Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X (Online)**
Sch. J. Eng. Tech., 2016; 4(5):211-214 **ISSN 2347-9523 (Print)**
©Scholars Academic and Scientific Publisher
(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Research Article

Problem of Transport Network Analysis for Identifying Bottlenecks

A. T. Akhmediyarova[1*], I. T. Utepbergenov[1, 2], Sh. N. Sagindykova[2], J. R. Kuandykova[3]
[1]Institute of Information and Computing Technologies, Republic of Kazakhstan, 050010 Almaty city, Pushkin str., 125
[2]Almaty University of Power Engineering end Telecommunication, Republic of Kazakhstan, 050013 Almaty city, Baytursynov str., 126
[3]Turan University, Republic of Kazakhstan, 050010 Almaty city, Dostyk av., 110

***Corresponding author**
A. T. Akhmediyarova
Email: aat.78@mail.ru

Abstract: The analysis was done and an algorithm was developed for calculating a route in the urban transport network. Based on the input data, this algorithm determines the capability to pass the given flow between two points during the time, not exceeding the given one, by calculating the travel time by alternative routes. The algorithm is based on the determination of the shortest path in the directed multigraph. The calculation results are given for different routes, obtained using the developed program.
Keywords: The road network of the city, Model of road network, Flow data, Bandwidth, Crossroads, A directed multigraph.

INTRODUCTION

Terms of reference

It is required that flow M passes from point S to point T in a time not exceeding Time. If this cannot be done due to the traffic jams, to identify their locations and transmit them to the output. (speed of all vehicles is averaged).

The following data are entered to solve the problem:

1) The road network of the city - roads, crossroads (ordinary), crossroads with traffic lights, roundabouts.

2) The parameters of the road:

a) Pmax (i, j) R^+ - maximum capacity of the arc (i, j).

b) Preal (i, j) R^+ - real flow, averaged over Time.

a) zerotime (i, j) R + - the time of passing the arc, if there is not a single machine.

3) δ (0,1) - traffic light parameter (one for each crossroad) indicates how much time the light is on. (i.e. the other light is on during time 1-δ)

4) Roundabout parameters:

a) Circle_number Z^+ - the number of incoming (outgoing) roads from the roundabout.

b) Circlemax R^+ - the maximum capacity of the roundabout.

in) Circlereal.1, ..., Circlereal. Circle_number $\in R$ + -. real flows between the first and second outgoing roads, between the second and third one, ..., and between Circle_number and the first one, averaged over Time.

g) Circletime.1, ..., Circletime. Circle_number $\in$ R + - by analogy with zerotime (i, j) and Circlereal.

5) Flow data:

a) S - the departure point of the flow.

b) T - the point of receiving the flow.

a) M $\in Z^+$ - flow rate (how many vehicles is supplied to the point S).

g) Time $\in R^+$ - time which should not be exceed by the flow from S to T.

THEORETICAL MODEL

The model where the problem is considered. The oriented weighted multigraph is used as a model [1-4]. Each road is a graph arc, each crossroad without traffic lights is a vertex. Crossroad with traffic lights and a roundabout are more complex structures, which, however, are reduced to the first two (road, crossroad without traffic lights) as follows [5, 6].

Crossroad with traffic lights. Initially intersection without traffic lights is an ordinary vertex (except that parameter δ is assigned to it), which is converted into a complete graph K_4 as shown in Figure 1.

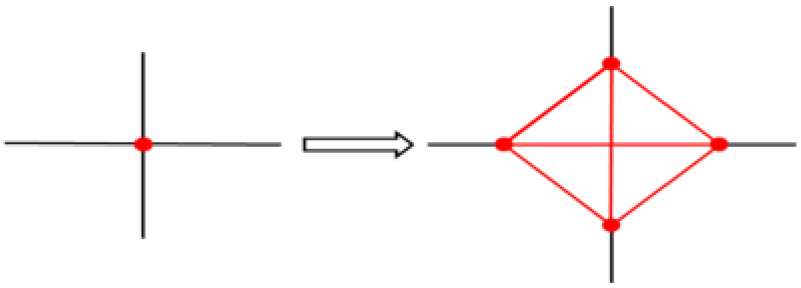

Fig-1: Converting the crossroad with traffic light

Capacity Pmax (i, j) of each of these six arcs is equal to the minimum capacity of the main incident arcs (not red) [5]. Since the traffic lights is on during time δ to one direction and (1-δ) to another, then, accordingly, actual capacities will be δ * Pmax and (1-δ) * Pmax (Figure 2).

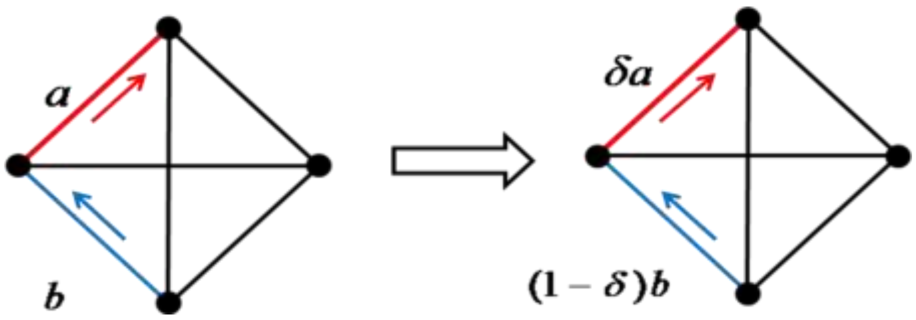

Fig-2: Explanation to the converting the crossroad with traffic light

The averaged flow Preal (i, j) of each arc is 0 (let's take that the vehicles pass the crossroad instantaneously). Accordingly, zerotime (i, j) from the previous statement is 0, as well.

Roundabout. Given a roundabout. Then it can be converted under the proposed model, as shown in Figure 3.

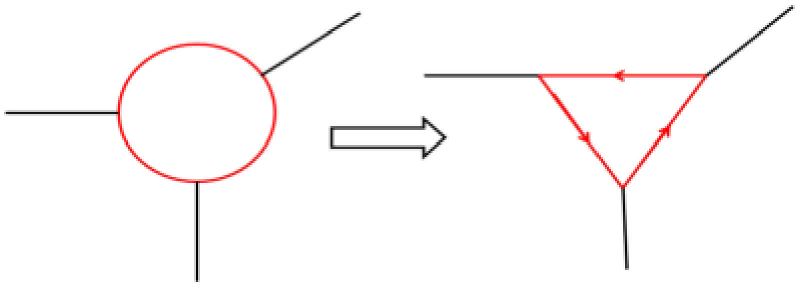

Fig-3: Converting the roundabout

The result is a structure made of simple vertices and arcs. Circlemax is taken as Pmax (i, j). As Preal (i, i + 1) and zerotime (i, i + 1), we take Circlereal.i and Circletime.i, where the path between i-th and i + 1st roads is considered.

In order to analyze the movement, we still need a special function to calculate the travel time of the flow unit along this arc [7, 8].

Function for calculating the travel time along the selected route. The proposed function for calculating the travel time along the given part of the route (arc) by the flow unit is presented below:

$$h(i,j) = h\,(Pmax\ (i, j), Preal\ (i, j), zerotime\ (i, j)) \in R^{+}. \text{---}(1)$$

At the input it receives the capacity of the arc, the averaged flow and time that would be required by unit of flow to pass along this arc if it is empty (there would be no other vehicles). This function is an empirical question, and can be defined in many ways. The following formula was used for implementation of the algorithm [9, 10]:

$$h(i,j) = \left(1 + \frac{Preal(i,j)}{Pmax(i,j)}\right) * zerotime(i,j) \quad \text{------------} (2)$$

1.1. DEVELOPMENT ALGORITHMS AND PROGRAMS

The proposed algorithm will test by input data the capability of points (from vertex S to vertex T) to transmit the flow for time not exceeding given one. At output, the algorithm will either state that the flow passes, or, if it is not possible, will identify paths (Memory.cost) and specific intersections (list Trafficway, structurally similar to Memory.path), where there are traffic jams that impede the given stream to pass successfully for a given time.

Algorithm TrafficWay

Step 1. To use the algorithm Mflow, where:

a) vertices S and T present the starting and ending points

b) Pmax (i, j) presents c_{ij},

c) h (Pmax (i, j), Preal (i, j), zerotime (i, j)) is taken as cost (i, j),

g) Time is taken as M.

Step 2. To transform the list obtained in the previous step in:

Memory.cost = round (Time - Memory.cost) + 1. Then we obtain F as the sum of all Memory.cost.

Step 3. To compare maximum flow F obtained in the second step with M:

a) If M <= F, then at output the algorithm terminates with the conclusion "Success."

b) if M> F, then go to step 4.

Step 4. To take the first path from the list Memory.path and start to go on it.

a) take the first arc - go to step b).

b) compare Preal (i, j) of the current arc with the sum Pmax (j, k) for all outgoing arcs from the vertex, which includes the current one. If this sum is more than Preal (i, j) (in Fig. 4. a <b + c + d), then go to the next arc, if there is one, and repeat step b). If there is not, go to step g). If the sum is less than Preal (i, j), then go to step c).

c) if the sum was less, the number of the vertex, where the current arc enters, will be included to the list TrafficWay (into the line with number analogic to the record number in the list Memory.path).

d) if the next arc in the path does not exist, go to step 5.

Step 5. To take the following path and go to step 4a. If such a path does not exist, then go to step 6.

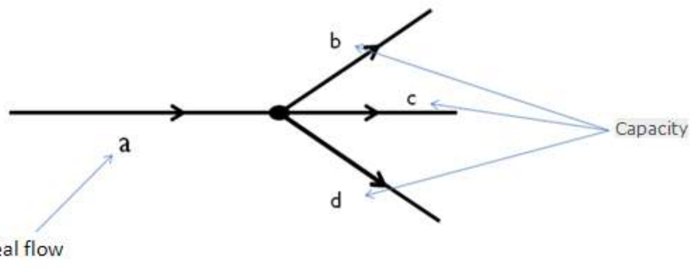

Fig-4: Explanation to step 4b

Step 6. If Traffic Way is empty, then the algorithm stops with the statement "Invalid input." If Traffic way contains records, then the algorithm terminates as well and the output provides two lists: Memory path, which keeps the path and Trafficway - which keeps the traffic jams.

These algorithms were developed in C ++. Input data is taken from a text file. The program analyzes the transport network on the passage of flow through it and looking for traffic jams and offers an optimum route option. If the flow can not pass, then to obtain a result that reflects the real state of the roads in solving related problems, this module should be used in an iterative conjunction with other (elimination of traffic jams, construction of interchanges - in this case it is necessary to the module, building interchanges before feed module proposed in this paper, has transformed the denouement in the form of who understands the model - the usual arc, the top and weights on them).

PROOF OF CORRECTNESS OF THE ALGORITHM

Statement: Traffic Way algorithm is correct, its time complexity is O ((n2 + m2) mC), where n - the number of vertices (V, E), m - the number of arcs, C = max {Pmaxij}.

Proof: This algorithm works as follows. First, he looks for a flow, which can be run from point S to point T for a time not exceeding Time (as F is the maximum flow per unit of time, then, to see how many vehicles can pass from S to T during the Time, it is necessary to take Time minus time spent at the elementary path plus one vehicle, which managed to get to the T at the last moment), and, after finding it, to start testing. If the input flow does not exceed the maximum, then we assume that it can safely pass from S to T during Time. If it exceeds, then we assume that there are jams on the path of our flow, and starting with the fourth step, we begin to look for them. To do this, take in sequence the paths from the list Memory path, which was formed even at implementation of the Mflow algorithm and for each of these paths do the following actions: Let's go over it and make sure that the actual (average) flow entering the vertex could pass fully through, i.e. capacities of all outgoing arcs allow it to pass. If, after passing all the paths we cannot find traffic jams – Traffic Way is empty, and then it is likely that input data (flow M or time Time) were set incorrectly.

If some jams are found, then we record via Traffic Way, in which path and in what node this jam was. After that, transmit these wo lists to the output.

Let's compute the time complexity of the given algorithm in steps:

Implementation of algorithm Mflow gives the time complexity O ((n^2+m)mC). Change of Memory. cost and further summing do not take more than O(mC) operations. Later, when we start to check all paths (not more than mC) and at each path we compare each arc (not more than m) with sum of outgoing ones (not more than m), then we obtain O(m^3C). as a result we have $O(n^2mC+m^2C+mC+m^3C) = O((n^2m+m^3+m+m^2)C) = O((n^2+m^2)mC)$.

The statement is proved.

CONCLUSION

(1) In result of the performed work the following has been developed:
—The algorithm for determining the shortest path in the oriented multigraph.
—Mflow algorithm for finding the maximum flow, the value of which does not exceed the specified one.
—TrafficWay algorithm for transport network analysis and identification of bottlenecks in it.

(2) To prove the correctness of the algorithm TrafficWay

(3) These algorithms were developed in C ++.

(4) Since only TrafficWay algorithm analyzes the transport network on the passage of flow through it and looking for traffic jams. If the flow can not pass, then to obtain a result that reflects the real state of the roads in solving related problems, this module should be used in conjunction with other iterative.

REFERENCES

1. Beckman J, McGuire CB, Winsten CB; Studies in the economics of transportation. CT: Yale University Press, 1956.
2. Chudak F, Dos Santos Eleuterio V; The traffic equilibrium problem, 2006.
3. Sukach EI; The use of simulation to study the dynamics of traffic flows in the region. Proceedings of the Gomel Fr. Skaryna State University, 2006; 4(37):96-99.
4. Cormen TH; Algorithms: construction and analysis. - 2nd ed. - M Williams, 2006; 1296.
5. Popkov VK; Mathematical models of connectivity. Novosibirsk. ICMMG SB RAS, 2006; 409.
6. Frank G, Frisch J; Networks, communications and flows. M. Svyaz, 1978; 448.
7. Zhogal SI, Maksimey IV; The objectives and models of operations research. Tutorial. – Gomel, BelSUT, 1999. 4.1: Analytical models of operations research. – p.109.
8. Zaichenko Yu P; Operations research: Tutorial. Kiev: Publishing House Slovo, 2002; 320
9. Polyakov Yu K; The theory of automatic control. St. Petersburg, 2008; 4-20.
10. Sobol IM; The Monte Carlo method. Moscow: Nauka, 1968; 64.