Minimisation of Network Covering Services with Predefined Centres

Milos Šeda[1], Pavel Šeda[2]
[1]Institute of Automation and Computer Science,
Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
Email: seda@fme.vutbr.cz
[2]Department of Telecommunications, Brno University of Technology, Brno, Czech Republic

KEYWORDS

Set covering, unicost problem, threshold, reachability matrix, genetic algorithm, repair operator

ABSTRACT

In this paper, we deal with a special version of the set covering problem, which consists in finding the minimum number of service centres providing specialized services for all customers (or larger units, such as urban areas) within a reasonable distance given by a threshold. If a suitable threshold is found that makes it possible to determine a feasible solution of the task, the task is transformed into a general set covering problem. In order to reflect the importance of the centers, we assign weights to them and, if some centers must be contained in the result, we can either add columns in the reachability matrix with link to these centres or add special constraints in the mathematical model. However, this is of a combinatorial nature and, because it belongs to the class of NP-hard problems, for a large instance of the problem, it cannot be used to find the optimal solution in a reasonable amount of time. In the paper, we present a solution that uses two heuristic methods: genetic algorithm and tabu search.

INTRODUCTION

There are numerous discussions on how to optimise a network of public facilities (e.g. hospitals and schools) that provide essential services (health, education) for the population so that the cost of their operation is as low as possible and each inhabitant or an urban district has at least one of the service centres in an affordable distance. It is clear that the question of what is an affordable distance is debatable and could be determined by agreement of the ruling political parties. In this text, however, we ignore the political aspects and address a formal mathematical approach to solve such tasks.

In the literature, the general set covering problem is studied that does not address any threshold of availability, but it is directly given by the matrix of binary values and a covering of all columns by suitable choice of rows is looked for. This task is an NP-hard problem [3] and, for a larger problem instance, can be solved in a reasonable time only by heuristic methods.

The problem that we investigate can be converted to a set covering problem because, by using a threshold, the distance matrix is changed to binary reachability matrix. However, if the threshold is chosen inadequately, the original task may have a number of degenerative cases, described in the following section, and we will show how setting an appropriate threshold makes it possible to find a solution using genetic algorithms and tabu search.

We also present modified data structures and model to guarantee that the results could not omit important service centres. Instead of the traditional weights, which only increase the probability that the important service centres will be included in the results, we propose additional colums in the reachability matrix, or additional constraints in the model.

PROBLEM FORMULATION

Assume that the transport network contains m vertices, that can be used as operating service centres, and n vertices to be served, and for each pair of vertices v_i (considered as service centres) and v_j (serviced vertex) their distance d_{ij} is given and D_{max} is the maximum distance which will be accepted for operation between the service centres and serviced vertices (Seda and Seda, 2015).

The aim is to determine which vertices must be used as service centres for each vertex to be covered by at least one of the centres and for the total number of operating centres to be minimal.

Remark 1.
1. A condition necessary to solve the task is that all of the serviced vertices are reachable from at least one place where an operating service centre is considered.
2. Serviced vertex v_j is reachable from vertex v_i, which is regarded as an operating service centre if $d_{ij} \leq D_{max}$. If this inequality is not satisfied, vertex v_j is unreachable from v_i.

Here, $a_{ij} = 1$ means that vertex v_j is reachable from v_i and $a_{ij} = 0$ means that it is not if v_i is operating service centre i. Similarly, $x_i = 1$ means that service centre i is selected while $x_i = 0$ means that it is not selected.

Then, the set covering problem can be described by the following mathematical model:

Minimise

$$z = \sum_{i=1}^{m} x_i \qquad (1)$$

Proceedings 32nd European Conference on Modelling and
Simulation ©ECMS Lars Nolle, Alexandra Burger,
Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

subject to

$$\sum_{i=1}^{m} a_{ij} \cdot x_i \geq 1, j = 1, \ldots, n \qquad (2)$$

$$x_i \ \in \{0, 1\}, i = 1, \ldots, m \qquad (3)$$

The objective function 1 represents the number of operating centres, constraint 2 means that each serviced vertex is assigned at least to one operating service centre. The parameter D_{max} represents a threshold of service reachability.

Example 1. Consider the following *distance matrix* which expresses service centres and serviced vertices (=customer locations) and D_{max}=40. Rows are service centres and columns are serviced vertices (customer locations).

$$\begin{array}{cc} & \text{serviced vertices (customer locations)} \\ & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \text{centres} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 5 & 41 & 50 & 26 & 38 & 60 & 44 & 59 \\ 49 & 82 & 13 & 67 & 68 & 20 & 32 & 31 \\ 45 & 17 & 61 & 45 & 67 & 48 & 53 & 127 \\ 37 & 170 & 195 & 32 & 77 & 88 & 90 & 30 \\ 58 & 42 & 25 & 101 & 133 & 32 & 21 & 78 \end{pmatrix} \end{array}$$

From D_{max} = 40 we get the reachability matrix of serviced vertices from sevice centres.

$$\begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

△

Special Cases

In this section, we will summarise cases for which the problem has no solution, or specified data need a modification. We will show this directly by using the below reachability matrices.

$$\begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

In the 2nd column of the previous matrix, we can see that the threshold distance is too low and the 2nd customer has no chance to visit a centre in a reachable distance. The threshold must be increased to get at least one 1 in each column. In the 3rd row of the previous matrix, is shown that service centre 3 can be omitted because it exceeds the threshold distance to all customers and nobody would visit it.

$$\begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

In the 4th row of the previous matrix, we can see that service centre 4 can be omitted because it exceeds the threshold distance to all customers and nobody would visit it.

$$\begin{array}{cc} & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

If a service centre must not be omitted, it represents only one centre for a customer, i.e., in the customer column, there is only one 1. Of course, we can have more necessary centres which cannot be omitted. However, if necessary centres cover all customers, then no centre needs to be added to the necessary ones and we immediately have a solution.

Computational Results

Since the mathematical model is simple, it seems that the problem could be solved by one of the optimisation toolboxes such as in GAMS (General Algebraic Modelling System) with the main part of code as follows:

```
LOOP(I,
      LOOP(J,
            IF (D(I,J) <= Dmax,
                 A(I,J)=1;
                ELSE
                 A(I,J)=0;
               );
          );
   );

VARIABLES
  X(I) decision variables
  XSum objective function;
BINARY VARIABLE X;

EQUATIONS
  EQ2(J) cover conditions
  EQ1   objective function (number of selected
     centres);
  EQ2(J) .. SUM(I,A(I,J)*X(I)) =G= 1;
  EQ1    .. XSum =E= SUM(I,C(I)*X(I));

MODEL COVER /ALL/;
SOLVE COVER USING MIP MINIMIZING XSum;
DISPLAY XSum.L, X.L;
```

This simple code in GAMS was tested here (and also in Excel Solver) for several cases such as pharmacies, employment offices and language schools in (Trchalíková, 2015).

However, if we apply the above procedure to minimising a network of service centres, we could get a solution where service centres in cities would be omitted.

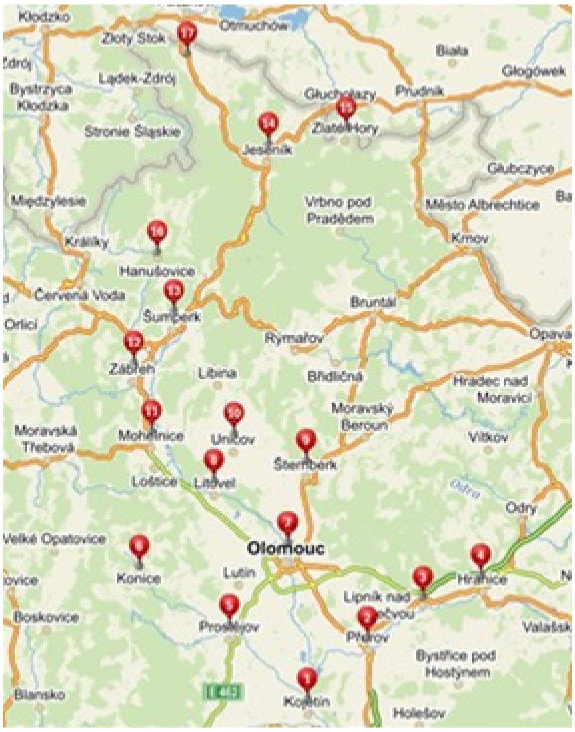

Fig. 1: Network of employment offices.

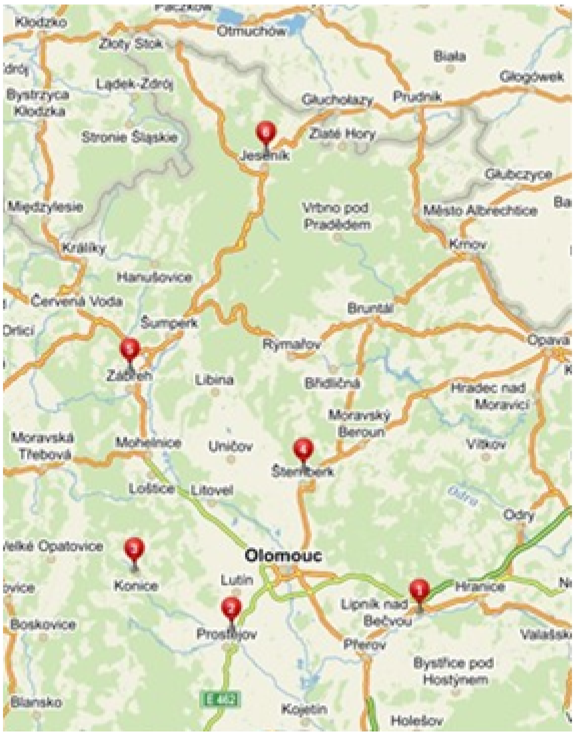

Fig. 2: Minimal cover in a reachable distance..

The first two figures taken from (Trchalíková, 2015) show the locations of employment offices in the city of Olomouc and its surroundings and the minimal cover of this area in a threshold distance. We can see that the number of these offices may be significantly reduced, but the office in the regional centre at Olomouc has also been cancelled. Of course, this situation is undesirable and, therefore, the model needs a modification.

In this case, it is appropriate to consider the importance of locations given by their size or necessity. As the objective function is minimised, it is necessary to determine the weights so that the lower the weight, the higher the priority.

It could even be suitable to classify facilities with high importance as necessary as if they represented the only choice for at least one of the customers.

If weights of service centres are expressed by coefficients c_j, the corresponding mathematical model would change as follows:

Minimise

$$z = \sum_{i=1}^{m} c_i \cdot x_i \tag{4}$$

subject to

$$\sum_{i=1}^{m} a_{ij} \cdot x_i \geq 1, j = 1, \ldots, n \tag{5}$$

$$x_i \in \{0, 1\}, i = 1, \ldots, m \tag{6}$$

From the point of view of the problem representation and parameter settings, there is no change with the exception of the objective function, for which equation 4 is used rather than equation 1.

However, if we want to ensure that the important centres in solving the problem will never be omitted, then the safer way is to extend the reachability matrix by columns containing only a single 1 in rows corresponding to these centres. Let us assume that the regional centres 2 and 5 must be contained in the result, then we will extend the reachability matrix by two additional columns (they represent dummy serviced vertices) as follows:

$$\begin{array}{c|cccccccc|cc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & & \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}$$

The advantage of this approach is that the model remains the same, only with the reachability matrix adapted. However, we can achieve the same result more simply without increasing the data structures by adding constraints to the model, assigning values 1 to the corresponding decision variables, here $x_2 = 1$ and $x_5 = 1$.

Another problem is that the GAMS software tool is useable only for "small" instances as in Figure 1 and in Figure 2. All computations leading to an optimum were performed in a few seconds, but for larger instances, they ended with a run time error with GAMS indicating "insufficient space to update U-factor …". It is caused by the fact that time complexity of the problem with m rows is $O(2^m)$ and, say, for an instance with 200 rows and 2000 columns tested in the following sections, its searching space has 2^{200} possible selections and $2^{200} = (1024)^{20} \approx 10^{60}$.

Therefore, for these cases, heuristics must be used. Two of them, genetic algorithm and tabu search, have been implemented and recommendations for their parameter settings are presented, based on many tests with various sets of possible operators (selection, crossover, mutation, etc).

Tabu-search

Definition 1. Tabu-search is a stochastic algorithm containing following parameters (Glover, 1989, 1990):

$$TS = (M, x_0, \Theta, f, t_{max}, TL, k), \tag{7}$$

where:
- M is a solution space,
- x_0 is the initial solution. If x is determined randomly, then the local search method is the stochastic algorithm,
- Θ denotes the set of permissible transformations generating a plurality of adjacent solutions,
- f is the objective function
- TL stands for a tabu list of forbidden transformations, and
- k is the size of TL, i.e., the capacity of the short-term memory.

■

The basic version of tabu search is represented by the following pseudopascal code 1. The empty list is denoted by $\varnothing$. The symbol $\oplus$ in binary operation between two lists represents the operation of connecting the second list to the end of the first one and, vice versa, $\ominus$ removes the first symbol of TL. The ϑ_1 symbol indicates the first element of the list (Glover and Taillard, 1993).

Obviously, the size of the forbidden transformation list affects the quality of the resulting solutions. With a small k, it may occur as a climbing algorithm, but not in the adjacent two steps. With a large k on the other hand, there is a risk of skipping promising local minima, among which there might be a local minimum. One of the possible modifications to the algorithm is *adapting the length of the tabu list.* Another modification of the tabu search algorithm is using a long-term memory. In proportion to the value, transformations are penalized. There are many other tabu search modifications. One of them is the reactive tabu-search described in (Battiti and Teccholli, 1994; Qingfu, 1993).

As to the neighbourhood operation in tabu search, we use the principle of the genetic algorithm shift mutation operator from the following paragraph with the length of the tabu list being 5.

Genetic Algorithms

Since the principles of GA's are well-known, we will only deal with GA parameter settings for the problems to be studied. Now we describe the general settings and the problem-oriented setting used in our application.

Individuals in the population (*chromosomes*) are represented as binary strings of length n, where a value of 0 or 1 at bit i (*gene*) implies that $x_i = 0$ or 1 in the solution respectively.

Algorithm 1 Tabu-search

```
procedure TabuSearch
    $t \leftarrow 1$;
    randomly select initial solution $x_{0t}$;
    $x^* \leftarrow x_{0t}$; $f_{min} \leftarrow f(x_{0t})$; $TL \leftarrow \varnothing$;
    while $t \leq t_{max}$ do
        $x_{loc} \leftarrow x_{0t}$; $f_{loc} \leftarrow f(x_{0t})$;
        for all $\vartheta \in \Theta$ do
            $y \leftarrow \vartheta(x_{0t})$;
            if $(f(y) < f_{loc})$ and $((\vartheta \notin TL)$ or $(f(y) < f_{min})$ then
                $x_{loc} \leftarrow y$;
                $f_{loc} \leftarrow f(y)$;
                $\vartheta_{loc} \leftarrow \vartheta$;
            end if
        end for
        if $f_{loc} < f_{min}$ then
            $f_{min} \leftarrow f_{loc}$;
            $f_{min} \leftarrow f_{loc}$;
        end if
        if $|TL| < k$ then
            $TL \leftarrow TL \oplus (\vartheta_{loc}^{-1})$;
        else
            $TL \leftarrow TL \ominus (\vartheta_1) \oplus (\vartheta_{loc}^{-1})$;
        end if
        $t \leftarrow t + 1$;
        $x_{0t} \leftarrow x_{loc}$;
    end while ▷ {$x_{min}$ is aproximation of minimal cover}
end procedure
```

The *population size* is usually set in the range [50, 200], in our programme, implemented in Java, 200 individuals in the population were used, because 50 individuals led to a reduction chromosome diversity and premature convergence.

Initial population is obtained by generating random strings of 0s and 1s in the following way: First, all bits in all strings are set to 0, and then, for each of the strings, randomly selected bits are set to 1 until the solutions (represented by strings) are feasible.

The *fitness function* corresponds to the objective function to be maximised or minimised; here, it is minimised.

Three of the most commonly used methods of *selection* of two parents for *reproduction*, roulette selection, ranking selection, and tournament selection, were tested.

As to *crossover*, uniform crossover, one-point and two-point crossover operators were implemented.

Mutation was set to 5, 10 and 15 %, exchange mutation, shift mutation, and mutation inspired by well-known *Lin-2-Opt change operator* usually used for solving the travelling salesman problem (Gutin and Punnen, 2007) were implemented.

In *replacement* operation two randomly selected individuals with below-average fitness were replaced by the children generated.

Termination of a GA was controlled by specifying a maximum number of generations t_{max}, e.g. $t_{max} \leq 10000$.

Repair Operator

The chromosome is represented by an m-bit binary string S where m is the number of columns in the SCP. A value of 1 for bit i implies that service centre i is in the solution and 0 that it is not. Since the SCP is a minimisation problem, the lower the fitness value, the more fit the solution is. The fitness of a chromosome for the unicost SCP is calculated by 8.

$$f(S) = \sum_{i=1}^{m} S_i \tag{8}$$

The binary representation causes problems with generating infeasible chromosomes, e.g., in the initial population, in crossover, and/or mutation operations. To avoid infeasible solutions, a *repair operator* (Seda and Seda, 2015) is applied.

Algorithm 2 Repair Operator for Set Covering Problem

```
Input: $I = \{1, \ldots, m\}$ = the set of all rows; $J = \{1, \ldots, n\}$ = the set of all columns; $S$ = the set of rows in a solution; $U$ = the set of uncovered columns; $w_i$ = the number of rows that cover column $j, j \in J$ in $S$; $\alpha_j\{i \in I\ |a_{ij} = 1\}$ = the set of rows that cover column $j, j \in J$; $\beta_i = \{j \in J|a_{ij} = 1\}$ = the set of columns that are covered by row $i$, $i \in I$;
procedure REPAIROPERATOR
    initialise $w_j = |S \cap \alpha_j|, \forall j \in J$;
    initialise $U = \{j|w_j = 0, \forall j \in J\}$;
    for all column $j$ in $U$ (increasing order of $j$) do
        find the first row $i$ (in increasing order of $i$) in $\alpha_j$ that minimises $1/$—$U \cap \beta_i$—;
        $S \leftarrow S + i$;
        $w_j \leftarrow w_j + 1, \forall j \in \beta_i$;
        $U \leftarrow U - \beta_i$;
    end for
    for all row $i$ in $S$ (in decreasing order of $i$ do
        if $w_j \geq 2, \forall_j \in \beta_i$ then
            $S \leftarrow S - i$;
            $w_j \leftarrow w_j - 1, \forall_j \in \beta_i$;
        end if
    end for
end procedure ▷ {$S$ is now a feasible solution to the SCP and contains no redundant rows }
```

The initialising steps identify the uncovered columns. Since the statements are "greedy" heuristics in the sense that in the 1st **for**, rows with low cost-ratios are being considered first and in the 2nd **for**, rows with high costs are dropped first whenever possible.

RESULTS

It is obvious that the tabu search converges to a very close approximation of the optimal solution, which is consistent with the well-known "No free lunch theorem". GA started with the objective function value 1225 and finished with the best value 166, and TS found the best value 172. Because the tabu search is a one-point method, only the dependence of the objective function for gradually updated points (centres for generating neighbours) in the search space is plotted. As in the genetic algorithm, it is necessary to apply the repair operator for the selected solution in the neighbourhood, as described in the previous section. The computational time of GA for the tested instance with 200 rows was only 19 seconds on a computer with a processor frequency of 2.4 GHz and operating memory of 4 GB while SA takes more than 1 minute. The reason is that, in each GA iteration, we generate only two children while TS creates the neighbourhood with 200 neighbours in each iteration.

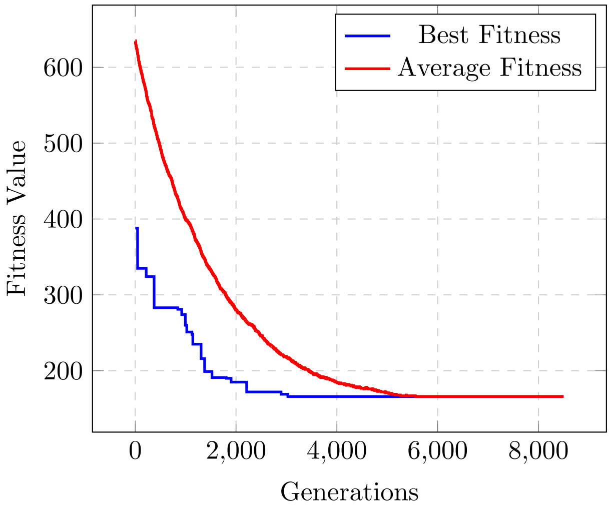

Fig. 3: Results from GA. 8500 generations, 5 % mutation, Roulette wheel selection

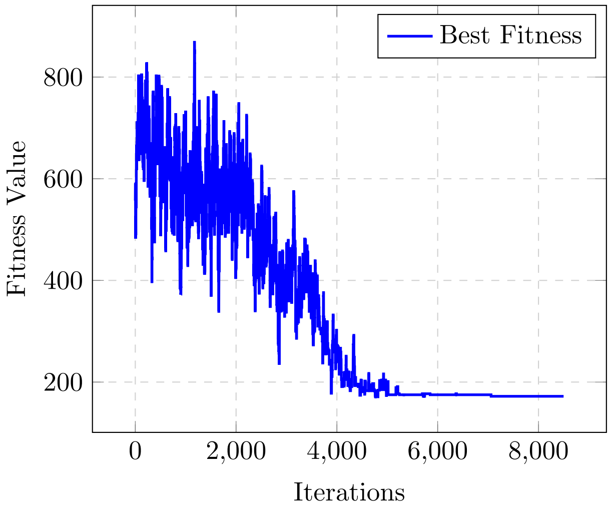

Fig. 4: Results from TS. TS – 8500 iterations

CONCLUSIONS

In this paper, we studied the set covering problem in a special case, in which a threshold is defined. This task may be used for optimising networks providing public services with operation costs being minimal.

We have shown how to increase the probability of selecting important centres with the addition of their weights, or how to directly ensure that some centres will not be missing in the result by adding columns to the reachability matrix, or by adding constraints to the model.

Due to the exponential time complexity, classical optimisation programs, often based on a branch and bound method, cannot be used to solve larger instances of (mixed-)integer programming problems. Therefore, a heuristic approach was proposed. The programme for solving this problem was implemented and parameter settings recommended based on testing many combinations of possible selections of their operators. It was shown that these methods yield very similar results when executed tens of times. In the future, we will try to implement other modern heuristic methods.

REFERENCES

R. Battitti and G. Teccholli. Simulated annealing and tabu search in the long run: A comparison on qap tasks. *Computers and Mathematics with Applications*, 28(6): 1–8, 1994. ISSN 08981221. doi: 10.1016/0898-1221(94)00147-2. URL `http://linkinghub.elsevier.com/retrieve/pii/0898122194001472`.

F. Glover. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, 1989. ISSN 0899-1499. doi: 10.1287/ijoc.1.3.190. URL `http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1.3.190`.

F. Glover. Tabu search—part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990. ISSN 0899-1499. doi: 10.1287/ijoc.2.1.4. URL `http://pubsonline.informs.org/doi/abs/10.1287/ijoc.2.1.4`.

F. Glover and E. Taillard. A user's guide to tabu search. *ORSA Journal on Computing*, 41(1):1–28, 1993. ISSN 0254-5330. doi: 10.1007/BF02078647. URL `http://link.springer.com/10.1007/BF02078647`.

G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations.* Discrete Mathematics in Computer Science. Springer US, 2007. ISBN 978-0-387-44459-8. doi: 10.1007/b101971.

Z. Qingfu. A user's guide to tabu search. *IEEE Computational Intelligence Magazine*, 5(1):59–60, 1993. ISSN 1556-603x. doi: 10.1109/MCI.2009.935312. URL `http://ieeexplore.ieee.org/document/5386108/`.

M. Seda and P. Seda. *A Minimisation of Network Covering Services in a Threshold Distance.* In Recent Advances in Soft Computing. Advances in Intelligent Systems and Computing. Heidelberg, Germany: Springer, 2015. ISBN 978-3-319-19823-1.

P. Trchalíková. *Optimisation of Coverage of Public Services (in Czech).* Diploma project. Přerov: VŠLG, 2015.