Skrzyżowania ++

Piotr Skibiak, Tomasz Kwiecień, Marcin Nowak, Ksawery Głaz, Paweł Łabno

AGH, Wydział IEiT, Informatyka

Abstrakt

Projekt ma na celu skrócenie oczekiwania samochodów na skrzyżowaniach, poprzez zastosowanie optymalnego ustawienia czasu świateł. Optymalne (albo prawie optymalne) ustawienie świateł wyznaczane jest poprzez użycie jednego z algorytmów stadnych - algorytmu kukułki.

Spis Treści

1 Przedstawienie problemu

W miastach (zwłaszcza tych wielkich) często spotyka się zjawisko jakim są zakorkowane ulice pełne samochodów. Jednak każda ulica ma inne natężenie ilości samochodów w różnych porach dnia, a światła na skrzyżowaniach mają stałe czasy świecenia. W czasie tak zwanych godzin szczytu przez jedne ulice przejeżdża więcej samochodów niż przez inne, co powoduje korki na drogach. Można by jednak upłynnić ten ruch poprzez wydłużenie czasu świecenia świateł zielonych dla bardziej uczęszczanych ulic, a skrócenie go dla tych mniej uczęszczanych. I właśnie tym zajmuje się nasz program. Czyli badaniem w danych godzinach natężenia ruchu na danych ulicach i optymalizowanie czasów świecenia świateł tak aby ten ruch był płynniejszy dla całego wybranego (narysowanego) obszaru. W tym celu korzystamy z algorytmu kukułki, który nam optymailizuje czasy świecenia świateł dla poszczególnych skrzyżowań.

2 Cele projektu

Projekt ma na celu umożliwienie użytkownikowi znalezienie optymalnego ustawienia czasów świecenia świateł (światła zielonego i czerwonego) w sieci skrzyżowań ustalonych przez użytkownika. Aplikacja powinna dać użytkownikowi możliwość ręcznego wprowadzenia sieci skrzyżowań, zapisu konfiguracji do pliku, lub wczytania wcześniej zapisanej.

3 Wstępne założenia

3.1 Środowisko Implementacji

Aby pogodzić potrzebę szybkiego wykonywania symulacji, oraz zdążyć wykonać projekt w przeznaczonym do tego czasie, językiem wybranym do implementacji jest Java. Ustaliliśmy, że aby zmniejszyć prawdopodobieństwo wystąpienia niespójności związanej z wykorzystywaniem różnych narzędzi, aplikacja będzie pisana przy pomocy Eclipse IDE.

3.2 Praca w zespole

Praca grupowa była wspomagana poprzez wykorzystanie sytemu kontroli wersji GIT, oraz poprzez utorzenie Google Doc'a projektu. W trakcie realizacji przeprowadziliśmy również kilka spotkań mających na celu, kontrolę przebiegu prac, wyjaśnienie watpliwości / niejasności wykrytych w trakcie iteracji, a także przydział kolejnych zadań.

3.3 Ogólne założenia

W naszym projekcie musieliśmy podjąć pewne złożenia (uproszczenia) gdyż nie jest możliwe aby zrobić program dla tego problemu (w tak krótkim czasie jaki mieliśmy) rozpatrując wszystkie najdrobniejsze szczegóły. Pominęliśmy

przedewszystkim czas świecenia żółtego światła przy zmianie z czerwonego na zielone oraz odwrotnie. Nie są rozpatrywane przypadki gdy samochód na skrzyżowaniu zawraca lub drogi jedno kierunkowe czy tak zwane ślepe uliczki. Pominęliśmy także dość ważny przypadek jakim są przejścia dla pieszych z światłami, które są prawie przy każdym skrzyżowaniu. Wiele osób za pewne znalazłoby jeszcze więcej szczegółów, które można by rozpatrzyć, ale aby to uwzględnić to przy naszym programie trzeba by pewnie spędzić jeszcze kilka miesięcy albo i dłużej. Kolejnym uproszczeniem jest fakt, że program działa tylko dla skrzyżowań, z których wychodzą jedynie cztery drogi. Można stworzyć sytuację dla mniejszej ilości dróg wychodzących ze skrzyżowania jeśli narysujemy drogę połączoną z generatorem, dla której jako parametry podamy 0 - maksymalna ilość samochodów oraz 0 - intensywność.

4 Opis logiki

Nasza logika składa się z kilku klas, za pomocą których możemy tworzyć potrzebne w naszym projekcie obiekty: Generatory, Skrzyżowania, Drogi i Samochody. Obiekty te tworzone są przez GUI, a następnie server wykonuje symulacje za ich pomocą.

4.1 Klasa Generator

Obiekt, który jest instancją tej klasy jest zakończeniem dróg, których jeden z końców nie jest podłączony do żadnego ze skrzyżowań. Zadaniem tego obiektu jest tworzenie nowych samochodów pojawiających się na mapie oraz usówanie tych wyjeżdżających z mapy. Klasa posiada metodę moveVehicles odpowiedzialną właśnie za pojawianie się i znikanie samochodów na mapie. Posiada także funkcję, która za pomocą losowości decyduje czy obiekt Generator powinien w danym momencie stworzyć obiekt samochód.

4.2 Klasa Junction

Jest to klasa abstrakcyjna, po której dziedziczy klasa Junction, której instancją jest obiekt Skrzyżowanie. Poza konstruktorem posiada ona głównie gettery, które ułatwiają dostęp do dróg wejściowych oraz wyjściowych, a także współrzędnych danego skrzyżowania. Posiada także metodę, która dodaje na każdą ulicę wychodzącą z tego skrzyżowania jakąś stałą liczbę samochodów, która jest zależna od ustalonego wcześniej natężenia ruchu na danej drodze.

4.3 Klasa Intersection

Klasa ta dziedziczy po abstrakcyjnej klasie Junction. Instancja te klasy odpowiada za obsługę świecenia świateł oraz za decyzję, w którą drogę dany samochód, który wjeżdża na to skrzyżowanie ma skręcić.

4.4 Klasa Road

Instancje tej klasy przechowują samochody, które znajdują się na danej drodze. Posiada ograniczenia w postaci maksymalnej liczby samochodów, które mogą się znajdować na tej drodze oraz natężenie ruchu, które określa ile samochodów wjeżdża na daną drogę w ciągu godziny. Każda droga ma współrzędne początkowe i końcowe. Pojedyńczy obiekt "Droga" odpowiada za drogę jednokierunkową, dlatego gdy rysujemy drogę łączącą dwa skrzyżowania lub skrzyżowanie i generator to tworzą się dwa obiekty typu "Road" (jeden w jedną stronę i drugi w drugą). Klasa ta posiada metodę getFirstVechicle, która umożliwia przekazanie najbliższego do skrzyżowania / generatora samochodu.

4.5 Klasa Simulator

W obiekcie typu "Simulator" przechowywane są obiekty, które reprezentują cały model sieci ulic. Zapewnia także wykorzystanie algorytmu kukułki, do której przekazujemy parametry - długość świecenia świateł, dzięki czemu mamy zapewnioną symulację.

4.6 Klasa Vehicle

Posiada metody i pola, które są ułatwiają ruch samochodów czyli gettery i settery dla współrzędnych danego samochodu, oraz czas oczekiwania, bądź informacja o danym samochodzie czy jest w konkretnym momencie w ruchu lub czeka przed skrzyżowaniem.

5 Algorytm kukułki

Optymalizacja ruchu na skrzyżowaniach jest dokonywana poprzez minimalizację funkcji f. Funkcja ta określona jest jako suma czasów oczekiwań samochodów na skrzyżowaniach. Minimalną (właściwie to pseudominimalną, ze względu na niedeterministyczny charakter funkcji f) wartość funkcji f obliczamy za pomocą algorytmu kukułek. Pseudokod obrazujący działanie algorytmu umieszczony jest poniżej

```
CuckooSearch(int MaxGenerations)
    population = generateInitialPopulationOfNests;
    t = 0;
    bestSolution = randomSolution;
    while (t < MaxGenerations)
        nest = randomNest(population);
        newSolution = generateRandomLevySolution(nest);
        if (Energy(nest) -  Energy(newSolution) > 0)
            nest.setSolution(newSolution)
        if (Energy(nest)  Energy(bestSolution)
            bestSolution = nest;
        sort(population);
        replaceWorstFraction(0.1);
        t++;
    return bestSolution;
```

Rozwiązanie funkcji f zostało określone jako para tablic R i G, takich że:
G[i] - czas świecenia światła zielonego na skrzyżowaniu numer i
R[i] - czas świecenia światła czerwonego na skrzyżowaniu numer i

6 Obsługa aplikacji

Uruchamiając nasz program wyświetla się okno, które jest podzielone na 2 seskcje obszar, w którym możemy rysowawać skrzyżowania i drogi oraz przedstawiona tam jest wizualnie symulacja. W drugim obszarze po prawej stronie znajdują się wszystkie przyciski do obsługi programu.

6.1 Dodaj Generator

Gdy przycisk ten zostanie naciśnięty możemy za pomocą kliknięcia lewym przyciskiem myszy na obszarze do rysowania stworzyć generator. Zostanie on narysowany w kliknętym miejscu jako czerwone kółko poprzez pobranie współrzednych kursora, a także zostanie stworzony obiekt Generator, który zostanie dodany do listy generatorów.

6.2 Dodaj Skrzyżowanie

Analogicznie jak w przypadku Generatora zostanie stworzony obiekt Skrzyżowanie oraz w kliknięty miejscu pojawi się czarne kółko.

6.3 Dodaj Drogę

Po naciśnięciu przycisku "Dodaj Drogę" użytkownik może poprzez kliknięcie na dwóch skrzyżowaniach lub generatorze i skrzyrzowaniu (najpierw jednym, a następnie na drugim), które zostało wcześniej narysowane, stworzyć nowy obiekt Droga. Gdy zostaną kliknięte dwa (wcześniej narysowane) obiekty typu Generator lub skrzyżowanie, pojawi się okno z zapytaniem o to ile samochodów może pomieścić droga, a następnie ile (w ciągu godziny) samochodów może przejechać przez tą drogę czyli intensywaność zatłoczenia. Jeżeli jeden z zapytanych przez program parametrów nie zostanie podany lub będzie podany, ale w złym formacie np. użytkownik wpisze jakąś literę zamiast cyfry to obiekt nie zostanie stworzony. Po poprawnym podaniu wszystkich parametrów zostanie stworzony i narysowany obiekt typu droga. Narysowany jest podwójnie by było lepiej widać poruszające się samochody w obie strony. Jedna linia oznacza drogę w jedną stronę, a druga w drugą.

6.4 Zacznij od nowa

Czyści całą narysowaną bądź wczytaną mapę.

6.5 Zapisz

Zapisuje w wybranym przez użytkownika miaejscu na dysku narysowaną wcześniej mapę w postaci rysunku i list stworzonych obiektów.

6.6 Wczytaj

Wczytuje z dysku wcześniej zapisaną mapę.

6.7 Wyślij do serwera

Gdy narysujemy już bądź wczytamy mapę, musimy ją wysłać do włączonego wcześniej serwera żądanie, podając IP hosta, na którego komputerze jest włączony serwer, lub wpisując localhost jeśli mamy postawiony serwer lokalnie u siebie. Serwer zwraca nam ID jakie nadał temu żądaniu.

6.8 Sprawdź rozwiązanie

Klikając ten przycisk, a następnie podając ID żądania, pytamy serwer czy dla danego żądania zostało wyliczone optymalne rozwiązanie. Serwer zwraca nam odpowiednią informację, która zostaje wyświetlna w okienku.

6.9 Rozpocznij Symulację

Rozpoczęta zostanie symulacja poruszających się samochodów przedstawionych w postaci zielonych kółek. Symulacja jest przedstawiona wizualnie na narysowanej mapie. Współrzędne samochodów są pobierane co jakiś (bardzo króki) czas przez co mamy wrażenie, że samochody poruszają się płynnie.

6.10 Zatrzymaj Symulacje

Wstrzymuje wcześniej uruchomioną symulację. Wstrzymaną symulację możemy kontynuoawać po przez ponowne wciśnięcie przycisku "Rozpocznij Symulacje".

6.11 Wyjdź

Przyciśnięcie tego przycisku powoduje całokowite wyjście z programu.

6.12 Szybkość Symulacji

Jest to suwak umożliwiający nam wybór, jak szybko nasza symulacja ma się wykonywać. Dzięki temu możemy ją zwolnić aby lepiej zobaczyć rezultaty.

7 Rezultaty