Vladimir Pribyl *

SOLUTION OF THE BUS ROUTE DESIGN PROBLEM

The paper deals with a single bus route design problem. It consists of two stages. The first one is to choose the set of stops fulfilling a defined constraint. The second one is the precising of the order of the stops on the route. Both exact and heuristic methods are proposed and verified on 9 randomly generated networks. Very high computational complexity of the exact method and some ways how to reduce it are discussed in the paper. Comparison of the experimental results is presented in the final table.

Keywords: bus, design, network, optimization, route, method, heuristics

1. Introduction

The problem we deal with belongs to the "family" of network reduction problems. The common feature of all of them is that a subgraph fulfilling some constraint has to be found for the given graph. The oldest member of the family is the minimum spanning tree problem, reducing only the edge set and keeping the vertex set unchanged.

A. Cerna et al [2] studied the shortest subgraph not lengthening any important trip more than the given percentage.

A. Cerna [5] introduced the problem of a bus route design in an area with a weak passenger demand. She outlined a basic idea of the exact method and of possible heuristics.

The main purpose of this paper is to present computational experience with exact and heuristic methods applied to randomly generated networks.

The bus route design problem in our formulation is different from the ones described in "classic" paper [6] or the "modern" [1]. E.g. in [1] the basic type of a bus route is "the shortest path between large demand vertex pairs". In [6] it is supposed that there exists a "wide" set of possible routes (constructed, e. g., manually), and the goal is to choose the optimal subset. We shall not use an "absolute" limit of accessibility, since in [3] and [4] it is shown that such an approach leads to extremely expensive or to user unfriendly solutions. We prefer a measure of a "mean" accessibility expressed in the constraint (1). Of course, the used methods can be modified for other accessibility measures.

2. Problem

Let $G = (V, E, q, d)$ be a (non-oriented) graph with the demand function q: $V \to \langle 0; \infty)$ and the length d: $E \to (0; \infty)$. Let $d(u, v)$ be the distance of u, $v \in V$ obtained by extension of the length of the edges. Let $\delta(S)$ be the length of the shortest path connecting the vertices of S on G for each $S \subset V$. Let $\lambda \in (0; \infty)$ and $q = \sum_{v \in V} q(v)$.

The problem is to find $S_{opt} \subset V$ such that

(1) $\mu(S_{opt}) = \frac{1}{q} \sum_{v \in V} q(v) d(v, S_{opt}) \leq \lambda$

(2) $\delta(S_{opt}) \to \min$

Appended problem

If there were several solutions of the problem 2, then the problem would be to take the set S_{opt} with a minimal number $|S_{opt}|$ of elements in it.

Note

The expression $\mu(S_{opt})$ in (1) represents the mean walking distance of passengers to the nearest stops.

The appended problem posses an alternative formulation: In the case of several solutions of the problem 2 first to take the one with the smallest $\mu(S_{opt})$ and only afterwards the one with minimal $|S_{opt}|$. Such a change would need small modification of the exact method of solution whereas the heuristics can remain unchanged.

3. Methods of Solution

Since the problem contains the open travelling salesman problem (OTSP) as a subproblem, it is obviously NP-hard. Therefore, we have to propose some heuristic method for its solution in addition to the exact method we shall start with. It is of the "Depth-First-Search" type.

* **Vladimir Pribyl,**
Faculty of Management, University of Economics, Prague, Czech Republic, E-mail: pribyl@fm.vse.cz

3.1 Exact Method (EM)

Initial step: We find first S fulfilling (1). Then we find a first record $\delta(S)$ solving OTSP.

Recursive step: Once a set S fulfilling (1) is found, each its extension is omitted since it cannot shorten the length $\delta(S)$. Then we look for the next S fulfilling (1) in the adjacent branch of solution structure.

3.2 Neighbor Greedy Heuristics (NGH)

We shall use these denotations:

$N(S') = \{w \in V: w \notin S', \exists\, s \in S', (w, s) \in E\}$ for the neighborhood of S' in G and

$m = m(G)$ for the median of G i.e. such $m \in V$ that $\sum_{v \in V} q(v)d(v,m) \longrightarrow \min$

Initial step: We put $S = S' = \{m\}$

Recursive step: If S fulfills (1), then we consider S the solution. If it does not then we look for such a $s' \in (V - S) \cap N(S')$ that

$$\sum_{v \in V} q(v)d(v, S \cup \{s'\}) = \min_{s \in (V-S) \cap (S')} \left\{ \sum_{v \in V} q(v)d(v, S \cup \{s'\}) \right\}$$

If such a vertex does not exist the heuristics is not able to solve the problem and we stop. If it does then we put $S = S \cup \{s'\}$ and further if $S' = \{m\}$, then we put $S' = \{m, s'\}$, if not (and thus the set S' contains at least two elements), then $S' = S' - \{s''\} \cup \{s'\}$ where $s'' \in S'$ and $d(s', s'') = \min_{s \in S'} \{d(s', s)\}$. After doing that we return to the recursive step.

4. Implementation of the methods and computational experience

Both above mentioned methods were implemented. The environment of the Visual Basic and database system Microsoft Access was used, even though it is highly probable that there are quicker environments. These are the reasons:

- built-in effective and easy to use algorithms and methods for importing, sorting, searching, exporting and presenting the data
- availability
- Visual Basic is a built-in programming environment of the MS Access and it is very simple to work with the Access objects in it

The both methods were tested on 9 randomly generated small networks. The diameter of each network is about 30 km and the number of vertices is 20. All the implemented algorithms were tested on the same hardware configuration.

4.1 Exact Method

The Exact Method consists of two basic stages. Each stage was implemented separately in order to gain possibility to work with intermediate data. It is very important for finding ways how to reduce the time consumption of the exact method.

The first stage of the method is the algorithm which creates the set of all subsets $S \subset V$ fulfilling the constraint (1). The number of all possible combinations of vertices is 2^n, where n is the number of vertices of graph G. Even for our small test networks it is more than 1 million combinations. The algorithm is of the "Depth-First-Search" type as mentioned above. Since it implements the principle described in 3.1, it is not necessary to test the constraint (1) for all the possible combinations of vertices. For our networks with 20 vertices, the number of tested combinations is about 300 thousands and finally the number of all subsets $S \subset V$ fulfilling the constraint (1) is about 150 thousands. It varies in accordance with the network topology and the value of the limit λ of course. The time consumption of this algorithm was several minutes for all our tested networks. But it is necessary to see that the computational complexity of this algorithm is $O(2^n)$ and, therefore, it can take tens of hours to finish this stage of the EM for the network with, say, 30 vertices.

The second stage of the method is the algorithm for solving the OTSP for each subset S found in the previous stage of the method. It finds the set S_{opt} fulfilling the constraint (2). The exact method for solving the OTSP on the set S is going through all the permutations of vertices in the set S in order to find $\delta(S)$. The computational complexity of this algorithm is $O(n!)$, where n is the number of vertices in S. The computational time of this algorithm is about 3 seconds for $n = 10$. However for one of our test networks (having set $\lambda = 4$) there are about 5000 subsets S fulfilling the constraint (1) with the $n \in \{11;12;13\}$. It means that solving the OTSP in this way can takes about 86 hours only for these 5000 subsets. The main goal of the EM is to obtain the optimal solutions of our problem. These are the platforms for testing and improving proposed heuristics. Computational times about 100 hours are unacceptable for this purpose. Therefore some ways to shorten the computational time were examined. These are the analyzed and tested ways:

- **Reducing the number of subsets *S*** fulfilling the constraint (1). The principle is to find such subset S fulfilling (1) that there exists another subset S' fulfilling (1) such that $S' \subset S$. Considering the principle mentioned in 3.1 it is possible to omit such subsets S. The experimental computations carried out on the test networks made out that it is possible to reduce the number of subsets S approximately two times. Unfortunately, this method is relatively time consuming. The computational time was several hours for our test networks. This is the reason why this method is not finally used in the EM.
- **Speeding up the algorithm for solving OTSP.** The algorithm for generating the permutations of vertices in the subset S was implemented as a recursive one. It means that this algorithm selects the vertex for a certain position and then it calls the same algorithm recursively to select the vertex for the next position. It is possible to calculate the length of the part of permutation before a recursive call. If this length is greater than the length of the best solution of the OTSP achieved till this time for all the tested subsets S then all the permutations starting with the same

sequence of vertices are skipped. In addition, the algorithm starts with the subsets S with the minimal number of vertices. This principle significantly reduces the computational time of the second stage of the EM and makes EM the suitable method for small networks with the maximum number of vertices about 25.

4.2 Heuristic Method

The heuristic method was implemented as described in 3.2. The initial step of the algorithm is very important. The described NGH starts with initial step $S = \{m\}$, because we assume $m \in S_{opt}$. However, this hypothesis does not have to be fulfilled in all cases. It is clear from the experimental results obtained on test networks, which are shown in Tab. 1. If $m \notin S_{opt}$, then the result of NGH cannot be an optimal solution. Tab. 1 shows that better results of NGH were achieved in the cases when $m \in S_{opt}$. However, the quality of solution obtained by NGH does not depend only on the initial step, improving the strategy of the starting vertex selection is one possible way how to improve the proposed heuristic method.

4.3 Experimental results

Table 1 summarizes the results obtained by using described methods on test networks. The distance limit λ was set to 4 (less than 1/7 of the diameter of the network). Having these parameters given, the computational time depends on the network topology. For the Exact Method it varies between 50 and 120 minutes. For the heuristic method it is less than 1 second. The optimal solutions obtained by the Exact Method (EM) are emphasized by the bold font.

The results of the described methods Tab. 1

Net No.	Method	Route length	NGH/EM route length ratio	The real distance mean value (limit was set to 4)	$m \in S_{opt}$
1	NGH	41.52	1.00	3.8379	YES
	EM	41.52		3.8379	
2	NGH	52.52	1.21	3.2148	YES
	EM	43.44		3.9476	
3	NGH	74.84	1.70	3.3527	NO
	EM	44.14		3.9500	
4	NGH	48.57	1.13	3.3288	YES
	EM	42.82		3.7987	
5	NGH	53.02	1.32	3.2731	NO
	EM	40.31		3.9646	
6	NGH	55.99	1.34	3.2126	YES
	EM	41.66		3.8744	
7	NGH	36.16	1.09	3.7522	YES
	EM	33.14		3.9798	
8	NGH	51.58	1.23	3.8771	YES
	EM	41.91		3.9822	
9	NGH	45.98	1.02	3.7727	YES
	EM	45.22		3.9604	

Fig. 1 and Fig. 2 depict the resulting routes for the selected test networks. The numbers in brackets are numbers of the passengers randomly generated between 0 and 100. Fig. 1 depicts the resulting routes in test network number 3. I have selected this example, because in this case the NGH gives the worst result.

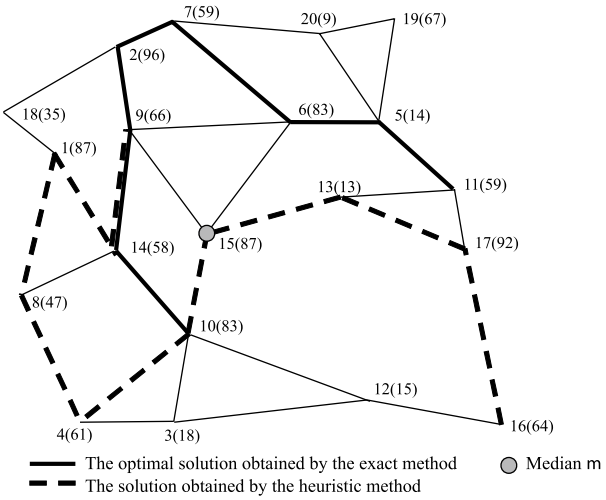

Fig. 1 The resulting routes in the network No. 3

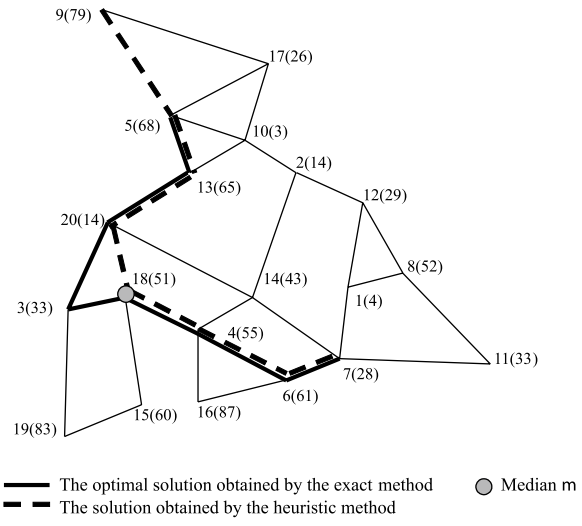

Fig. 2 The resulting routes in the network No. 7

5. Conclusion

Two methods for solving the problem described in 2 were discussed in this paper. The first one was the EM. Computational complexity of this method is very high. It is suitable only for a small network. But the optimal solutions achieved by this method are very important, especially for testing the heuristic method.

The heuristic method called "Neighbor Greedy Heuristics" was proposed too. Experimental computations carried out on the test networks made out that NGH is a very quick method. But it turned out that this method does not give good results in some cases. The main goal for my further work is to find ways for improving the heuristics in order to gain the solutions closer to optimal ones.

References

[1] CIPRIANI, E., FUSCO, G., GORI, S., PETRELLI, M.: *A procedure for the solution of the urban bus network design problem with elastic demand,* E-Proc. of 10th EWGT Meeting and 16th Mini-EURO Conference, Poznan, Poland, 2005.

[2] CERNA, A., CERNY, J., PESKO, S:, CZIMERMAN, P.,: *Network Reduction Problems, Journal of Information,* Control and Management Systems, Vol. 5, No 2/1,2007, pp. 139–145, ISSN 1336-1716

[3] CERNA, A.: *Intensification MHD and her Potential conflict with attendance limit (in Slovak)*, Proc. 6. medzinarodnej konferencie o verejnej osobnej doprave, Bratislava, 2003, pp. 43–47, ISBN 80-233-0485-2

[4] CERNA, A.: *Note to Transit Stops Accessibility Constraint (in Slovak),* Horizonty dopravy 2/2003, VUD Zilina, pp. 23–24

[5] CERNA, A.: *Bus Route Design in a Weak Demand Area,* Presented on Czech-Slovak Seminar on Transport Optimization, Jindrichuv Hradec, 2007, can be requested from cerna@fm.vse.cz.

[6] ERLANDER, S., SCHEELE, S.: *A Mathematical Programming Model for Bus Traffic in a Network.* Transportation and Traffic Theory (ed. Buckley), REED, Sydney, 1974, pp. 581–605.