Badania Operacyjne
Informatyka 2024/2025

Antoni Kucharski
Piotr Rusak
Paweł Prus
Kacper Garus

1 czerwca 2025

Spis treści

1 Wstęp

Celem projektu jest analiza i implementacja algorytmu mrówkowego do rozwiązania problemu znalezienia najlepszej trasy przejazdu z przystanku A do przystanku B w sieci linii komunikacyjnych. Szukanie trasy uwzględnia czasy odjazdu pojazdów z przystanków, czas przejazdu pomiędzy przystankami i możliwość przesiadek. Użytkownik może podać czas, o której chce rozpocząć podróż jak i przystanek początkowy i końcowy.

2 Opis zagadnienia

2.1 Model matematyczny

2.1.1 Mapa

Mapa komunikacyjna jest grafem nieskierowanym $G = (V, E)$, gdzie $V \subset \mathbb{N}^2$ to zbiór wierzchołków reprezentujących punkty na mapie jako para współrzędnych (x, y), a $E \subset \{\{u, v\} : u, v \in V\}$ to zbiór krawędzi. Ponadto zachodzi zależność

$$\forall\ \{(x_u, y_u), (x_v, y_v)\} \in E\ \ |x_u - x_v| + |y_u - y_v| = 1$$

To oznacza, że sąsiednie wierzchołki różnią się dokładnie o jedną współrzędną, co odpowiada ruchowi w górę, w dół, w lewo lub w prawo.

2.1.2 Zbiór linii komunikacyjnych

Każda linia komunikacyjna jest opisana jako zbiór S_n, gdzie n to numer linii, a $S \subset V \times \{0, 1\}$ to zbiór par wierzchołków i flagi określające czy dany punkt jest przystankiem czy nie. Zakładamy, że punkt $(v, 0) \in S_n$ może występować kilka razy w zbiorze S_n, czyli może przejeżdżać przez ten sam punkt kilka razy w różnych momentach czasowych. Natomiast $(v, 1) \in S_n$ występuje tylko raz. Oznacza to, że dana linia nie zatrzymuje się na danym przystanku więcej niż raz.

2.1.3 Harmonogramy odjazdu

Dla danego przystanku v definiujemy zbiór

$$H_v = \{(n_0, h_0), (n_1, h_1), \ldots, (n_n, h_n)\}$$

określającą harmonogram odjazdu, gdzie n oznacza liczbę odjazdów z przystanku v, l_i to numer linii komunikacyjnej, a h_i to czas odjazdu tej linii.

2.1.4 Funkcja trasy

Jeśli przez N oznaczymy liczbę tras z z przystanku v_0 do przystanku v_m, to k-tą z nich definiujemy jako sekwencję

$$P_k = \left\{(v_0, v_1^k, n_0^k), (v_1^k, v_2^k, n_1^k), \ldots, (v_{m-1}^k, v_m, n_{m-1}^k)\right\}$$

gdzie m to liczba przystanków na trasie, $v_0, v_1^k, \ldots, v_{m-1}^k v_m$ to kolejne przystanki, a n_i^k to linia komunikacyjna, którą jedziemy z przystanku v_i^k do przystanku v_{i+1}^k. Definiujemy funkcję

$$f(v_0, v_m) = \{P_1, P_2, \ldots, P_N\}$$

zwracającą zbiór wszystkich tras z przystanku v_0 do przystanku v_m rozpoczynających się w chwili t_0.

2.1.5 Funkcja celu

Niech

$$t(n, v, t_0) = \min(\{h - t_0 : (n, h) \in H_v, h \geqslant t_0\})$$

oznacza czas oczekiwania na linię n będąc na przystanku v w chwili t_0, a

$$d(n, u, v) = \min(\{h_v - h_u : (n, h_u) \in H_u, (n, h_v) \in H_v, h_v > h_u\})$$

oznacza czas przejazdu linią n z przystanku u do przystanku v. Wtedy funkcja celu dla danej trasy $P = \{(v_0, v_1, n_0), \ldots, (v_{m-1}, v_m, n_{m-1})\} \in f(v_0, v_m)$ jest zdefiniowana jako

$$T(P, t_0) = \sum_{i=0}^{m-1} (t(n_i, v_i, t_i) + d(n_i, v_i, v_{i+1}))$$

gdzie t_i to czas przyjazdu na przystanek v_i.

2.2 Szukana wartość

Szukana przez nas wartość to trasa $P_{opt} \in f(v_0, v_m)$ z przystanku $v_0 \in V$ do $v_m \in V$ zaczynając o godzinie t_0 o minimalnej wartości funkcji celu czyli

$$P_{opt} = \underset{P \in f(v_0, v_m)}{\arg\min}\ T(P, t_0)$$

2.3 Możliwe zastosowania

Zastosowanie algorytmu mrówkowego do problemu komunikacji miejskiej może być przydatne w wielu sytuacjach, takich jak:

- Planowanie codziennych dojazdów do pracy lub szkoły, uwzględniając czas odjazdu pojazdów i przesiadek.
- Organizacja transportu publicznego w miastach, aby zoptymalizować trasy i czasy przejazdu.
- Pomoc turystom w znalezieniu najdogodniejszej trasy zwiedzania miasta z uwzględnieniem komunikacji miejskiej.

3 Opis algorytmu

3.1 Idea algorytmu

Problem znalezienia najlepszej trasy przejazdu z przystanku A do przystanku B w sieci linii komunikacyjnych można rozwiązać przy pomocy algorytmu mrówkowego, który jest inspirowany zachowaniem mrówek w poszukiwaniu najkrótszej drogi do źródła pożywienia. Algorytm ten polega na symulacji ruchu mrówek, które eksplorują graf komunikacyjny, pozostawiając feromony na krawędziach, co pozwala innym mrówkom na znalezienie lepszej trasy. W kontekście komunikacji miejskiej, mrówki będą reprezentować użytkowników transportu publicznego, którzy szukają optymalnej trasy przejazdu, uwzględniając czas odjazdu pojazdów, czas przejazdu pomiędzy przystankami i możliwość przesiadek.

3.2 Adaptacja algorytmu

3.2.1 Reprezentacja rozwiązania

Rozwiązaniem naszego problemu jest sekwencja krotek postaci (v, t, n), gdzie n to numer linii komunikacyjnej i $(v, 1) \in S_n$ to przystanek, na którym się znajdujemy w chwili t. Krotki te są uporządkowane rosnąco według czasu t i reprezentują kolejne przystanki na trasie, na których mrówka się zatrzymuje lub wsiada do pojazdu komunikacji miejskiej. Rozwiązaniem początkowym jest wartość pusta `None`.

3.2.2 Opis procedur

Pierwszym krokiem jest wygenerowanie struktury danych zawierającej informację o czasie przejazdu pomiędzy każdymi kolejnymi przystankami dla każdej linii komunikacyjnej jak i czasów oczekiwania na przystankach na każdą linię.

Następnie uruchamiamy pętlę, która będzie wykonywana przez określoną liczbę iteracji. W każdej iteracji algorytm generuje tyle tras ile mamy mrówek, które będą symulowane. Dla każdej mrówki póki nie znajdzie się w przystanku końcowym algorytm waży krawędzie wychodzące z przystanku, na którym się znajduje, i wybiera jedną z nich na podstawie prawdopodobieństwa, które jest obliczane na podstawie ilości feromonu na krawędzi oraz odległości do przystanku końcowego. Dokładny wzór na wagę sąsiada jest następujący:

$$w = \frac{p^\alpha}{d^\beta}$$

, gdzie p to ilość feromonu na krawędzi, d to odległość do przystanku końcowego, a α i β to metaparametry wpływające na wagę krawędzi.

Następnie wybierana jest trasa o najmniejszym koszcie przejazdu, czyli o najmniejszym czasie przejazdu. Po przejściu wszystkich mrówek, algorytm aktualizuje ilość feromonu na krawędziach poprzez parowanie feromonu na wszystkich krawędziach (mnożąc je przez $1 - r$, gdzie r to współczynnik parowania feromonu) i dodanie do nich ilości feromonu pozostawionego przez mrówki, które

znalazły trasę. Ilość feromonu pozostawionego przez mrówki jest obliczana jako

$$p = \frac{q}{T(P, t_0)}$$

gdzie q to stała określająca ilość feromonu, który mrówki zostawiają na krawędziach grafu po znalezieniu trasy, a $T(P, t_0)$ to czas przejazdu znalezionej trasy.

3.3 Pseudokod algorytmu

4 Aplikacja

4.1 Wymagania wstępne

Aplikacja jest napisana w języku Python i wymaga zainstalowania biblioteki `pygame` do obsługi interfejsu graficznego. Uruchomienie aplikacji odbywa się poprzez wykonanie polecenia:

```
python3 run.py
```

w katalogu głównym projektu.

4.2 Charakterystyka danych wejściowych

Dane wejściowe do aplikacji są trzymane w katalogu `data` i są to pliki pythonowe z rozszerzeniem `.py`. Każdy plik zawiera listę linii komunikacyjnych zgodnych z definicją zbioru S_n z sekcji (2.1.2). Plik ten również zawiera listę godzin odjazdów z początkowego przystanku dla każdej linii komunikacyjnej. Pełny harmonogram odjazdu jest generowany na etapie wykonania programu. Dodatkowo plik zawiera listę kolorów, które będą używane do rysowania linii komunikacyjnych.

4.3 Ustawianie parametrów

W celu ustawienia mapy i linii komunikacyjnych należy z wybranego pliku w katalogu `data` zaimportować zmienne

- `lines`
- `starting_times`
- `colors`

nic więcej nie jest wymagane.

Ustawienie przystanku początkowego i końcowego odbywa się poprzez przypisanie do zmiennych `start_stop` i `end_stop` krotek reprezentujących współrzędne przystanków w postaci (x, y). **Punkty muszą być przystankami komunikacyjnymi!** To oznacza, że musi istnieć linia komunikacyjna w odpowiednim pliku z `data`, która zawiera ten punkt z flagą `True`. Dodatkowo można

ustawić czas rozpoczęcia podróży poprzez przypisanie wartości do zmiennej `start_time` w formie liczby całkowitej.

Dodatkowo można ustawić parametry algorytmu mrówkowego poprzez przypisanie wartości do następujących zmiennych w konstruktorze klasy `AntColony`:

- `n_ants` — liczba mrówek, które będą symulowane w algorytmie,
- `n_iterations` — liczba iteracji algorytmu, czyli liczba cykli, w których mrówki będą eksplorować graf,
- `evaporation_rate` — współczynnik parowania feromonów, który określa, jak szybko feromony znikają z krawędzi grafu,
- `alpha` — współczynnik wpływu feromonów na wybór trasy przez mrówki,
- `beta` — współczynnik wpływu odległości na wybór trasy przez mrówki,
- `q` — stała określająca ilość feromonu, który mrówki zostawiają na krawędziach grafu po znalezieniu trasy,

4.4 Interpretacja logów i wyników

Wyniki działania algorytmu mrówkowego są wyświetlane w konsoli w postaci logów, które zawierają informacje o obecnej iteracji oraz najlepszej trasie znalezionej do tej pory. Logi te zawierają jedynie czas potrzebny do pokonania znalezionej trasy.

Wynikiem algorytmu jest rozwiązanie opisane w sekcji (3.2.1), czyli sekwencja krotek reprezentujących przystanki, na których mrówka się zatrzymuje lub wsiada do pojazdu komunikacji miejskiej.

5 Eksperymenty

Do napisania po eksperymentach Piotra Rusaka.

6 Podsumowanie

6.1 Wnioski

Algorytm mrówkowy okazał się skuteczny w znajdywaniu optymalnych tras przejazdu w sieci linii komunikacyjnych. Mimo probabilistycznego charakteru algorytmu, udało się znaleźć trasy o minimalnym czasie przejazdu.

6.2 Możliwości rozwoju

Algorytm mrówkowy może być dodatkowo rozszerzony o uwzględnienie dodatkowych czynników, takich jak:

- Ruch drogowy i jego wpływ na czas przejazdu.

- Preferencje użytkowników dotyczące komfortu podróży, takie jak unikanie przesiadek.
- Dodanie możliwości pokonania trasy pieszo lub rowerem.

6.3 Podział zadań

Podział zadań w projekcie był następujący:

- Antoni Kucharski: Implementacja narzędzia do tworzenia mapy linii komunikacyjnych, dokumentacja projektu.
- Piotr Rusak: Implementacja algorytmu mrówkowego, przeprowadzenie eksperymentów.
- Paweł Prus: Implementacja algorymu mrówkowego i interfejsu graficznego aplikacji.
- Kacper Garus: Wytworzenie linii komunikacyjnych i harmonogramów, pomoc przy dokumentacji.