Marek Żurawski

Bartosz Zalewski

Projekt

Współczesne metody heurystyczne

Temat: *Wizualizacja Algorytmu Mrówkowego*

Prowadzący: dr inż. Sebastian Kozłowski

# Opis algorytmu

## 1.1 Idea

Inspiracją do stworzenia algorytmu była obserwacja zachowania społeczności mrówek i pytanie jak, prawie ślepe zwierzęta odnajdują najlepszą drogę pomiędzy mrowiskiem i pożywieniem, w jaki sposób się między sobą komunikują i co sprawia, że podążają jedna za drugą. Odpowiedzialne za to wszystko są feromony, czyli substancje chemiczne wydzielane i rozpoznawane przez większość gatunków mrówek. Mrówki jako jednostki, nie są stworzeniami obdarzonymi inteligencją , zatem ich działania nie wynikają z wcześniejszych umów, czy nawet doświadczeń. Ich „mądrość” jest wynikiem pośredniej „komunikacji” opartej na prostej reakcji na feromony. Mrówki wędrując pozostawiają na podłożu tzw. Ślad feromonowy. Ten ślad jest swego rodzaju informacją dla towarzyszek, ale również dla siebie. Mrówki, mając do dyspozycji wiele dróg, wybierają raczej te, które wcześniej zostały oznakowane feromonami. Im większe ich stężenie, tym chętniej , ta właśnie ścieżka będzie wybierana. Oczywiście, jeśli kolejne mrówki, powędrują tą samą drogą, to ślad fermonowy nie będzie zanikał lecz zostanie wzmocniony. Jeżeli mrówka nie odnajduje żadnego znaku fermonowego, to wybiera drogę w sposób zupełnie przypadkowy lub o jej wyborze zdecydują inne względy (np. brak przeszkód). Jak widać, ślad feromonowy umożliwia mrówkom zarówno znalezienie drogi powrotnej do mrowiska, jak i dotarcie do źródeł pożywienia znalezionych wcześniej przez innych członków społeczności.

Bardzo interesującym zachowaniem kolonii jest jej umiejętność przystosowania się do zmian w środowisku np. znalezienie alternatywnej drogi w sytuacji, gdy stara przestaje spełniać swoje zadanie. Równocześnie okazuje się, że podążając raz odnalezioną, skuteczną drogą, mrówki nie zmieniają jej nawet wtedy, gdy z pewnych podwodów pojawi się możliwość skorzystania z lepszej, krótszej drogi.

## 1.2 Założenia

* Tworzona jest populacja mrówek. Jej rozmiar jest jednym z parametrów algorytmu
* Pojedyncza mrówka generuje swoją ścieżkę niezależnie od swoich towarzyszek
* Mrówka porusza się po grafie, szukając sekwencji wierzchołków grafu tworzącej najkrótszą drogę od wierzchołka startowego do końcowego
* Mrówka jest wyposażona w pamięć w której przechowuje listę odwiedzonych wierzchołków. Na starcie na liście znajduje się wierzchołek z którego mrówka startuje.
* Mrówka zostawia feromon na krawędziach dopiero w momencie, gdy zbuduje cała drogę.
* Decyzja, do którego z wierzchołków się udać w kolejnym kroku jest podejmowana w sposób probabilistyczny, na podstawie funkcji i zależy od koncentracji śladu feromonowego na dostępnych drogach.

# Działanie algorytmu

Poniżej przedstawiono pseudokod, określający ogólne działanie algorytmu.

DO WHILE iteracja<i

FOR EACH ant DO

     target=ant.move();

     IF target==mrowisko THEN DO

          updateFeromon(ant.scieżka);

ELSE IF target==NULL THEN DO

ant.die();

END

END

END WHILE

ant.move():

IF ant.hasFood() THEN DO

      target=ant.sciezka.previesNode(ant.position);

ELSE DO

      target=ant.chooseNextNode();

IF target==NULL THEN DO

RETURN NULL;

END

      ant.sciezka.apend(target)

END

ant.go(target);

RETURN target;

updateFeromon(scieżka):

FOR EACH node DO

      node.feromon= (1-p)\*node.feromon

END

FOR EACH node IN sciezka DO

      node.feromon=node.feromon+q/sciezka.lenght

END

ant.chooseNextNode():

Funkcja realizująca wybór kolejnego węzła obliczanego zgodnie z podanym niżej wzorem. Prawdopodobieństwo obliczamy tylko dla węzłów, których mrówka jeszcze nie odwiedzała. Jeśli mrówka z danego punktu może przejść jedynie do punktu który już wcześniej znajdował się na jej ścieżce (mrówka wykonała pętle), uśmiercamy taką mrówkę.

p – współczynnik redukcji feromonu (0;1)

q - ilość feromonu do pozostawienie na każdej poprawnej ścieżce

i – liczba iteracji algorytmu

Równianie określające prawdopodbieństwo z jakim mrówka wybierze kolejny wierzchołek grafu. Ze względu na to, że odległości pomiędzy wierzchołkami są jednakowe, parametr β jest równy 0.

gdzie,

τij(t) – ślad feromonowy w iteracji t pomiędzy wierzchołkami i oraz j,

α – parametr określający wpływ feromonów na zachowanie mrówki,

Kolejnym równaniem niezbędnym w algorytmie mrówkowym jest równanie określające aktualizację śladu feromonowego po każdej iteracji na całej długości trasy.

Gdzie,

m – liczba mrówek,

ρ - współczynnik odpowiadający za utlenianie się feromonów ( 0 < ρ<1),

# Manager Animacji

Manager animacji będzie modułem odpowiedzialnym za zbieranie kolejnych wyników pracy algorytmu i wydawanie poleceń wizualizacji zmian. Wyróżniliśmy następujące elementy podlegające animacji:

* Ruch mrówki do kolejnego węzła
* Zmianę stężenia feromonu w węzłach grafu

Wynikach prac menadżer animacji będzie informowany poprzez zmiany zachodzące w modelu danych w wyniku pracy algorytmu. Na podstawie otrzymanego zdarzenia menadżer stworzy odpowiedni obiekt/polecenie wykonania kolejnego kroku animacji i wstawi je do kolejki oczekującej na wizualizacje. Puszczone zostanie odpowiednie zdarzenie informujące o tym że na animacje czeka nowy element. Kolejne polecenia będą wizualizowane osobno, kolejno po sobie. Komponent odpowiedzialny za animacje, będzie pobierał polecenia z kolejki i odtwarzał animacje.

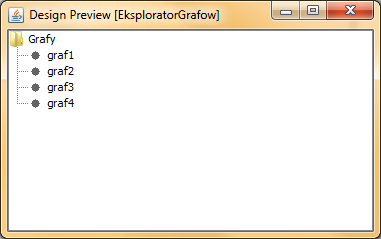
# Interfejs aplikacji

Aplikacja będzie posiadała prosty interfejs graficzny umożliwiający:

* Ładowanie plików z grafami wejściowymi
* Panel pozwalający na określenie parametrów wejściowych dla algorytmu
* Okno animacji, pozwalające na śledzenie wyników prac algorytmu
* Okno podglądu grafu

## Eksplorator grafów

Eksplorator grafów będzie prostym komponentem pozwalającym na przeglądanie załadowanych przez użytkownika grafów i podejrzenie ich zanim uruchomiony zostanie algorytm.



## 4.2 Okno parametrów

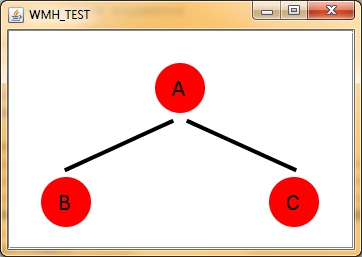
Ten komponent będzie zbiorem prostych kontrolek pozwalających na określenie wszystkich parametrów wejściowych algorytmu. Z tego panelu będziemy również uruchamiać algorytm. W momencie uruchomienia algorytmu automatycznie otworzy się okno animacji.

## 4.3 Okno animacji

Komponent będzie pozwalał na śledzenie przebiegu algorytmu.

## 4.4 Okno podglądu

Ten komponent będzie pozwalał na dojrzenie grafów, które zostały już wczytane. Będzie on dostępny z poziomu eksploratora.



# Ładowanie grafu z pliku

Grafy na których będzie można uruchamiać algorytm, będą ładowane z pliku xml. W pliku będą znajdowały się jedynie informacje o postaci grafu. Poprzez plik nie definiujemy żadnych parametrów dla algorytmu. Plik będzie zawierał jedynie informacje o węzłach i połączeniach w grafie. Jako pierwsze w pliku zdefiniowane są węzły, a następnie połączenia między nimi. W pliku definiujemy grafy nieskierowane o krawędziach bez wag. Węzły zawierają tylko jeden parametr, swoją nazwę, a krawędzie są definiowane poprzez podanie dwóch węzłów, które dana krawędź łączy. Kolejność podania węzłów niema znaczenia. Przykładowy plik definiujący graf ma następującą postać:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<graph>

<node label="A"/>

<node label="B"/>

<node label="C"/>

<node label="D"/>

<node label="E"/>

<node label="F"/>

<relation node1="A" node2="B"/>

<relation node1="A" node2="D"/>

<relation node1="A" node2="C"/>

<relation node1="D" node2="B"/>

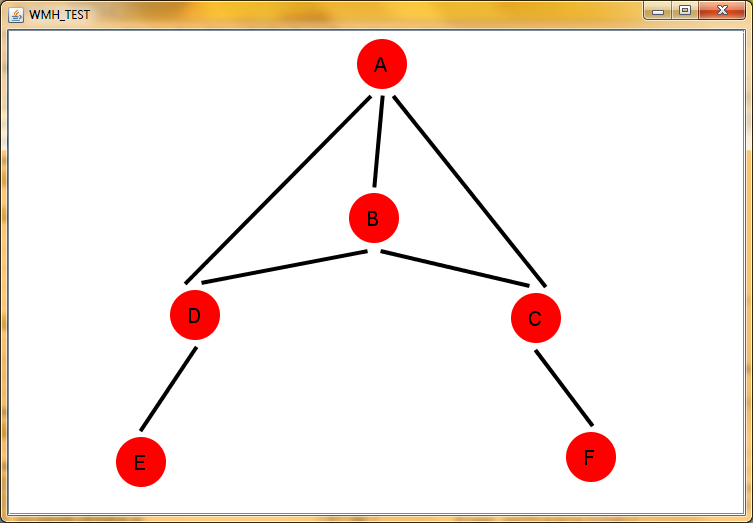
<relation node1="D" node2="E"/>

<relation node1="B" node2="C"/>

<relation node1="C" node2="F"/>

</graph>

Po wczytaniu takiego pliku otrzymamy następujący graf:



# Założenia programu

## 7.1 Złożoność obliczeniowa

Złożoność dla jednej iteracji algorytmu

* O(n) - O(n\*m) – Zależnie od tego czy graf jest postaci prostej ścieżki czy jest on grafem pełnym. Złożoność dotyczy obliczania możliwych ruchów dla każdej z mrówek gdzie:
* n – liczba mrówek
* m – liczba wierzchołków w grafie
* O(m) – Złożoność aktualizacji feromonu na każdej z krawędzi. Gdzie m – liczba wierzchołków grafu

## 7.2 Parametry wejściowe algorytmu

* n – ilość mrówek w mrowisku,
* p – współczynnik odparowania feromonu ( 0 < p < 1),
* q – ilość feromonu do pozostawienia na każdej poprawnej krawędzi,
* I – punkt wejściowy (mrowisko),
* O – punkt wyjściowy (jedzenie),
* f – początkowa wartość feromonu na każdej krawędzi grafu,
* i – liczba iteracji,

## 7.3 Kryteria stopu

* Liczba wykonanych iteracji,

## 7.4 Sytuacje awaryjne

* Mogą wystąpić przypadki w, których mrówka wykona pętle na swojej ścieżce. W takim wypadku będziemy uśmierać mrówkę a jej miejsce zastąpi nowa, która wyjdzie z mrowiska.

# Technologia

Program zostanie napisany w języku *Java* z wykorzystaniem bibliotek zewnętrznych. Do wizualizacji wykorzystana zostanie biblioteka *Netbeans Visual API*, która służy między innymi do wizualizacji grafów. Do testów wykorzystamy biblioteke *JUnit*, która pozwoli na stworzenie odpowiednich testów automatycznych dla naszej aplikacji. Nie wykluczamy że w trakcie tworzenia aplikacji będziemy posiłkować się dodatkowymi bibliotekami.

# Wykaz literatury

* R. Grzymkowski, K. Kaczmarek, S. Kiełtyka, I. Nowak „Wykłady z modelowania matematycznego: Wybrane algorytmy optymalizacji, algorytmy genetyczne, algorytmy mrówkowe, 2008
* <http://www.cs.put.poznan.pl/mradom/resources/labs/OptKomb/CI_wyklad_ewoluc_4.pdf>