International Journal of Computers, Communications & Control
Vol. I (2006), No. 1, pp. 41-46

An Exact Algorithm for Steiner Tree Problem on Graphs

Milan Stanojević, Mirko Vujošević

Abstract: The paper presents a new original algorithm for solving Steiner tree problem on graph. The algorithm is simple and intended for solving problems with relatively low dimensions. It is based on use of existing open source software for solving integer linear programming problems. The algorithm is tested and shown very efficient for different randomly generated problems on graphs of up to 50 nodes, up to 10 terminals and average node degree 7.
Keywords: Steiner tree problem on graph, branch and cut, algorithm, optimization

1 Introduction

The Steiner tree problem (STP) is met in telecommunication and energetic systems, VLSI technologies and in other network planning tasks. The problem is to find a minimal length tree which connects all terminal nodes of a given graph, and contains arbitrary number of other nodes. The problem is similar to the well known shortest spanning tree problem (SSTP), but unlike that it doesn't necessary contain all nodes of given graph. Also, a very important difference is that STP is much harder problem than SSTP.

The usual formulation of STP is the following: a connected undirected graph $G = (N, E)$, where $N = \{1, \ldots, n\}$ is a set of nodes and $E \subseteq \{\{i, j\} \mid i \in N, j \in N, i < j\}$ denotes a set of edges, is given. A positive value (length, weight, etc.) c_e is associated to every edge $e \in E$. Also, a set $T \subset N$ of so called terminal (Steiner) nodes is given.

Definition 1. Steiner tree for T in G is a subgraph $St = (N', E')$, $N' \subseteq N$, $E' \subset E$ which satisfies one of the following statements:

1. $T \subseteq N'$ and St is a tree;
2. $\forall s, t \in T$ in St exists exactly one path from s to t.

The length of Steiner tree is the sum of lengths of all edges which it consists of. The Steiner tree problem is to find the shortest Steiner tree. This is a NP-hard problem and its decision problem variant belongs to NP-complete problem class [6]. The problem is presented in details in surveys: [4], [5], [9] and [11]. In papers [2], [8] and [7] the problem was solved to optimality using sophisticated branch and cut methods.

In this paper an original exact algorithm, generally based on branch and cut procedure, for solving Steiner tree problems on graph is proposed. It uses code from an available open source project which develops software for solving linear and integer linear programming problems. An intention was to formulate an algorithm which can be implemented in relatively short time and which will be able to solve STP of "reasonable" dimensions in "reasonable" time. To be more precise, the implementation of proposed algorithm lasted about 3-4 days. The program could solve problems of dimensions which are larger of the most of real life problems in, at most, several minutes.

In this paper, in the next section, a mathematical model of the Steiner tree problem and its explanation are given. In Section 3, the algorithm and the process of implementation are explained in details. In Section 4 the experiments are described and some conclusions about the algorithm behavior are made. In Section 5, the conclusion of the whole paper is given.

2 Mathematical model

A Steiner tree can be represented by a vector of binary variables $x = (x_e)_{|E|}$. Each element of the vector is assigned to one edge from the set E. The value of a variable x_e indicates whether the corresponding edge e is in the Steiner tree ($x_e = 1$) or not ($x_e = 0$). A mathematical model of the STP on undirected graphs has the following form:

$$\begin{array}{rlll} \min & \sum\limits_{e\in E} c_e x_e & & \\ \text{subject to} & & & \\ (i) & \sum\limits_{e\in\delta(M)} x_e \geq 1 & \forall & M \subset N, \\ & & & M \cap T \neq \emptyset, \\ & & & (N\backslash M) \cap T \neq \emptyset \\ (ii) & x_e \in \{0,1\} & \forall & e \in E \end{array} \tag{1}$$

where N, E and T are as in Definition 1, c_e is a positive value associated to each edge e and $\delta(M)$ denotes a graph cut defined by subset of nodes $M \subset N$, i.e. the set of edges with one end node in M and the second one in complement set $N \setminus M$, $\delta(M) = \{\{i,j\} \in E \mid i \in M,\, j \in N \setminus M\}$.

The mathematical model (1) is linear and integer. Constraints (i) ensure that in every cut, for which terminal nodes are on the both sides (M and $N\backslash M$), at least one edge exists. In other words, they ensure that between each two terminal nodes exists at least one path. A feasible solution of model (1) is not necessary a Steiner tree. But the optimal solution will be a Steiner tree because any edge constructing a contour would violate the optimality condition. So, although the formulation is rather comprehensive, it can be applied only for those problems where the goal is to minimize the length of the Steiner tree.

A disadvantage of model (1) is that the number of constraints grows exponentially in the problem size. On the other hand, in branch and cut methods, the relaxation of the formulation may give acceptable results.

3 Implementation

Two main challenges in solving the model (1) are: (i) the exponential number of constraints, and (ii) the exponential time needed to solve the integer program even with smaller number of constraints. To overcome the first challenge, the proposed algorithm uses a relaxation of model (1). The idea was inspired by the paper [7] and partly published at [10]. Namely, because of a big number of constraints, solving a model which includes all the constraints is very hard and in many cases impossible. On the other hand, in the most cases it is not necessary to include all the constraints in order to obtain an optimal solution. Only the constraints which are active in the optimal solution are really necessary. Of course, we cannot predict which of them will be active, but we can start with some smaller number of constraints, giving priority to those which are “more likely” to be necessary to obtain a feasible solution (Steiner tree). If we solve a model with smaller number of constraints and the solution is a Steiner tree of the given graph, then this solution will be the optimal for the starting problem, i.e. by adding more constraints we cannot improve the solution. Otherwise, the obtained solution will consist of two or more subtrees. Then, we iteratively add those constraints for which we find they are violated and which would probably lead to feasible solution until we finally obtain a Steiner tree.

The most common way to solve a binary linear programming problem is implementation of branch and bound method in combination with simplex method. According to the one of main features of the algorithm we intended to formulate – quick and easy implementation, development of simplex and branch and bound algorithms from a scratch wouldn’t be appropriate. Although it would finally give better performance if they would be incorporated in the essence of complete procedure, the development of these procedures would last very long. An alternative was found among Open Source projects. The project used in this implementation was **lp_solve**.

Lp_solve [1] is an open source project that realize very robust procedures and techniques for solving linear programming problems. Beside that, it implements the branch and bound method for solving binary, integer and mixed integer linear problems. It has a lot of options by which it is possible to influence the branch and bound procedure changing its strategies, so it is possible to significantly improve its performance [13]. Lp_solve can be used as independent application when it can read problem files in *lp* and *mps* formats, and as a set of functions, when it can be incorporated into other programs and controlled from the host code.

The project itself doesn’t have any restrictions of the problems dimensions. Some successful applications on mixed integer programming problems with several thousands variables were reported. The license of the project is GLGPL (GNU Lesser General Public License) [12] and it allows free download, using, changing and redistribution of the source code of lp_solve project.

The algorithm is formally formulated as follows:

Algorithm 1 Simplified branch and cut algorithm for STP

1. Formulation of initial integer linear mathematical model:

   (a) Goal function formulation: one variable is introduced for each edge and corresponding edge length is associated as a parameter to each variable.

   (b) For each terminal node, one constraint of type (i) is formulated, so the terminal node is a single node on one side of the cut and the rest of the nodes are on the other side, i.e. $\sum_{e\in\delta(M)} x_e \geq 1\ \forall M \in \{\{t\} \mid t \in T\}$.

   The number of constraints after Step 1 will be $|T|$.

2. Solve the current mathematical model.

3. Check if the obtained solution is a Steiner tree, i.e. if there is a path between all pairs of terminal nodes. If so, the optimal solution was found in Step 2; the end of the procedure. Otherwise, next step.

4. If solution is not a Steiner tree, then it represents two or more unconnected subtrees. For every subtree add one type (i) constraint defined by cut $\delta(M)$ where the nodes of that subtree belong to set M. Go to Step 2.

The proposed algorithm can be qualified as a simplified version of branch and cut method, i.e. a combination of branch and bound and cutting planes. The steps of the algorithm will be illustrated by an example. Suppose, we have to obtain the minimal Steiner tree for the given graph, illustrated in Figure 1, where four terminal nodes are marked with bigger circles.

The mathematical model created in Step 1 and updated in Step 4 will have one column for every edge. The constraints added in the first step are necessary to provide that every terminal is connected to, at least, one edge. In the example, four constraints will be added in the first step – one for every terminal. They will ensure that at least one edge is connected to each terminal. In Figure 2, the four cuts are marked as open curves surrounding each terminal node, and the edges marked by dashed lines are candidates to be in the first solution.

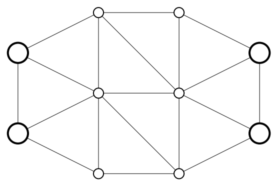
Figure 1: Initial graph

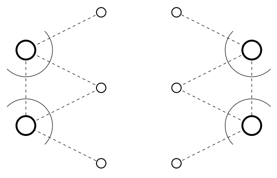
Figure 2: First step

The possible solution after Step 2 could be like one in Figure 3. As mentioned above, if the obtained solution is not a Steiner tree, it will consist of several subtrees. The number of subtrees generally can be between 2 and $|T|$. For determining if the obtained solution is a Steiner tree (in Step 3), Dijkstra's shortest path algorithm was used.

In Step 4 a new constraint for each subtree is added. In the example, three new constraints, corresponding to three subtrees shown in Figure 3, are added. On each image of Figure 4, one cut (represented by the curve surrounding the subtree) and edges (dashed lines) among which, at least one will be in the next solution are shown.

After the next optimization, a possible solution could be like the one shown in Figure 5. The solution satisfies all added constraints and the graph structure is a Steiner tree. Without further checking, we can claim that it is an

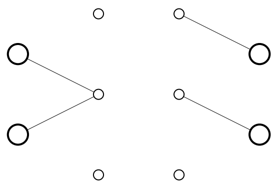
Figure 3: Possible solution after the first iteration

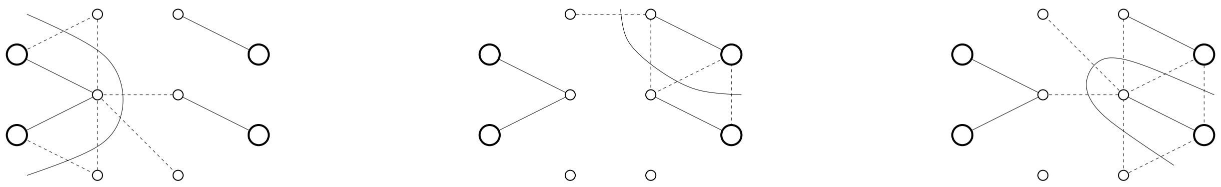
Figure 4: Three new constraints

optimal solution of the given Steiner tree problem.

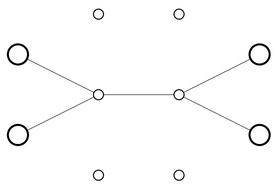
Figure 5: Final – feasible and optimal solution

The main criteria for determining data structures were access speed and simplicity of implementation. The amount of used memory was not considered because of relatively small graph dimensions of target instances. The logical choice were static structures (vectors and matrices). The realized structures enabled fast data access and mapping between graph structure (realized through neighborhood matrix) and vector of edges with all corresponding attributes.

The algorithm was implemented in C language and it is compiled and tested on Linux operation system (with gcc - GNU C Compiler) on PC with Pentium® III processor on 600 MHz and 384 MB of RAM. The implemented program supports so called *STP* format [15] of Steiner tree problems, and it is compatible with the library of standard Steiner tree problems [14].

4 Experimental results

The developed program was tested on different problem instances. A characteristic of the Steiner tree problem is that the complexity of a procedure for its solving depends on three attributes: number of nodes (n), number of edges and number of terminal nodes (t). The largest dimensions of instances successfully solved by the program were [n/t]: 20/10, 32/8 and 50/5, with the average node degree 7. The solving procedures lasted between 2 seconds and 2 minutes for the most of the instances. These dimensions may look modest in comparison with those with several thousands nodes which were successfully solved as it was reported in papers [2] and [7]. However, comparing the procedure complexity, simplicity of implementation and the fact that in many real-life telecommunication planning processes, even smaller size problems may appear, the proposed implementation could be very useful.

To get a more precise insight in the behavior of the algorithm, experiments were performed with instances where some parameters were varied. In the following table, results (mean values and standard deviation) obtained by experiments where every dimension was tested on 15 randomly generated instances are given. In the columns named “No. of rows” the number of constraints needed to obtain an optimal solution (the optimization in the last iteration) is given. The columns named “No. of iterations” represent a number of iterations of Algorithm 1 when passing through Steps 2-4, i.e. the total number of solved binary subproblems (in Step 2). The columns "Time" represent CPU time spent to solve the problem. The column "ANCAI" shows the average number of constraints added per iteration. It's obtained by formula: $\frac{No.rows-t}{No.iterations}$ where t represents the number of terminal nodes, i.e. the number of constraints added in initial mathematical model in Step 1.

Table 1: The complexity analysis of the algorithm

Instance dimensions		No. of rows		No. of iterations		Time [sec.]		AN-CAI
$[n/t]$		$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	
1		2	3	4	5	6	7	8
1	20/5	25	8.9	7.5	3.0	0.2	0.3	2.7
2	32/5	38	22.1	12.3	8.0	3.0	9.7	2.7
3	50/5	60	28.3	20.0	11.3	7.2	13.3	2.8
4	20/8	65	20.3	19.5	8.5	11.5	27.5	2.9
5	32/8	88	31.9	26.3	12.0	34.5	46.7	3.0
6	20/10	83	15.5	23.6	5.2	13.4	9.9	3.1

On the basis of data given in Table 1, some conclusions can be made. The fact that complexity of STP grows with the number of nodes is obvious from the first three rows. Although the speed of the growth cannot be determined exactly on the basis of so small sample, it is obviously nonlinear – probably exponential. The more interesting conclusion is that the growth of complexity is faster by changing the number of terminal nodes than the total number of nodes in graph. Comparing rows 1, 4 and 6, a kind of "explosion" of complexity can be seen: when the number of terminals was increased two times, execution time was increased 67 times. Similar conclusion can be made observing rows 2 and 5: addition of three terminals resulted in execution time increase of more than 11 times.

The most important analysis here concerns the number of constraints needed to get an optimal solution. According the Column 2, that growth is almost linear in problem size. We cannot be certain if it is linear, but it is definitely not exponential. Finally, we can conclude that, although the number of constraints in model (1) grows exponentially, the number of constraints necessary to obtain an optimal solution grows much slowlier. The number of iterations grows even less. The parameter in Column 8 is also interesting. The average number of constraints added in each iteration also represents the average number of subtrees obtained in each sub solution. It seems that the relatively small values in Column 8 doesn't depend much on the number of terminal nodes. The explanation could be that the current solution of the solving procedure relatively quickly forms a structure which consists of a few subtrees each containing several terminals. This may contradict to the previous statement that complexity depends more on the number of terminals than on the total number of nodes, because, what influences the number of iterations is the number of unconnected subtrees, and not the number of terminals. One possible explanation is that the structure of subtrees changes, so subtrees contain different terminals in different iterations. It is also important to have in mind that every iteration lasts more than the previous one because every mathematical model has more rows (constraints) than the previous one.

Yet another interesting thing from Table 1 is relatively big dispersion (represented by standard deviation) of data obtained by different randomly generated instances with same characteristics. This is a consequence of a nature of the algorithm (which is nondeterministic polynomial). It is impossible to predict the number of iterations necessary to obtain final solution. In the worst case it can be exponential.

5 Conclusion

The first impressions and experiment conclusions indicate that the proposed algorithm can be efficiently used when there is a need for rapid development of an algorithm for solving smaller size Steiner tree problems. Although the worst case number of constraints needed to obtain a final solution remains exponential, the algorithm have shown a kind of "good behavior" – in all solved examples the number stayed relatively low. Although the exponential complexity of the branch and bound method (Step 2 of the algorithm) remains, instances with acceptable dimensions can be solved in real time.

The idea of successive adding violated constraints could be also applied to some other problems whose mathematical models have an exponential number of constraints. Concerning that, some new researches have been planed, where the idea would be applied to the traveling salesman problem (TSP). Namely, the so called DFJ formulation of TSP [3] also has exponential number of constraints, but it has shown a good behavior in relaxation based algorithms.

References

[1] M. Berkelaar, K. Eikland, P. Notebaert, Lp_solve, Files and Discussion Group, ftp://ftp.es.ele.tue.nl/pub/lp_solve, http://groups.yahoo.com/group/lp_solve/, 1994-2006.

[2] S. Chopra, E. Gorres, M. R. Rao, "Solving a Steiner tree problem on a graph using branch and cut", *ORSA Journal on Computing*, Vol. 4, pp. 320-335, 1992.

[3] G. B. Dantzig, D. R. Fulkerson, S. M. Johnson, "Solution of a large-scale traveling-salesman problem", *Operations Research*, Vol. 2, pp. 393-410, 1954.

[4] F. K. Hwang, D. S. Richards, "Steiner tree problems", *Networks*, Vol. 22, pp. 55-89, 1992.

[5] F. K. Hwang, D. S. Richards, P. Winter, *The Steiner tree problem*, North-Holland, Amsterdam, 1992.

[6] R. M. Karp, "Reducibility among combinatorial problems", R. E. Miller, J. W. Thatcher (Ed.), *Complexity of Computer Computations*, pp. 85-103, Plenum Press, New York, 1972.

[7] T. Koch, A. Martin, "Solving Steiner tree problems in graphs to optimality", *Networks*, Vol. 32, pp. 207-232, 1998.

[8] A. Lucena, J.E. Beasley, "A branch and cut algorithm for the Steiner problem in graphs", *Networks*, Vol. 31, pp. 39-59, 1998.

[9] N. Maculan, "The Steiner tree problem in graphs", Surveys in Combinatorial Optimization, S. Martello, G. Laporte, M. Minoux, C. C. Ribeiro (Ed.), *Annals of Discrete Mathematics*, Vol. 31, pp. 185-212, 1987.

[10] M. Stanojević, M. Vujošević, "A new algorithm for solving Steiner tree problem on graph" (in Serbian), 12th Telecommunications Forum TELFOR 2004, Belgrade, http://www.telfor.org.yu/telfor2004/e-index.html (http://www.telfor.org.yu/telfor2004/radovi/TM-2-4.pdf), 2004.

[11] P. Winter, "Steiner problem in networks: A survey", *Networks*, Vol. 17, pp. 129-167, 1987.

[12] GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html

[13] Lp_solve Reference Guide, http://www.geocities.com/lpsolve/

[14] SteinLib TestSets – The Library of Standard Steiner Problems, http://elib.zib.de/steinlib/testset.php

[15] STP – Description of the STP Data Format, http://elib.zib.de/steinlib/format.php

Milan Stanojević, Mirko Vujošević
University of Belgrade
Faculty of Organizational Sciences
Address: Jove Ilića 154, Belgrade, Serbia and Montenegro
E-mail: {milans,mirkov}@fon.bg.ac.yu