Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 107–120

DOI: 10.1515/ausi-2015-0014

Empirical study of the greedy heuristic as applied to the link selection problem

Pál PUSZTAI
Széchenyi István University
email: pusztai@sze.hu

Tamás HAJBA
Széchenyi István University
email: hajbat@sze.hu

Abstract. Behind the link selection problem there is a practical problem that aims to check efficiently the vehicles on a road network. The checking process is to be realized with license plate reading cameras for checking the valid vignette of vehicles using that part of the network. However this problem should be defined generally and the methods of obtaining a solution can be applied to a wider range of problems independent of the original problem. This paper defines the link selection problem with directed graph, it shows the NP-hard complexity and it gives a heuristic and binary integer programming models to solve the problem. These two kinds of approaches allow us to examine and qualify the heuristic. The computational results of the methods are compared with different sizes of problems.

1 Introduction

The problem of link selection as an effective traffic check was introduced in [6]. The input data of a real life situation were produced with a traffic assignment model [5], and the problem was solved with an algorithm that is based on the greedy heuristic of set cover [1, 2, 3]. It was focused on the efficiency of the

Computing Classification System 1998: G.2.3
Mathematics Subject Classification 2010: 90B20, 90C10, 90C59
Key words and phrases: link selection, greedy algorithm, binary integer programming, traffic check

monitoring when the checking process has no influence on the flow of traffic, i.e. it does not stop nor slow down the vehicles. Present work is also related to this case, the loads of the network are given.

2 The link selection problem

The link selection problem has got two similar but slightly different optimization tasks. We define the problem and its heuristic more generally than it was described in [6].

Let us suppose that G is a directed graph and P is a finite, nonempty set that contains acyclic paths in G. Every path has got a positive integer number called weight.

Task 1: For a given ratio $r \in (0, 1]$ let us select minimal number of edges from G so that $x/y \geq r$ satisfied, where x is the sum of the weights of the paths that contain at least one selected edge, and y is the sum of the weights of all paths.

Task 2: For a given integer $k > 0$ let us select k edges from G so that the sum of the weights of the paths that contain at least one selected edge is maximum.

The first task is a special set cover problem in case of $r = 1$. The speciality comes from the data. Let us consider the weights of the paths as the same number of vehicles that are using the related paths. All vehicles are considered as the set to cover and the vehicles that use an edge are considered as a subset. The second task is a max k-cover problem with the same subsets of vehicles. The set cover problem and the max k-cover problem are NP-hard [2].

The link selection problem is a special case of them, where the weights of the paths of P correspond to the set to cover, the edges correspond to the subsets, and the paths make a kind of relationship between the subsets.

3 The complexity of the link selection problem

It is shown that the 3-SAT problem can be reduced to the Task 1 and Task 2 problem as well, thus the link selection problem is NP-hard. For an arbitrary 3-SAT problem we give a proper Task 1 and Task 2 link selection problem such that the solution of the 3-SAT problem can be obtained from the solution of the link selection problem.

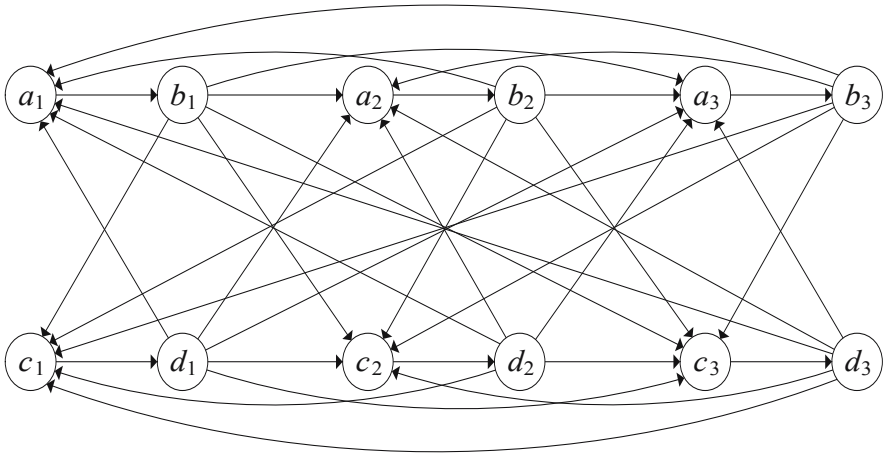

Figure 1: The G graph constructed for a three variable 3-SAT problem

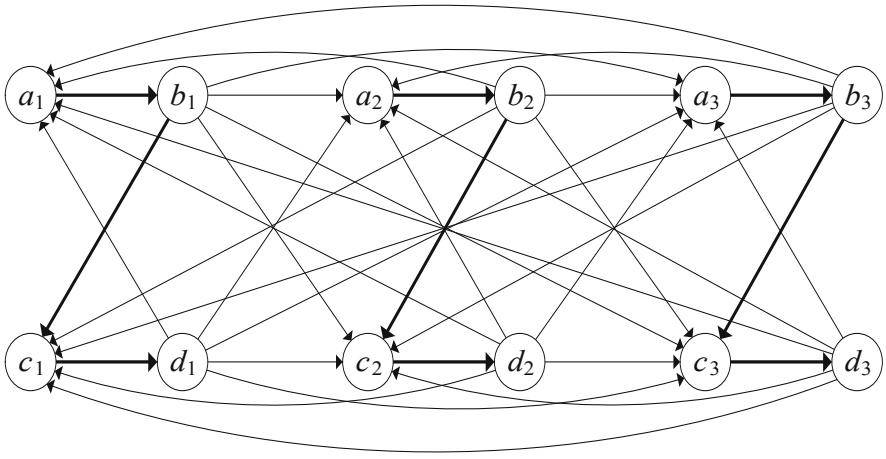

Figure 2: The 3 edge paths related to the three variables

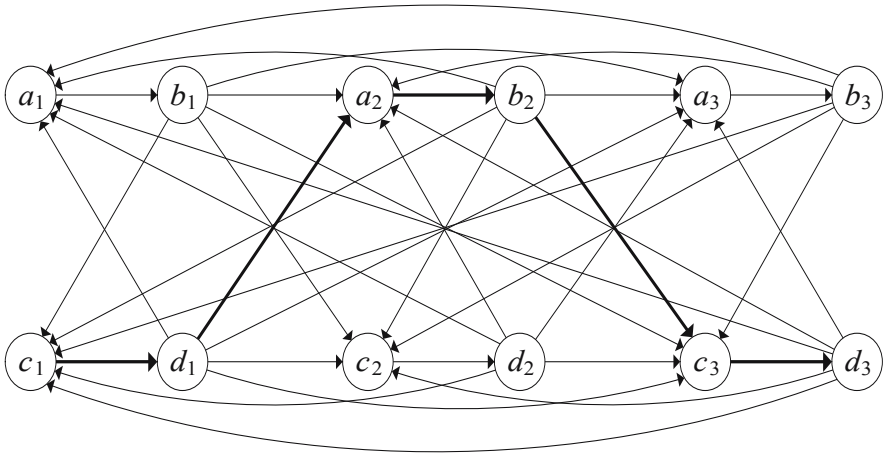

Figure 3: The 5 edge path related to the $\overline{x_1} \vee x_2 \vee \overline{x_3}$ clause

Construction: Let us suppose that there is a given 3-SAT problem with n variables and m clauses. Let $x_1, x_2, \cdots, x_n$ be the variables of the 3-SAT problem. Construct the $G = (V, E)$ directed graph as follows: for every x_i variable add 4 vertices a_i, b_i, c_i, d_i into V (so V will contain $4n$ vertices); for every i ($i = 1, \cdots, n$) add an edge from a_i to b_i (this edge corresponds to x_i) and add an edge from c_i to d_i (this corresponds to $\overline{x_i}$) into E; in addition, for every i, j, $i \neq j$ add (b_i, a_j) and (d_i, c_j) edges into E, and for every (i, j) add (b_i, c_j) and (d_i, a_j) edges as well. Note that in this G graph there is exactly one edge from every a_i and c_i vertices (see Fig. 1).

Now we make P, a set of directed paths in G. For every x_i variable add the $a_i - b_i - c_i - d_i$ 3 edges path into P (see Fig. 2). In addition, for every clause of the 3-SAT problem relate a 5 edges path as follows: let $(X \vee Y \vee Z)$ be an arbitrary clause of the 3-SAT problem; add into P the path in which the 1st edge corresponds to X, the 3rd corresponds to Y, the 5th corresponds to Z, and the 2nd and 4th ones are the edges between the proper vertices. For example for the $\overline{x_1} \vee x_2 \vee \overline{x_3}$ clause the $c_1 - d_1 - a_2 - b_2 - c_3 - d_3$ path is related (see Fig. 3). After this P will contain $n + m$ directed paths.

Lemma 1 *Let it be given an arbitrary 3-SAT problem and let* G *and* P *be the graph and the set of paths constructed before. The 3-SAT problem is satisfiable if and only if there exists a set of edges* $C \subseteq E, |C| = n$ *such that every path of* P *contains at least one edge from* C *(shortly* C *covers* P*).*

Proof. $\Rightarrow$ If the 3-SAT problem is satisfiable then we give a set of edges $C \subseteq E, |C| = n$ that covers P. If an x_i variable of the 3-SAT problem is true then add the (a_i, b_i) edge into C, otherwise (if x_i is false) add the (c_i, d_i) edge into C (thus $|C| = n$ is satisfied). Let us notice that C contains exactly one edge from every 3 edge paths and at least one edge from every 5 edge paths (as every clause has got at least one true literal and the edge related to this literal is in C), thus C covers P.

$\Leftarrow$ Let us suppose that there exists $C \subseteq E, |C| = n$ that covers P. As P has got all 3 edges $a_i - b_i - c_i - d_i$ $(i = 1, \cdots, n)$ paths and these paths are edge disjunct (they have no common edges), so every 3 edges path has got exactly one edge in C. On the other hand all paths of P start from an a_i or c_i vertex, furthermore only one edge goes to every b_i vertex, so if there is a (b_i, c_i) edge in C then replacing it with the (a_i, b_i) edge we get such a set of n edges that still covers P. Therefore it can be supposed that every edge in C is (a_i, b_i) or (c_i, d_i) type, and for every i there is exactly one edge from these (a_i, b_i) and (c_i, d_i) edges in C. Let x_i be true if the (a_i, b_i) edge is in C, otherwise (if the (c_i, d_i) edge is in C) let it be false. For these variables the 3-SAT problem will be satisfiable because every path of the 5 edge paths of the clauses contains at least one edge from C (due to the assumption) that is (a_i, b_i) or (c_i, d_i) type edge, thus the literal related to this edge (and the clause itself) will be true. □

Theorem 2 *Task 1 of the link selection problem is NP-hard.*

Proof. Let it be given an arbitrary 3-SAT problem. Let G and P be the graph and the set of paths constructed before, let the weights of all paths equal to 1, and let $r = 1$ (i.e. we want to cover all paths of P). Due to the lemma if the solution of this link selection problem contains n edges then the 3-SAT problem is satisfiable, otherwise it is not. □

Theorem 3 *Task 2 of the link selection problem is NP-hard.*

Proof. Let it be given an arbitrary 3-SAT problem. Let G and P be the graph and the set of paths constructed before, let the weights of all paths equal to 1, and let $k = n$ (i.e. we want to cover with n edges as much as possible paths of P). Due to the lemma if the solution of this link selection problem covers all paths of P then the 3-SAT problem is satisfiable, otherwise it is not. □

4 The greedy heuristic

A greedy algorithm can be used to solve the link selection problem.

GREEDY($G, P, task, r, k$)

1. $L \leftarrow \{\}$
2. **if** $task = 1$
3. $\quad y \leftarrow$ sum up the weights of paths of P
4. **repeat**
5. $\quad$ For every edge e of G sum up the weights of paths of P that contain e
6. $\quad$ Select the edge e that has got the maximum weight
7. $\quad L \leftarrow L \cup \{e\}$
8. $\quad P \leftarrow P \setminus \{$paths that contain edge $e\}$
9. $\quad$ /* Stopping criteria */
10. $\quad$ **if** $task = 1$ /* reaching r ratio */
11. $\quad\quad u \leftarrow$ sum up the weights of paths of P
12. $\quad\quad x \leftarrow y - u$
13. $\quad\quad stop \leftarrow x/y \geq r$
14. $\quad$ **else** /* selecting k number of edges */
15. $\quad\quad stop \leftarrow |L| = k$
16. **until** $stop$
17. **return** L

Describing the complexity of the algorithm let $G = (V, E)$ graph be given and $n = |V|$. Let us suppose that $|P| = O(n^2)$. In this case we can handle all the paths between all different origin-destination pairs. If G represents a road network, the degree of vertices is limited with a small constant, thus $|E| = O(n)$. The lengths of the (acyclic) paths of P are $O(n)$. The algorithm repeats a greedy selection (line 6) until the stopping criteria becomes true. This iteration (line 4–16) runs $O(n)$ times. The most complex step of the iteration is in line 5. Based on the previous assumptions this step can be done in $O(n^3)$ time, thus we get $O(n^4)$ complexity in both cases ($task = 1, task = 2$).

Unfortunately this polynomial time algorithm does not guarantee the optimal solution. The greedy algorithm is an $\mathcal{H}(d)$-approximation algorithm for the set cover problem, where $\mathcal{H}(d) = \sum_{i=1}^{d} \frac{1}{i}$, and d is the size of the largest subset [3, 4]. It means that the solution of the greedy algorithm (the number of the selected subsets) is at most $\mathcal{H}(d)$ times larger than the optimum (the number of subsets selected by the optimal solution).

The ratio of the greedy algorithm is $1-(1-\frac{1}{k})^k \geq 1-\frac{1}{e} \approx 0.632$ for the max k-cover problem [2], which means that the number of elements covered by the greedy algorithm divided by the number of elements covered by the optimal solution is at least 0.632.

These general approximation ratios are valid for our algorithm too, namely it is $\mathcal{H}(d)$ approximation algorithm for Task 1 with $r = 1$, and it is 0.632 approximation algorithm for Task 2.

Because of the speciality of the link selection problem, the greedy algorithm gives much better solution than it is guaranteed by these ratios. In section 6 it will be shown that the greedy heuristic produces close to optimal solution for the link selection problem.

5 Binary integer programming models

The link selection problem can be solved with binary integer programming models too.

List of symbols

Parameters

n	number of the edges of G
m	number of the paths of P
w_j	weight of the path j $(j = 1, 2, \ldots, m)$
n_j	number of the edges of path j $(j = 1, 2, \ldots, m)$
j_l	the index of the lth edge of path j $(j = 1, 2, \ldots, m; l = 1, 2, \ldots, n_j)$
r	the required checking rate for Task 1 $(r \in (0, 1])$
k	the number of required edges for Task 2 $(k > 0)$

Binary variables

x_i	$= 1$, if edge i is selected (0, otherwise) $(i = 1, 2, \ldots, n)$
y_j	$= 1$, if at least one edge of path j is selected (0, otherwise) $(j = 1, 2, \ldots, m)$

The following models can be given according to the tasks. Both models contain $n + m$ binary variables and $m + 1$ equations. Equations (1) and (4) express that a path will be selected if an edge of that path is selected. Equation (2) ensures that it reaches the required monitoring rate. Equation (5) ensures that it selects the given number of edges.

Model 1: checking with a given ratio

$$y_j \leq \sum_{l=1}^{n_j} x_{j_l} \qquad (j = 1, \ldots, m) \tag{1}$$

$$r \cdot \sum_{j=1}^{m} w_j \leq \sum_{j=1}^{m} w_j y_j \tag{2}$$

$$z = \sum_{i=1}^{n} x_i \longrightarrow \min \tag{3}$$

Model 2: checking with a given number of edges

$$y_j \leq \sum_{l=1}^{n_j} x_{j_l} \qquad (j = 1, \ldots, m) \tag{4}$$

$$\sum_{i=1}^{n} x_i \leq k \tag{5}$$

$$z = \sum_{j=1}^{m} w_j y_j \longrightarrow \max \tag{6}$$

6 Computational outcomes

To compare the solution of our algorithm with the optimal solution we used three test networks with 10, 20 and 80 junctions (vertices) (see Fig. 4, where the thickness of the links (e-dges) corresponds to their total weights). The greedy heuristic and Model 2 were compared for all possible values of k.

The results are shown in the figures, where the values of the X axes are the number of selected links. The checked rates of the optimal solution are shown in Fig. 6, 8, 10 corresponding to the test networks. The checked rates are given in percentage of all traffic (the sum of the weights of all paths). The differences between the checked rate of the approximate and the optimal solution are given in percentage too and shown in Fig. 5, 7, 9.

We used all shortest paths between all different junctions of the networks. The weights of the paths were generated in two different ways. In the first case the weights were the demands of travel and in the second case the weights were calculated by the demands of travel multiplied by the lengths of the paths (that resulted larger weights and larger differences between them). In the first case the weights give the *traffic* (see Traffic data in the figures) and in the second

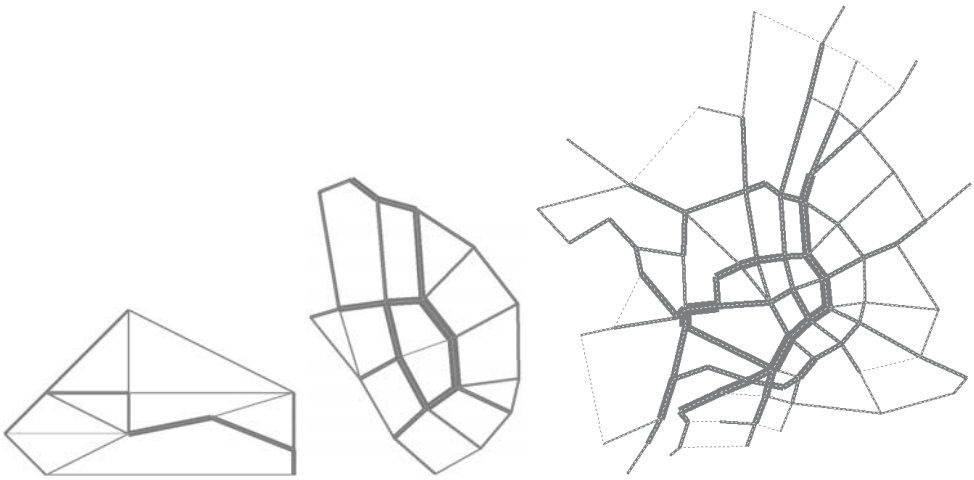

Figure 4: Test networks (10, 20 and 80 vertices) with loaded links

case the weights give the *traffic performance* (see Traf. Perf. data in figures). With using traffic performance we prefer to check those vehicles that travel longer distances on the networks (as it has been done in [6]).

In our tests the demands of travel were a random integer number in [1, 10]. The largest test network models a small part of the downtown of Budapest. It contains 80 junctions (vertices), 244 links (edges), 6320 (80*79) paths, i.e. 6564 binary variables. We used GAMS software and CPLEX solver on an average PC to solve Model 2. On our largest test network the solver required about 2 hours to compute the optimal solution for all 244 values of k, while the heuristic ran only 1 second.

	10 junctions		20 junctions		80 junctions	
Difference	Traffic	Traf.Perf.	Traffic	Traf.Perf.	Traffic	Traf.Perf.
Maximum	1.879	1.442	2.477	2.247	1.819	2.394
Average	1.037	0.671	0.587	0.596	0.397	0.250

Table 1: The maximum and the average differences between the solutions

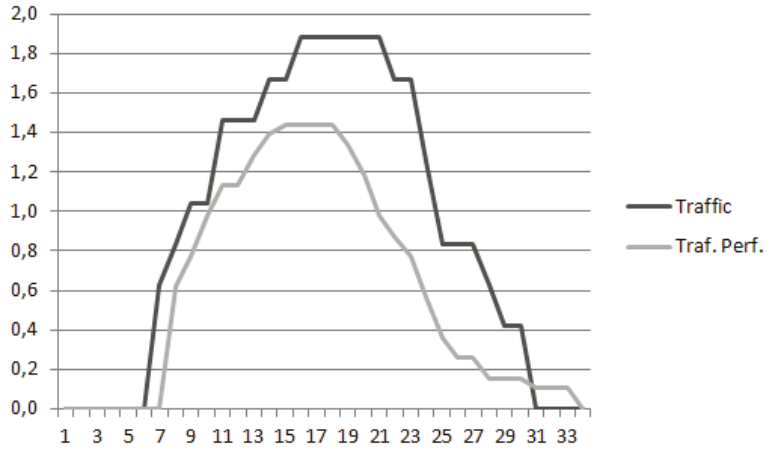

Figure 5: The difference of the solutions on a network with 10 junctions

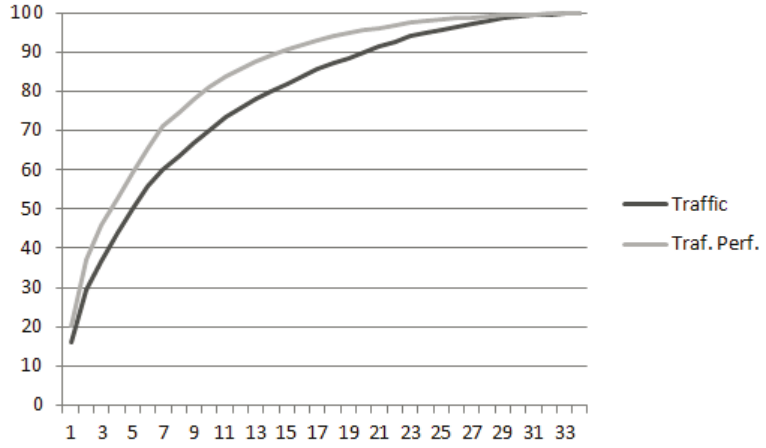

Figure 6: The optimal solution on a network with 10 junctions

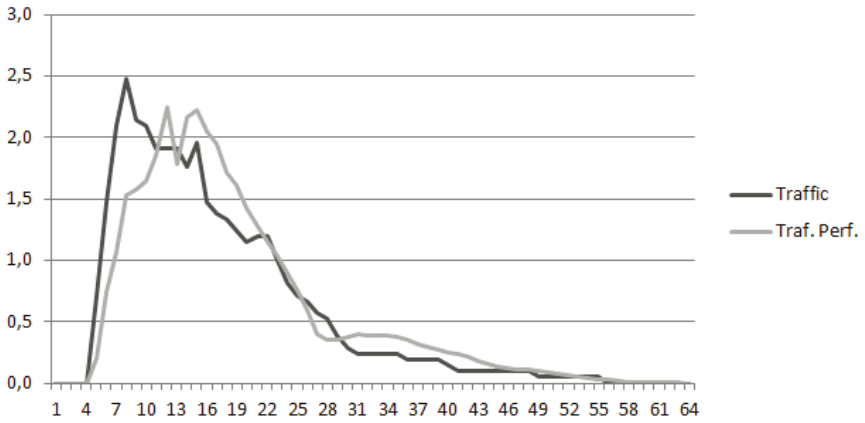

Figure 7: The difference of the solutions on a network with 20 junctions

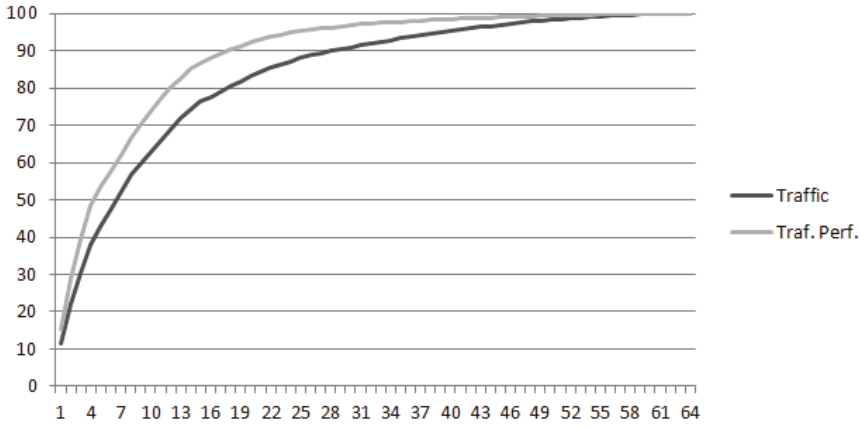

Figure 8: The optimal solution on a network with 20 junctions

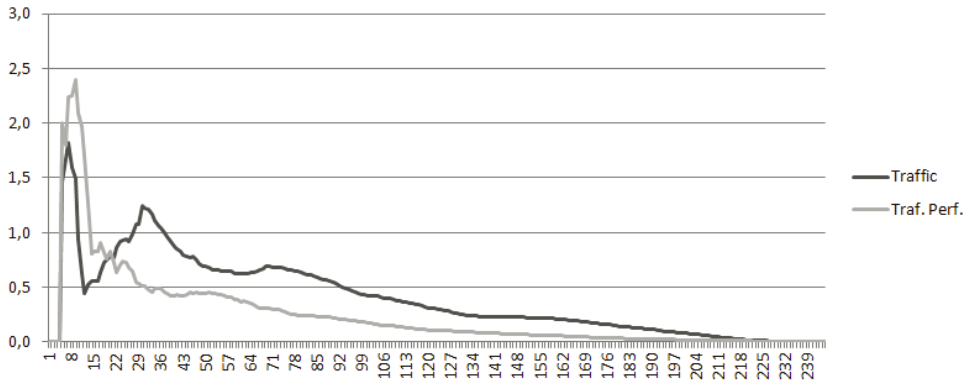

Figure 9: The difference of the solutions on a network with 80 junctions

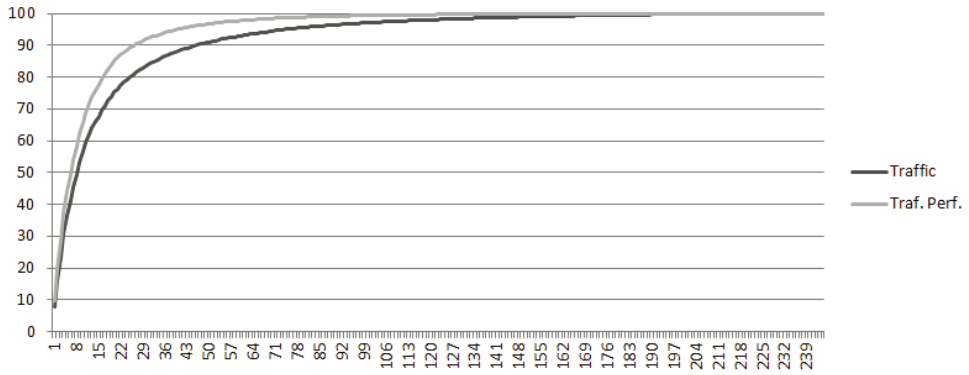

Figure 10: The optimal solution on a network with 80 junctions

The results show that:

- The maximum difference between the solution of the greedy algorithm and the optimal solution was below 2.5%, but the average difference was usually below 1% (see Tab. 1).
- Increasing the size of the problem or the weights of the paths, the functions of the optimal solution became steeper (see Fig. 6, 8, 10). It means that we need proportionally less number of links to reach the same checked ratio.
- Selecting one or all links obviously gives optimal solution, so it is expected that selecting some or almost all links also results good solution. The shapes of the functions of the maximum difference have proved this idea. Additionally it can be seen that the parts of the worst cases are not too long and they occurred earlier with increasing the size of the problem.

7 Conclusion

The link selection problem is an NP-hard optimization problem. The goal of the original real life problem was to check efficiently the vehicles on a road network. The checking process is realized in such way that it does not change the flow of traffic. A greedy heuristic was applied to solve a large size problem, but the qualification of this heuristic was demonstrated only on small (10 loaded links) networks [6].

The optimal solution of the link selection problem can be calculated with binary integer programming models as well. With this method the size of the investigated problems could be increased, but it is bounded by the capability of the applied solver contrary to the heuristic that is always usable.

In this study the greedy heuristic has been described generally for the link selection problem and it was tested on different sized problems. The results were compared with the optimal solutions that were calculated with binary integer programming models presented here. The comparison shows that the solution of this fast heuristic is much closer to the optimal solution than it is guaranteed by the general approximation ratio.

References

[1] V. Chvátal, A greedy heuristic for the set-covering problem, *Math. Oper. Res.,* **4,** 3, (1979) 233–235. ⇒107

[2] D. S. Hochbaum (Ed.), *Approximation Algorithms for NP-hard Problems,* PWS Publishing Company, Boston, Mass., 1997. ⇒107, 108, 113

[3] D. B. Johnson, Approximation algorithms for combinatorial problems, *J. Comp. System Sci.*, **9,** 3 (1974) 256–278. ⇒107, 112

[4] L. Lovász, On the ratio of optimal integral, and fractional covers *Discrete Math.,* **13,** 4 (1975) 383–390. ⇒112

[5] L. Marton, P. Pusztai, On modelling and computing traffic assignment, *Proc. EURO XVII. European Conf. Operational Research,* Budapest, Hungary, 2000, pp. 114. ⇒107

[6] P. Pusztai, An application of the greedy heuristic of set cover to traffic checks, *CEJOR Cent. Eur. J. Oper. Res.*, **16,** 4 (2008) 407–414 ⇒107, 108, 115, 119

Received: January 9, 2015 • Revised: May 1, 2015