Projektowanie Efektywnych Algorytmów Projekt

27/01/2023

252892 Jan Kucharski

(7) Ant Colony Optimalization

1. Sformułowanie zadania

Zadanie polega na opracowaniu i implementacji algorytmu rozwiązywania problemu komiwojażera przy użyciu algorytmu wykorzystującego metodę Ant Colony (przeszukiwanie mrówkowe).

2. Metoda

Metoda użyta do rozwiązania problemu komiwojażera to „Ant Colony Optimalization". Jest to probabilistyczna technika rozwiązywania problemów. Jak sama nazwa wskazuje, metoda ta inspirowana jest zachowaniem mrówek. W rzeczywistości mrówki szukają pożywienia w sposób losowy, w trakcie przemieszczania się zostawiają za sobą ślad, który składa się z feromonów. Jeżeli jakakolwiek mrówka napotka taki ślad to podąża za nim.

W prawdziwym świecie, po pewnym czasie feromony wyparowują, więc siła działania tych substancji maleje. Z tego powodu istnieje większe prawdopodobieństwo, że mrówki dotrą do pożywienia, które znajduje się bliżej (feromony mają mniej czasu aby wyparować).

W metodzie „Ant Colony Optimalization" wykorzystane są mrówki sztuczne, które znacząco różnią się od tych naturalnych. Mogą one np.: aktualizować ślad feromonu w dowolnym momencie, określać jakość trasy, rozpoznawać różnicę w jakości dróg w danym węźle grafu, podejmować decyzję.

Ponadto każda mrówka posiada pamięć odwiedzonych wierzchołków (lista tabu). Prawdopodobieństwo wybrania wierzchołka j, z wierzchołka i (krawędzi i-j) przez k-tą mrówkę wyraża się w sposób następujący:

Prawdopodobieństwo wyboru miasta j przez k-tą mrówkę w mieście i dane jest wzorem:

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{c_{i,l} \in \Omega}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}} & \bigvee c_{i,l} \in \Omega \\ 0 & \bigvee c_{i,l} \notin \Omega \end{cases}$$

gdzie:

c - kolejne możliwe (nie znajdujące się na liście $tabu_k$ miasto),

Ω - dopuszczalne rozwiązanie (nieodwiedzone miasta, nienależące do $tabu_k$),

η_{ij} -wartość lokalnej funkcji kryterium; np. $\eta = \dfrac{1}{d_{ij}}$ (*visibility*), czyli odwrotność odległości pomiędzy miastami,

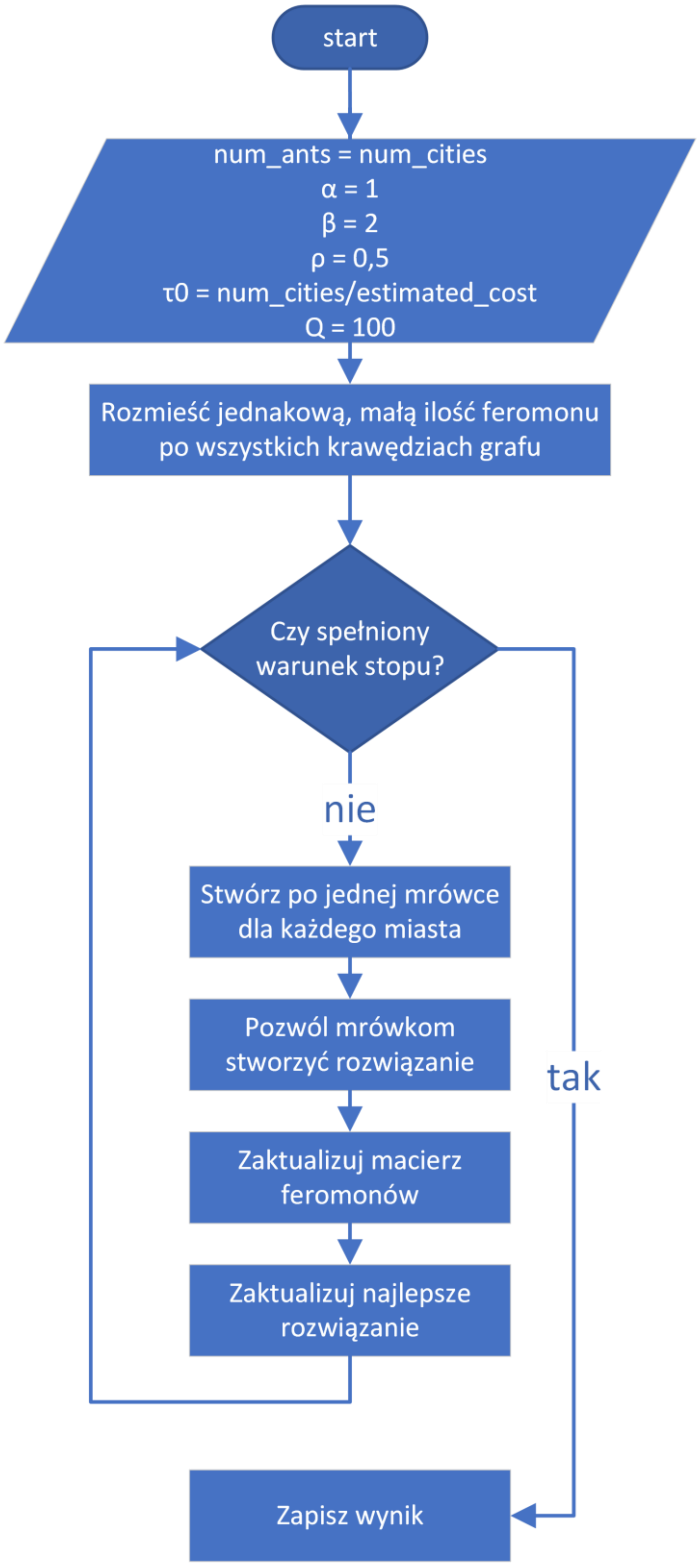
α - parametr regulujący wpływ τ_{ij},

β - parametr regulujący wpływ η_{ij}.

Dodatkowo w systemie mrówkowym wyróżniamy trzy rodzaje algorytmów:
- Algorytm gęstościowy (ant-density, DAS)
- Algorytm Ilościowy (ant-quantity QAS)
- Algorytm Cykliczny (ant-cycle CAS)

W zaimplementowanym przeze mnie programie został użyty algorytm cykliczny (CAS).

3. Algorytm



Rysunek 1 – algorytm programu

Początkowymi danymi potrzebnymi do działania algorytmu są liczba miast, liczba mrówek (w mojej implementacji równa liczbie miast), parametr α (ustawiony na 1), parametr β (ustawiony na 2), parametr ρ odpowiedzialny za stopień wyparowywania feromonu (ustawiony na 0,5), początkowa ilość feromonu w krawędziach grafu τ_0, która powstaje z podzielenia liczby miast przez oszacowany koszt rozwiązania problemu komiwojażera (za pomocą metody najbliższego sąsiada) oraz wartość Q, która wpływa na ilość zostawianego feromonu przez mrówkę.

Na początku inicjalizujemy macierz reprezentującą ilość feromonu na krawędziach grafu poprzez umieszczenie w niej początkowej ilości τ_0 feromonu na każdej krawędzi.

Następnie rozpoczyna się pętla, która wykonuje się aż do spełnienia warunku stopu. W mojej implementacji warunkiem stopu jest osiągnięcie rozwiązania o dopuszczalnym błędzie nie przekraczającym 20%.

W samej pętli co iterację tworzymy zestaw mrówek i rozmieszczamy po jednej mrówce do każdego miasta. Następnie mrówki znajdują rozwiązanie na podstawie funkcji, której logika jest przedstawiona na diagramie na następnej stronie.

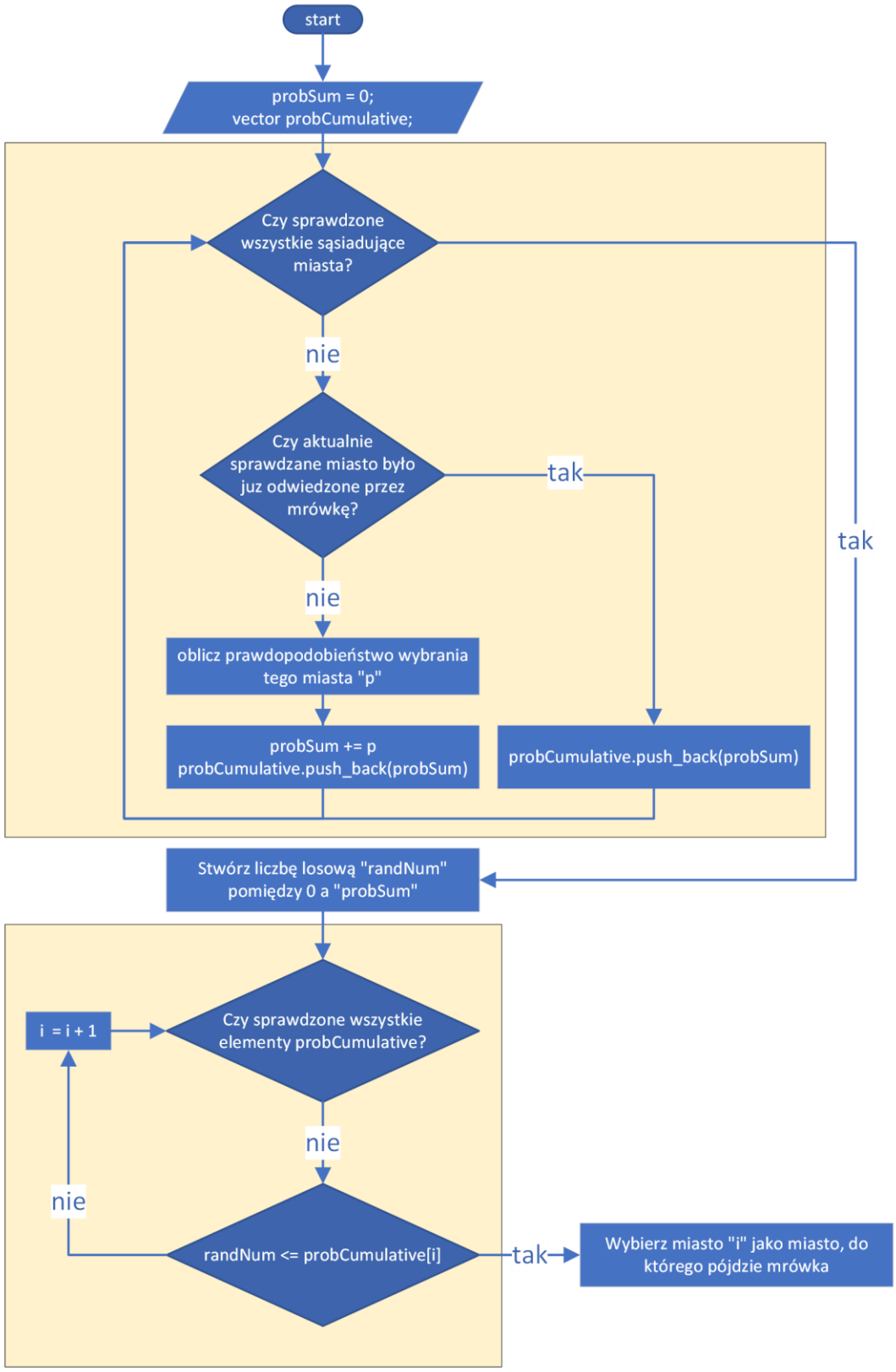
Po znalezieniu rozwiązań przez wszystkie mrówki następuje aktualizacja feromonu w macierzy. Na początku wszystkie wartości w macierzy mnożone są przez współczynnik ρ, aby zasymulować wyparowanie feromonu, a następnie we wszystkich ścieżkach pokonanych przez mrówki dodawane są odpowiednie wartości. W zaimplementowanym przeze mnie algorytmie cyklicznym CAS ta wartość wynosi:

2. ACO i TSP (13)(CAS)(1)(aktualizacja feromonu)

W CAS stała ilość feromonu Q_{Cycl} dzielona jest przez długość trasy L znalezionej przez k-tą mrówkę - L^k.

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} \dfrac{Q_{Cycl}}{L^k} & \text{jeżeli } k\text{-ta mrówka przeszła z } i \text{ do } j \text{ na swojej trasie} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Następnie zapisywane jest najlepsze dotychczasowe rozwiązanie i wracamy na początek pętli.



Rysunek 2 – przedstawienie logiki stojącej za wyborem następnego miasta przez mrówkę

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:

1. tsp_6_1.txt (132)
2. tsp_6_2.txt (80)
3. tsp_10.txt (212)
4. tsp_12.txt (264)
5. tsp_13.txt (269)
6. tsp_14.txt (282)
7. tsp_15.txt (291)
8. tsp_17.txt (39)

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Do wykonania badań wybrano następujący zestaw instancji:

1. tsp_6_1.txt (132)
2. tsp_6_2.txt (80)
3. tsp_10.txt (212)
4. tsp_12.txt (264)
5. tsp_13.txt (269)
6. tsp_14.txt (282)
7. tsp_15.txt (291)
8. tsp_17.txt (39)
9. gr21.txt (2707)
10. gr24.txt (1272)
11. gr48.txt (5046)
12. gr96.txt (55209)
13. kroA100.txt (21282)
14. gr120.txt (6942)
15. kroB150.txt (26130)
16. kroB200.txt (29437)
17. rbg323.txt (1326)
18. gr431.txt (171414)

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (pliki zmodyfikowane w celu ujednolicenia sposobu zapisu i odczytu grafu z pliku. Zmodyfikowane pliki znajdują się w folderze „data")

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu mrówkowego występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. Poniższe testy uruchomione zostały z następującymi parametrami:

- dopuszczalny błąd: 20%
- liczba mrówek: liczba miast
- parametr α: 1
- parametr β: 2
- parametr $\tau_{0:}$ liczba miast podzielona przez oszacowany koszt rozwiązania problemu komiwojażera (za pomocą metody najbliższego sąsiada)
- Q: 100

Procedura badawcza polegała na uruchomieniu programu sterowanego plikiem inicjującym .INI po ustawieniu wyżej zapisanych parametrów w kodzie programu.

Treść pliku config.ini w poszczególnych testach:

```
1.
plik_do_otworzenia      data/tsp_6_1.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
2.
plik_do_otworzenia      data/tsp_6_2.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
3.
plik_do_otworzenia      data/tsp_10.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
4.
plik_do_otworzenia      data/tsp_12.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
5.
plik_do_otworzenia      data/tsp_13.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
6.
plik_do_otworzenia      data/tsp_14.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
7.
plik_do_otworzenia      data/tsp_15.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
8.
plik_do_otworzenia      data/tsp_17.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
9.
plik_do_otworzenia      data/gr21.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
10.
plik_do_otworzenia      data/gr24.txt
liczba_powtorzen            100
plik_do_zapisania       same_as_plik_do_otworzenia
11.
plik_do_otworzenia      data/gr48.txt
```

```
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia

12.
plik_do_otworzenia      data/gr96.txt
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia
13.
plik_do_otworzenia      data/kroA100.txt
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia
14.
plik_do_otworzenia      data/gr120.txt
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia
15.
plik_do_otworzenia      data/kroB150.txt
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia
16.
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        50
plik_do_zapisania       same_as_plik_do_otworzenia
17.
plik_do_otworzenia      data/rbg323.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
18.
plik_do_otworzenia      data/gr431.txt
liczba_powtorzen        1
plik_do_zapisania       same_as_plik_do_otworzenia
```

Wpływ schematu rozkładu feromonu na jakość i czas rozwiązania

Aby sprawdzić wpływ schematu rozkładu feromonu na jakość i czas rozwiązania uruchomiłem poniższe testy z takimi samymi parametrami jak w poprzednich testach oprócz zamienionego schematu rozkładu feromonu z CAS na QAS:

2. ACO i TSP (12)(QAS)(1)(aktualizacja feromonu)

W QAS przy przejściu po krawędzi (i, j), stała ilość feromonu Q_{Quan} dzielona jest przez długość krawędzi d_{ij}.

$$\Delta\tau_{ij}^{k}(t, t+1) = \begin{cases} \dfrac{Q_{Quan}}{d_{ij}} & \text{jeżeli } k\text{-ta mrówka przechodzi z } i \text{ do } j \text{ w jednostce czasu} \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Treść pliku config.ini w poszczególnych testach:

```
1.
plik_do_otworzenia      data/tsp_6_1.txt
liczba_powtorzen        100
plik_do_zapisania       same_as_plik_do_otworzenia
```

2.
plik_do_otworzenia data/tsp_6_2.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
3.
plik_do_otworzenia data/tsp_10.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
4.
plik_do_otworzenia data/tsp_12.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
5.
plik_do_otworzenia data/tsp_13.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
6.
plik_do_otworzenia data/tsp_14.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
7.
plik_do_otworzenia data/tsp_15.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
8.
plik_do_otworzenia data/tsp_17.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
9.
plik_do_otworzenia data/gr21.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
10.
plik_do_otworzenia data/gr24.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
11.
plik_do_otworzenia data/gr48.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia

12.
plik_do_otworzenia data/gr96.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
13.
plik_do_otworzenia data/kroA100.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
14.
plik_do_otworzenia data/gr120.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
15.
plik_do_otworzenia data/kroB150.txt
liczba_powtorzen 100
plik_do_zapisania same_as_plik_do_otworzenia
16.
plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 50
plik_do_zapisania same_as_plik_do_otworzenia
17.
plik_do_otworzenia data/rbg323.txt
liczba_powtorzen 10

plik_do_zapisania same_as_plik_do_otworzenia
18.
plik_do_otworzenia data/gr431.txt
liczba_powtorzen 1
plik_do_zapisania same_as_plik_do_otworzenia

Wpływ parametrów α i β na zachowanie się algorytmu

Aby sprawdzić wpływ czasu kadencji zakazanego ruchu na jakość i czas rozwiązania uruchomiłem poniższe testy z parametrem β równym 2 oraz:

- parametr α: 1

plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 10
plik_do_zapisania same_as_plik_do_otworzenia

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 14,7947s

- parametr α: 2

plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 10
plik_do_zapisania same_as_plik_do_otworzenia

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 3,0924s

- parametr α: 3

plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 10
plik_do_zapisania same_as_plik_do_otworzenia

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 2,32149s

- parametr α: 4

plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 10
plik_do_zapisania same_as_plik_do_otworzenia

Pomimo, że w pierwszych próbach udało się znaleźć rozwiązanie to w trzeciej próbie program utknął w minimum lokalnym. Najlepsza ścieżka jaką wtedy znalazł miała koszt 36285 co daje błąd w okolicy 23%.

Aby sprawdzić wpływ czasu kadencji zakazanego ruchu na jakość i czas rozwiązania uruchomiłem poniższe testy z parametrem α równym 1 oraz:

- parametr β: 1

plik_do_otworzenia data/kroB200.txt
liczba_powtorzen 10
plik_do_zapisania same_as_plik_do_otworzenia

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 68,1482s

- parametr β: 2

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 14,0515s

- parametr β: 3

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 3,95856s

- parametr β: 4

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 2,00538s

- parametr β: 5

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 1,53106s

- parametr β: 6

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 1,37509

- parametr β: 7

```
plik_do_otworzenia      data/kroB200.txt
liczba_powtorzen        10
plik_do_zapisania       same_as_plik_do_otworzenia
```

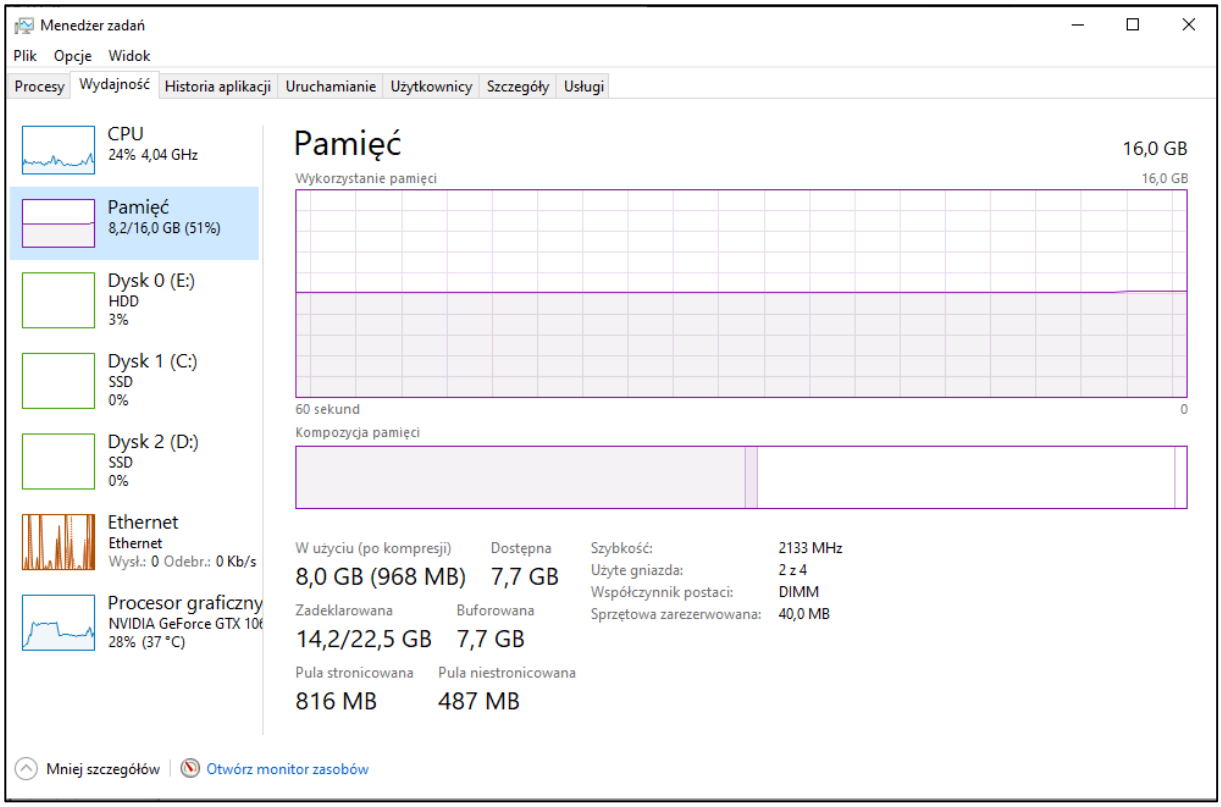
Program znalazł rozwiązanie mieszczące się w dopuszczalnym błędzie po średnim czasie 1,03938s

6. Wyniki

Opis sprzętu:
Procesor: Intel Core i7-7700 CPU @ 3.60GHz
Zainstalowana pamięć RAM: 16,0 GB
Typ systemu: 64-bitowy system operacyjny, procesor x64

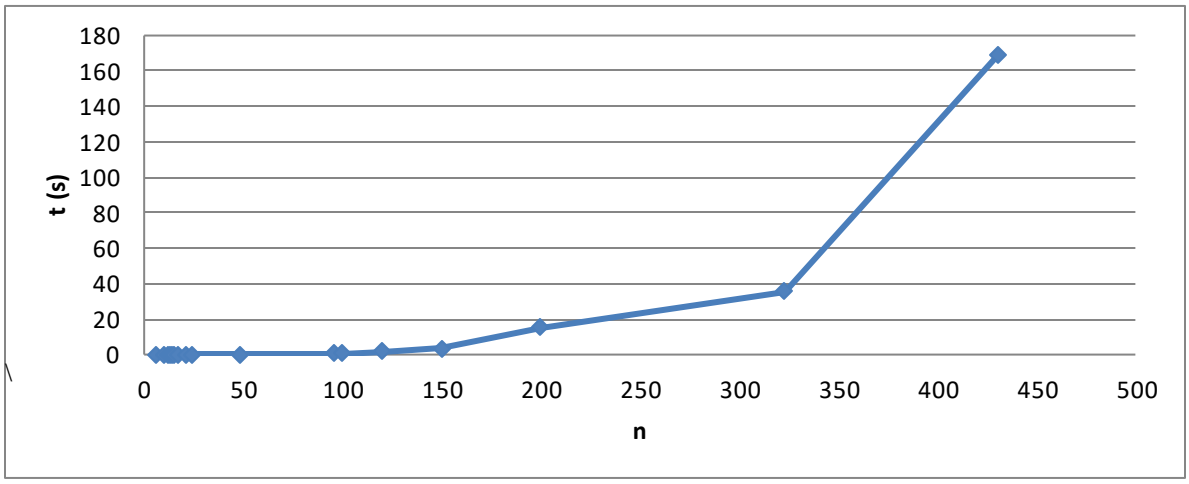
Wyniki zgromadzone zostały w folderze „results”

Program nie miał dużego wpływu na pamięć. Zużycie pamięci pozostawało na podobnym poziomie przed i w trakcie działania programu (rysunek 3)



Rysunek 3 - wykres pamięci komputera w trakcie działania programu

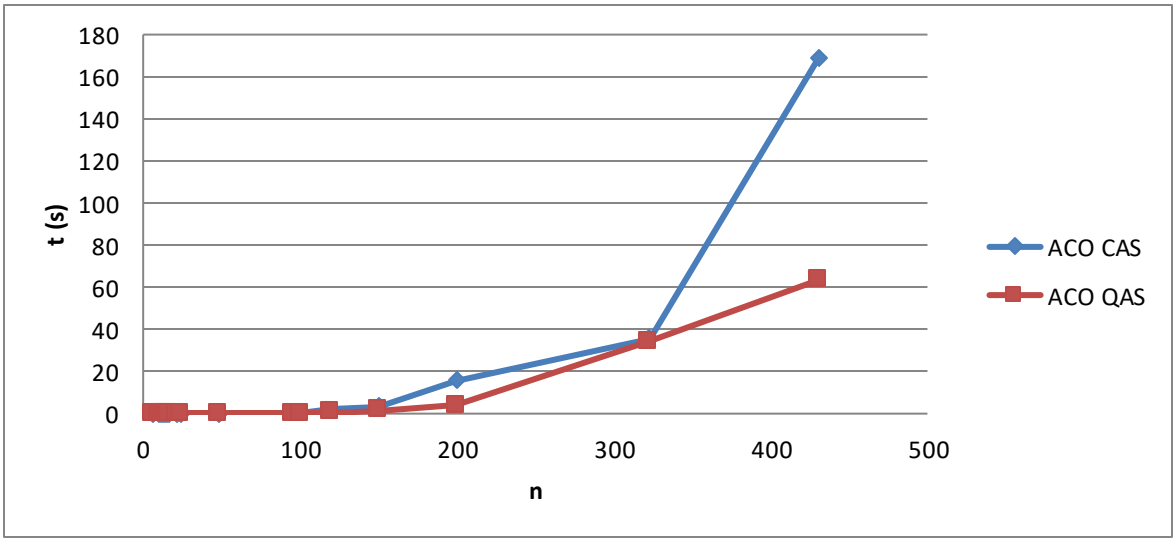
Wyniki w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji (rysunek 2).



Rysunek 4 - zależność czasu rozwiązania problemu od wielkości instancji

Metoda Ant Colony dobrze sobie radzi nawet dla dużych rozmiarów instancji. Wraz ze wzrostem rozmiaru instancji rośnie również czas. Warto zauważyć, że metoda „algorytm mrówkowy" nie jest deterministyczna tzn. przy dwóch próbach rozwiązania problemu przy takich samych danych wejściowych i parametrach możemy uzyskać inne wyniki. Związane jest to z losowością wyboru miasta przez mrówkę. Dlatego też ważne było wykonanie kilkakrotnie testów tej samej instancji, aby wykluczyć „szczęśliwe" znalezienie optymalnego rozwiązania w początkowym etapie rozwiązywania. Wszystkie testy oprócz największego zostały wykonane wielokrotnie, natomiast przed ostatecznymi poprawkami również i ostatnia instancja była sprawdzana i została rozwiązywana w podobnym czasie, co zwiększa wiarygodność wyniku.

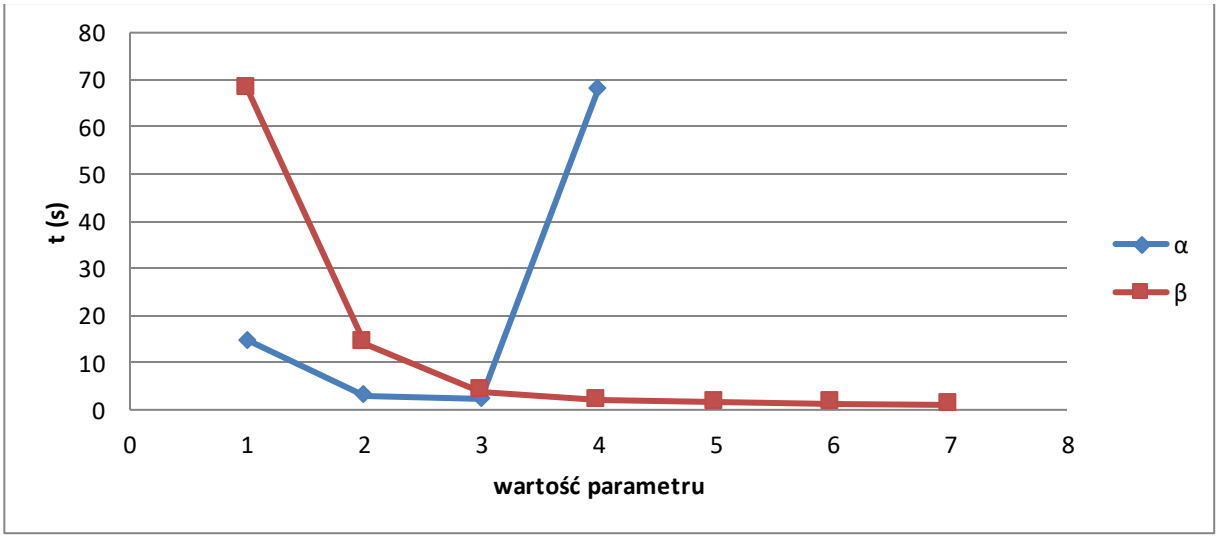
Wpływ schematu rozkładu feromonu na jakość i czas rozwiązania



Rysunek 5 – zależność czasu rozwiązania problemu od wybranego schematu rozkładu feromonu

Wybór schematu rozkładu ma duże znaczenie na jakość i czas rozwiązania. Po analizie wyników wynika, że schemat rozkładu QAS jest lepszy dla instancji o liczbie miast większej niż 14. Dla wyników mniejszych i równych dla niektórych instancji czas rozwiązywania był często nawet rząd dłuższy od CAS, jednak mówimy tu o różnicach wielkości paru milisekund. Lepsze wyniki mogą być spowodowane tym, że CAS rozkłada feromony na podstawie całego rozwiązania, co może nie być przydatną informacją dla mrówki, która często korzysta tylko z fragmentu tej ścieżki, który akurat może okazać się słaby. Natomiast QAS zmienia to podejście analizując każde przejście osobno, co jest wartościową informacją nawet dla mrówki, która nie skorzysta z całej ścieżki poprzednika.

Wpływ parametrów α i β na zachowanie się algorytmu



Rysunek 6 - zależność czasu od wartości parametrów α (przy β=2) i β (przy α=1)

Jak widać po wykresie zarówno parametr α jak i β mają duży wpływ na zachowanie algorytmu. Przy zbyt małym α program dla danej instancji wykonywał się wolno, ale również przy zbyt dużej wartości utknął w minimum lokalnym. Może to być spowodowane tym, że ten parametr reguluje wpływ feromonu na wybór miasta przez mrówkę:

Prawdopodobieństwo wyboru miasta j przez k-tą mrówkę w mieście i dane jest wzorem:

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{c_{i,l} \in \Omega}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}} & \bigvee c_{i,l} \in \Omega \\ 0 & \bigvee c_{i,l} \notin \Omega \end{cases}$$

gdzie:

c - kolejne możliwe (nie znajdujące się na liście $tabu_k$ miasto),

Ω - dopuszczalne rozwiązanie (nieodwiedzone miasta, nienależące do $tabu_k$),

η_{ij} -wartość lokalnej funkcji kryterium; np. $\eta = \frac{1}{d_{ij}}$ (*visibility*), czyli odwrotność odległości pomiędzy miastami,

α - parametr regulujący wpływ τ_{ij},

β - parametr regulujący wpływ η_{ij}.

Jako że ilość feromonu często była reprezentowana jako ułamek to zwiększenie potęgi zmniejsza wpływ feromonu na wybór miasta, co przy dostatecznie dużej wartości parametru powoduje, że algorytm zachowuje się głównie zachłannie przez co utyka w minimach lokalnych. Dla tej instancji maksymalna wartość α wynosi 3 (dla większej czasem się udawało, a czasem program utykał w minimum lokalnym).

β kieruje algorytm w drugą stronę, gdzie przy zwiększonej wartości wybór miasta zależy głównie od feromonu. Po wynikach można wnioskować, że im większa wartość β tym lepszy uzyskany czas rozwiązania. Jednak przy wartości 5 szybkość rozwiązania zwiększa już nieznacznie, a bazując na zaleceniach Dorigo w sprawie doboru parametrów uważam, że odpowiednio zachowany balans jest kluczowy do rozwiązania różnych instancji problemu.

7. Analiza wyników i wnioski

Algorytm zachowuje się ładnie i w dużej części przypadków znajduje prawidłowe rozwiązanie w relatywnie krótkim czasie. Jednakże należy pamiętać, że jest to algorytm heurystyczny i nie zawsze jest w stanie dojść do optymalnego rozwiązania.

Wpływ schematu rozkładu feromonu na jakość i czas rozwiązania

Schemat rozkładu feromonu QAS okazał się być lepszy dla badanych instancji od schematu CAS.

Wpływ parametrów α i β na zachowanie się algorytmu

Zarówno α jak i β regulują wpływ wartości kolejno feromonów i lokalnej funkcji kryterium. Mają wpływ na szybkość i jakość rozwiązania. W przypadku korzystania z kryterium visibility w tym projekcie warto jest ustawić wartość β na 5, aby zwiększyć prędkość rozwiązania i dalej trzymać się w ramach ustalonych przez Dorigo.