RC 21846 (98327) 29 September 2000 Computer Science/Mathematics

IBM Research Report

Solving Steiner Tree Problems in Graphs with Lagrangian Relaxation

Laura Bahiense
Univ. Federal do Rio de Janeiro
P.O. Box 68511
Rio de Janeiro
RJ 21945-970, Brazil

Francisco Barahona
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Oscar Porto
PUC-Rio
Predio Cardeal Leme, Sala 401
CEP 22453-900
Rio de Janeiro, RJ, Brazil

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

SOLVING STEINER TREE PROBLEMS IN GRAPHS WITH LAGRANGIAN RELAXATION

LAURA BAHIENSE, FRANCISCO BARAHONA, AND OSCAR PORTO

ABSTRACT. This paper presents an algorithm to obtain near optimal solutions for the Steiner tree problem in graphs. It is based on a Lagrangian relaxation of a multicommodity flow formulation of the problem. An extension of the subgradient algorithm, the *volume algorithm*, has been used to obtain lower bounds and to estimate primal solutions. It was possible to solve several difficult instances from the literature to proven optimality without branching. Computational results are reported for problems drawn from the *SteinLib* library.

1. INTRODUCTION

Given an undirected graph $G = (V, E)$ and a subset of the nodes $T \subseteq V$ called *terminals*, a *Steiner tree* for T in G is a tree that spans T. Let c_{ij}, for each $(i, j) \in E$, be nonnegative *costs* associated to the edges of G. The *Steiner tree problem in graphs* (STPG) asks for the Steiner tree of minimum total edge cost. The vertices in $V \backslash T$ are called *non-terminals*. Non-terminal vertices that end up in an optimum Steiner tree are called *Steiner vertices*. The STPG is known to be *NP*-Hard [21] for a general graph, remaining *NP*-Hard for particular classes of graphs such as grid graphs [14], among others. The STPG has been widely applied in the design of communication, distribution and transportation systems. It is also applied to problems such as the wire routing phase in physical VLSI design [24], network design [28], etc.

Different integer programming formulations of the STPG can be found in the works of Maculan [11, 27] and Goemans & Myung [15]. In the latter, the equivalence between some of the given formulations is shown. A variety of solution methods have been proposed including: branch-and-cut [8, 9, 10, 26, 22]; dual ascent [37]; Lagrangian relaxation [6, 5], Lagrangian relaxation and polyhedral methods [25]. Stand alone heuristics for the STPG have also been developed; see for example [32, 33, 35, 13]. Due to its outstanding performance, the one developed in [32] by Takahashi & Matsuyama is the basis of several meta-heuristics and primal heuristics developed for the STPG.

Many reduction techniques for the STPG have been developed, several of them can be found in [22]. A good reference is the book [19]. Very good surveys are also given in [34, 19, 27, 20].

This paper presents a new algorithm to obtain near optimal solutions for the STPG. This algorithm is based on a multicommodity flow formulation. Due to the large size of the model obtained, it is tackled with Lagrangian relaxation, and the *Volume Algorithm* (VA) is used in the relaxed problem. The VA is an extension of the subgradient algorithm that produces primal as well as dual solutions. The primal vectors are used as inputs for several heuristics that construct integer solutions. It was possible to solve several difficult instances from the literature to proven optimality, without branching.

Computational results are reported for problems drawn from the *SteinLib* library.

This paper is organized as follows. Section 2 describes the integer programming formulation of the STPG used in this work. Dual bounds and the VA are presented in Section 3. A comparison between the VA and a modified subgradient method is shown in Section 4. Section 5 describes the primal heuristics used in this work. Section 6 provides computational results. Section 7 contains conclusions.

2. Formulation

In this section, two formulations of the STPG are discussed. As mentioned above, the STPG is defined on an undirected graph. A directed version of the problem, the Steiner arborescence problem, is defined as follows. Given a digraph $D = (V, A)$, a set of terminals $T \subseteq V$ and a root vertex $r \in T$, a *Steiner arborescence* is a tree directed away from the root r spanning T. The *Steiner arborescence problem* (SAP) asks for the minimum total arc cost Steiner arborescence. Any instance of the STPG can be formulated as a bidirected SAP (see [9] for properties of both formulations).

In this study, the bidirected SAP is formulated as a *non-simultaneous multi-commodity flow problem* in a digraph $D = (V, A)$. This directed graph has the same set of vertices of the original graph $G = (V, E)$ and its set of arcs A is obtained as follows: for each edge $(i, j) \in E$, two arcs, $(i,\vec{j})$ and $(j,\vec{i})$, both with the same weight c_{ij} of edge (i, j), are included in A.

The formulation below was simultaneously introduced by Claus & Maculan [11] and Wong [37]. A vertex $r \in T$ is chosen to be the *source* offering $|T_0| := |T \backslash \{r\}|$ commodities, one demanded by each of the remaining $|T_0|$ terminal vertices. Variables f_{ij}^k, $(i,\vec{j}) \in A$ and $k = 1, \ldots, T_0$, indicate the flow amount of commodity k going through the arc $(i,\vec{j})$. Binary variables x_{ij}, $(i,\vec{j}) \in A$, control the inclusion ($x_{ij} = 1$) or not ($x_{ij} = 0$) of arc $(i,\vec{j})$ in the solution. Naming $I(i)$ the set of vertices $j \in V$ such that $(j,\vec{i}) \in A$ and $O(i)$ the set of vertices $j \in V$ such that $(i,\vec{j}) \in A$, the SAP can be formulated as follows:

$$
(1) \quad \left\{
\begin{array}{ll}
\min\limits_{x:=(x_{ij})_{(i,\vec{j})\in A}} & cx \\
\displaystyle\sum_{j\in O(r)} f_{rj}^k - \sum_{j\in I(r)} f_{jr}^k = 1\,, \quad k \in T_0 & (a_1) \\
\displaystyle\sum_{j\in O(k)} f_{kj}^k - \sum_{j\in I(k)} f_{jk}^k = -1\,, \quad k \in T_0 & (a_2) \\
\displaystyle\sum_{j\in O(i)} f_{ij}^k - \sum_{j\in I(i)} f_{ji}^k = 0\,, \quad i \in V \setminus \{r,k\}, \quad k \in T_0 & (a_3) \\
x_{ij} \in \{0,1\}\,, \quad (i,\vec{j}) \in A\,. & (a_4) \\
0 \le f_{ij}^k \le x_{ij}\,, \quad (i,\vec{j}) \in A\,, \quad k \in T_0\,. & (a_5)
\end{array}
\right.
$$

Equations $(a_1), (a_2)$ and (a_3) represent flow conservation for the terminal vertex chosen to be the source, for the remaining terminal vertices and for the non-terminal vertices,

respectively. Constraints (a_5) allow a non-zero flow f_{ij}^k of any commodity k through an arc $(i,\vec{j})$ only if the latter is included in the solution. Finally, in the objective function, the total sum of the costs for the arcs included in the solution (a Steiner tree) is minimized.

The last part of this section analyzes the relationship between the continuous relaxations of two different integer programming formulations for the SAP, the one used in this work and the classical dicut formulation for the problem. Notice that the formulation above has a polynomial number of constraints and a polynomial number of variables. Let P_{xf} be the polytope defined by $(a_1), (a_2), (a_3)$, the continuous relaxation of (a_4) and (a_5). It can be shown (see [27]) that the polytope P_x obtained by projecting P_{xf} onto the x variables can be characterized as follows:

$$P_x = \{x : x(\delta^+(S)) \geq 1, \text{for all} S \subset V \text{with} r \in S \text{and} (V \backslash S) \cap T \neq \emptyset;\ 0 \leq x_{ij} \leq 1,\ (i,\vec{j}) \in A\}$$

with $\delta^+(S) := \{(i,\vec{j}) \in A : i \in S, j \in V \backslash S\}$ and $x(\delta^+(S)) := \sum_{(i,\vec{j}) \in \delta^+(S)} x_{ij}$.

The *dicut formulation* for the SAP, based on the polytope P_x and proposed first in [1], is as follows:

$$(2) \quad \begin{cases} \min\limits_{x:=(x_{ij})_{(i,\vec{j}) \in A}} & cx \\ x(\delta^+(S)) \geq 1\ , \quad \text{for all } S \subset V\ ,\ r \in S\ ,\ (V \backslash S) \cap T \neq \emptyset & (a_6) \\ x_{ij} \in \{0,1\}\ , \quad (i,\vec{j}) \in A \end{cases}$$

Since P_x is the projection onto the x variables of P_{xf}, the continuous relaxations of (1) and (2) are equivalent. It is important to mention that even with an exponential number of constraints, the continuous relaxation of the dicut formulation for the SAP can be solved in polynomial time with the ellipsoid algorithm. The reason for that is the polynomiality of the separation problem over the *Steiner dicut inequalities* (a_6); it can be tackled by solving a maximum flow problem for each terminal vertex in the set T_0. The dicut formulation for the SAP is the one most used in the literature. Examples include the branch and cut algorithms presented in [8, 22]. The latter combine a minimum cut algorithm and the Simplex method, used iteratively. Due to its size, formulation (1) has been mostly ignored. To our knowledge, only Wong [37] has used it with a dual ascent method. He reports results on graphs with up to 60 nodes and 120 edges.

In this paper, Lagrangian relaxation is used to tackle (1), and subgradient techniques are used in such a way that at every iteration the resulting subproblem is trivial to solve. A difference that should be stressed is that the algorithm presented in this study does not require the use of the Simplex method at any stage.

3. Dual Bounds

This section describes how lower bounds for the SAP are computed. Flow conservation constraints (a_1)-(a_3) of (1) are dualized in a Lagrangian way with multipliers π_i^k, for $i \in V$ and $1 \leq k \leq |T_0|$. Let $\pi = (\pi_i^k) \in \mathbb{R}^{|V||T_0|}$ be the resulting vector of Lagrangian multipliers. Finally the integrality constraints (a_4) are linearly relaxed and the following *Lagrangian subproblem* is obtained:

$$(3) \qquad (\theta(\pi)) \begin{cases} \min\limits_{\substack{x:=(x_{ij})_{(i,\vec{j})\in A} \\ f:=(f_{ij})_{(i,\vec{j})\in A}}} cx + \sum\limits_{k\in T_0} \ell^k f^k + \xi(\pi) \\ x_{ij} \in [0,1]\,, \quad (i,\vec{j}) \in A\,. \qquad (a_4') \\ 0 \le f_{ij}^k \le x_{ij}\,, \quad (i,\vec{j}) \in A\,, \quad k \in T_0\,, \quad (a_5) \end{cases}$$

where $\ell^k := (\ell_{ij}^k) = (\pi_i^k - \pi_j^k)$ is the vector of *Lagrangian costs* of variables f_{ij}^k and $\xi(\pi)$ is a constant factor easily computed for any fix π. For any value of π, the solution of $(\theta(\pi))$ gives a valid lower bound on the optimal solution of (1). The best of these bounds is given by a solution of the following Lagrangian dual problem:

$$(4) \qquad \max_{\pi\in\mathbb{R}^{|V||T_0|}} \theta(\pi)$$

Subsections 3.2 and 3.3 show how the volume algorithm is used to solve the Lagrangian dual problem (4). As mentioned before, special care has been taken to use Lagrangian relaxation and subgradient techniques in such a way that the resulting Lagrangian subproblems are trivial to solve. The next subsection describes a simple inspection algorithm that solves the Lagrangian subproblems.

3.1. **Solving the Lagrangian subproblem.** For any fixed vector of Lagrangian multipliers π, the resulting Lagrangian subproblem (3) decomposes into a set of independent problems, one for each arc $(i,\vec{j}) \in A$. After dropping the indices i, j the solution is:

- If $\sum\limits_{k:\,\ell^k<0} |\ell^k| > c$, then:
  - $x = 1$,
  - $f^k = 1$, for every k such that $\ell^k < 0$,
  - $f^k = 0$, for every k such that $\ell^k \ge 0$;
- If $\sum\limits_{k:\,\ell^k<0} |\ell^k| \le c$, then:
  - $x = 0$,
  - $f^k = 0$, for every k.

The rest of this section is dedicated to the solution of the Lagrangian dual problem (4).

3.2. **Solving the Lagrangian dual problem.** A similar relaxation has been used in [18], where the subgradient method [16] is used in the traditional way. Since this only produces dual variables, branching decisions are based on this dual information. In this paper the Lagrangian dual problem (4) is solved with the volume algorithm. This is an extension of the subgradient method that produces primal solutions as well as dual solutions, see [3]. It can be seen as a fast way to approximate Dantzig-Wolfe decomposition [12]. The name "volume" is inspired by the fact that the primal values come from computing the volumes below the faces of the dual problem. See [2] for a

study of its convergence. A similar approach for uncapacitated facility location problems appears in [4]. A description of the algorithm is below.

Volume algorithm (VA)

STEP 0. [*Initialization*]

Let $\pi^* \in \mathbb{R}^m$ be a vector of Lagrangian multipliers.
Solve (3) with $\pi := \pi^*$. Let $(\hat{x}_0, \hat{f}_0)$ be an optimal solution of (3), and z^* the value of the objective function. Set $t := 0$ and $(\bar{x}_0, \bar{f}_0) := (\hat{x}_0, \hat{f}_0)$.

STEP 1a. [*"Supergradient" Displacement*]

Let $\bar{v}_t = b - A\bar{f}_t$, where $Af = b$ is the system (a_1), (a_2), (a_3).
Perform a *''supergradient'' displacement*:

$$\bar{\pi} := \pi^* + s_t \bar{v}_t ,$$

where the step size s_t is given by (7).

STEP 1b. [*Solving the Subproblem*]

Solve (3) with $\pi := \bar{\pi}$.
Let: $(\hat{x}_{t+1}, \hat{f}_{t+1})$ be an optimal solution of (3), and z_{t+1} the value of the objective function.

STEP 2a. [*Primal Update*]

Compute

$$(\bar{x}_{t+1}, \bar{f}_{t+1}) := \alpha\,(\hat{x}_{t+1}, \hat{f}_{t+1}) \;+\; (1-\alpha)\,(\bar{x}_t, \bar{f}_t)\,. \tag{5}$$

The value of α is discussed below.

STEP 3. [*Dual-Test*]

Perform the *Dual-Test*:

$$\text{if } z_{t+1} > z^*, \text{ then update } z^* := z_{t+1},\ \pi^* := \bar{\pi}. \tag{6}$$

STEP 4. [*Loop*]

Check stopping criteria. Do $t := t+1$ and go to STEP 1a. □

The step size is given by

$$\lambda \frac{T - z^*}{\|\bar{v}_t\|^2}, \tag{7}$$

where $0 < \lambda < 2$, and T is a *target* value. A small value of T is used at the beginning and each time that z^* is within 5% of T, the value of T is increased by 5%. In order to set the value of lambda, three types of iterations are defined:

- If there is no improvement in the value of z^*, the iteration is called *red.* A sequence of red iterations suggest the need for a smaller step-size.
- If $z_{t+1} > z^*$ the following number is computed:

  $$a = \bar{v}_t \cdot (b - A\hat{f}_t).$$

  If $a < 0$, it means that a longer step-size would have given a smaller value for z_{t+1}. This iteration is called *yellow.*
- If $a \geq 0$, this iteration is called *green.* A green iteration suggests the need for a larger step-size.

The initial value of λ was 0.1. After a sequence of 20 consecutive red iterations and if $\lambda > 10^{-4}$, the value of λ is multiplied by 0.67. After each green iteration and if $\lambda < 1$, this value is multiplied by 2. Extensive experimental work has been done in [3, 4] to determine good values for these parameters.

To chose the value of α, an upper bound $\alpha_{\max} = 0.001$ is set. Then the following value is computed:

$$\gamma_{\text{opt}} := \underset{\gamma \in \mathbb{R}}{\text{Argmin}} \|\gamma \hat{v}_{t+1} + (1-\gamma)\bar{v}_{t+1}\|^2 .$$

Then if $\gamma_{\text{opt}} < 0$ the value of α is given by $\alpha := \frac{\alpha_{\max}}{10}$; otherwise $\alpha := \min\{\gamma_{\text{opt}}, \alpha_{\max}\}$. After every 250 iterations, the progress in the objective value is monitored. If the increase is less than 1% and $\alpha_{\max} > 10^{-5}$, then $\alpha_{\max}$ is divided by 2. This idea was first used in conjugate subgradient methods (see [23, 36]). However, one difference here is the use of $\alpha_{\max}$, and another important difference is that this is used to produce primal solutions.

As can be seen in step [*Dual-Test*] the dual vector π^* is updated only when a dual improvement has been obtained. In a regular subgradient method the update is done at every iteration. In the VA, the multiplier π^* plays the same role as the *stability center* in *bundle methods* [17].

Note that in [***Primal Update***], $(\bar{x}_{t+1}, \bar{f}_{t+1}) := \alpha(\hat{x}_{t+1}, \hat{f}_{t+1}) + (1-\alpha)(\bar{x}_t, \bar{f}_t) = \alpha(\hat{x}_{t+1}, \hat{f}_{t+1}) + (1-\alpha)\alpha(\hat{x}_t, \hat{f}_t) + \ldots + (1-\alpha)^{t+1}(\hat{x}_0, \hat{f}_0)$. This should be seen as a convex combination of $\{(\hat{x}_0, \hat{f}_0), \ldots, (\hat{x}_{t+1}, \hat{f}_{t+1})\}$. The assumption that this sequence approximates an optimal solution of the continuous relaxation of (1) is based on a theorem in linear programming duality that appears in [3]. Due to the exponential decrease of the coefficients of this convex combination, later vectors receive a much larger weight than those that appeared only in earlier iterations.

Three stopping criteria were used, and they are as follows:

1. *Linear-Optimality.* If the maximum absolute value of the components of $b - A\hat{f}$ was less than 0.001 and
$$\frac{|z^* - c\bar{x}_t|}{z^*} < 0.001.$$
2. *Integer-Optimality.* Under the assumption that all costs are integer, if the difference between the best upper bound and z^* is less than 1.
3. *Time.* If the CPU time was greater than 10000 seconds.

4. Comparison with a Subgradient algorithm

A variation of the Subgradient algorithm that has been very effective in practice was proposed by Camerini et al. [7]. Instead of just using the subgradient direction

$v_t = b - A\hat{f}_t$ at iteration t, they suggest $d_t = v_t + \beta_t d_{t-1}$. The scalar β_t is given by

$$\beta_t = \begin{cases} -\gamma\, v_t \cdot d_{t-1}/\|d_{t-1}\|^2 & \text{if } v_t \cdot d_{t-1} < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Here $0 \leq \gamma \leq 2$, the value $\gamma = 1.5$ is suggested in [7]. This is a way of avoiding the "zig-zag" behaviour of the subgradient method. The modified subgradient algorithm is as follows.

STEP 0.
Let $\pi^0 \in \mathbb{R}^m$ be a vector of Lagrangian multipliers. Set $d_{-1} = 0$, $t := 0$.
STEP 1.
Solve (3) with $\pi := \pi^t$. Let $(\hat{x}_t, \hat{f}_t)$ be an optimal solution of (3).
Let $v_t = b - A\hat{f}_t$, and $d_t = v_t + \beta_t d_{t-1}$.
Set

$$\pi^{t+1} := \pi^t + s_t d_t\,,$$

where the step size s_t is given by (7).
STEP 2.
Check stopping criteria. Do $t := t + 1$ and go to STEP 1

We modified our code to implement this algorithm. The stopping criterium was a limit of 10000 secs. The same formula (7) was used to define the step-size and the value λ was updated as in Section 3. In order to conduct some comparisons we chose a small set of instances from those treated in Section 6. All of them took larger computing times in [22] relative to others of the same type. Table 1 below shows the lower bounds produced by the VA and by the modified subgradient method. We list the name, then the number of nodes, edges and terminals, and the lower bounds produced by the VA and the subgradient method. These instances are studied in more detail in Section 6. As one can see, the lower bound given by the VA was tighter in all cases.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Volume	Subgradient
c18	465	1361	80	112.10	85.58
c19	433	1057	116	145.01	100.52
d19	901	2453	234	308.39	213.67
d20	379	684	226	536.02	177.306
dmxa0368	1926	3532	18	1016.99	605.18
dmxa0454	1707	3101	16	913.05	668.78
msm0654	1131	2062	10	822.99	427.52
msm2152	1938	3453	37	1589.63	1015.72
es40a	1169	2268	39	44837297.00	18080701.00
es40b	1123	2173	39	46806023.00	12152722.00

TABLE 1. Comparison between Volume and Subgradient.

5. Upper Bounds

Let $(\hat{x}_t, \hat{f}_t)$ be the primal approximation of a solution of the linear programming relaxation of (1) computed in the STEP 2A of VA. The idea in this section is to use this primal information in order to derive an upper bound (an integer feasible point) to the problem.

Denote by $\hat{x}$ the vector $\hat{x}_t$. For every edge $(i,j) \in G(V,E)$ set $\hat{y}_{ij} = \hat{x}_{ij} + \hat{x}_{ji}$. Three heuristics for finding Steiner trees are presented below.

I. Minimum spanning tree with volumetric weights (MSTV).

- Find a minimum spanning tree $MSTV$ in the graph $G = (V,E)$ with arc weights:
  - c_{ij} , when $t = 0$;
  - $-(\hat{y}_{ij})$, when $t > 0$.
- Prune all non-terminal leaves of $MSTV$.
- Find a minimum spanning tree MST in the subgraph induced by the vertices remaining in $MSTV$ after the pruning, considering the original costs c_{ij} as weights.
- Prune all non-terminal leaves of MST.

II. Minimum spanning tree in a modified graph (MSTM).

The second heuristic is applied in a subgraph of $G(V,E)$. For a given $\beta, 0 \leq \beta < 1$, let $G_\beta = (V_\beta, E_\beta)$ be the graph induced by the set of vertices V_β consisting of all terminal vertices and the nonterminal vertices i such that

$$\sum_j \hat{y}_{ij} \geq \beta ;$$

.

The second heuristic is as follows:

- Find the largest value of $\beta \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$, for which the graph $G_\beta(V_\beta, E_\beta)$ obtained is connected.
- Find a minimum spanning tree $MSTM$ in the graph $G_\beta = (V_\beta, E_\beta)$ with arc weights:
  - c_{ij} , when $t = 0$;
  - $(1 - (\hat{y}_{ij})\, c_{ij}$, when $t > 0$.
- Prune all non-terminal leaves of $MSTM$.
- Find a minimum spanning tree MST in the graph induced by the vertices remaining in $MSTM$ after the pruning, considering the original costs c_{ij} as weights.
- Prune all non-terminal leaves of MST.

III. Takahashi & Matsuyama heuristic with volumetric weights (T&MV).

Given a graph $G = (V, E)$ with nonnegative arc costs c_{ij}, for each $(i, j) \in E$, the *Takahashi & Matsuyama* heuristic [32] is as follows:

1. Chose an initial terminal vertex v_i; let $k = 1$;
2. Connect by the shortest path v_i with the terminal vertex v_j, $j \neq i$, that is closest to v_i; let T_1 be the subtree obtained;
3. Connect by the shortest path the subtree T_k with the terminal vertex v_l, $l \notin T_k$, that is closest to T_k; let T_{k+1} be the subtree obtained;
4. Stop if all terminals are connected; otherwise do $k \leftarrow k + 1$ and go to step 3.

The third heuristic is below:

- Given a starting vertex v_i, run the *Takahashi & Matsuyama* heuristic for the digraph $D = (V, A)$ with arc weights:
  - $c_{\vec{ij}} = c_{\vec{ji}} = c_{ij}$, when $t = 0$;
  - $c_{\vec{ij}} = c_{\vec{ji}} = (1 - \hat{y}_{ij})c_{ij}$, when $t > 0$.
- Find a minimum spanning tree in the subgraph induced by the vertices included in the tree obtained in the previous step, considering original costs c_{ij} as weights.
- Prune all non-terminal leaves.

The first two heuristics are fast, so they are run every 50 iterations of the VA. The third heuristic is similar to the one used in [22], it takes longer computing time but in most cases gives better solutions. This is run every 300 iterations of the VA. All of them use the primal approximation as an input.

6. Computational Results

This section presents extensive computational experience. The code was implemented in C++. The instances were obtained from the *SteinLib* library, available at `http://elib.zib.de/steinlib/steinlib.php`.

This set consists of the following series:

- C, D e E: tests cases introduced by J. Beasley [5]. They are defined on sparse random graphs with Euclidean costs.
- MC: Instances created by Francois Margot.
- Instances in complete graphs.
- P: tests cases introduced by E. Gorres.

- R: test cases introduced by J. Soukup and W. F. Chow [31], defined on a grid with no holes.
- DIW, TAQ, DMXA, MSM, GAP, ALUE, ALUT: cases derived from VLSI circuits introduced by T. Koch e A. Martin [22]. They are defined on grids with holes.
- ES: rectilinear problems.

The appendix contains tables for all cases treated. Each line corresponds to an instance. The first column gives the name of the instance and the next three columns give the number of nodes, edges and terminals respectively. We have used the preprocessing procedure of [22] with the default values. We report the size of the instances *after* preprocessing. The next column reports the time in seconds. The sixth column (LB) gives the lower bound. The next column (UB) contains the best upper bound found by the primal heuristics. The eighth column (GAP %) gives the gap in percentage defined by $100(UB - LB)/LB$. The ninth column (Vol $\times$ KM) compares performance between the algorithm proposed in this paper and the one presented in [22]. The comparison is based on the bounds produced as follows:

- $*^+$ means that optimality was proved by this algorithm and not by [22];
- $*^-$ means that optimality was proved by [22]; and not in the present paper;
- $+$ means that neither algorithm proved optimality, but the one in this paper gave a smaller gap.
- $-$ means that neither algorithm proved optimality, and the algorithm of [22] produced a smaller gap.
- $\sim$ implies that both algorithms gave similar results: either both proved optimality, or both obtained the same gap.

The tenth column displays the computing reported in [22]. We used a workstation RS-6000/640e with a 332 MHz processor. A time limit of 10000 secs. was set for all cases. In [22] a Sun SPARC 20 Model 71 was used, with also a limit of 10000 secs. Since these are different computers it is not possible to do have a direct comparison of CPU times, however we give approximate comparisons below.

Table 2 presents our results with Series C and D. All problems were solved to optimality with the exception of one whose gap from optimality was 0.8 %. Our computing times are larger than those reported in [22].

Table 3 deals with Series E. From a total of 20 instances, our method could prove optimality in 11 cases, they all had less than 200 terminals. Our computing times are higher than those reported in [22]. Two interesting exceptions were e16 and e17 that have 5 and 10 terminals respectively. Our times were 116 and 251 secs., and the times reported in [22] were about 1500 secs.

The series MC is shown in Table 4. Our code proved optimality in 3 out of 6 cases. In the remaining cases the largest gap was 3.2%. Optimality was proved for all cases in [22]. Our computing times are larger than those from [22].

Table 5 displays three cases in complete graphs. We could prove optimality in all of them, with computing times larger than those from [22].

Series P is shown in Table 6. All cases were solved to optimality. Most of them took a few seconds, and only one took more than 1 minute. A similar situation arises for Series R in Table 7.

Series DIW is displayed in Table 8. Our method proved optimality in 17 out of 21 cases. The instance diw0234 was solved in 8282 secs. while [22] reports 24000 secs. The instances diw0559, diw0795 and diw0801 were left unsolved in [22] and were solved to optimality by our algorithm.

Table 9 deals with Series TAQ. Our method proved optimality in 7 out of 14 cases. Our method failed to prove optimality in 2 cases for which it was proved in [22]. For the cases where optimality was proved, the computing times are higher than those of [22].

Series DMXA appears in Table 10. Optimality was proved in 13 out of 14 cases. For dmxa0368 the lower bound gives the optimal value, but the primal heuristics did not produce the right upper bound. The instance dmxa1010 was left unsolved in [22], our method proved optimality in 7335 secs. Our computing times were comparable to those in [22].

Table 11 presents our results with Series MSM. Our algorithm proved optimality in 26 out 30 instances. Optimality was proved for msm2601 that had been left unsolved in [22]. From the other 4 cases left unsolved in [22], our method gave a better gap from optimality in 3 of them. Our computing times were comparable to those in [22].

Series GAP is shown in Table 12. Our method proved optimality in 11 instances from a total of 13. Optimality for gap2007 had been proved in [22], our method failed in this case. Our computing times are larger that those of [22].

Table 13 presents our results with Series ALUE. Our method could prove optimality in 4 out of 15 instances. There are 3 instances for which optimality was proved in [22] and our method failed. There are 2 instances that could not be treated because they required more than 700 MB of memory.

Table 14 shows our results with Series ALUT. Our code could prove optimality for 3 out of 9 cases. These are the smallest instances in this Series. The algorithm of [22] proved optimality for the same 3 cases. Our code could not handle the 2 largest instances in this series because they required more than 700 MB of memory. Our computing times were larger than those reported in [22].

Finally Table 15 deals with Series ES. These instances come from rectilinear Steiner tree problems that have been transformed into Steiner tree problems in graphs. Our

code could not prove optimality in any of them. The main difficulty here is that the objective function value consists of 8 digits. An approximate method like ours can not attain such precision. However the gap from optimality was less than 1% in most cases, with the exception of 4 instances whose largest gap was 1.5%. The algorithm of [22] proved optimality for all cases with less than 40 terminals and for 9 out 15 instances with 40 terminals.

7. Conclusions

We have presented an approximate method to solve Steiner tree problems in graphs based on a Lagrangian relaxation of a multicommodity flow formulation. We used the VA to tackle the linear programming relaxation. We compared the VA with the modified subgradient method of [7] and in all cases the VA gave better bounds. Then we studied the test-set in Steinlib and compared our approach with the cutting plane algorithm of [22]. These computational results show that our method performs poorly in cases with a large number of terminals. This seems to be the main limitation of this approach. On the other hand, our method proved optimality in 5 cases left unsolved by the cutting plane algorithm of [22]. These are diw0559, diw0795, diw0801, dmxa1010 and msm2601. All these instances have a small number of terminals. Our approach seems to be a viable alternative for those cases that have a small number of terminals and for which the cutting plane approach happens to be too slow.

Acknowledgments. We are grateful to Dr. T. Koch for providing us with his preprocessing code. We are also indebted to both referees whose comments have improved our presentation.

APPENDIX

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
c01	114	203	5	2	84.51	85	0.0	∼	1
c02	93	166	8	3	143.22	144	0.0	∼	1
c03	76	109	42	54	753.31	754	0.0	∼	1
c04	76	111	47	49	1078.15	1079	0.0	∼	1
c05[a]	0	0	0	0	1579.00	1579	0.0	∼	1
c06	350	804	5	10	54.55	55	0.0	∼	3
c07	356	814	9	18	101.21	102	0.0	∼	4
c08	272	517	67	594	509.00	509	0.0	∼	13
c09	217	378	86	1949	707.00	707	0.0	∼	13
c10	78	111	56	76	1092.31	1093	0.0	∼	3
c11	496	2039	5	9	31.47	32	0.0	∼	17
c12	487	1772	10	50	45.41	46	0.0	∼	13
c13	386	878	78	1055	257.01	258	0.0	∼	21
c14	266	491	96	661	322.56	323	0.0	∼	15
c15	122	188	78	194	555.26	556	0.0	∼	7
c16	500	3504	5	13	10.10	11	0.0	∼	20
c17	499	3000	10	30	17.12	18	0.0	∼	29
c18	465	1361	80	1120	112.10	113	0.0	∼	105
c19	433	1057	116	922	145.01	146	0.0	∼	42
c20	106	179	68	93	266.22	267	0.0	∼	14
d01	234	443	5	6	105.77	106	0.0	∼	2
d02	259	479	10	17	219.62	220	0.0	∼	2
d03	135	201	72	202	1564.96	1565	0.0	∼	4
d04	113	163	66	156	1934.02	1935	0.0	∼	3
d05	51	69	41	22	3249.28	3250	0.0	∼	1
d06	742	1708	5	29	67.00	67	0.0	∼	17
d07	721	1663	10	46	102.46	103	0.0	∼	15
d08	555	1076	147	6172	1071.03	1072	0.0	∼	80
d09	413	714	179	6826	1447.11	1448	0.0	∼	78
d10	140	206	101	464	2109.15	2110	0.0	∼	22
d11	985	4376	5	53	28.18	29	0.0	∼	53
d12	989	3808	10	105	41.08	42	0.0	∼	48
d13	753	1700	153	7156	499.06	500	0.0	∼	88
d14	610	1202	201	1548	666.02	667	0.0	∼	163
d15	258	401	163	2277	1115.03	1116	0.0	∼	50
d16	1000	8621	5	45	12.13	13	0.0	∼	141
d17	1000	8035	10	102	22.20	23	0.0	∼	198
d18	939	2901	159	6081	222.41	223	0.0	∼	309
d19	901	2453	234	10000	308.39	311	0.8	$*^{-}$	869
d20	379	684	226	926	536.02	537	0.0	∼	209

[a] This instance was solved by the preprocessing procedure.

TABLE 2. Series C and D.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
e01	655	1248	5	7	110.41	111	0.0	$\sim$	5
e02	682	1273	10	24	214.00	214	0.0	$\sim$	10
e03	358	540	199	786	4012.70	4013	0.0	$\sim$	114
e04	214	316	147	214	5100.61	5101	0.0	$\sim$	52
e05	47	64	40	3	8127.22	8128	0.0	$\sim$	6
e06	1820	4278	5	35	72.12	73	0.0	$\sim$	26
e07	1863	4334	10	106	145.00	145	0.0	$\sim$	84
e08	1344	2623	361	10000	2630.10	2651	0.8	$*^{-}$	1071
e09	1056	1849	431	10000	3598.48	3608	0.3	$*^{-}$	1246
e10	410	600	283	10000	5599.00	5600	0.0	$*^{-}$	449
e11	2495	11861	5	73	33.05	34	0.0	$\sim$	200
e12	2487	11362	10	734	67.00	67	0.0	$\sim$	394
e13	1884	4300	395	10000	1164.01	1317	13.1	$*^{-}$	2817
e14	1514	3058	484	10000	1664.55	1753	5.3	$*^{-}$	1611
e15	536	831	361	10000	2781.45	2784	0.1	$*^{-}$	1294
e16	2500	25184	5	116	14.20	15	0.0	$\sim$	1592
e17	2500	21508	10	251	24.11	25	0.0	$\sim$	1509
e18	2382	7368	408	10000	469.80	600	27.7	$*^{-}$	68950
e19	2127	5352	574	10000	679.79	780	14.8	$*^{-}$	4582
e20	904	1618	534	10000	1336.70	1343	0.5	$*^{-}$	2317

TABLE 3. Series E.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
mc11	153	218	90	69	11688.12	11689	0.0	$\sim$	11
mc13	149	692	80	10000	90.13	93	3.2	$*^{-}$	740
mc2	120	489	60	10000	70.37	72	2.3	$*^{-}$	138
mc3	97	1203	45	10000	45.87	47	2.5	$*^{-}$	675
mc7	142	219	74	44	3416.81	3417	0.0	$\sim$	15
mc8	200	291	112	297	1565.43	1566	0.0	$\sim$	23

TABLE 4. Series MC.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
berlin52	48	144	15	4	1043.99	1044	0.0	$\sim$	1
brasil58	30	94	9	1	13655.00	13655	0.0	$\sim$	1
world666	570	3054	131	7083	122467.00	122467	0.0	$\sim$	330

TABLE 5. Problems in complete graphs.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
p401	66	148	5	1	155.00	155	0.0	~	1
p402	59	121	5	1	115.28	116	0.0	~	1
p403	71	166	5	1	178.13	179	0.0	~	1
p404	53	100	9	1	269.44	270	0.0	~	1
p405	49	91	9	1	269.85	270	0.0	~	1
p406	67	139	10	1	289.33	290	0.0	~	1
p407	67	129	17	2	590.00	590	0.0	~	1
p408	45	84	16	1	541.98	542	0.0	~	1
p409	19	24	14	1	962.23	963	0.0	~	1
p410[a]	0	0	0	0	1010.00	1010	0.0	~	0
p455	100	1057	5	6	1137.94	1138	0.0	~	6
p456	100	880	5	3	1227.03	1228	0.0	~	8
p457	99	654	10	7	1609.00	1609	0.0	~	6
p458	99	591	10	4	1867.41	1868	0.0	~	5
p459	93	405	19	6	2344.70	2345	0.0	~	2
p460	93	436	20	9	2958.98	2959	0.0	~	5
p461	40	80	21	2	4473.97	4474	0.0	~	1
p463	200	2213	10	46	1509.99	1510	0.0	~	45
p464	194	1757	18	41	2544.61	2545	0.0	~	62
p465	184	794	38	58	3852.99	3853	0.0	~	16
p466	112	234	56	29	6234.00	6234	0.0	~	7
p601	53	96	5	1	10230.00	10230	0.0	~	1
p602	48	81	5	1	8083.00	8083	0.0	~	1
p603	50	88	5	1	5022.00	5022	0.0	~	1
p604	51	88	8	1	11397.00	11397	0.0	~	1
p605	50	84	8	1	10355.00	10355	0.0	~	1
p606	45	74	7	1	13048.00	13048	0.0	~	1
p607	24	38	9	1	15358.00	15358	0.0	~	1
p608	27	43	11	1	14439.00	14439	0.0	~	1
p609	41	64	16	1	18262.20	18263	0.0	~	1
p610	24	35	12	1	30160.71	30161	0.0	~	1
p611	15	19	11	1	26903.00	26903	0.0	~	1
p612	14	17	10	1	30258.00	30258	0.0	~	1
p613	118	210	9	2	18428.04	18429	0.0	~	3
p614	118	201	19	6	27276.00	27276	0.0	~	3
p615	105	168	36	14	42473.10	42474	0.0	~	3
p616	37	48	25	2	62262.50	62263	0.0	~	1
p619	78	147	5	1	7484.25	7485	0.0	~	1
p620	76	142	5	2	8745.07	8746	0.0	~	1
p621	73	137	5	2	8687.06	8688	0.0	~	1
p622	75	135	10	2	15971.10	15972	0.0	~	1
p623	68	124	10	4	19495.15	19496	0.0	~	1
p624	63	109	14	2	20245.40	20246	0.0	~	1
p625	71	130	18	4	23077.10	23078	0.0	~	1
p626	62	109	19	5	22345.20	22346	0.0	~	1
p627	21	33	10	1	40647.00	40647	0.0	~	1
p628	35	56	21	3	40007.30	40008	0.0	~	1
p629	21	30	13	1	43286.80	43287	0.0	~	1
p630	174	329	10	8	26124.20	26125	0.0	~	4
p631	166	308	19	18	39066.10	39067	0.0	~	4
p632	155	286	34	87	56216.90	56217	0.0	~	4
p633	46	72	23	5	86267.20	86268	0.0	~	2

[a] This instance was solved by the preprocessing procedure.

TABLE 6. Series P.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
r01	4	5	3	1	186.41	187	0.0	≈	1
r02[a]	0	0	0	0	164.00	164	0.0	≈	0
r03	8	10	4	1	235.38	236	0.0	≈	1
r04	28	44	8	1	253.04	254	0.0	≈	1
r05[a]	0	0	0	0	226.00	226	0.0	≈	0
r06	8	10	6	1	241.05	242	0.0	≈	1
r07	8	10	6	1	247.05	248	0.0	≈	1
r08	8	9	6	1	235.55	236	0.0	≈	1
r09[a]	0	0	0	0	164.00	164	0.0	≈	0
r10	9	12	4	1	176.43	177	0.0	≈	1
r11[a]	0	0	0	0	144.00	144	0.0	≈	0
r12	13	18	6	1	179.53	180	0.0	≈	1
r13	24	37	8	1	149.02	150	0.0	≈	1
r14[a]	0	0	0	0	260.00	260	0.0	≈	0
r15	41	64	14	1	147.32	148	0.0	≈	1
r16[a]	0	0	0	0	160.00	160	0.0	≈	0
r17	16	25	6	1	199.22	200	0.0	≈	1
r18	128	221	45	41	404.00	404	0.0	≈	3
r19	44	72	10	1	187.39	188	0.0	≈	3
r20[a]	0	0	0	0	112.00	112	0.0	≈	0
r21[a]	0	0	0	0	192.00	192	0.0	≈	0
r22[a]	0	0	0	0	63.00	63	0.0	≈	0
r23[a]	0	0	0	0	65.00	65	0.0	≈	0
r24[a]	0	0	0	0	30.00	30	0.0	≈	0
r25[a]	0	0	0	0	23.00	23	0.0	≈	0
r26[a]	0	0	0	0	15.00	15	0.0	≈	0
r27[a]	0	0	0	0	133.00	133	0.0	≈	0
r28[a]	0	0	0	0	24.00	24	0.0	≈	0
r29[a]	0	0	0	0	200.00	200	0.0	≈	0
r30	8	10	5	1	109.05	110	0.0	≈	1
r31	69	122	14	2	258.19	259	0.0	≈	2
r32	136	249	19	17	313.00	313	0.0	≈	18
r33	116	216	18	8	267.31	268	0.0	≈	3
r34	189	349	19	15	240.02	241	0.0	≈	70
r35	133	239	18	8	150.68	151	0.0	≈	20
r36[a]	0	0	0	0	90.00	90	0.0	≈	0
r37[a]	0	0	0	0	90.00	90	0.0	≈	0
r38	60	102	12	1	165.11	166	0.0	≈	1
r39	60	102	12	1	165.44	166	0.0	≈	1
r40	40	66	10	1	154.11	155	0.0	≈	1
r41	113	203	19	6	223.20	224	0.0	≈	3
r42	24	35	11	1	152.58	153	0.0	≈	1
r43	125	223	16	7	254.92	255	0.0	≈	15
r44	136	244	17	11	251.96	252	0.0	≈	19
r45	196	361	19	29	219.99	220	0.0	≈	64
r46[a]	0	0	0	0	150.00	150	0.0	≈	0

[a] This instance was solved by the preprocessing procedure.

TABLE 7. Series R.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
diw0234	5258	9968	25	8282	1995.02	1996	0.0	≃	24002
diw0250	297	529	11	10	349.12	350	0.0	≃	6
diw0260	499	924	11	66	467.27	468	0.0	≃	6
diw0313	398	717	11	19	396.09	397	0.0	≃	7
diw0393	183	339	11	9	301.18	302	0.0	≃	3
diw0445	1709	3186	33	8981	1362.04	1363	0.0	≃	2828
diw0459	3491	6595	25	3738	1361.02	1362	0.0	≃	8629
diw0460	280	504	13	6	344.06	345	0.0	≃	3
diw0473	2097	3986	25	2028	1097.03	1098	0.0	≃	1869
diw0487	2250	4168	25	6556	1423.94	1424	0.0	≃	785
diw0495	874	1572	10	119	615.05	616	0.0	≃	41
diw0513	851	1599	10	86	604.00	604	0.0	≃	130
diw0523	1007	1920	10	67	560.10	561	0.0	≃	189
diw0540	225	384	10	4	373.53	374	0.0	≃	3
diw0559	3597	6837	18	9524	1569.05	1570	0.0	$*^{+}$	10070
diw0778	7134	13612	24	10000	1956.74	2175	11.2	$+$	10136
diw0779	11680	22346	50	10000	2948.35	4608	56.3	$-$	10168
diw0795	3076	5752	10	6313	1549.90	1550	0.0	$*^{+}$	10075
diw0801	2848	5350	10	3329	1586.03	1587	0.0	$*^{+}$	10037
diw0819	10427	19914	32	10000	2565.77	3441	34.1	$+$	10268
diw0820	11604	22202	37	10000	2793.61	4287	53.5	$-$	10139

TABLE 8. Series DIW.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
taq0014	5886	10360	128	10000	4247.82	5475	28.9	$-$	10537
taq0023	484	847	11	33	620.99	621	0.0	≃	70
taq0365	3751	6569	22	10000	1880.31	1914	1.8	$+$	10024
taq0377	6316	11135	136	10000	4909.88	6632	35.1	$-$	10422
taq0431	961	1696	13	884	896.03	897	0.0	≃	559
taq0631	458	759	10	61	581.00	581	0.0	≃	30
taq0739	740	1313	16	10000	845.49	848	0.3	$*^{-}$	361
taq0741	607	1074	16	77	846.03	847	0.0	≃	407
taq0751	904	1600	16	10000	937.00	939	0.2	$*^{-}$	517
taq0891	253	449	10	6	318.14	319	0.0	≃	8
taq0903	5503	9690	130	10000	4139.71	5234	26.4	$-$	10485
taq0910	230	402	15	6	369.09	370	0.0	≃	3
taq0920	31	49	7	1	209.19	210	0.0	≃	1
taq0978	647	1087	10	33	565.28	566	0.0	≃	20

TABLE 9. Series TAQ.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
dmxa0296	170	298	11	4	343.14	344	0.0	$\sim$	3
dmxa0368	1926	3532	18	10000	1016.99	1019	0.2	$*^{-}$	1274
dmxa0454	1707	3101	16	1769	913.05	914	0.0	$\sim$	591
dmxa0628	115	204	9	3	274.13	275	0.0	$\sim$	3
dmxa0734	610	1088	11	51	505.10	506	0.0	$\sim$	94
dmxa0848	433	777	16	42	594.00	594	0.0	$\sim$	36
dmxa0903	543	968	10	46	580.00	580	0.0	$\sim$	105
dmxa1010	3680	6733	23	7335	1487.04	1488	0.0	$*^{+}$	10020
dmxa1109	278	475	17	14	453.88	454	0.0	$\sim$	8
dmxa1200	698	1290	21	80	749.07	750	0.0	$\sim$	132
dmxa1304	253	445	10	9	310.99	311	0.0	$\sim$	4
dmxa1516	651	1179	11	26	507.06	508	0.0	$\sim$	43
dmxa1721	880	1573	18	142	779.99	780	0.0	$\sim$	51
dmxa1801	2087	3840	17	5165	1364.03	1365	0.0	$\sim$	7260

TABLE 10. Series DMXA.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
msm0580	247	417	11	7	466.61	467	0.0	$\sim$	8
msm0654	1131	2062	10	852	822.99	823	0.0	$\sim$	264
msm0709	1220	2113	16	131	883.04	884	0.0	$\sim$	247
msm0920	608	1076	26	152	805.80	806	0.0	$\sim$	47
msm1008	340	609	11	13	493.41	494	0.0	$\sim$	24
msm1234	846	1519	11	611	550.00	550	0.0	$\sim$	121
msm1477	1011	1822	31	830	1067.03	1068	0.0	$\sim$	262
msm1707	207	368	11	6	563.84	564	0.0	$\sim$	2
msm1844	44	71	10	1	187.48	188	0.0	$\sim$	1
msm1931	776	1395	8	75	603.04	604	0.0	$\sim$	76
msm2000	797	1433	9	100	594.00	594	0.0	$\sim$	114
msm2152	1938	3453	37	6981	1589.63	1590	0.0	$\sim$	3012
msm2326	367	656	14	11	398.06	399	0.0	$\sim$	7
msm2492	3761	6733	12	10000	1456.40	1459	0.2	$+$	10003
msm2525	2726	4868	12	4574	1289.95	1290	0.0	$\sim$	1887
msm2601	2668	4759	16	3906	1439.04	1440	0.0	$*^{+}$	10044
msm2705	1274	2351	13	171	713.04	714	0.0	$\sim$	371
msm2802	1537	2756	18	435	925.04	926	0.0	$\sim$	429
msm2846	3091	5564	89	10000	3008.37	3145	4.5	$-$	10269
msm3277	1536	2793	12	1244	868.02	869	0.0	$\sim$	615
msm3676	763	1307	10	67	606.04	607	0.0	$\sim$	96
msm3727	4354	7942	21	8625	1375.03	1376	0.0	$\sim$	7666
msm3829	3849	6811	12	10000	1570.89	1574	0.2	$+$	10071
msm4038	149	264	11	5	352.90	353	0.0	$\sim$	4
msm4114	323	582	16	11	392.16	393	0.0	$\sim$	7
msm4190	298	540	16	11	380.30	381	0.0	$\sim$	9
msm4224	127	214	10	2	310.11	311	0.0	$\sim$	3
msm4312	4733	8374	10	10000	2015.93	2047	1.5	$+$	10020
msm4414	186	307	10	3	407.05	408	0.0	$\sim$	3
msm4515	718	1285	13	82	630.00	630	0.0	$\sim$	167

TABLE 11. Series MSM.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
gap1307	262	452	17	13	548.56	549	0.0	∼	5
gap1413	443	777	10	17	456.59	457	0.0	∼	17
gap1500	140	253	9	2	253.73	254	0.0	∼	2
gap1810	331	568	17	19	481.97	482	0.0	∼	5
gap1904	643	1135	21	57	762.90	763	0.0	∼	26
gap2007	1860	3325	17	10000	1099.00	1104	0.5	$*^{-}$	2410
gap2119	1512	2701	29	2796	1243.03	1244	0.0	∼	818
gap2740	1057	1902	14	241	744.98	745	0.0	∼	624
gap2800	287	506	11	8	385.08	386	0.0	∼	14
gap2975	137	240	10	3	244.08	245	0.0	∼	2
gap3036	298	518	13	16	456.95	457	0.0	∼	16
gap3100	756	1338	11	426	640.00	640	0.0	∼	179
gap3128	9612	17171	104	10000	3388.46	4437	30.9	−	10022

TABLE 12. Series GAP.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
alue2087	907	1578	34	200	1048.08	1049	0.0	∼	83
alue2105	854	1430	34	172	1031.04	1032	0.0	∼	69
alue3146	2951	5088	64	10000	1988.72	2250	13.1	−	10086
alue5067	2766	4693	68	8063	2585.03	2586	0.0	∼	3810
alue5345	4168	7060	68	10000	3116.43	3567	14.5	−	10265
alue5623	3499	5869	68	10000	3220.38	3453	7.2	−	10216
alue5901	9650	16415	68	10000	3187.77	4029	26.4	−	10082
alue6179	2630	4400	66	10000	2418.31	2461	1.8	$*^{-}$	4574
alue6457	3167	5307	68	10000	2887.20	3091	7.1	−	10098
alue6735	3422	5910	68	10000	2669.69	2701	1.2	$*^{-}$	4143
alue6951	2188	3723	67	10000	2381.73	2409	1.1	$*^{-}$	1601
alue7065[a]	28711	49226	543					−	10043
alue7066	5444	9400	14	10000	2145.74	2275	6.0	+	10051
alue7080[a]	34479	55494	2344					∼	
alue7229	682	1158	34	76	823.15	824	0.0	∼	33

[a] This instance could not be solved because it required more than 700 MB of memory.

TABLE 13. Series ALUE.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
alut0787	1068	1967	34	1698	981.04	982	0.0	∼	185
alut0805	824	1498	34	3562	957.02	958	0.0	∼	207
alut1181	2894	5486	64	10000	2303.95	2397	4.0	+	10211
alut2010	5710	10535	68	10000	2996.85	3363	12.2	−	10070
alut2288	8758	16227	68	10000	2972.85	3946	32.7	−	10201
alut2566	4697	8656	67	10000	2854.37	3128	9.6	−	10351
alut2610[a]	33088	61910	204					−	10470
alut2625[a]	35833	67086	879					∼	10300
alut2764	263	455	32	35	639.97	640	0.0	∼	7

[a] This instance could not be solved because it required more than 700 MB of memory.

TABLE 14. Series ALUT.

Name	$\|V\|$	$\|E\|$	$\|T\|$	Time	LB	UB	Gap %	Vol × KM	Time [22]
es10a	49	84	10	10000	22920276.00	22920776	0.0	$*^{-}$	1
es10b	47	78	10	10000	19133177.00	19134077	0.0	$*^{-}$	1
es10c	50	85	10	10000	26003607.00	26003707	0.0	$*^{-}$	1
es10d	44	73	10	10000	20461004.00	20461104	0.0	$*^{-}$	1
es10e	49	85	10	10000	18818746.00	18818946	0.0	$*^{-}$	1
es10f	45	75	9	10000	26540391.00	26540791	0.0	$*^{-}$	1
es10g	36	59	8	10000	26023615.00	26025115	0.0	$*^{-}$	1
es10h	47	81	10	10000	25054492.00	25056192	0.0	$*^{-}$	1
es10i	46	77	9	10000	22062219.00	22062319	0.0	$*^{-}$	1
es10j	44	73	9	10000	23934204.00	23936104	0.0	$*^{-}$	1
es10k	45	76	10	10000	22237136.00	22239536	0.0	$*^{-}$	1
es10l	43	72	9	10000	19626130.00	19626330	0.0	$*^{-}$	1
es10m	35	58	9	10000	19483268.00	19483868	0.0	$*^{-}$	1
es10n	26	42	7	10000	21855164.00	21856164	0.0	$*^{-}$	1
es10o	35	58	7	10000	18641554.00	18641954	0.0	$*^{-}$	1
es20a	265	495	20	10000	33703712.00	33703912	0.0	$*^{-}$	34
es20b	271	510	20	10000	32639371.00	32639471	0.0	$*^{-}$	33
es20c	216	398	20	10000	27847298.00	27847398	0.0	$*^{-}$	12
es20d	260	486	20	10000	27501919.00	27624419	0.4	$*^{-}$	107
es20e	260	489	20	10000	33918996.00	34040296	0.4	$*^{-}$	60
es20f	262	492	19	10000	36014033.00	36014233	0.0	$*^{-}$	21
es20g	263	494	20	10000	34934619.00	34934819	0.0	$*^{-}$	49
es20h	229	428	19	10000	37884741.00	38016341	0.3	$*^{-}$	54
es20i	277	521	20	10000	36739715.00	36739915	0.0	$*^{-}$	28
es20j	241	451	20	10000	34024611.00	34024711	0.0	$*^{-}$	22
es20k	260	487	20	10000	27123730.00	27123930	0.0	$*^{-}$	42
es20l	251	466	20	10000	30451321.00	30451421	0.0	$*^{-}$	42
es20m	235	436	19	10000	34438683.00	34438683	0.0	$\sim$	17
es20n	238	444	20	10000	34060777.00	34062377	0.0	$*^{-}$	40
es20o	239	444	19	10000	32303511.00	32303711	0.0	$*^{-}$	20
es30a	654	1256	30	10000	40692112.00	40693012	0.0	$*^{-}$	834
es30b	633	1215	30	10000	40890379.00	40900079	0.0	$*^{-}$	1316
es30c	649	1246	30	10000	43119968.00	43120468	0.0	$*^{-}$	4006
es30d	669	1283	30	10000	42149957.00	42150957	0.0	$*^{-}$	1571
es30e	652	1249	30	10000	41738493.00	41739793	0.0	$*^{-}$	1538
es30f	589	1126	30	10000	39944145.00	39955145	0.0	$*^{-}$	1271
es30g	659	1267	29	10000	43759208.00	43761408	0.0	$*^{-}$	1461
es30h	577	1101	29	10000	41688569.00	41691169	0.0	$*^{-}$	1400
es30i	641	1232	29	10000	37122969.00	37133669	0.0	$*^{-}$	878
es30j	617	1188	29	10000	42686525.00	42686625	0.0	$*^{-}$	535
es30k	665	1275	30	10000	41647626.00	41648026	0.0	$*^{-}$	1259
es30l	543	1031	30	10000	38416697.00	38416697	0.0	$*^{-}$	100
es30m	586	1120	28	10000	37406279.00	37406679	0.0	$*^{-}$	520
es30n	685	1317	29	10000	42896615.00	42897015	0.0	$*^{-}$	710
es30o	616	1180	29	10000	43031247.00	43035547	0.0	$*^{-}$	2716
es40a	1169	2268	39	10000	44837297.00	44996097	0.4	$*^{-}$	5328
es40b	1123	2173	39	10000	46806023.00	46908023	0.2	$*^{-}$	3792
es40c	1147	2225	40	10000	49928951.00	50022651	0.2	$*^{-}$	9887
es40d	1120	2166	40	10000	45287974.00	45289874	0.0	$*^{-}$	5074
es40e	1280	2488	40	10000	51544583.00	52042683	1.0	$+$	10080
es40f	1098	2129	40	10000	49752134.00	50035934	0.6	$-$	10031
es40g	1162	2251	39	10000	45638861.00	45660861	0.0	$*^{-}$	3176
es40h	1248	2424	40	10000	48725000.00	49198400	1.0	$-$	10030
es40i	1220	2369	40	10000	51230137.00	51983737	1.5	$-$	10062
es40j	1246	2419	40	10000	56962238.00	57817538	1.5	$-$	10067
es40k	1184	2293	40	10000	46672936.00	47018736	0.7	$*^{-}$	8850
es40l	1252	2430	40	10000	43839197.00	44103997	0.6	$*^{-}$	4478
es40m	1373	2671	40	10000	51783391.00	51963291	0.3	$*^{-}$	5669
es40n	1300	2528	40	10000	49122981.00	49329481	0.4	$+$	10030
es40o	1303	2530	40	10000	50466994.00	50910694	0.9	$*^{-}$	8848

TABLE 15. Series ES.

REFERENCES

[1] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[2] L. Bahiense, N. Maculan and C. Sagastizábal. On the convergence of the volume algorithm. To appear.

[3] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87:385-399, 2000.

[4] F. Barahona and F. Chudak. Solving large scale uncapacitated facility location problems. In: *Approximation and Complexity in Numerical Optimization*, P. Pardalos (ed.), Kluwer, 48-62, 2000.

[5] J. Beasley. An sst-based algorithm for the Steiner problem in graphs. *Networks*, 19:1–16, 1989.

[6] J. Beasley. An algorithm for the Steiner problem in graphs. *Networks*, 14:147–159, 1984.

[7] P.M. Camerini, L. Frata and F. Maffioli. On improving relaxation methods by modified gradient techniques. *Mathematical Programming Study*, 3:26–34, 1975.

[8] S. Chopra, E. R. Gorres and M. R. Rao. Solving the Steiner tree problem in graphs using branch-and-cut. *ORSA J. Comput.*, 4:320–335, 1992.

[9] S. Chopra and M. R. Rao. The Steiner tree problem I: formulations, compositions and extension of facets. *Mathematical Programming*, 64:209–229, 1994.

[10] S. Chopra and M. R. Rao. The Steiner tree problem II: properties and classes of facets. *Mathematical Programming*, 64:231–246, 1994.

[11] A. Claus and N. Maculan. Une nouvelle formulation du Problème de Steiner sur un graphe. Technical Report 280, Centre de Recerche sur les Transports, Université de Montréal, 1983.

[12] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[13] C. W. Duin and S. Voß. Efficient path and vertex exchange in Steiner tree algorithms. *Networks*, 29:89–105, 1997.

[14] M. R. Garey and D. S. Johnson. The Rectilinear Steiner tree problem is *NP*-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

[15] M. X. Goemans and Y s. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993.

[16] M. Held, P. Wolfe and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[17] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms*. Springer Verlag, 1991.

[18] K. Holmberg and J. Hellstrand. Solving the uncapacitated network design problem by a Lagrangian heuristic and branch-and-bound. *Operations Research*, 46:247–259, 1998.

[19] F. K. Hwang, D. S. Richards and P. Winter. The Steiner tree problem. *Annals of Discrete Mathematics*, volume 53. North Holland, Amsterdam, 1992.

[20] F. K. Hwang and D. S. Richards . Steiner tree problems. *Networks*, 22:55–89, 1992.

[21] R. M. Karp. Reducibility among combinatorial problems. R. E. Miller and J. W. Thatcher, eds., *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

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

[23] C. Lemaréchal. An extension of Davidon methods to nondifferentiable problems. *Mathematical Programming Study*, 3:95–109, 1975.

[24] T. Lengauer. Combinatorial Algorithms for integrated circuit layout. Wiley, Chichester.

[25] A. Lucena. Steiner problem in graphs: Lagrangian relaxation and cutting-planes. *COAL Bull.*, 21:2–7, 1992.

[26] A. Lucena. and J. Beasley A branch-and-cut algorithm for the Steiner problem in graphs. *Networks*, 31:39–59, 1998.

[27] N. Maculan. The Steiner Problem in Graphs. *Annals of Discrete Mathematics*, 31:185–212, 1987.

[28] T. L. Magnanti and T. Wong. Network design and transportation planning: models and algorithms. *Transp. Science*, 18:1–55, 1984.

[29] V. J. Rayward-Smith and A. Clare. On finding Steiner vertices. *Networks*, 16:283–294, 1986.

[30] N. Shor. *Minimization methods for non-differentiable functions.* Springer-Verlag, Berlin, 1985.

[31] J. Soukup and W. F. Chow. Set of test problems for the minimum length connection networks. ACM/SIGMAP *Newsletters*, 15:48–51, 1973.

[32] H. Takahashi and A. Matsuyama. An approximated solution for the Steiner tree problem in graphs. *Math. Japonica*, 254:573–577, 1980.

[33] S. Voß. Steiner's problem in graphs: heuristic methods. *Discrete Applied Mathematics*, 40:45–72, 1992.

[34] P. Winter. Steiner problems in networks: a survey. *Networks*, 17:129–167, 1987.

[35] P. Winter and J. M. Smith. Path-distance heuristics for the Steiner problem in undirected networks. *Algorithmica*, 7:309–327, 1992.

[36] P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. *Mathematical Programming Study*, 3:145–173, 1975.

[37] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

(L. Bahiense) UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COPPE-SISTEMAS E COMPUTAÇÃO, P.O. BOX 68511, RIO DE JANEIRO, RJ 21945-970, BRAZIL

SUPPORTED BY GRANT FROM BRAZILIAN AGENCY CNPQ

E-mail address, L. Bahiense: laura@psr-inc.com

(F. Barahona) IBM T. J. WATSON RESEARCH CENTER, YORKTOWN HEIGHTS, NY 10589, USA (CORRESPONDING AUTHOR)

E-mail address, F. Barahona: barahon@us.ibm.com

(O. Porto) PUC-RIO, DEPT. DE ENGENHARIA ELÉTRICA, RUA MARQUÊS DE SÃO VICENTE 225, PREDIO CARDEAL LEME, SALA 401, CEP 22453-900, RIO DE JANEIRO, RJ, BRAZIL

PARTIALLY SUPPORTED BY BRAZILIAN AGENCY CNPQ, GRANT 301261/91-1

E-mail address, O. Porto: oscar@ele.puc-rio.br