A Minimisation of Network Covering Services in a Threshold Distance

Miloš Šeda and Pavel Šeda

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. However, this has 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 by means of two stochastic heuristic methods - genetic algorithms and simulated annealing – and, using a test instance from OR-Library, we present the parameter settings of both methods and the results achieved.

Keywords Set covering · Unicost problem · Threshold · Reachability matrix · Genetic algorithm · Repair operator

1 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

M. Šeda (✉)
Faculty of Mechanical Engineering, Brno University of Technology,
Technická 2, 616 69 Brno, Czech Republic
e-mail: seda@fme.vutbr.cz

P. Šeda
IBM Global Services Delivery Center, Technická 21, 616 00 Brno, Czech Republic
e-mail: pavelseda@email.cz

P. Šeda
Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1,
613 00 Brno, Czech Republic

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_13

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 simulated annealing.

2 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.

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

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

subject to

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

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

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:

Consider the following *distance matrix* which expresses service centres and serviced vertices (= customer locations) and $D_{\max} = 40$.

	serviced vertices (customer locations)							
service centres	1	2	3	4	5	6	7	8
1	5	41	50	26	38	60	44	59
2	49	82	13	67	68	20	32	31
3	45	17	61	45	67	48	53	127
4	37	170	195	32	77	88	90	30
5	58	42	25	101	133	32	21	78

From $D_{\max} = 40$ we get the *reachability matrix* of serviced vertices from service centres.

	1	2	3	4	5	6	7	8
1	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	1	1
3	0	1	0	0	0	0	0	0
4	1	0	0	1	0	0	0	1
5	0	0	1	0	0	1	1	0

Since only service centre 3 is reachable from the second serviced vertex (serviced vertex 2 is covered by the 3rd service centre) and only service centre 1 is reachable from serviced vertex 5, these service centres must not be omitted. These two centres cover serviced vertices 1, 4, 5, and 2.

This means that the service centres must be found that cover the remaining uncovered vertices 3, 6, 7 and 8. This can be achieved either by selecting the service centres 2 and 5, or 4 and 5.

Thus, the example has two solutions, where four vertices are sufficient to cover serviced vertices (either 1, 3, 2, 5, or 1, 3, 4, 5)

In the general case, however, the selection of service centres for k uncovered vertices has 2^k possibilities and, thus, the task complexity increases exponentially with the number of uncovered vertices.

For large k, we must solve the task by a heuristic method, e.g., by a genetic algorithm [2, 4, 7, 8], ant colonies [6], differential evolution, SOMA (Self Organizing Migrating Algorithm) [12], HC12 meta-heuristic algorithm [13].

2.1 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.

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

In the 3rd row of the previous matrix, we can see that service centre 3 can be omitted because it exceeds the threshold distance to all customers and nobody would visit it.

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

In the 5th column of the previous matrix, we can see that the threshold distance is too low and the 5th 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.

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.

3 Computational Results

Since the mathematical model is simple, it seems that the problem could be solved by one of the optimisation toolboxes.

E.g. in GAMS (General Algebraic Modelling System) the main part of code is as follows:

```
VARIABLES
  X(I) decision variables
  XSum objective function;
BINARY VARIABLE X;
EQUATIONS
  EQ6(J) cover conditions
  EQ5    objective function (number of selected centres);
  EQ6(J) .. SUM(I,A(I,J)*X(I)) =G= 1;
  EQ5    .. XSum =E= SUM(I,C(I)*X(I));

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

However, this software tool is successful for only "small" instances. All computations leading to 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, a heuristic approach must be used. Two of them, genetic algorithm and simulated annealing, have been implemented and recommendations of their parameter settings are presented, based on many tests with various sets of possible operators (selection, crossover, mutation, etc.).

3.1 Genetic Algorithm

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.

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 0 s and 1 s 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 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 [5] were implemented.

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

Termination of a GA was controlled by specifying a maximum number of generations *tmax*, e.g. $tmax \leq 10000$.

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 otherwise.

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 (4).

$$f(S) = \sum_{i=1}^{m} S_i \tag{4}$$

The binary representation causes problems with generating infeasible chromosomes, e.g., in initial population, in crossover, and/or mutation operations. To avoid infeasible solutions, a *repair operator* [10] is applied.

Let
$I = \{1, \ldots, m\}$ = 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_j = the number of rows that cover column j, $j \in J$ in S.
$\alpha_j = \{i \in I \mid a_{ij} = 1\}$ = the set of rows that cover column j, $j \in J$;
$\beta_i = \{j \in J \mid a_{ij} = 1\}$ = the set of columns that are covered by row i, $i \in I$;

The repair operator for the unicost SCP has the following form:
initialise $w_j := |S \cap \alpha_j|, \forall j \in J$;
initialise $U := \{j \mid w_j = 0, \forall j \in J\}$;
for each column j in U (in increasing order of j) **do**
begin find the first row i (in increasing order of i) in α_j
that minimises $1/|U \cap \beta_i|$;
$S := S + i$;
$w_j := w_j + 1, \ \forall j \in \beta_i$;
$U := U - \beta_i$
end;
for each row i in S (in decreasing order of i) **do**
if $w_j \geq 2, \forall j \in \beta_i$
then begin $S := S - i$;
$w_j := w_j - 1, \ \forall j \in \beta_i$
end;
{ S is now a feasible solution to the SCP and contains no redundant rows }

Initialising steps identify the uncovered columns. **For** 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.

The genetic algorithm was tested using an instance from OR-Library [1] with 200 rows and 2000 columns.

Figure 1 shows the computation of a fitness function as depending on the sequence of generations of the algorithm. The upper characteristic shows the average value of the fitness function of all individuals in the population and the lower one shows the fitness function of the best individual in the population. It can be seen that the algorithm converges quite rapidly to a pseudo-optimal solution (no optimal solution is known for such a large instance).

The test results showed that one-point crossover gave worse results than two-point crossover, which, in turn, was even worse than a uniform crossover. Obviously, it would also be appropriate to monitor the width of the middle part of a chromosome in the two-point crossover because, when its width is very small, it brings very little change in comparison with parental chromosomes and reduces the diversity of genetic material in the population, which increases the risk of small or almost no changes in the fitness function after only a small number of generations.

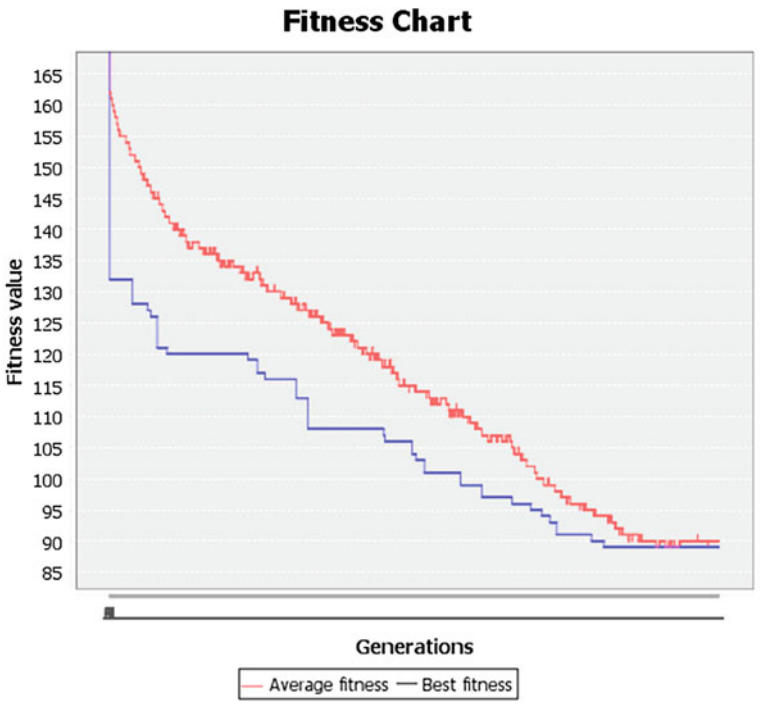

Fig. 1 Computation by genetic algorithm (instance with 200 rows and 2000 columns, roulette wheel selection, uniform crossover, mutation probability 5 %, 6500 generations, the best solution from 10 runs)

3.2 Simulated Annealing

The simulated annealing (SA) technique has been given much attention as a general-purpose way of solving difficult optimisation tasks. Its idea is based on simulating the cooling of a material in a heat bath - a process known as annealing. More details may be found in [9].

As in the previous section we only mention the parameter settings:

- *initial temperature* $T_0 = 10000$,
- *final temperature* $T_f = 1$,
- *temperature reduction function* $\alpha(t) = t * decrement$, where $decrement = 0.999$,
- *neighbourhood* for a given solution $\mathbf{x}_0$ is generated so that each neighbour is given by the inverse of one position in the binary string representing $\mathbf{x}_0$. (This means that the number of neighbours of $\mathbf{x}_0$ is equal to the number of positions in this string.) (Fig. 2)

It is obvious that simulated annealing converges to the same, or a very close approximation of the optimal solution, which is consistent with the well-known "No free lunch theorem" [11]. Because the simulated annealing is a one-point

method, only the dependence of the objective function for gradually updated points $\mathbf{x}_0$ (centres for generating neighbours) in the search space is plotted. The chart also shows that the likelihood of transfer to a worse neighbour is decreasing steadily as temperature decreases, which is also in line with the principle of the method.

As in the genetic algorithm, it is necessary to apply the repair operator for the

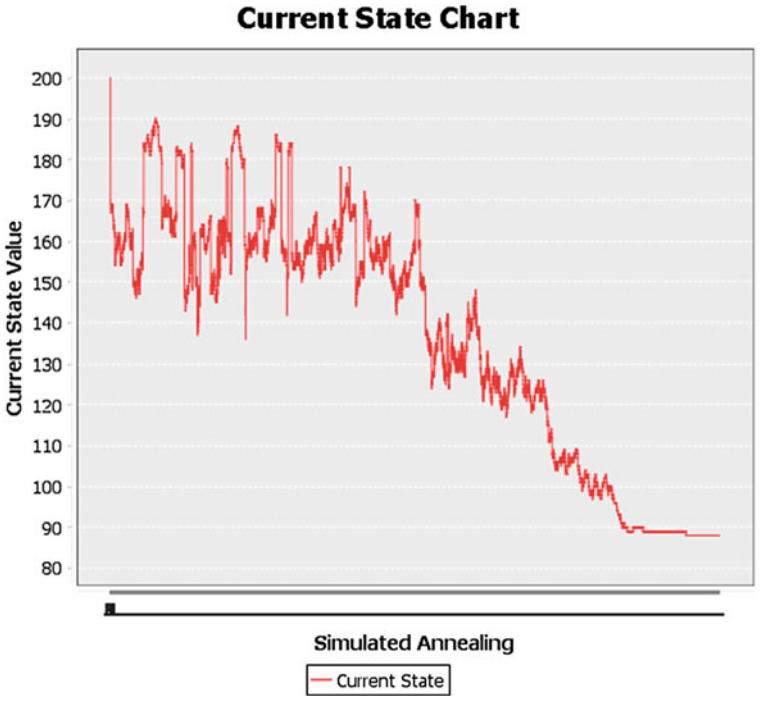

Fig. 2 Computation by simulated annealing (the same instance as in GA)

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 s on a computer with a processor frequency of 2.4 GHz and operating memory of 4 GB while SA takes more than 1 min. The reason is that, in each GA iteration, we generate only two children while SA creates the neighbourhood with 200 neighbours in each iteration when a new position is chosen.

An even worse situation would be for a hill-climbing algorithm, where each individual would have to be modified by the repair operator in order to determine the best neighbour. For simulated annealing, it is sufficient to apply the repair operator only to a randomly selected individual from the neighbourhood.

4 Practical Aspects

In the foregoing, we investigated the set covering problem in a version that did not take into account the importance of individual centres. If we apply the above procedure to minimising a network of hospitals and schools, we could get a solution where hospitals and schools respectively in cities would be omitted. In this case, it is appropriate to consider their importance 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 represent unique choice for some customers.

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

Minimise

$$z = \sum_{i=1}^{m} c_i x_i \tag{5}$$

subject to

$$\sum_{i=1}^{m} a_{ij} x_i \geq 1, \quad j = 1, \ldots, n \tag{6}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, m \tag{7}$$

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 Eq. (5) is used rather than Eq. (1).

5 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 [12].

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 include weights of service centres in our considerations and try to implement other modern heuristic methods [13, 14].

References

1. Beasley, J.E.: OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **11**, 1069–1072 (1990)
2. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. Eur. J. Oper. Res. **94**, 392–404 (1996)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
4. Goldberg, D.E.: The Design of Innovation (Genetic Algorithms and Evolutionary Computation). Kluwer Academic Publishers, Dordrecht (2002)
5. Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers, Dordrecht (2002)
6. Lessing, L., Dumitrescu, I., Stützle, T.: A comparison between ACO algorithms for the set covering problem. In: Dorigo, M. (ed.) ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 1–12. Springer-Verlag, Berlin (2004)
7. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin (1996)
8. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer, Berlin (2002)
9. Reeves, C.R.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford (1993)
10. Šeda, M., Roupec, J., Šedová, J.: Transportation problem and related tasks with application in agriculture. Int. J. Appl. Math. Inf. **8**, 26–33 (2014)
11. Wolpert, D.H., McReady, W.G.: No Free Lunch Theorems for Optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
12. Zelinka, I., Snášel, V., Abraham, A. (eds.): Handbook of Optimization: From Classical to Modern Approach. Berlin. Springer, Berlin (2013)
13. Matousek, R.: HC12: the principle of CUDA implementation. In: Proceedings of 16th International Conference on Soft Computing – MENDEL 2010. Mendel series vol. 2010, pp. 303–308, Brno (2010), ISSN: 1803-3814
14. Matousek, R., Zampachova, E.: Promissing GAHC and HC12 algorithms in global optimization tasks. J. Optim. Methods Softw. **26**(3), 405–419 (2011)