International Journal of Current Engineering and Technology
©2016 INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

Traffic Control System using Maximum Flow Algorithm

Yamuna M.[#], Amber Raza[#], Kanav Anand[#*], Sakthi Kumar P[#], Sangam Kumar[#]

[#]School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, 632014, India

Accepted 18 Oct 2016, Available online 19 Oct 2016, **Vol.6, No.5 (Oct 2016)**

Abstract

Due to increase in use of vehicles, one major problem that the modernized society is facing is traffic congestion. Specifically in metro cities, this problem is high, because development of the city does not provide room for construction of new roads. Efficient use of the available facilities will be a positive approach in this situation. Any network is best represented using a weighted graph. In this paper we propose a method of using the available roads more efficient by using sensors in the signal junctions, using weighted graphs.

Keywords: *Graph, path, weighted graph, sensors, junction, threshold value.*

Introduction

In present era, traffic congestion has become one of the major problems of the modern cities. Due to the exponentially increasing population and a drastic increase in the vehicle count, the problem of congestion in traffic has become a major concern for the authorities. It is not only slowing down an individual's development but is also dragging down the cumulative development. There are a number of solutions for such problem like construction of new roads. But the cost of such solution would be very high. Also there is not enough space left in urban cities for the construction of new roads in order to minimize traffic congestion. But if the current road networks are used efficiently, we can still reduce the problem of traffic congestion to a great deal.

The traffic problem between the junctions could be minimized with the use of the weighted graphs. Suppose a vehicle 'X' has to reach destination 'E' from source 'A'. There are 3 possible paths available namely 'B', 'C', and 'D'. There will be sensors present on each path detecting the amount of traffic going through.As the count of vehicles passing, the more the flow of the path will be. The flow of each path will be displayed at source 'A'. The driver could then choose the path with the minimum flow available at that moment from the list of available paths.

An infinite flow rate is impossible as every path has limited capacities to carry traffic flow. There are usually multiple paths to reach a particular destination. Finding the maximal flow involves looking at all the possible routes between the starting and ending point.

We would like to propose a method of using the available roads more efficiently, during heavy traffic congestion time. Maximum flow minimum cut is a famous problem in graph theory, which helps to fine the maximum flow possible in any given network. This can be used to overcome the traffic congestion problem, and we propose a method here.

*Corresponding author: **Kanav Anand**

Preliminary Note

In this section we provide the definitions and results required for the proposed method.

A graph G has two types of elements, vertices and edges. Edge can be written as a set of two vertices, A and B, starting and the ending point. A vertex is expressed as a node or a dot. The vertex set of G is usually denoted by V(G), or V.

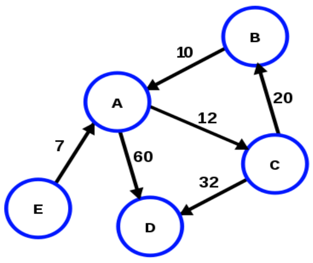
A path is said to be walk if it never includes any vertex twice except for the cycle where first vertex will be repeated. A directed arc, or directed edge, is represented graphically with an arrow drawn between the vertices.

A directed graph is analogous to an undirected graph except that it contains only arcs.

A weighted graph associates a label (weight) with every edge in the graph. Weights are mostly real numbers. They may be restricted to rational numbers or integers (http:// en.wikipedia. org/ wiki/ Glossary_of_ graph_ theory).

The following graph represents a weighted directed graph (http://en.m.wikibooks.org/ wiki/ File:CPT-Graphs-directed-weighted-ex1.svg).

In optimization theory, the max-flow min-cut theorem states that in a network, the maximum amount of flow passing from the source to the sink is equal to the minimum capacity that when removed in a specific way from the network causes the situation that no flow can pass from the source to the sink.



Maximal Flow Algorithm

Let **G** = (V, E) be a directed graph with source s ∈ V and sink t ∈ V, where edge (i, j) ∈ E has capacity c (i, j) > 0, flow f (i, j) > 0 and f (i, j) ≤ c (i, j) in the given flow network.

A flow is mapping f: E → R^+ which is denoted by f (i, j) or F_{ij}.

The capacity of an edge can be considered as the mapping c: E → R^+ and is denoted by c (i, j) or C_{ij}. It is used to represent the maximum amount of flow that can pass through an edge in the given flow network.
The Maximal Flow Algorithm maximizes the value of flow, i.e. the value of flow passing from the source to the sink. The main aim of the algorithm is route as much flow as possible from source 's' to sink 't'.

Theorem

Let P be a path from a to f in a network G satisfying the following conditions:

(a) For each properly oriented edge (i, j) in P, $F_{ij} < C_{ij.}$
(b) For each improperly oriented edge (i , j) in P,
$0 < F_{ij}$

Let Δ = min X.

Where X consists of the numbers C_{ij} - $F_{ij,}$ for properly oriented edges (i, j) in P, and $F_{ij,}$ for improperly edges (i,j) in P. Define

$F^*_{i,j}$ = $F_{i,j}$ if (i, j) is not in P

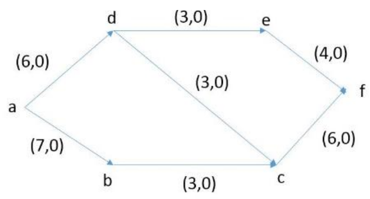
$F^*_{i,j}$ = F_{ij} + Δ if (i, j) is properly oriented in P.

$F^*_{i,j}$ = F_{ij} - Δ if (i, j) is not properly oriented in P.

Then $F^*$is a flow whose value is Δ greater than the value of F. If there are no paths satisfying the above conditions, the flow is maximum. To find the maximal flow, we follow the following steps:
(a) We start with a flow in which flow in each edge is zero initially.
(b) We search for the path satisfying the conditions mentioned in the theorem above. If no paths are found, then the flow is maximal.
(c) Increase the flow through the path by a value 'Δ'.

Example

In this section we provide a small example with six nodes a, b, c, d, e,f. As seen in the figure initially the flow value is assumed to be zero. Let us now proceed on implementing our proposed method.

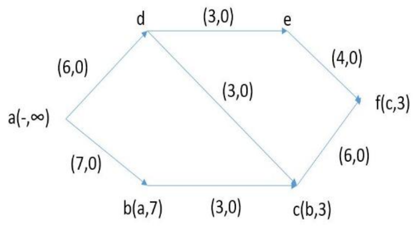


Consider, the above directed graph representing the traffic flow between junctions of a small area. The source could be taken as 'a' and the destination to be reached is 'f'.

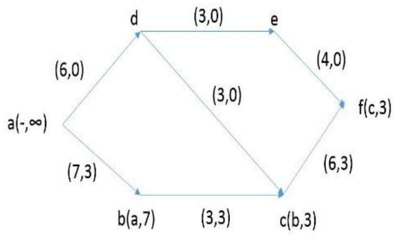
Each edge is labelled as (capacity (i), flow (j)) and can be represented as e (i,j) where 'i' depicts the edge's maximum capacity and 'j' represents the flow through that edge. The initial flow is considered zero here.

Each vertex above is labelled as (predecessor (v), value (v)).

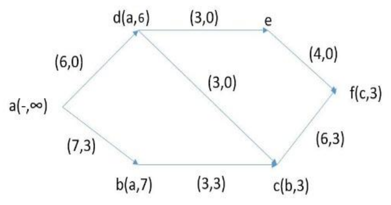
The source vertex (a) is labelled as (-, ∞).
Now consider vertex b,
Predecessor (b) = a
and
value (b) = min(∞, (7 - 0))
= 7
Similarly for vertex c,
Predecessor (c) = b and
Value (c) = min(7, (3-0))
= 3
Predecessor (f) = c and
value(f) = min (3, (6 – 0))
= 3
= Δ



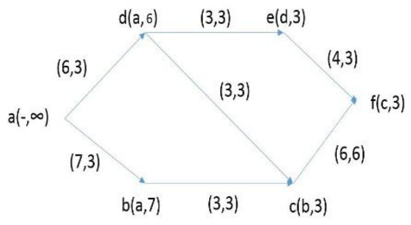
Path P(a, b, c, f)'s all edges are properly oriented; the flow is increased by Δ = 3 in all edges.



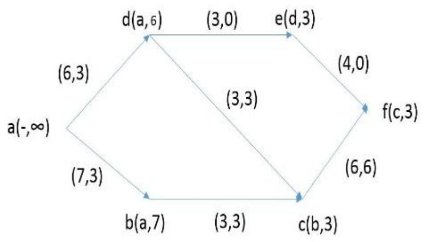
Now, we consider the second path which is P (a, d, c, f).Now consider vertex d,
Predecessor (d) = a and
Value (d) = min(∞, (6 – 0))
= 6
Similarly for vertex c
Predecessor (c) = d and
Value (c) = min(6, (3 – 0))
= 3
Predecessor (f) = c and
Value (f) = min(3, (6 - 3))
= 3
= Δ



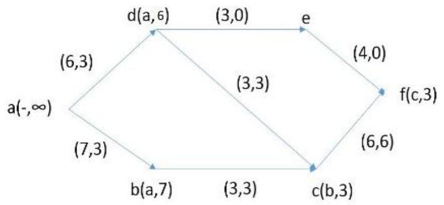
Path P(a, d, c, f)'s all edges are properly oriented; the flow is increased by Δ = 3 in all edges.



Now, we consider the third path which is
P (a, d, e, f).
Now consider vertex d,
Predecessor (d) = a and
Value (d) = min(∞, (6 – 0))
= 6
Similarly for vertex e
Predecessor (e) = d and
Value (e) = min(6, (3 – 0))
= 3
Predecessor (f) = e and
Value (f) = min(3 , (4 – 0))
= 3
= Δ



Since each edge in the path P(a, d, e, f) is properly oriented; the flow in each edge in P is increased by Δ = 3.



Now, since the flow is maximal; we consider the total flow of each path.
For path P (a, b, c, f):
Total flow = $F_{ab} + F_{bc} + F_{cf}$
= 3+3+6
= 12
Similarly, for path P (a, d, c, f):
Total flow = $F_{ad} + F_{dc} + F_{cf}$
= 6+3+6
= 15
For path P (a, d, e, f):
Total flow = $F_{ad} + F_{de} + F_{ef}$
= 6 + 3 + 3
= 12.

It is to be noted that in this example the optimal value 12 is attained for two different paths, that is for paths a, b, c, f and a, d, e, f.

Explanation

In order to collect accurate traffic data, we have to place sensors on the roads and streets to get accurate traffic flow details. And the data ingested could be used effectively to direct the traffic without adding new roads.

The sensors placed on each path will calculate the maximal flow of traffic via different paths available and the corresponding information will be displayed on each junction. Based on the information displayed at the junctions, the drivers will choose the optimal path that is with minimum cost.

The method enables us to know information about each optimal paths. This provides space for choosing either of them depending on the need of the individual. As in the above case, the two optimal path that could be taken are P (a, d, e, f) or P (a, b, c, f).The path P (a, d, c, f) should not be considered, as the maximal flow of this path is greater than that of the other two paths.

In emergency situations this method helps us to choose the optimal one, and when required for minimizing the cost, choice would be displayed at the traffic junctions. Moreover since this is displayed at every junction, we can change our choice and need not stick on to the one, that we had already picked. This provides us space to decide our choice timely and need not plan for the entire travel, particularly for long trips where you may have to change your travel plans.

NP Completeness of Maximum Flow Problem

In computational complexity theory, NP-complete is the complexity class which belongs to a class of decision problems whose solution can be verified in polynomial time. NP refers to Non deterministic Polynomial time. A decision problem L is said to be NP-complete if it belongs to the set of NP problems as well as NP-hard problems.

Formal definition of NP-complete Problem

#A decision problem L is NP-complete if

1. L is in NP, and
2. Every problem in NP is reducible to L in polynomial time.

Maximum Flow Problem is a NP-complete problem. The solution to such problem can be verified quickly but it is difficult to find its solution in the first place. Also the time required to solve such problem increases with the size of the problem.

As the Maximum Flow Problems belong to NP-complete set of problems, their running time is at most exponential. This is one of the drawbacks of this problem.

Limitation

The algorithm requires the computation of traffic on all the paths. Suppose, the optimal path in the previously mentioned example is found in the very first step. We still need compute the traffic through all the remaining paths. This requires extra time and computation which is not necessary; as the optimal path is already obtained.

Another factor that might limit the effectiveness of system is path's length. The system might not be able to compute the accurate cost if the path length is very large. Moreover, the larger path would mean more time for computation of the path cost.

How to overcome the Limitations

In order to overcome the limitation mentioned above, we will have to set a threshold value for all the paths. The traffic on any path will not go below this threshold value. Now, once the threshold value is obtained while calculating the traffic we stop our calculation at that point. The flow for the remaining paths need not to be calculated. The threshold value thus obtained will be considered as the optimal flow.

The upper bound of the path that the sensors would compute could be fixed at each junction that is, in order to avoid the problem of larger computation due to longer path, the sensors would only compute the cost/traffic up to a certain distance. The succeeding junctions would further display the cost of the oncoming paths.

Another solution to the computation of longer paths could be, with a particular junction computing the optimal costs up to its maximum capacity. If the length of the path to be computed is greater than the maximum length that the sensors could compute, then the calculated cost from further sensors could be relayed back to the preceding sensors. The current sensor could then add up the cost and display the summation at that given junction.

Application

The maximal flow obtained can be used to direct the traffic at the junctions. The sensors placed on different paths will capture the traffic flow and relay the flow information to the junctions. This will help the drivers in choosing the optimal path with minimum traffic.

Conclusion

The theorem stated above can find out the maximal flow of each path. The flow thus obtained can be used for selecting the optimal path and minimize the traffic congestion. This can be achieved by placing the sensors on different paths and relaying the corresponding information to the junctions of roads.

Future Work

The algorithm stated above could be modified according to the prevailing scenario and could eventually be implemented to effectively manage traffic congestion.

The problem of having larger paths for traffic computation could also be minimized with the application of improved sensors having greater efficiency and accuracy.

Overcoming the problem of longer paths with the sensors relaying the cost to the preceding sensors would require constant updating as the traffic would be changing forever. Thus, more sophisticated sensors would be required to perform this task.

In future we plan to modify and design an algorithm, which suits the proposed method.

References

Discrete Mathematics by Richard Johnsonbaugh, fifth edition, Prentice Hall Publisher.
NikyBaruah, A. K.Baruah, On A Traffic Control Problem Using Cut-Set *of Graph,*Int. J. Advanced Networking and Applications
http://en.wikipedia.org/wiki/NP-complete.
http://xlinux.nist.gov/dads/HTML/weightedGraph.html.
http://en.wikipedia.org/wiki/Glossary_of_graph_theory.
http://en.m.wikibooks.org/wiki/File:CPT-Graphs-directed-weighted-ex1.svg

Authors' Profile

M. Yamuna received her Ph. D in Mathematics from Alagappa University, Karaikudi, India. Currently she is working as Asst Prof (Sr) at VIT University Vellore. She is interested in graph theory.

Amber Raza 2014 B.Tech under-graduate in Computer Science and Technology from Vellore Institute of Technology, Vellore, India.

Kanav Anand 2014 B.Tech under-graduate in Computer Science and Technology from Vellore Institute of Technology, Vellore, India.

Sakthi Kumar R 2014 B.Tech under-graduate in Computer Science and Technology from Vellore Institute of Technology, Vellore, India.

Sangam Kumar 2014 B.Tech under-graduate in Computer Science and Technology from Vellore Institute of Technology, Vellore, India.