Aplikacja usprawniająca pracę kuriera – problem komiwojażera w praktyce

Aleksander Krzeszowski

18 stycznia 2017

Spis treści

Wstęp

W literaturze można odnaleźć liczne prace skupiające się na analizie wydajności i optymalizacji różnych algorytmów rozwiązujących problem komiwojażera. Celem poniższej pracy jest stworzenie łatwej w obsłudze aplikacji, umożliwiającej odnalezienie optymalnej trasy łączącej wprowadzone miejsca docelowe. Założenie dostępności dla zwykłego użytkownika nakłada pewne wymagania na aplikację: interfejs musi być prosty w obsłudze, a aplikacja powinna być dostępna na różnych urządzeniach (komputery, tablety, urządzenia przenośne).

Wymagania te spełnia aplikacja internetowa – dostępna przez przeglądarkę. Taki rodzaj aplikacji pozwala na realizację architektury klient-serwer. W tym modelu obliczenia są wykonywane po stronie serwera, nie obciążając klienta, który wyświetla tylko interfejs aplikacji.

Kolejną konsekwencją przeznaczenia aplikacji dla zwykłych użytkowników jest konieczność zapewnienia odpowiedniej wydajności. Przetwarzanie danych powinno przebiegać w możliwie krótkim czasie, tak by użytkownik nie musiał czekać na zakończenie obliczeń. Znalezienie dokładnego rozwiązania problemu komiwojażera w czasie wielomianowym jest niemożliwe [2].

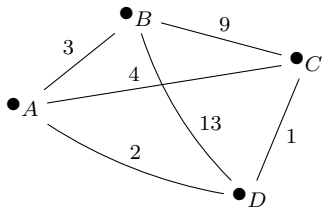
Poszukiwanie optymalnego rozwiązania już dla małej liczby miejsc docelowych wymagałoby znacznego czasu, nawet na wydajnym komputerze. Z tego powodu aplikacja powinna wykorzystywać algorytm sub-optymalny, który pozwala odnaleźć rozwiązanie w stosunkowo krótkim czasie, a odnalezione trasy są wystarczająco optymalne dla praktycznych zastosowań.

Rozdział 1

Definicja i rozwiązanie problemu

1.1 Określenie problemu

Problem komiwojażera polega na wyznaczeniu trasy łączącej wybrane punkty[1], przy dodatkowych warunkach: każdy punkt może zostać odwiedzony wyłącznie raz, poza wybranym punktem będącym początkiem i końcem trasy. Można więc powiedzieć, że rozwiązanie stanowi permutacja n punktów, a optymalnym rozwiązaniem jest permutacja o minimalnej sumie odległości między punktami[1].



Rysunek 1.1: Przykład symetryczny: Reprezentacja grafowa

[1]Pierwotnie problem dotyczył tras między miastami, przez co w opracowaniach lub algorytmach punkty pośrednie często nazywa się miastami

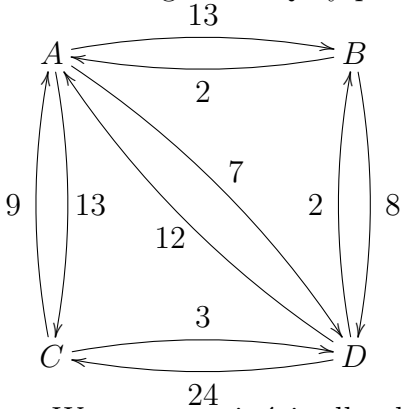
	A	B	C	D
A	0	3	4	2
B	3	0	9	13
C	4	9	0	1
D	2	13	1	0

Tabela 1.1: Przykład symetryczny: Macierz sąsiedztwa

Przykładowy problem został przedstawiony na grafie 1.1. Przyjmując B za punkt startowy, optymalną trasą dla takiego zbioru punktów jest na przykład: $B \to A \to D \to C \to B$ o długości 15.

Punkty pośrednie stanowią wierzchołki grafu, a trasy je łączące to krawędzie o wagach równych odległościom między punktami. Powyższy prosty przykład to wariant symetryczny problemu komiwojażera – odległości między dwoma punktami są identyczne w każdym kierunku. Dla takiego grafu wystarczy odnaleźć wagi dla $\frac{n(n-1)}{2}$ krawędzi, ponieważ są nieskierowane.

Jednak w rzeczywistym zastosowaniu (a także w zrealizowanej aplikacji) mamy do czynienia z asymetryczną wersją problemu, a więc ze skierowanym grafem. Jest to spowodowane tym, że co prawda z dowolnego punktu możemy dotrzeć do innego, jednak trasy między dwoma punktami mogą być inne (na przykład ulice jednokierunkowe). W rezultacie konieczne jest odnalezienie $n^2 - n$ odległości między punktami.



Warto zauważyć że dla algorytmów nie ma znaczenia w jakiej jednostce jest wyrażona waga – można więc tym samym algorytmem optymalizować

zarówno odległość, jak i czas przejazdu.

1.2 Algorytmy genetyczne

1.3 Selekcja

1.4 Krzyżowanie

Rozdział 2

Implementacja

2.1 Struktura projektu

(Opis najważniejszych klas, projektów w solucji)

2.2 Integracja systemów zewnętrznych

2.2.1 Google Maps

2.2.2 OSRM: Open Source Routing Machine

Rozdział 3

Testowanie aplikacji

3.1 Testy manualne

(prezentacja walidacji, uruchomienie na duzych danych, innych systemach)

3.2 Testy jednostkowe

3.3 Badanie wydajności

(Porównanie poprawności i szybkości rozwiązania dla różnych konfiguracji Solvera)

Podsumowanie

Bibliografia

[1] Zbigniew Michalewicz. *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Wydaw. Naukowo-Techniczne, Warszawa, 2003.

[2] Christos H Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237, 1977.