Steiner Tree Problem in Graphs and Mixed Integer Linear Programming-Based Approach in GAMS

MILOŠ ŠEDA
Institute of Automation and Computer Science
Brno University of Technology
Technická 2, 616 69 Brno
CZECH REPUBLIC

Abstract: - The Steiner tree problem in graphs involves finding a minimum cost tree which connects a defined subset of the vertices. This problem generalises the minimum spanning tree problem, in contrast, it is NP-complete and is usually solved for large instances by deterministic or stochastic heuristic methods and approximate algorithms. In this paper, however, we focus on a different approach, based on the formulation of a mixed integer programming model and its modification for solving in the professional optimization tool GAMS, which is now capable of solving even large instances of problems of exponential complexity.

Key-Words: - Steiner tree problem, NP-completeness, heuristic, performance ratio, network flow, mixed-integer linear programming, GAMS

Received: September 14, 2021. Revised: May 9, 2022. Accepted: June 5, 2022. Published: July 2, 2022.

1 Introduction

In practice, e.g., in telecommunication applications [7], [9], [2], [18], we frequently solve problems based on finding minimal networks. The criterion of minimality can be represented by the total costs for the implementation of the network or by the total length of connections. These problems include, e.g., the problem of finding optimal location of a source with respect to given sinks.

The *Network Steiner tree problem* (NSTP) (or *Steiner tree problem in graphs*) [10], [14], [22] is concerned with connecting a subset of vertices at a minimal cost. More precisely, given an undirected connected graph $G = (V, E)$ with vertex set V, edge set E, nonnegative weights associated with the edges, and a subset B of V (called *terminals* or *customer vertices*), the problem is to find a subgraph, T, which connects the vertices in B so that the sum of the weights of the edges in T is minimised. It is obvious that the solution is always a tree and it is called a *Steiner minimum tree* for B in G.

If $|B| = 2$ then the problem reduces to the *shortest path problem* and can be solved by Dijkstra's algorithm. In the case of $B = V$ the NSTP reduces to the *minimum spanning tree* (MST) *problem* and can be solved by Jarník's (Prim's), Borůvka's or Kruskal's algorithm. All these algorithms are polynomial. However, in the general case the NSTP is NP-complete [15], [20] and it cannot therefore be solved exactly for larger instances, i.e. heuristic or approximate methods must be used. A Steiner minimum tree is not a minimum spanning tree only, it can also span some nonterminals, called *Steiner vertices*, as shown in Fig. 1.

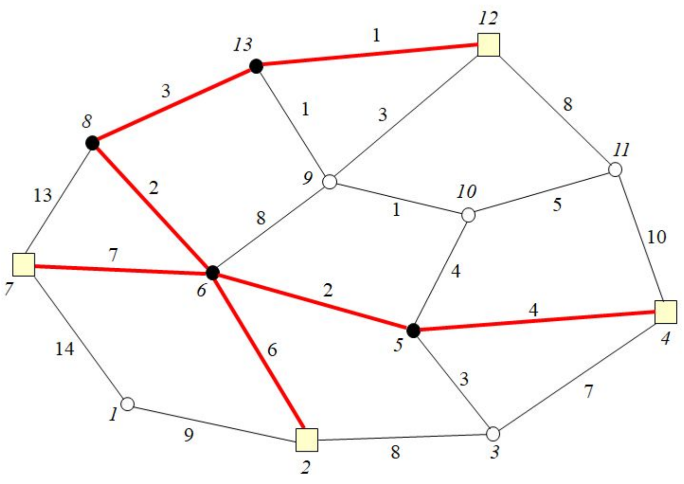

Figure 1: An example of the Streiner tree problem in graphs, = terminals, • = Steiner vertices)

2 Approximation algorithms

As was mentioned above, the network Steiner tree problem is NP-complete. It means that solving the problem requires a computational time that grows (in the worst case) *exponentially* with the problem size. Therefore we must use approximation or heuristic approaches for large scaled instances. Many approaches are based on a reformulation of the problem [5], [8] [12], branch and bound method, dynamic programming, etc. [11], [19].

Aproximation algorithms are algorithms that guarantee that the "distance" of their solutions from the optimum is in the worst case restricted by a mathematically proved upper bound. The quality of an approximation algorithm is mostly measured by its *performance ratio*, which is given by the ratio of then achieved solution and the optimum. If a Steiner tree for a set of terminals B is computed by an approximation algorithm and has a total length $ASMT(B)$, Steiner minimum tree for B is $SMT(B)$, then the Steiner tree problem performance ratio $\varrho_{\text{graph}}(B)$ is given by

$$\varrho_{\text{graph}}(B) = \frac{w(ASMT(B))}{w(SMT(B))}$$

The simplest approximation algorithms for the NSTP are *distance network approximation* [17], *minimum path approximation* [24], and *contraction approximation* [21]. We will briefly summarise only the first one.

A distance network approximation [3] can be described as follows:

[*To determine a Steiner mimimum tree in a connected weighted graph $G = (V, E)$ with a set of terminals B*]

1. Construct the complete graph $K_B = (B, F)$ in which the weight of $\{i, j\} \in F$ is the shortest distance between i and j in G.
2. Obtain a minimum spanning tree $MST(K_B)$ of the graph K_B.
3. Replace each edge $\{i, j\}$ of $MST(K_B)$ by a shortest path between i and j in G. (The resulting graph, G', is a Steiner subgraph of G since it is connected and contains B.)
4. Obtain a minimum spanning tree $MST(G')$ in G'. ($MST(G')$ is a Steiner tree.)
5. If v is a Steiner vertex of degree 1 in $MST(G')$, delete v from the tree $MST(G')$ with its incident edge. Continue this process by deleting one Steiner vertex at a time.

Theorem 2.1 *The distance network approximation algorithm runs in $\mathcal{O}(|B||V|^2)$ time.*

Proof. Running times of the steps are:

1. If the distance graph is not known, Step 1 requires time $\mathcal{O}(|B||V|^2)$ to compute the shortest paths from each of the $|B|$ vertices.
2. Kruskal's minimum spanning tree algorithm requires time $\mathcal{O}(|B| \log |B|)$.
3. Each of the $|B| - 1$ edges of $MST(K_B)$ may correspond to a path in G of up to $|V| - 1$ edges. Hence, Step 3 requires time $\mathcal{O}(|B||V|)$.
4. $\mathcal{O}(|B||V| \log(|B||V|))$ time using Kruskal's algorithm again.
5. The final step is done in time $\mathcal{O}(|V|)$.

Step 1 is the most expensive and gives the distance network approximation algorithm a time complexity of $\mathcal{O}(|B||V|^2)$. •

Unfortunately, the performance ratio of the above algorithm is equal to 2, and thus in the worst case the length of the obtained Steiner tree is 2 times longer than the length of the minimum Steiner tree.

The approximation ratios of other deterministic heuristic algorithms for solving the Steiner problem in graphs are only slightly better, so we will not discuss them further and move on to the formulation of the mathematical model and its solution in the optimization tool GAMS.

3 Mathematical formulation

Let $V = \{1, 2, \dots, n\}$ and S be a set of Steiner vertices. For every edge (i, j), c_{ij}, $c_{ij} \geq 0$ is a weight of the edge. The aim is to find a connected graph $G' = (B \cup S, E')$ (Steiner tree), $E' \subset E$, so that the sum of weights to be minimal.

In other words, the Steiner minimum tree problem can be described as a problem of finding a set of edges that connects terminals. Therefore we can define a bivalent variable x_{ij} for each edge $(i, j) \in E$ indicating whether the edge (i, j) is included into the Steiner tree ($x_{ij} = 1$) or not ($x_{ij} = 0$) and similarly a bivalent variable f_i indicating whether vertex i is included in the Steiner tree ($f_i = 1$) or not ($f_i = 0$). For terminals, $i \in B$, it is satisfied $f_i = 1$, and $f_i \in \{0, 1\}$ for the other vertices, $i \in (V - B)$.

Each vertex of $B \cup S$ must be reachable via edges of E' from any other vertex of $B \cup S$. Therefore, it must be an endpoint of at least one edge in E'. For the sake of mathematical formulation, we select a vertex r, called *root*, with the *orientation of edges* to r, and by this way we get a *rooted tree*. Due to the selected orientation the root has no output edges, and thus $x_{rj} = 0$ for $j \in V - \{r\}$. So as the final graph G' would be a tree, i.e. it does not contain a cycle, just one path must lead to the root from the vertices included into the Steiner tree, and thus exactly one edge must leave these vertices. If a vertex is not included into the

Steiner tree, then no edge leaves it. The number of edges leaving vertex $i \in (V - \{r\})$ is defined by $\sum_{j=1}^{n} x_{ij}$, i.e. it corresponds to f_i.

Finally, we must provide the *connectedness* of G', that means the edges of E' in the selected orientation must enable a connection of any vertex with the root. For this reason we define a *network flow* with respect to E' so that all vertices of $(B \cup S) - \{r\}$ are *sources* of 1 unit of flow [1], and the root is a *sink*, supplied by flows of all vertices from the set $(B \cup S) - \{r\}$ of G'.

Let y_{ij} be a flow through the edge $(i, j) \in E'$. If i is a source of 1 unit of flow, then the difference between the total flow leaving i and the total flow entering i must be equal to 1. As vertices unselected into the Steiner tree have no incident edges, their flows are equal to 0, and for all vertices $i \in (V - \{r\})$ the equation $\sum_{j=1}^{n} y_{ij} - \sum_{j=1, j\neq r}^{n} y_{ji} = f_i$ is satisfied.

For each edge $(i, j) \notin E'$, i.e. edge unselected into the Steiner tree ($x_{ij} = 0$), the flow $y_{ij} = 0$. On the other hand, if the edge is selected ($x_{ij} = 1$), then since this edge is in the rooted tree is the only output edge of vertex i, the vertex i is the source of 1 unit of flow and the total inflow to i is at most equal to the flow from $n-2$ other vertices ($|V| - |\{i, r\}|$), it follows that the flow y_{ij} through edge (i, j) is at most equal to $(n-2)+1$, i.e., $(n-1)$. Both cases can be expressed by a single relation as follows: $0 \leq y_{ij} \leq (n-1)x_{ij}$ for $i, j \in V$.

Summarizing all the considerations so far, we can formulate the Steiner problem in graphs as a mixed integer linear programming problem of the following form:

$$\text{Minimise } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\forall j \in (V - \{r\}): \quad x_{rj} &= 0 \\
\forall i \in (V - \{r\}): \quad \sum_{j=1}^{n} x_{ij} &= f_i \\
\forall i \in (V - \{r\}): \quad \sum_{j=1}^{n} y_{ij} - \sum_{j=1, j\neq r}^{n} y_{ji} &= f_i \\
\forall i, j \in V: \quad 0 \leq y_{ij} &\leq (n-1)x_{ij} \\
\forall i, j \in V: \quad x_{ij} &\in \{0, 1\} \\
\forall i \in B: \quad f_i &= 1 \\
\forall i \in (V - B): \quad f_i &\in \{0, 1\}
\end{aligned}$$

Another formulation can be found in [16]. A survey of Steiner tree problem formulations is presented in [13].

4 Modification of the model and computing in GAMS

The mathematical model derived in the previous section can be used for computing Steiner trees by means of the optimisation package *GAMS* (*General Algebraic Modelling System*) [6] developed by Alexander Meeraus and Anthony Brooke at the World Bank in the 80s especially for the tasks of linear, nonlinear and mixed integer programming.

The question is how to choose the root from the given terminals. Since it can be any terminal, we can choose the first terminal in the list of terminals, or the smallest terminal number, as the root without loss of generality.

From the input data we need to create a matrix with edge lengths $\mathbf{C}=(c_{ij})$, that is, to somehow define the weights of the nonexisting edges. The usual approach, if it is a minimization problem, is to set these weights equal to ∞. However, this value is not defined in the computer, and it is not advisable to choose the largest displayable number in the computer either, as this may be dependent on the software used and, in addition, the sum in the objective function could cause overflow of the range. For these reasons, we assign zero weights to non-existing edges. Without further modification, however, this setting would lead to erroneous results, because the minimum Steiner tree would be constructed from zero-ranked edges, i.e., edges that in the specified graph do not exist at all. However, this shortcoming is easily remedied by adding constraints to ensure that for those pairs of vertices that are not connected by an edge, the values of x_{ij} and y_{ij} are zero.

In order to speed up the calculation, it will also be useful to limit the range of values of the y_{ij} variables. The model is based on the fact that all vertices except the root are sources of flow of size 1, and hence those vertices of the root graph where no edges enter are also sources of flow of size 1. It follows that all flows must be equal to integer values.

If we denote the root by r, then the model can be written as follows:

$$\text{Minimise } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
r := \min\{k \mid k \in B\} \\
\forall j \in (V - \{r\}): \quad x_{rj} &= 0 \\
\forall i \in (V - \{r\}): \quad \sum_{j=1}^{n} x_{ij} &= f_i
\end{aligned}$$

$$\begin{aligned}
\forall i \in (V-\{r\}):\ \sum_{j=1}^{n} y_{ij} - \sum_{j=1, j\neq r}^{n} y_{ji} &= f_i \\
\forall i,j \in V:\ y_{ij} &\leq (n-1)x_{ij} \\
\forall i,j \in V:\ y_{ij} &\in \mathbf{Z}_+ \\
\forall i,j \in V, c_{ij}=0:\ x_{ij} &:= 0 \\
\forall i,j \in V:\ x_{ij} &\in \{0,1\} \\
\forall i \in B:\ f_i &= 1 \\
\forall i \in (V-B):\ f_i &\in \{0,1\}
\end{aligned}$$

where $\mathbf{Z}_+$ denotes the set of nonnegative integers.

GAMS is a high level language for formulating models with concise algebraic statements that are easily read by modellers and computers alike, easily modified, and easily moved from one computer environment to another - it is independent on the solution algorithms of specific solvers. The basic components of a GAMS model are:

- SETS statement declares sets of indices of arrays and assigns values to them.
- In input data section, parameters are declared and values are assigned to them (SCALAR statement is defined for constants, PARAMETERS statement for arrays, TABLE statement simplifies matrix declarations). PARAMETERS statement can also be used for a definition of a data object by a formula which expresses how to calculate object values from the values of the other already defined objects.
- VARIABLES statement declares decision variables and objective function and enables to specify the type of variables (BINARY, INTEGER, POSITIVE, NEGATIVE, FREE (default)).
- EQUATIONS section describes constraints of the optimisation problem. MODEL statement builds a model of the task and assigns a name to the list of constraints.
- SOLVE statement calls the solver and specifies the way of optimisation (MINIMIZING or MAXIMIZING), variable optimised and runs calculations.
- DISPLAY statement is determined to output of results.

The source code in GAMS for data from Fig. 1 can be expressed by the following way:

```
$TITLE Network Steiner tree problem
$OFFSYMXREF
$OFFUELLIST
$OFFUELXREF

OPTION ITERLIM=200000
* ITERLIM maximal number of iterations

* Section defining index sets
SETS
  K indices of terminals /1*4/
  I start vertices of edges /1*13/;
  ALIAS(I,J);
* J - end vertices of edges

******************************************
* Input data section
PARAMETERS
  B(K) indices and numbers of terminals
       /1 2, 2 4, 3 7, 4 12/;
PARAMETERS
  N,R;
  N=CARD(I);
  R=B("1");
TABLE C(I,J) edge weights
*  (0 = corresponding edge does not exist)
    1  2  3  4  5  6  7  8  9 10 11 12 13
 1  0  9  0  0  0  0 14  0  0  0  0  0  0
 2  9  0  8  0  0  6  0  0  0  0  0  0  0
 3  0  8  0  7  3  0  0  0  0  0  0  0  0
 4  0  0  7  0  4  0  0  0  0  0 10  0  0
 5  0  0  3  4  0  2  0  0  0  4  0  0  0
 6  0  6  0  0  2  0  7  2  8  0  0  0  0
 7 14  0  0  0  0  7  0 13  0  0  0  0  0
 8  0  0  0  0  0  2 13  0  0  0  0  0  3
 9  0  0  0  0  0  8  0  0  0  1  0  3  1
10  0  0  0  0  4  0  0  0  1  0  5  0  0
11  0  0  0 10  0  0  0  0  0  5  0  8  0
12  0  0  0  0  0  0  0  0  3  0  8  0  1
13  0  0  0  0  0  0  0  3  1  0  0  1  0;
******************************************

* Variable section (decision variables
*                   and objective function)
VARIABLES
  X(I,J) equals to 1 if the edge ij is
   included into Steiner tree, 0 otherwise
  Y(I,J) edge flow
  F(I)   equals to 1 if the vertex i is
   included into Steiner tree, 0 otherwise
  St     total sum of edge weights in
         Steiner tree;
BINARY VARIABLE X;
INTEGER VARIABLE Y;
BINARY VARIABLE F;

* Equation section
EQUATIONS
  EQ1(I,J)
```

```
    EQ2(I)
    EQ3(I)
    EQ4(I,J)
    EQ5(I,J)
    EQ6(I,K)
    EQ7(I,J)
    DELKA;
    EQ1(I,J)$((ORD(I) EQ R)
          AND (ORD(J) NE R)) .. X(I,J) =E= 0;
    EQ2(I)$(ORD(I) NE R)
     .. SUM(J,X(I,J)) =E= F(I);
    EQ3(I)$(ORD(I) NE R)
     .. SUM(J,Y(I,J))
       -SUM(J$(ORD(J) NE R),Y(J,I))
       =E= F(I);
    EQ4(I,J) .. Y(I,J) =G= 0;
    EQ5(I,J) .. Y(I,J) =L= (N-1)*X(I,J);
    EQ6(I,K)$(ORD(I) EQ B(K)) .. F(I) =E= 1;
    EQ7(I,J) .. X(I,J)$(C(I,J) EQ 0) =E= 0;
    DELKA .. St =E= SUM((I,J),C(I,J)*X(I,J));

* Model description, solver execution
*           and display of results
MODEL STEINER /ALL/;
SOLVE STEINER USING MIP MINIMIZING St;
DISPLAY X.L, F.L, Y.L, St.L;
```

Since INTEGER in GAMS is restricted only for nonnegative integers the declaration of `INTEGER VARIABLE Y;` is sufficient to specify constraint assigning y_{ij} to $\mathbf{Z}_+$ domain.

The relational operator `=G=` has the meaning of $\geq$ and `=E=` corresponds to $=$.

To avoid large changes of GAMS program according input data structures, it is more efficient to prepare the data structures in a separate text file and include it in GAMS program by `$INCLUDE` directive, e.g. as follows:

```
$INCLUDE "B1.TXT"
```

Since more complex instances have (c_{ij}) tables with hundreds of columns and the line width is limited, the tables must be expressed in a continued form. Similarly long sequences of terminals must be written into several lines. Both these modifications were provided by a special conversion program.

5 Statistics and computational results

Traditional approaches to calculations by heuristic methods use statistical tests, which examine at a certain significance level (e.g. $\alpha = 0.05$), to what extent the mean value of the results obtained by different methods and different settings of their parameters at a larger number of runs is the same or different (and therefore one of the methods gives better results). For the t-test, we assume that the sets of values have a normal distribution. However, this assumption may not be met and then one of the non-parametric tests, such as the Wilcoxon test, must be used.

However, given the validity of the No Free Lunch Theorem [25], [26], one should not expect a general conclusion that any of the heuristics for each problem instance gives better results than other heuristics.

Here, we do not explore heuristics in this paper and instead use a mixed integer programming model using the CPLEX software tool built as a solver in the GAMS environment to find an exact solution by deterministic computation. Statistical evaluations are therefore meaningless here. What can be said, however, is that the power of this software today is considerably greater than it was 20 years ago, when in our experience for a problem with complexity $\mathcal{O}(20!)$ the system ended up with a runtime error and the message "`insufficient space to update U-factor ...`"; today it computes an exact solution on a laptop with a 2.4 GHz processor in about 4.5 minutes.

Table 1 summarizes the results of computing Steiner minimum trees for test problems with vertex counts of 50 and 100, defined sets of terminals and computation times, with all tree lengths corresponding to the optimal solutions. It can be seen that even for such large instances GAMS is able to obtain an accurate solution in a time of a few seconds at most.

test	$\lvert V\rvert$	$\lvert B\rvert$	time [sec]	length
B1	50	9	0.11	82
B2	50	13	0.56	84
B3	50	25	0.39	142
B7	75	13	1.14	116
B8	75	19	1.64	105
B9	75	38	0.51	225
B13	100	17	2.00	173
B14	100	25	1.88	244
B15	100	50	1.53	335

Table 1: Computational results

6 Conclusions

The paper proposed a general mathematical model of the network Steiner tree and its modification to use it in GAMS. According to our experience from another application area, presented in

[27], this tool is already able to successfully solve exponentially complex problems with data structures with thousands of rows and columns. Here, we verified that it is able to find exact solutions for graphs with up to 100 vertices.

However, more tests will need to be performed in the future to compare the results with benchmark data from combinatorial optimization libraries as [4], [23].

References:

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network Flows. Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs, New Jersey, 1993. ISBN 0-13-617549- X.

[2] S. Balaji, K. Kannan, and Y. B. Venkatakrishnan. Total Dominating Set Based Algorithm for Connected Dominating Set in Ad hoc Wireless Networks. WSEAS Transactions on Mathematics, 12:1164–1172, 2013.

[3] V. K. Balakrishnan. Network Optimization. Chapman & Hall Mathematics, London, 1995.

[4] J. E. Beasley. OR-Library. Report, The Management School Imperial College, London, 1996. http://mscmga.ms.ic.ac.uk/info.html.

[5] C. Bentz, M.-C. Costa, and A. Herty. On the Edge Capacitated Steiner Tree Problem. Discrete Optimization, 38:1–25, 2020.

[6] A. Brooke, D. Kendrick, and A. Meeraus. GAMS Release 2.25. A User's Guide. The Scientific Press. Boyd & Fraser Publishing Company, Massachussets, 1992.

[7] J.-F. Camacho-Vallejo and C. Garcia-Reyes. Co-Evolutionary Algorithms to Solve Hierarchized Steiner Tree Problems in Telecommunication Nnetworks. Applied Soft Computing Journal, 84:1–12, 2019.

[8] C.-Y. Chen and S.-Y. Hsieh. An Improved Algorithm for the Steiner Tree Problem with Bounded Edge-Length. Journal of Computer and System Sciences, 123:20–36, 2022.

[9] D. Du and X. Hu. Steiner Tree Problems in Computer Communication Networks. World Scientific, Singapore, 2008.

[10] D.-Z. Du, J. M. Smith, and J. H. Rubinstein. Advances in Steiner Trees. Kluwer Academic Publishers, Dordrecht, 2000. ISBN 0-7923- 6110-5.

[11] C. W. Duin and A. Volgenant. The Partial Sum Criterion for Steiner Trees in Graphs and Shortest Paths. European Journal of Operational Research, 97:172–182, 1997.

[12] D. Gaul and D. R. Schmidt. Chv´atal–Gomory Cuts for the Steiner Tree Problem. Discrete Applied Mathematics, 291:188–200, 2021.

[13] M.X. Goemans and Y. Myung. A Catalog of Steiner Tree Formulations. Networks, 23:19– 28, 1993.

[14] F. K. Hwang, D. S. Richards, and P. Winter. The Steiner Tree Problem. North-Holland, Amsterdam, 1992.

[15] S. Khuller. Design and Analysis of Algorithms. Lecture Notes, University of Maryland, Department of Computer Science, 1994. 112 pp.

[16] T. Koch and A. Martin. Solving Steiner Tree Problems in Graphs to Optimality. Networks, 32:207–232, 1998.

[17] L. Kou, G. Markowsky, and L. Berman. A Fast Algorithms for Steiner Trees. Acta Informatica, 15:141–145, 1981.

[18] Y.-C. Lin, H.-A. Chien, C.-C. Shih, and H.- M. Chen. A Multi-layer Obstacles-Avoiding Router Using X-Architecture. WSEAS Transactions on Circuits and Systems, 7:879–888, 2008.

[19] A. Lucena and J. E. Beasley. A Branch and Cut Algorithm for the Steiner Problem in Graphs. Networks, 31:39–59, 1998.

[20] J. Plesn´ık. Grafov´e algoritmy. VEDA, vydavate´lstvo Slovenskej akad´emie vied, Bratislava, 1983.

[21] J. Plesn´ık. Heuristics for the Steiner Problem in Graphs. Discrete Applied Mathematics, 37/38:451–463, 1992.

[22] H. J. Pr¨omel and A. Steger. The Steiner Tree Problem. A Tour through Graphs, Algorithms, and Complexity. Vieweg Verlag, Braunschweig, 2002.

[23] G. Skorobohatyj. Testsets. Report, Konrad-Zuse-Zentrum f¨ur Informationstechnik, Berlin, 2000. ftp://ftp.zib.de/pub/Packages /mp-testdata/index.html.

[24] H. Takahashi and A. Matsuyama. An Approximate Solution for the Steiner Problem in Graphs. Mathematica Japonica, 24(6):573–577, 1980.

[25] D. H. Wolpert and W. G. McReady. No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997.

[26] D. H. Wolpert and W. G. McReady. Coevolutionary Free Lunches. IEEE Transactions on Evolutionary Computation, 9(6):721–735, 2005.

[27] P. ˇSeda, M. ˇSeda, and J. Hoˇsek. On Mathematical Modelling of Automated Coverage Optimization in Wireless 5G and beyond Deployments. Applied Sciences, 10(24):1–25, 2020.