gdzie:

$$\Delta\tau_{ij}(t, t+n) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t, t+n)$$

$$\Delta\tau_{ij}^{k}(t, t+n) = \begin{cases} \frac{Q}{L_k} & \text{jeżeli mrówka } k \text{ wybierze krawędź } (i,j) \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

gdzie:
Q - pewna stała wartość,
L_k - długość trasy mrówki k.

Parametry

Na podstawie przedstawionych informacji wyłaniają się następujące parametry algorytmu, którymi możemy sterować:

- m - liczba mrówek,
- α - waga śladu feromonowego,
- β - waga heurystyki,
- ρ - współczynnik wyparowania,
- τ_0 - feromon startowy,
- Q - stała użyta do nakładania feromonu.

Parametr alfa i beta

Zgodnie z zaleceniami autora:

- $\alpha = 1$,
- $\beta =$ liczba z zakresu [2;4].

Z racji tego, że zarówno τ_{ij} i η_{ij} są mniejsze od 1, zwiększanie tych parametrów powoduje zmniejszenie znaczenia odpowiedniego czynnika przy wyborze miasta. Jeżeli $\alpha = 0$, to wtedy mamy do czynienia z przpyadkiem, gdy wartość feromonu jest ignorowana (algorytm zachłanny).

Pozostałe parametry

- $\rho = 0.5$ (koniecznie $0 < \rho < 1$, aby nie kumulować zanadto feromonu),
- $m =$ liczba miast,
- $\tau_0 = \frac{m}{C}$, gdzie C to szacowana długość trasy,
- Q - dobieramy odpowiednio mając na uwadzę, że feromon będzie dążył do $\frac{m*Q}{\rho}$ jeśli $0 < \rho < 1$.

Możemy dobrać Q tak, aby prawdopodobieństwo pomiędzy 'najlepszą' drogą, a 'najgorszą' miało stosunek 100:1 - pewien odpowiednik mutacji w GA.

ACS

ACS - ant colony system - jedna z wielu modyfikacji ACO, która lepiej sprawdza się dla problemu TSP w przypadku dużej ilości miast.

- Feromon lokalny oraz globalny (każdy nakładany inaczej niż wcześniej zaprezentowano).
- Nie ma parametru Q.
- Nowy parametr q_0, który determinuje proporcje pomiędzy wzmacnianiem feromonu na najlepszej trasie, a eksploracją nowych tras.
- Wprowadzono listę miast kandytujących.

Moja implementacja

Moja implementacja:

- dystans pomiędzy miastami przedstawiony za pomocą macierzy dystansów,
- feromon jest reprezentowany w macierzy o takich samych wymiarach jak macierz dystansów, tylko zamiast odległości między punktami, znajduje się tam wartość feromonu pomiędzy nimi,
- dla problemów asymetrycznych wykorzystana jest cała macierz, dla symetrycznych tylko górna część,
- wykorzystanie własnej klasy Ant jako struktury danych, zawierającej miasto docelowe oraz listę tabu,
- implementacja w języku python, z pakietem numba i dekoratorem @jit(nopython=True, parallel=True) w celu przyśpieszenia kodu (ponad 30 razy szybciej).

Rezultaty z pracy Marco Dorigo

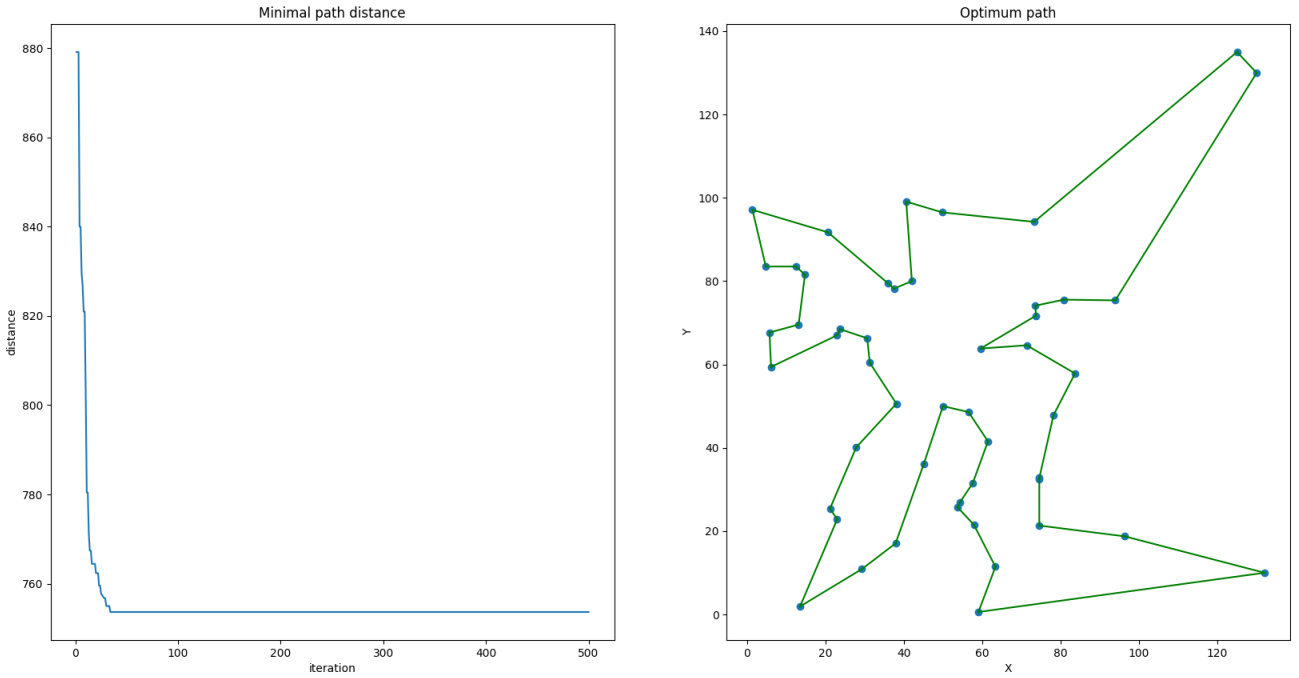
- GA - algorytm genetyczny
- EP - expectation propagation
- SA - symulowane wyżarzanie

Problem name	ACS	GA	EP	SA	AG	Optimum
Oliver30	**420**	421	**420**	424	**420**	**420**
(30-city problem)	**(423.74)**	(N/A)	**(423.74)**	(N/A)	(N/A)	**(423.74)**
	[830]	[3200]	[40 000]	[24 617]	[12 620]	
Eil50	**425**	428	426	443	436	**425**
(50-city problem)	(427.96)	(N/A)	**(427.86)**	(N/A)	(N/A)	(N/A)
	[1830]	[25 000]	[100 000]	[68 512]	[28 111]	
Eil75	**535**	545	**542**	580	561	535
(75-city problem)	**(542.31)**	(N/A)	(549.18)	(N/A)	(N/A)	(N/A)
	[3480]	[80 000]	[325 000]	[173 250]	[95 506]	
KroA100	**21 282**	21 761	N/A	N/A	N/A	21 282
(100-city problem)	**(21 285.44)**	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)
	[4820]	[103 000]	[N/A]	[N/A]	[N/A]	

Rysunek: Ant colonies for the travelling salesman problem, Marco Dorigo.

Rezultat mojej implementacji

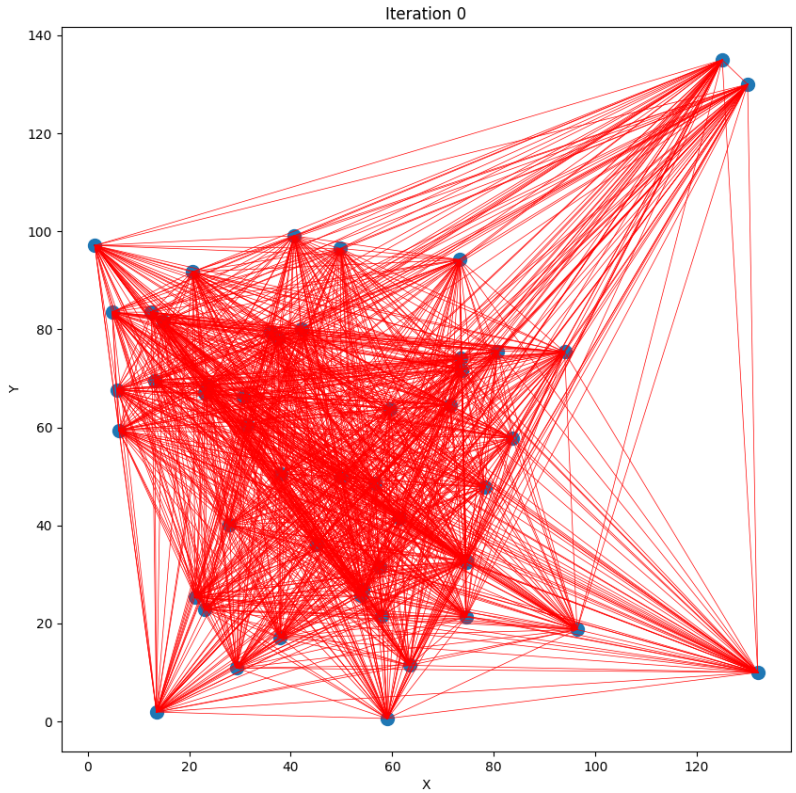
Czas znalezienia 'optymalnego' (753.6) rozwiązania - 30 sekund, ponad 30 iteracji.
Czas wykonania wszystkich iteracji - 11.5 minuty.
Parametry: $\alpha = 1$, $\beta = 2$, $\rho = 0.2$, $q_0 = 0.4$, $\tau_0 = 1e-3$, $m = 1000$



Rysunek: Przebieg szukania rozwiązania i jego prezentacja - 50 miast.

Kilka klatek z pracy algorytmu

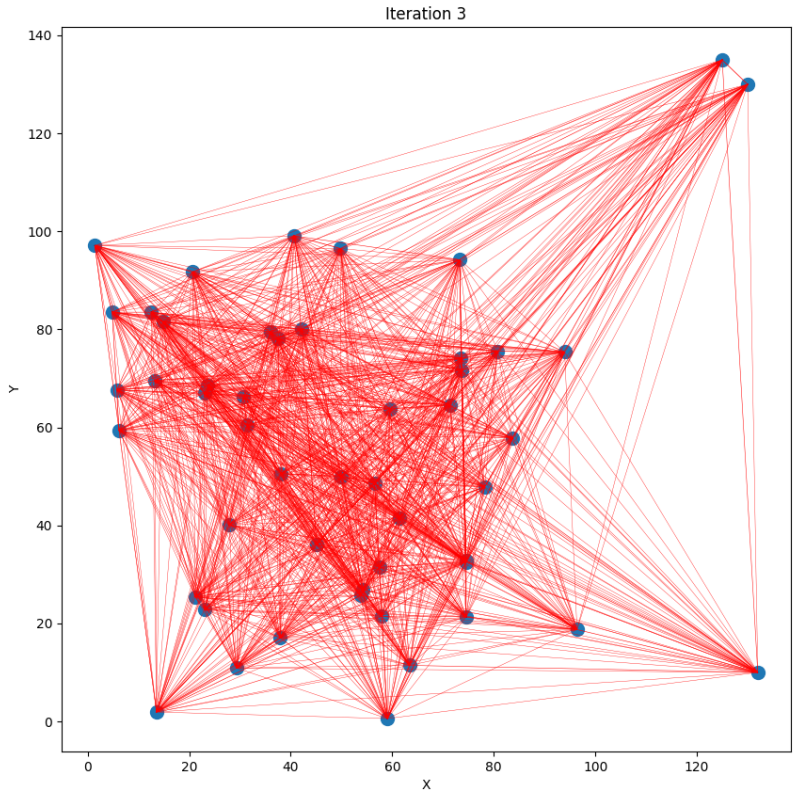
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

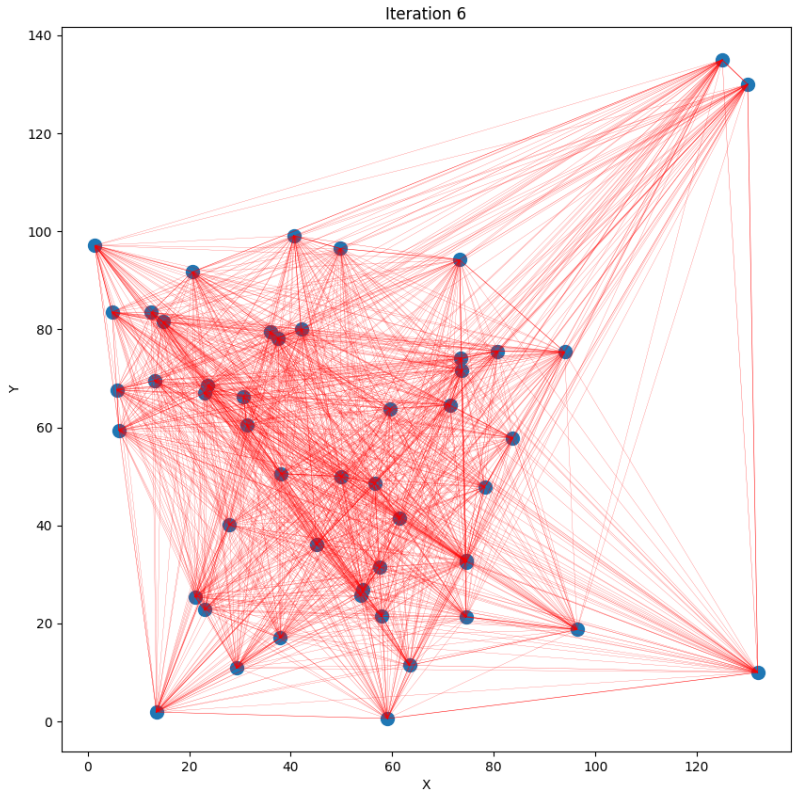
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

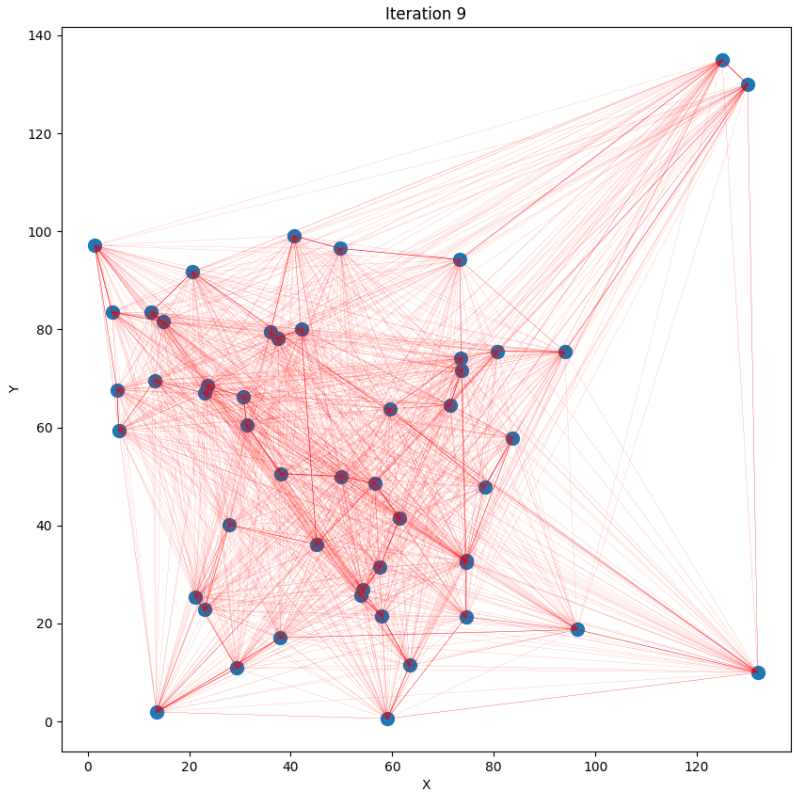
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

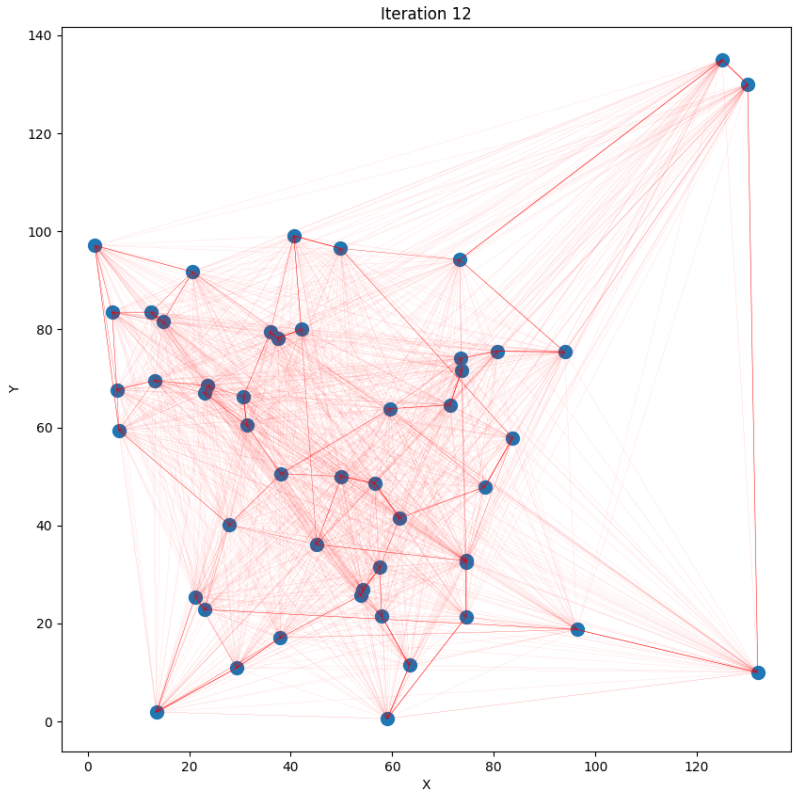
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

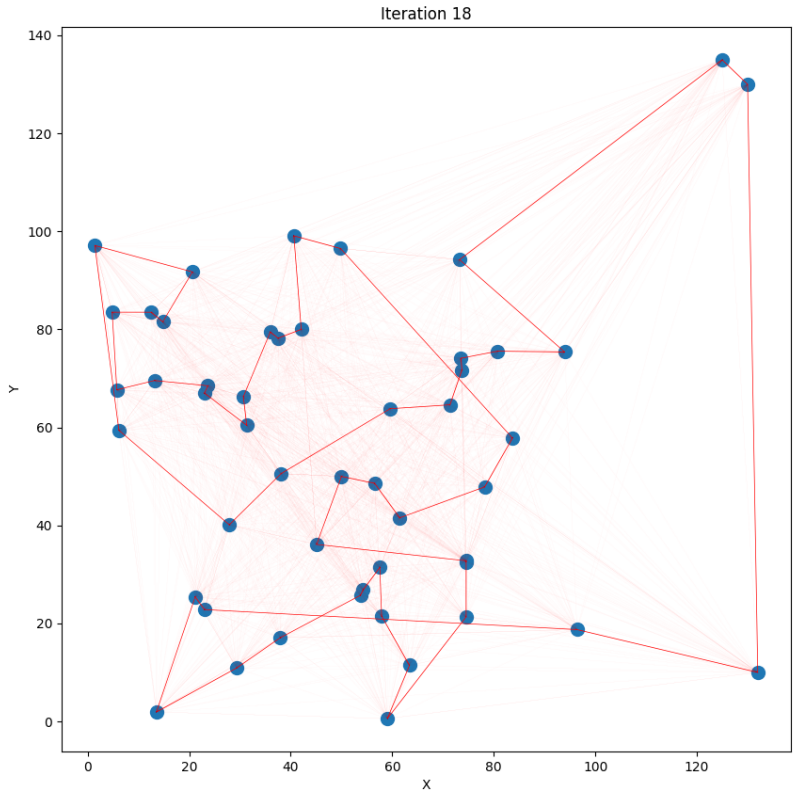
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

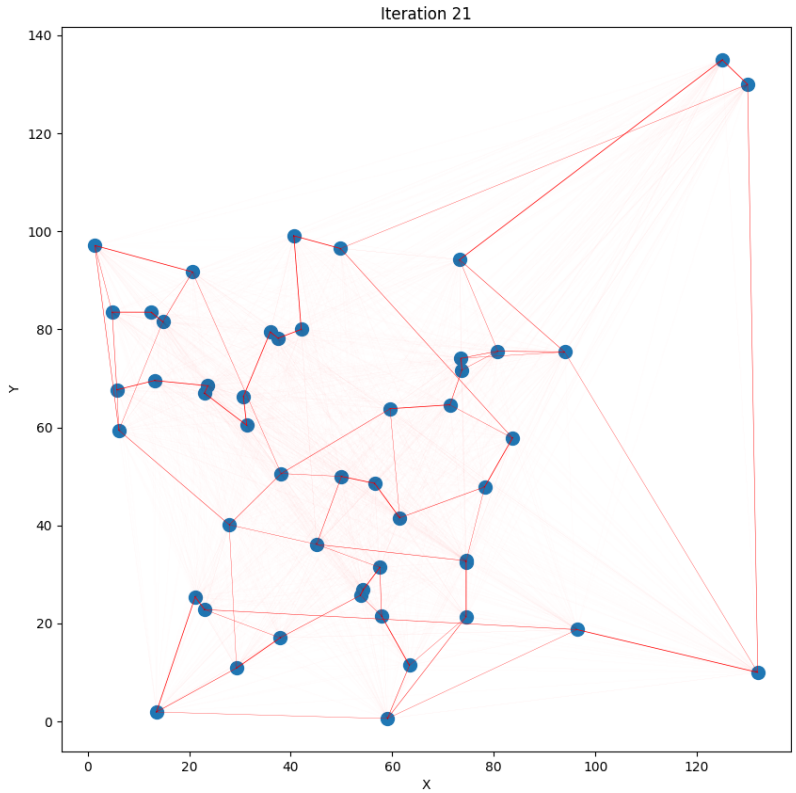
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

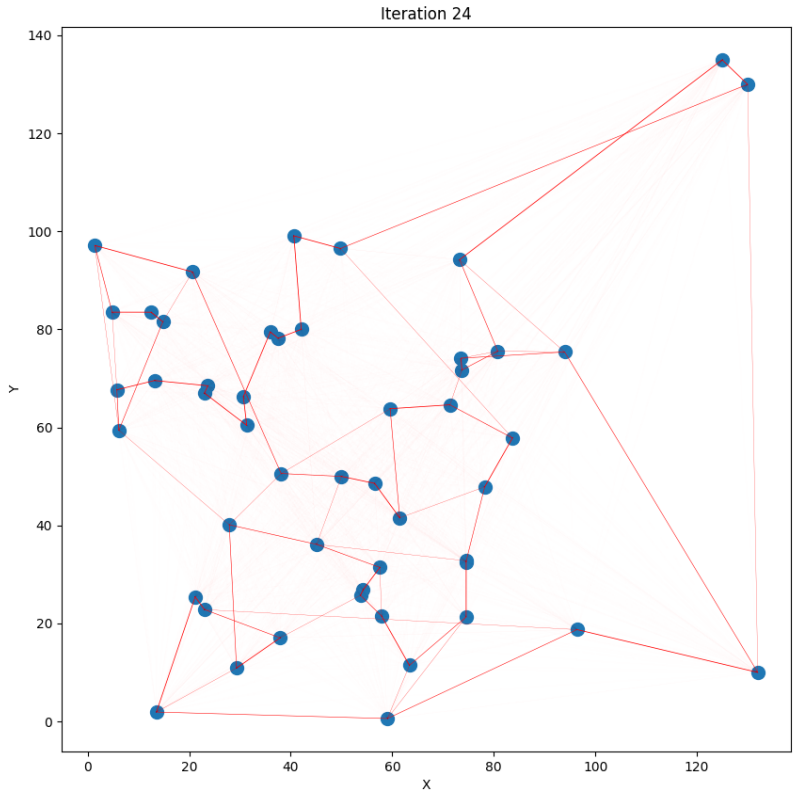
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

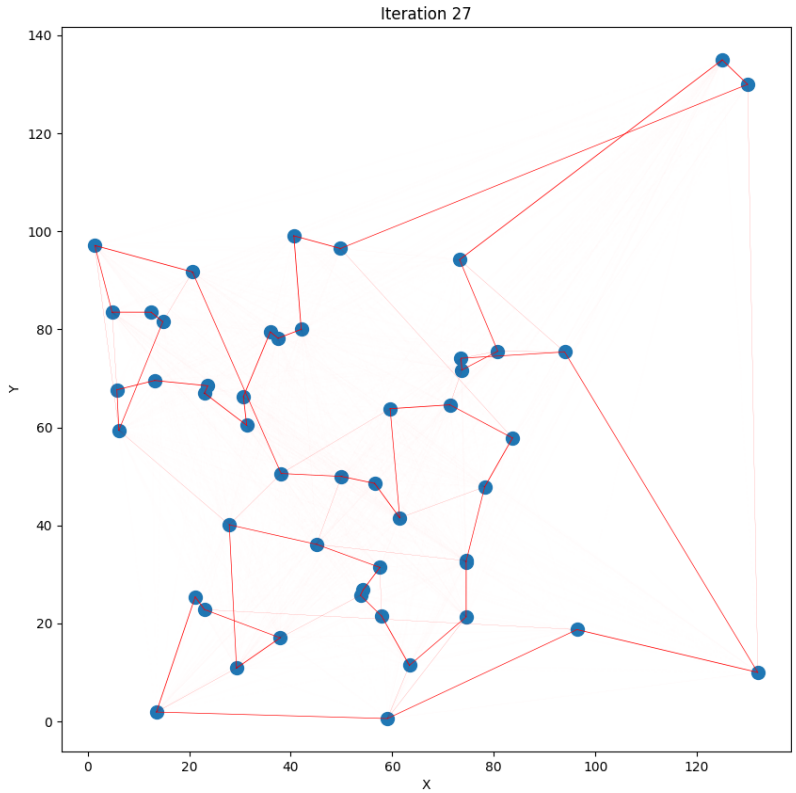
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

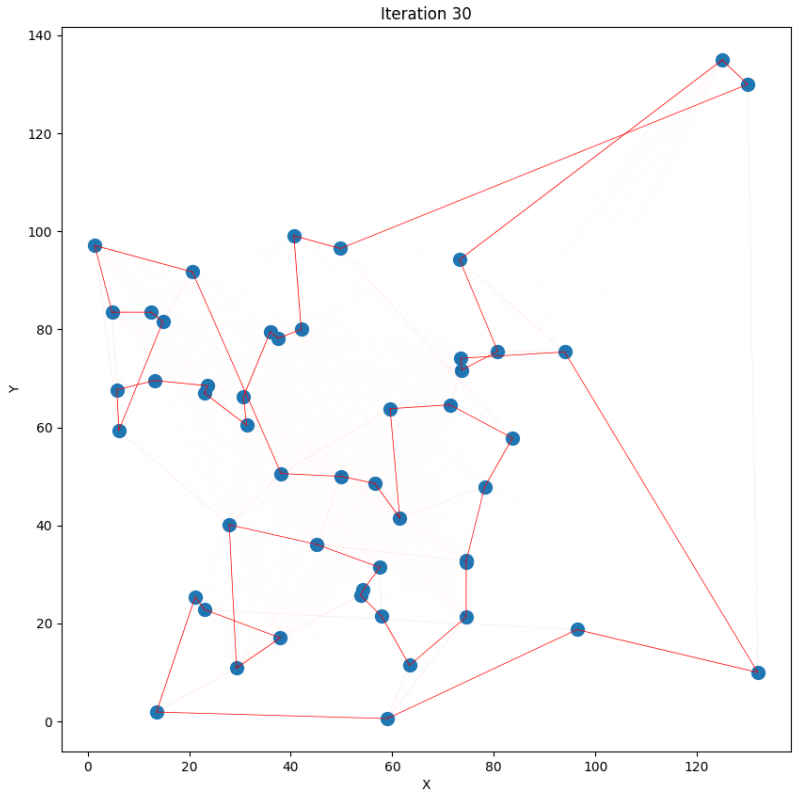
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

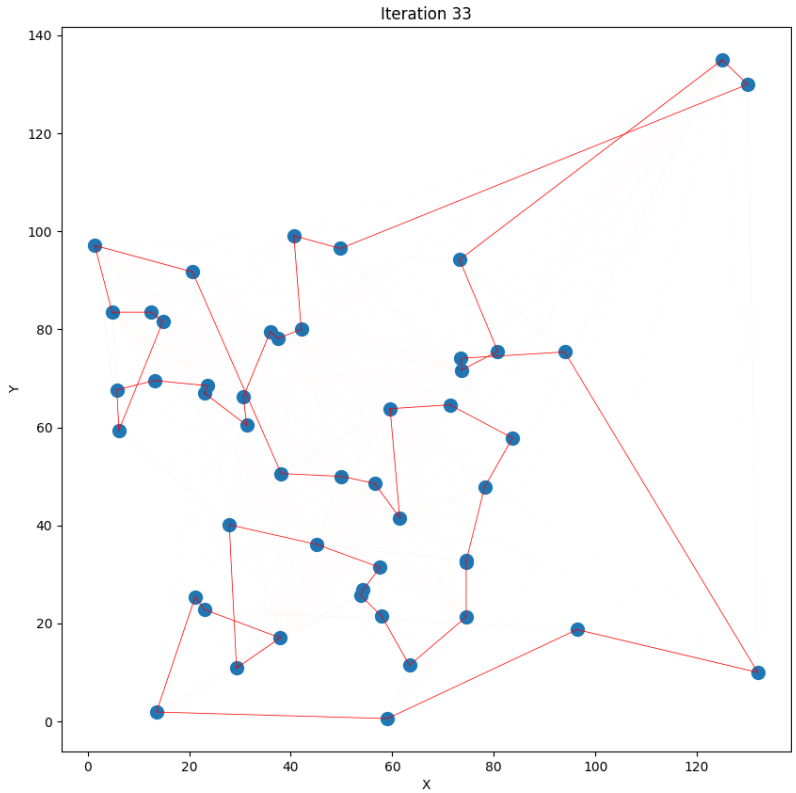
Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)



Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Kilka klatek z pracy algorytmu

Kilka rezultatów z pracy algorytmu (inne parametry, niż w poprzednim przykładzie, $\alpha = 0$)

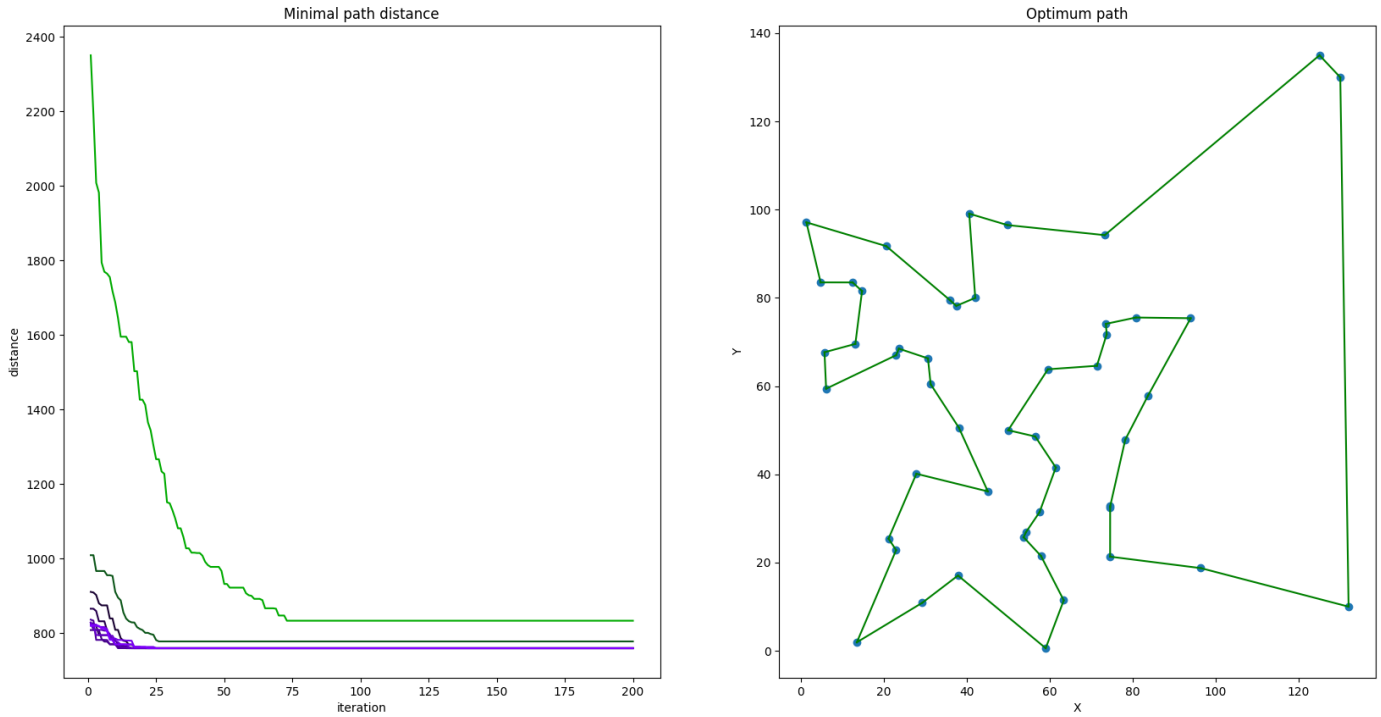


Rysunek: Praca algorytmu - 50 miast, czerwone linie - feromon.

Losowe Permutacje

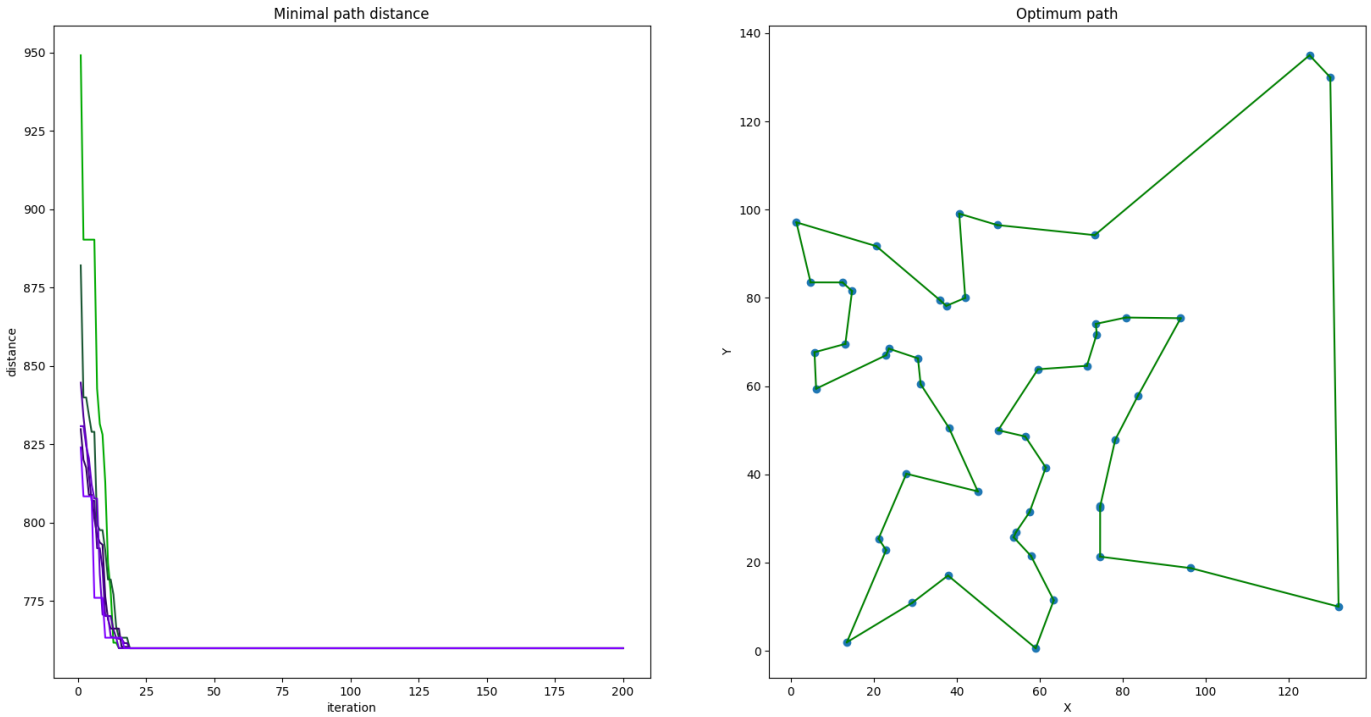
W wyniku losowania miliarda permuatcji najlepszy wynik jaki udało mi się uzyskać to 1824.16. Wszystkich kombinacji jest 3e62.

Badanie parametru β



Rysunek: β 0-10, zielone-fioletowe.

Badanie parametru β



Rysunek: β 2-6, zielone-fioletowe.

Zalety i wady algorytmu

Zalety:

- Problem marszrutyzacji (ang. VRP) i jemu podobne (np. TSP) są rozwiązywane szybciej niż przez algorytmy genetyczne (ang. GA).
- Bez problemu odnajdzie się w środowisku, które ulega zmianom w czasie (np. zdarzenia drogowe, które wymuszają zmianę trasy).
- Poradzi sobie z optymalizacją wielokryterialną.

Wady:

- Trudność implementacji dla niektórych problemów optymalizacyjnych (głównie nadaje się dla zagadnień, które dają się przedstawić w postaci grafów).
- Mniej jasny wpływ parametrów na działanie algorytmu niż przy GA.