

ALHE - Dokumentacja

Maciej Kapuściński, Sebastian Pietras

Treść zadania

SK.ALHE.12

Dla sieci o nazwie *india35* ze strony <http://sndlib.zib.de/home.action> zastosować algorytm mrówkowy (*Ant Colony*) do znalezienia n najlepszych (wg. ustalonej metryki) ścieżek w danej sieci. Porównanie z innym algorytmem będzie dodatkowym atutem.

Przyjęte założenia i doprecyzowanie treści

Sieć zostanie potraktowana jako graf ważony nieskierowany. Wagi zostaną przypisane krawędziom zgodnie z kolumną *module cost*, która znajduje się w pliku definiującym sieć.

Ścieżki będą wyszukiwane między wierzchołkiem początkowym a końcowym. Te wierzchołki będą podawane jako parametr programu.

Jako metryka zostanie wykorzystany koszt ścieżki. To znaczy, że najlepsza jest ta ścieżka, dla której suma wag jej krawędzi jest najmniejsza.

Implementacja zostanie wykonana z użyciem języka programowania Python.

Opis algorytmu

Znajdowanie jednej najlepszej ścieżki

Algorytm wysyła m mrówek (a raczej ich abstrakcyjnych reprezentacji) przez graf I razy, gdzie I to ustalona ilość iteracji, po czym zapamiętywana jest najkrótsza znaleziona przez mrówki ścieżka.

Mrówki, które znalazły ścieżkę zostawiają na niej feromony. Lepsze ścieżki powinny częściej zawierać większą ilość feromonu.

W każdym kroku mrówki, które nie dotarły jeszcze do wierzchołka końcowego, wybierają wierzchołek, do którego się przemieszczą. Brane pod uwagę są jedynie wierzchołki sąsiadujące z wierzchołkiem, w którym aktualnie znajduje się mrówka. Decyzja o wybranym wierzchołku zostanie podjęta losowo, lecz zgodnie z prawdopodobieństwami przypisanymi do każdej krawędzi.

Prawdopodobieństwo wyboru krawędzi jest opisane wzorem
$$p_{xy} = \frac{\tau_{xy}^\alpha \eta_{xy}^\beta}{\sum_{z \in N(x)} \tau_{xz}^\alpha \eta_{xz}^\beta},$$

gdzie:

τ_{xy} - ilość feromonu na krawędzi xy

α - parameter kontrolujący wpływ τ

η_{xy} - "atrakcyjność" krawędzi xy (w naszym przypadku odwrotność wagi tej krawędzi)

β - parametr kontrolujący wpływ η

$N(x)$ - wierzchołki sąsiadujące z wierzchołkiem x

Gdy wszystkie mrówki znajdą ścieżkę lub przekroczą limit kroków, nastąpi aktualizacja feromonów na krawędziach grafu. Do wartości feromonu danej krawędzi będzie

przypisywana wartość $\tau_{xy} \leftarrow (1 - \rho) \tau_{xy} + \sum_k^m \Delta \tau_{xy}^k$, gdzie:

τ_{xy} - wartość feromonu krawędzi xy

ρ - współczynnik parowania (zanikania) feromonów

m - liczba mrówek

$\Delta \tau_{xy}^k$ - ilość feromonu pozostawiona na krawędzi xy przez mrówkę k

$\Delta \tau_{xy}^k$ wynosi 0 jeśli mrówka k nie przeszła przez krawędź xy , a w przeciwnym przypadku $\frac{Q}{L_k}$, gdzie:

Q - współczynnik nakładania feromonów

L_k - koszt całej drogi mrówki k w tej iteracji

Znajdowanie n najlepszych ścieżek

Aby znaleźć $(i + 1)$ -szą najlepszą ścieżkę należy zablokować poprzednie i ścieżek w grafie, a następnie wyszukać w takim grafie najlepszą ścieżkę. Można tak iteracyjnie postępować aż do znalezienia n najlepszych ścieżek.

Zablokowanie ścieżek w grafie będzie polegać na usunięciu po jednej krawędzi z każdej ścieżki. Ta czynność zostanie powtórzona dla wszystkich możliwych doborów krawędzi. We wszystkich powstałych grafach zostanie wyszukana najlepsza ścieżka w danym grafie. Z wyszukanych ścieżek zostanie wybrana ta, która była ogółem najlepsza.

Uruchomienie programu

```
python -m antcolony [--n N] [--n_ants N_ANTS] [--n_iter N_ITER]
                  [--max_steps MAX_STEPS]
                  [--alpha ALPHA] [--beta BETA] [--ro R0] [--q Q]
graph start_node end_node
```

Argumenty:

- graph - ścieżka do pliku z definicją grafu
- start_node - wierzchołek początkowy
- end_node - wierzchołek końcowy
- --n N - liczba ścieżek do znalezienia (domyślnie 1)
- --n_ants N_ANTS - liczba "mrówek" w algorytmie mrówkowym (domyślnie 50)
- --n_iter N_ITER - liczba iteracji w algorytmie mrówkowym (domyślnie 10)
- --max_step MAX_STEP - maksymalna liczba kroków mrówki w jednej iteracji (domyślnie 1000)
- --alpha ALPHA - współczynnik α (domyślnie 0.5)
- --beta BETA - współczynnik β (domyślnie 1.2)
- --ro R0 - współczynnik ρ (domyślnie 0.6)
- --q Q - współczynnik Q (domyślnie 10)

Program wypisze na standardowe wyjście ścieżki znalezione przez algorytm mrówkowy oraz, dla porównania, przez algorytm Dijkstry.

Wykorzystane narzędzia

- Python
- NetworkX (do reprezentacji grafu)
- NumPy (do operacji na macierzach)

Testy

Przeprowadzone zostały testy dla ścieżek między różnymi wierzchołkami i dla różnych liczby mrówek.

1. Wywołania z większą liczbą mrówek (50)

```
[2021-01-12 23:04:51,853] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:05:35,658] [INFO] [antcolony] Shortest paths according to ant colony: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
[2021-01-12 23:05:35,658] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:05:35,675] [INFO] [antcolony] Shortest paths according to Dijkstra: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
[2021-01-12 23:05:35,678] [INFO] [antcolony] Shortest paths according to NetworkX: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
```

```
[2021-01-12 23:12:55,368] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:13:21,744] [INFO] [antcolony] Shortest paths according to ant colony: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
[2021-01-12 23:13:21,744] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:13:21,763] [INFO] [antcolony] Shortest paths according to Dijkstra: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
[2021-01-12 23:13:21,766] [INFO] [antcolony] Shortest paths according to NetworkX: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
```

2. Wywołanie z mniejszą liczbą mrówek (30)

```
[2021-01-12 23:25:47,260] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:26:03,096] [INFO] [antcolony] Shortest paths according to ant colony: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (650.0, ['1', '21', '17', '15'])]
[2021-01-12 23:26:03,096] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:26:03,112] [INFO] [antcolony] Shortest paths according to Dijkstra: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
[2021-01-12 23:26:03,114] [INFO] [antcolony] Shortest paths according to NetworkX: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
```

```
[2021-01-12 23:26:10,749] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:26:27,715] [INFO] [antcolony] Shortest paths according to ant colony: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
[2021-01-12 23:26:27,715] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:26:27,729] [INFO] [antcolony] Shortest paths according to Dijkstra: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
[2021-01-12 23:26:27,733] [INFO] [antcolony] Shortest paths according to NetworkX: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
```

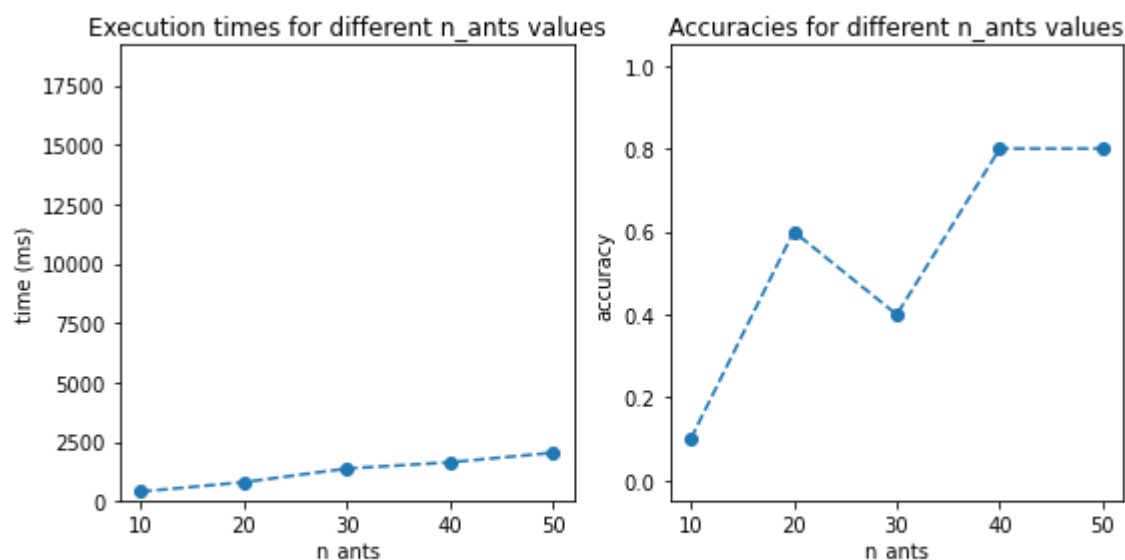
3. Wywołania z najmniejszą liczbą mrówek (10)

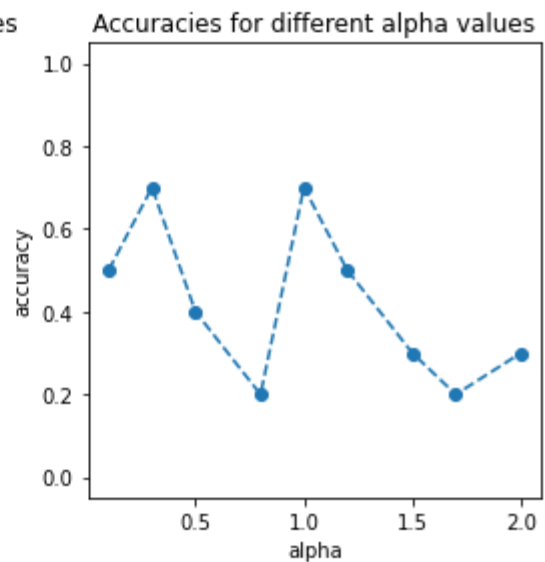
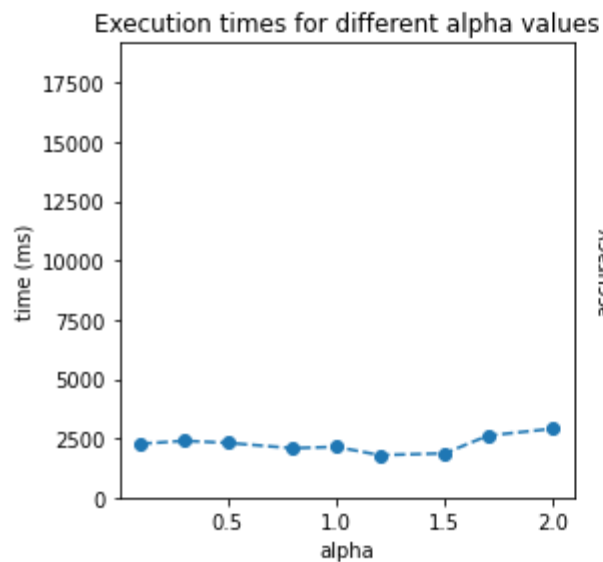
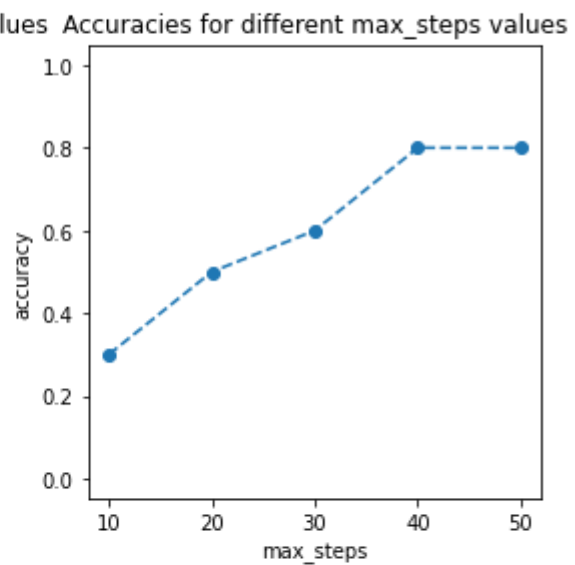
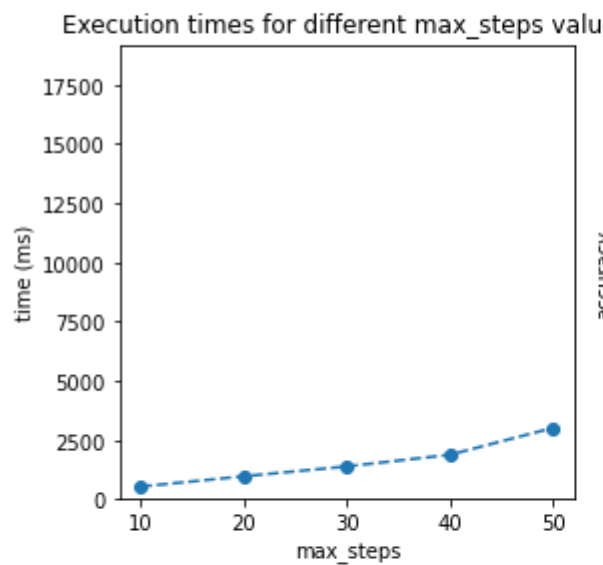
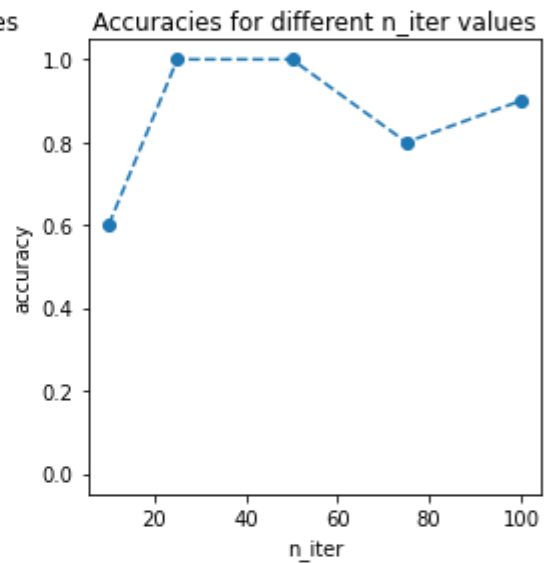
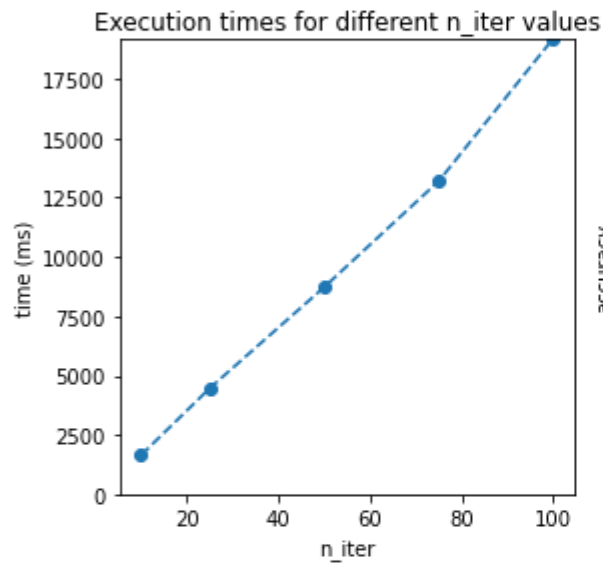
```
[2021-01-12 23:10:56,759] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:11:04,334] [INFO] [antcolony] Shortest paths according to ant colony: [(570.0, ['1', '6', '32', '30']), (870.0, ['1', '6', '21', '34', '29', '30']), (590.0, ['1', '21', '32', '30'])]
[2021-01-12 23:11:04,335] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:11:04,349] [INFO] [antcolony] Shortest paths according to Dijkstra: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
[2021-01-12 23:11:04,352] [INFO] [antcolony] Shortest paths according to NetworkX: [(570.0, ['1', '6', '32', '30']), (590.0, ['1', '21', '32', '30']), (690.0, ['1', '28', '32', '30'])]
```

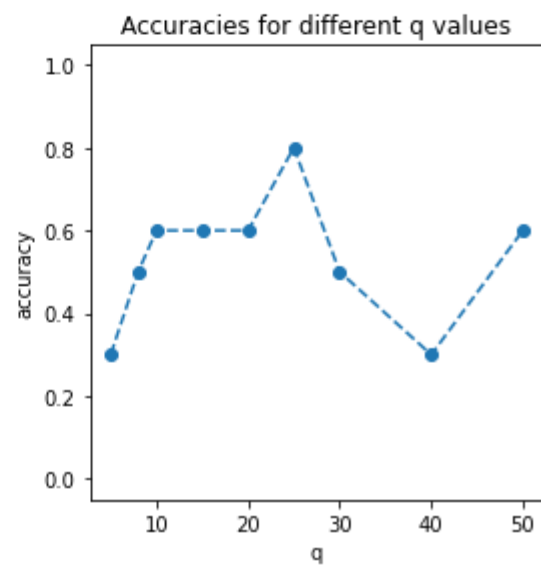
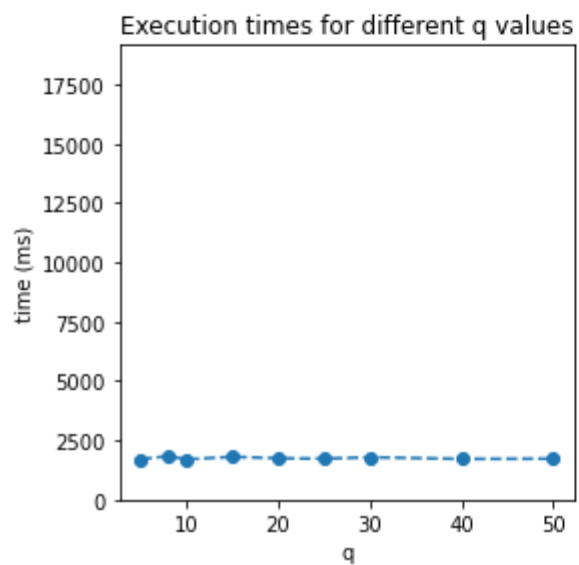
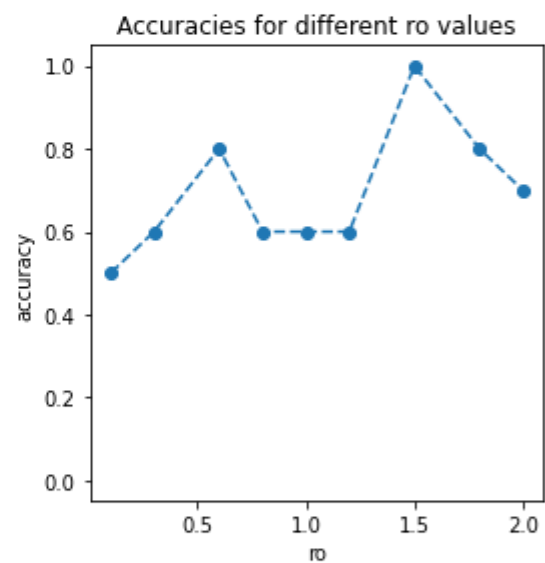
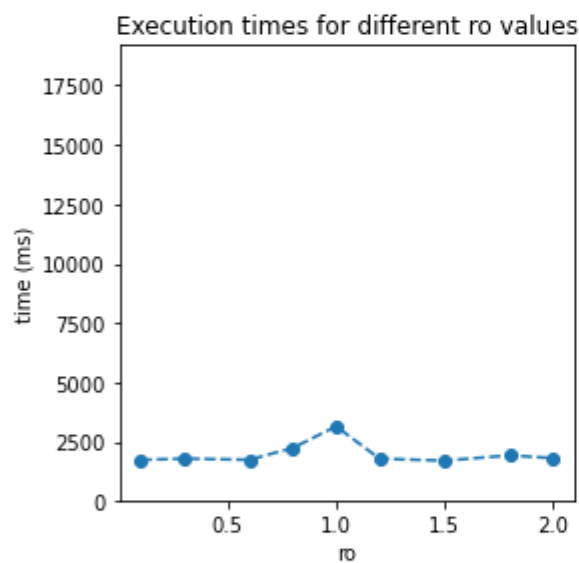
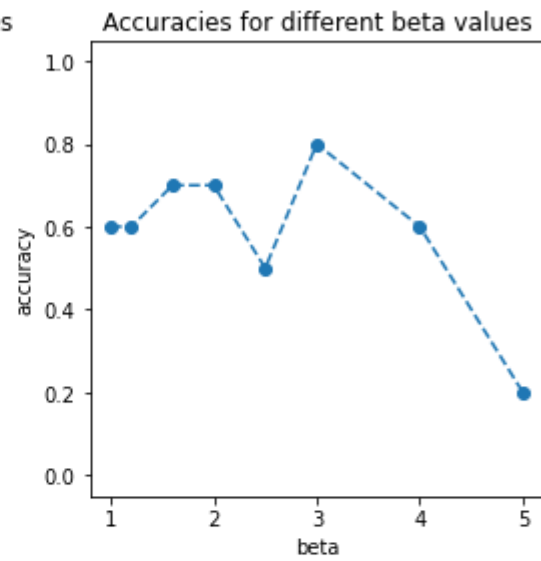
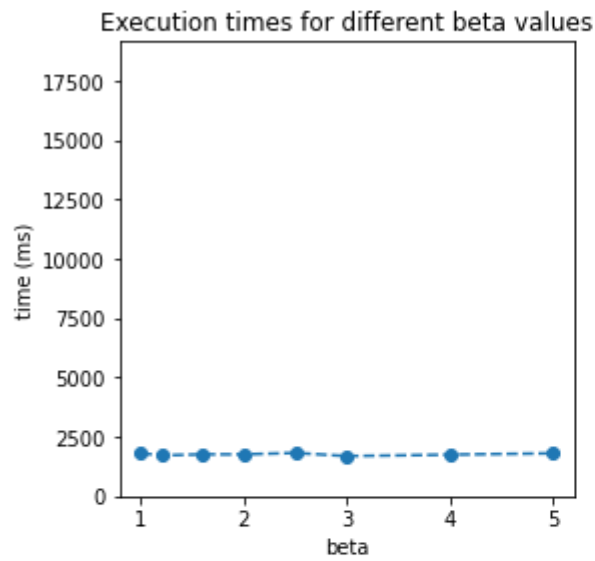
```
[2021-01-12 23:12:21,490] [INFO] [antcolony] Starting search using ant colony
[2021-01-12 23:12:30,053] [INFO] [antcolony] Shortest paths according to ant colony: [(910.0, ['1', '14', '25', '5', '27', '15']), (490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15'])]
[2021-01-12 23:12:30,054] [INFO] [antcolony] Starting search using Dijkstra
[2021-01-12 23:12:30,076] [INFO] [antcolony] Shortest paths according to Dijkstra: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
[2021-01-12 23:12:30,079] [INFO] [antcolony] Shortest paths according to NetworkX: [(490.0, ['1', '3', '17', '15']), (530.0, ['1', '3', '27', '15']), (630.0, ['1', '3', '17', '2', '15'])]
```

Z wykonanych testów wynika, że dla grafu o takiej wielkości nie warto używać algorytmu mrówkowego. Dla większej liczby mrówek czas znalezienia najkrótszej ścieżki potrafi być przesadnie długi w porównaniu do algorytmu Dijkstry, którego wywołanie trwa jedynie setne sekundy. Dla mniejszej liczby mrówek jest zaś większe ryzyko znalezienia niewłaściwej ścieżki (co jest niemożliwe przy korzystaniu z algorytmu Dijkstry), a zysk czasowy nie jest na tyle znaczący, żeby było to warte.

Testy dla różnych parametrów







Przeprowadzone zostały testy dla różnych parametrów algorytmu mrówkowego. Po lewej stronie znajduje się wykres czasu wykonania algorytmu, a po prawej precyzja algorytmu (procent wykonań, które znalazły ścieżkę, która faktycznie była najlepsza). Zarówno czas wykonania jak i precyzja zostały zmierzone dla 10 wykonań algorytmu.

Z przeprowadzonych testów wynika, że zwiększanie liczby mrówek, liczby iteracji oraz maksymalnej liczby kroków mrówki pozytywnie wpływa na precyzję. Z drugiej strony niestety wpływa też na czas wykonania algorytmu.

Dla parametrów α , β , ρ , Q nie zaobserwowano wpływu ani na czas wykonania ani na precyzję.