A specialized branch-and-bound algorithm for the Euclidean Steiner tree problem in n-space

Marcia Fampa · Jon Lee · Wendel Melo

16 November 2014

Abstract We present a specialized branch-and-bound (b&b) algorithm for the Euclidean Steiner tree problem (ESTP) in $\mathbb{R}^n$ and apply it to a convex mixed-integer nonlinear programming (MINLP) formulation of the problem, presented by Fampa and Maculan. The algorithm contains procedures to avoid difficulties observed when applying a b&b algorithm for general MINLP problems to solve the ESTP. Our main emphasis is on isomorphism pruning, in order to prevent solving several equivalent subproblems corresponding to isomorphic Steiner trees. We introduce the concept of representative Steiner trees, which allows the pruning of these subproblems, as well as the implementation of procedures to fix variables and add valid inequalities. We also propose more general procedures to improve the efficiency of the b&b algorithm, which may be extended to the solution of other MINLP problems. Computational results demonstrate substantial gains compared to the standard b&b for convex MINLP.

Marcia Fampa
Instituto de Matemtica and PESC/COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil.
E-mail: fampa@cos.ufrj.br

Jon Lee
University of Michigan. Ann Arbor, Michigan, USA.
E-mail: jonxlee@umich.edu

Wendel Melo
PESC/COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil.
E-mail: wendelmelo@cos.ufrj.br

1 Introduction

The Euclidean Steiner tree problem (ESTP) is to find a tree of minimal Euclidean length that spans a set of given points in $\mathbb{R}^n$, using or not additional points in its construction. Given points are called *terminals*, and additional points are *Steiner points*. The ESTP is NP-Hard (see [9]) and is notoriously difficult to solve in dimension greater than 2. An optimal solution of ESTP, known as a *Steiner minimal tree (SMT)*, satisfies some well-known properties (see [12]) as, for example:

– A Steiner point in an SMT has degree equal to three. The Steiner point and its 3 adjacent nodes lie in a plane, and the included angles between the arcs connecting the point to its adjacent nodes are all 120 degrees.
– A terminal node in an SMT has degree between one and three.
– An SMT for a problem with p terminal nodes has at most $p-2$ Steiner points.
– All Steiner points lie in the convex hull of the given terminal nodes.

The *topology* of a Steiner tree is an important concept for solution methods for the ESTP and is defined as the tree where the connections between the terminals and Steiner points are specified, but the locations of the Steiner points are not. A *full Steiner topology (FST)* for p terminals is a topology with $p-2$ Steiner points, where all the terminals have degree one and Steiner points have degree three. Any SMT with non-full topology can be related to a FST where some edges have zero length. In this case the Steiner tree is called *degenerate*.

The exact solution of the ESTP in $\mathbb{R}^2$ has been successfully addressed in the past decades. Very impressive numerical results for the problem in the plane were obtained with the GeoSteiner algorithm proposed by Warne, Winter and Zachariasen [22]. Nevertheless, this algorithm, specifically developed for $\mathbb{R}^2$, cannot be applied to problems in higher dimensions and only a few papers have considered the exact solution for the ESTP in $\mathbb{R}^n$, for $n \geq 3$. Gilbert and Pollak [11] proposed solving the problem in $\mathbb{R}^n$ by enumerating all Steiner topologies and computing the minimal length for the tree associated with each topology. The extremely fast growth of the number of Steiner topologies with the number of given terminals, however, restricts the application of the algorithm only to very small instances of the problem. A branch-and-bound (b&b) algorithm for finding Steiner minimal trees in $\mathbb{R}^n$ was proposed by Smith [19]. The algorithm applies an interesting implicit enumeration scheme for all FSTs on a given set of terminals, where each level $k \geq 0$ of the enumeration tree has nodes corresponding to all FSTs on a subset of $k+3$ terminal nodes. Fampa and Anstreicher [7] used the enumeration scheme presented by Smith and proposed a conic formulation for the problem of locating the Steiner points, for a given topology, to obtain a lower bound on the minimal tree length and also to implement a "strong branching" technique. Van Laarhoven and Anstreicher [21] also applied the enumeration scheme proposed by Smith to solve the problem, using some geometric conditions satisfied by an SMT to eliminate candidate topologies from the b&b tree.

To our best knowledge, only two methematical-programming formulations for the ESTP have been presented in the literature. Maculan, Michelon and Xavier [15] formulated the problem as a non-convex mixed-integer nonlinear programming (MINLP) problem and proposed a b&b algorithm using Lagrangian dual bounds. Fampa and Maculan [8] presented a convex MINLP formulation that could be addressed with a b&b algorithm using bounds computable from conic problems (we use the standard terminology of *convex* MINLP for an MINLP that has a convex continuous relaxation). Both formulations (that is, from [15] and [8] use binary variables to indicate whether or not the edge connecting two nodes is present in a Steiner topology. The presence of these binary variables leads to a natural branching scheme, however neither [15] nor [8] present computational results based on the proposed formulations.

Our objective is to consider the formulation for the ESTP in [8], presented in Section 2, and take advantage of the recent advances in convex MINLP to solve the ESTP. Numerical experiments with the model, where we applied the b&b algorithms for convex MINLP described in [18], revealed the difficulty in solving even small instances of the problem, and motivated us to develop a b&b algorithm specialized for the ESTP. The MINLP b&b algorithms in [18] were coded in the solver `Muriqui`, and the specialized b&b algorithm presented in this work is named `SAMBA` (**S**teiner **A**daptations on **M**uriqui **B**&b **A**lgorithm). The main difficulty observed in the experiments with `Muriqui` comes from the large number of isomorphisms among the trees that span terminals and Steiner points, leading the b&b algorithm to solve equivalent or symmetric subproblems several times. We present in this work a procedure to avoid the consideration of isomorphic trees in the b&b enumeration scheme, based on the idea of representatives for classes of isomorphic Steiner topologies. The difficulty related to symmetry in the solution of integer-programming problems is well known (see [16]), and our approach is based on the idea of representatives for classes of isomorphic subproblems introduced in [17].

The concept of representatives for classes of isomorphic Steiner topologies and the algorithms to compute them are introduced in Section 3. In Section 4, we discuss how representative topologies are used to improve the b&b algorithm. We propose a procedure to fix variables in a preprocessing phase of the b&b, develop valid inequalities, and a pruning-by-isomorphism strategy. Still in preprocessing, we use geometric conditions for a SMT to reduce the number of representative topologies. Besides the procedures related to the concept of representative topologies, we present a more general procedure, called Dynamic Constraints Set (DCS); it eliminates nonlinear redundant constraints from subproblems and was very effective in our experiments. The DCS strategy can be extended to a variety of MINLPs having disjunctive constraints.

We also propose a heuristic procedure to generate upper bounds on the optimal solution value that leads to valid combinatorial cuts. The strategy is based on the solution approach proposed by Gentilini, Margot and Shimada for the “traveling salesman problem with neighborhoods” (see [10]).

Because most solvers for the nonlinear continuous relaxation of the problem requires the functions in the model to be differentiable, in Section 5 we discuss how to deal with the non-differentiability of the Euclidean norm at points where the solution degenerates. In Section 6, we present computational results demonstrating the effectiveness of `SAMBA`, and in Section 7 we make some final remarks.

2 A convex MINLP formulation for the ESTP

In [8], Fampa and Maculan formulate the ESTP as a convex MINLP. The model, reproduced below, was derived from the nonconvex MINLP formulation proposed in [15], and the convexification of the continuous relaxation of the problem was obtained with the introduction of a “Big-M” parameter, that represents an upper bound on the distance between any two nodes in a SMT.

Let $P = \{1, 2, ..., p-1, p\}$ be the set of indices associated with the given terminals $a^1, a^2, ..., a^{p-1}, a^p$ and $S = \{p+1, p+2, ..., 2p-3, 2p-2\}$ be the set of indices associated with the Steiner points $x^{p+1}, x^{p+2}, ..., x^{2p-3}, x^{2p-2}$. Let $V = P \cup S$. Denote by $[i, j]$ an edge of the graph $G = (V, E)$, with $E = E_1 \cup E_2$, where $E_1 = \{[i, j] \ : \ i, j \in S, i < j\}$ and $E_2 = \{[i, j] \ : \ i \in P, j \in S\}$.

The ESTP is then formulated as

$$
\begin{aligned}
\text{(FM)}\ \min \quad & \sum_{[i,j]\in E} d_{ij}, && && (1)\\
\text{s.t.:} \quad & d_{ij} \geq \|x^i - x^j\| - M(1 - y_{ij}), && [i,j] \in E_1, && (2)\\
& d_{ij} \geq \|a^i - x^j\| - M(1 - y_{ij}), && [i,j] \in E_2, && (3)\\
& \sum_{j\in S} y_{ij} = 1, && i \in P, && (4)\\
& \sum_{i\in P} y_{ij} + \sum_{k<j, k\in S} y_{kj} + \sum_{k>j, k\in S} y_{jk} = 3, && j \in S, && (5)\\
& \sum_{i<j, i\in S} y_{ij} = 1, && j \in S - \{p+1\}, && (6)\\
& y_{ij} \in \{0, 1\}, && [i,j] \in E, && (7)\\
& d_{ij} \in \mathbb{R}, && [i,j] \in E, && (8)\\
& x^i \in \mathbb{R}^n, && i \in S. && (9)
\end{aligned}
$$

where $\|v\| := \sqrt{\sum_{l=1}^{n} v_l^2}$ is the Euclidean norm of $v \in \mathbb{R}^n$.

The binary variable y_{ij} indicates if $[i, j]$ is in the SMT, and the continuous variable d_{ij} represents the Euclidean length of the edge $[i, j]$. Constraints (2–3) ensure that the length is only considered in the objective function if the edge is in the tree. Based on the property of SMTs, that establishes that all Steiner points lie in the convex hull of the given terminal nodes, in [8] the authors suggest the use of the Big-M parameter value given by $M = \max\{\|a^i - a^j\|, \forall 1 \leq i < j \leq p\}$. We note that for constraints (3), this value can be improved, and we have replaced it by $M_i = \max\{\|a^i - a^j\|, \forall j \in P\}$.

The other constraints of the formulation model a FST: (4) enforces degree 1 for terminals, (5) enforces degree 3 for Steiner points, and (6) prevents subcycles.

3 Isomorphism and the concept of representative FSTs

Due to the large number of isomorphic FSTs, the b&b algorithm, when applied to formulation (FM) of the ESTP, generally solves equivalent subproblems many times and, because of this, the solution of even moderate-sized instances may become very challenging.

In order to prevent the algorithm from solving subproblems associated with isomorphic FSTs, we introduce in this section the concept of a representative for each class of isomorphic FSTs, and in the next section we discuss how to use it in the b&b algorithm. The concept introduced is based on the idea of representatives for classes of isomorphic subproblems in a b&b algorithm, proposed by Margot in [17].

3.1 Building representative FSTs

We present a procedure for generating non-isomorphic representatives for the FSTs, i.e. for the spanning trees on p terminals and $p - 2$ Steiner points, such that terminals have degree one and are connected to Steiner points, and Steiner points have degree 3. The procedure is designed to consider all FSTs in the feasible set of problem (FM), and save one representative for each class of isomorphic topologies. Therefore, all the representatives generated represent a feasible topology to (FM), and every the feasible topology to (FM) is either a representative FST or is isomorphic to one of them.

The representative topologies are built in two steps described in what follows. In the first step, representatives for topologies of spanning trees on the Steiner points are constructed, and in the second step, they are extended to FSTs. For simplicity, we use the expression “spanning tree” in this section to mean “topology of the spanning tree”.

3.1.1 First step - Building representatives for spanning trees on Steiner points

Consider the bijection

$$f : E_1 \to L = \left\{1, \ldots, \frac{(p-2)(p-3)}{2}\right\},$$

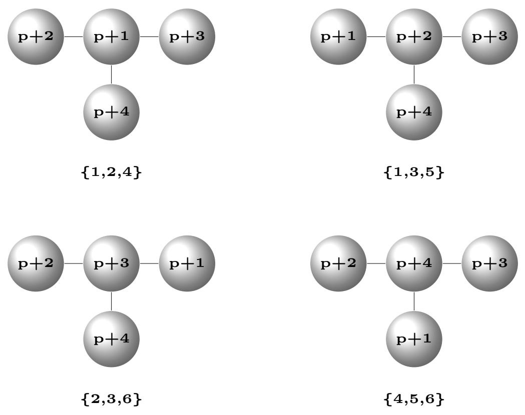

Fig. 1 Isomorphic trees for 4 Steiner points and their representations

which assigns a natural number to every edge in E_1 in the following way

$$f([p+i,p+j]) := i + \frac{(j-1)(j-2)}{2}. \tag{10}$$

Let T be a spanning tree of $p-2$ Steiner points, $V(T) := \{p+1,\ldots,2p-2\}$ be the set of nodes of T, and $E(T) := \{[i_1,j_1],\ldots,[i_{p-3},j_{p-3}]\}$ be the set of edges of T. We represent T by $f(E(T))$ defined as $f(E(T)) := \{f([i_1,j_1]),\ldots, f([i_{p-3},j_{p-3}])\}$ and we consider a tree T_1 to be lexicographically smaller than a tree T_2 if $f(E(T_1))$ is lexicographically smaller than $f(E(T_2))$. In Figure 1, we exhibit four isomorphic spanning trees on four Steiner points, as well as their representatives. The tree $\{1,2,4\}$ is lexicographically smaller than all the others. This is an important concept for our definition of a representative for each class of isomorphic trees. The representative is selected as the lexicographically smallest tree in the class. In Figure 1, for example, the tree $\{1,2,4\}$ is the representative of the isomorphic class of the four given trees.

Algorithm 1 is a procedure for constructing a representative tree for each class of isomorphic spanning trees with degree 3 or less, connecting the Steiner points $\{p+1,\ldots,p+j\}$, for all $j = 2,\ldots,p-2$. The representatives generated are the lexicographically smallest trees in the classes of isomorphic trees to which they belong. In the algorithm, $nclass(j)$ is the number of classes of isomorphic spanning trees on the j Steiner points $\{p+1,\ldots,p+j\}$ and $RT(j,k)$ is the representative tree of class k of the isomorphic trees with j Steiner points, where $k = 1,\ldots,nclass(j)$. The representative $R(2,1)$ is defined as $\bar{T}$, which is initialized as the only tree connecting the first two Steiner points. Then, at each iteration of the outer loop, an additional Steiner point $p+j$ is added to a different representative tree. For each representative tree $RT(j-1,k)$, node $p+j$ is connected to each node already in the tree. We verify for each constructed tree if it is isomorphic to any of the representatives already defined. If not, the tree becomes a representative of a new isomorphic class.

```
   Input: $p$.
   Output: $nclass(j)$, $RT(j,k)$, $k = 1,\ldots,nclass(j)$, $j = 2,\ldots,p-2$.
1  $V(\bar{T}) := \{p+1, p+2\}$; $E(\bar{T}) := \{[p+1, p+2]\}$ ;
2  $nclass(2) := 1$ ;
3  $RT(2,1) := \bar{T}$ ;
4  for $j = 3,\ldots,p-2$ do
5      $nclass(j) := 0$ ;
6      $V(\bar{T}) := V(\bar{T}) \cup \{p+j\}$ ;
7      for $k = 1,\ldots,nclass(j-1)$ do
8          for $i = 1,\ldots,j-1$ do
9              $E(\bar{T}) := E(RT(j-1,k)) \cup \{[p+i, p+j]\}$ ;
10             if $degree(\bar{T}) \leq 3$ then
11                 for $l = 1,\ldots,nclass(j)$ do
12                     if $\bar{T}$ is isomorphic to $RT(j,l)$ then
13                         GOTO (*) ;
14                 $nclass(j) := nclass(j) + 1$ ;
15                 $RT(j, nclass(j)) := \bar{T}$ ;
16             (*) ;
```

Algorithm 1: Defining representatives for classes of isomorphic spanning trees with degree three or less, on the set of Steiner points

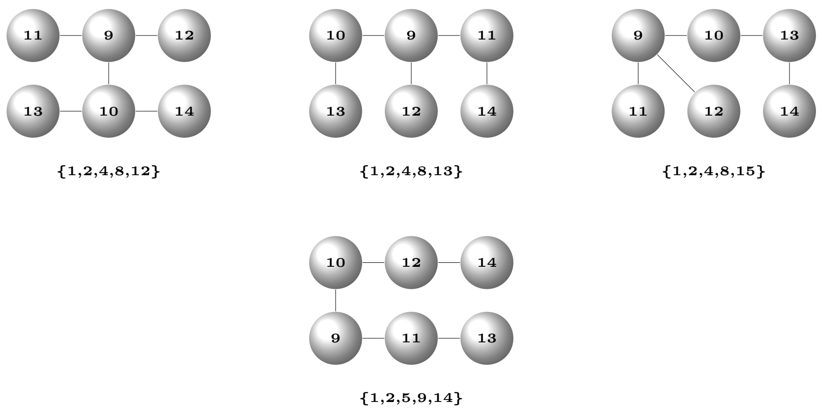

Fig. 2 Representatives for spanning trees on Steiner points for $p = 8$

Figure 2 shows the 4 representative non-isomorphic spanning trees on 6 Steiner points ($p = 8$) constructed by Algorithm 1, as well as their representatives.

It is important to note that verifying if the tree $\bar{T}$ is isomorphic to $RT(j,l)$ in step 12 of Algorithm 1 can be done in polynomial time (see [13]). We can use, for example, Algorithm 2, which is described in [14] and determines whether or not two given trees T_1 and T_2 are isomorphic.

```
   Input: $T_1$, $T_2$.
1  In each tree, label all the leaves with 1. If the numbers of leaves in $T_1$ and $T_2$ do not
   coincide, stop with "non-isomorphic" ;
2  In each tree, determine the sets of unlabeled vertices $S_1$ and $S_2$ such that every
   neighbor of $v \in S_i$, except at most one, has a label. Tentatively, label $v$ with the list
   $[l_1, \ldots, l_k]$ of labels of its labeled neighbors, sort in non-decreasing order. Compare
   the respective labels for the vertices in $S_1$ and $S_2$. If these labels do not agree (as
   multisets), stop with "non-isomorphic" ;
3  In each tree, substitute the tentative labels (which are list of numbers) by new labels
   which are just numbers: order the tentative labels of both trees in non-decreasing
   order. The vertices with the smallest labels get the new label $m$, where $m$ is the
   smallest number that has not been used as a label before ;
4  If not all vertices have labels, go back to step 3 ;
5  Stop with "isomorphic".
```

Algorithm 2: Detecting isomorphism between two trees

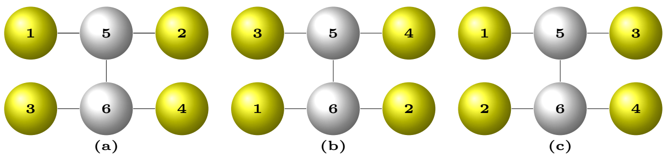

Fig. 3 FSTs for $p = 4$

3.1.2 Second step - Extension for the terminals inclusion

The procedure described in the previous subsection constructs a representative to each class of isomorphic spanning trees with degree three or less, on $p - 2$ Steiner points. This is the initial step for the construction of representatives for FSTs. We discuss in this subsection how to extend the partial trees constructed by Algorithm 1 to FSTs, connecting the p terminals to them. There are two basic differences between what is done in this subsection and what was done in the previous one: (i) the terminals have degree one and are not connected to each other (only to Steiner points), (ii) unlike Steiner points, the terminals are distinct from each other, as their positions are previously fixed. Therefore, to verify if two given FSTs are isomorphic, we need to consider the terminals as labeled nodes. Figure 3 shows three FSTs for $p = 4$ and exemplifies the relevance of the distinction of terminals for isomorphism detection. Nodes 1 to 4 are terminals, and 5, 6 are Steiner points. Considering that the terminals are distinct from each other and the Steiner points are not, we conclude that FSTs (a) and (b) are isomorphic while (a) and (c) are not.

To construct one representative of each class of isomorphic FSTs, we initially consider each representative tree on the Steiner points generated by Algorithm 1. For each tree, we verify all possibilities of connecting the terminals to the Steiner points. Each FST possibly generated with these connections is uniquely represented by a vector (denoted by v_1 in the following), which now takes into account the labels of the terminals, and defines once more a lexical ordering for the trees. The lexicographically smallest tree in each class of isomorphic FSTs is defined as the representative of its class. The procedure is detailed in Algorithm 3, and the detection of isomorphism between FSTs is based on the well-known AHU algorithm [2].

```
   Input: Representative trees on $p-2$ Steiner points constructed by Algorithm 1.
   Output: Representative FSTs for $p$ terminals.
1  for each representative tree constructed by Algorithm 1 do
2      Denote (arbitrarily) the $l$ Steiner points in the tree with degree equal to 2, by
       $s_1,\ldots,s_l$ the $k$ Steiner points with degree equal to one by $s_{l+1},\ldots,s_{l+k}$, and
       the others $p-2-l-k$ Steiner points (with degree equal to three) by
       $s_{l+k+1},\ldots,s_{p-2}$ ;
3      Create a vector $w$ with $l+2k$ components. The first $l$ components of $w$ represent
       the $l$ terminals that will be connected to the Steiner points $s_1,\ldots,s_l$.
       Components $l+1$ and $l+2$ (such that, component $l+1 <$ component $l+2$)
       represent the two terminals that will be connected to Steiner point $s_{l+1}$. The
       idea is repeated until the two last components, which represent the two
       terminals to be connected to Steiner point $s_{l+k}$ .
4      for each $\bar{w}$ generated by permuting the components of $w$ do
5          Obtain vector $\bar{v}_1$ that represents the FST corresponding to $\bar{w}$, with the
           procedure described in Algorithm 4 and denote the FST as $\bar{T}$ ;
6          if $\bar{T}$ is not isomorphic to any representative FST already saved then
7              save $\bar{T}$ as a new representative FST ;
8          if $\bar{T}$ is isomorphic to a representative FST $\tilde{T}$ already saved and $\bar{T}$ is
           lexicographically smaller than $\tilde{T}$ then
9              discard $\tilde{T}$ and save $\bar{T}$ as a new representative FST ;
10         if $\bar{T}$ is isomorphic to a representative FST $\tilde{T}$ already saved and not
           lexicographically smaller than $\tilde{T}$ then
11             discard $\bar{T}$ .
```

Algorithm 3: Defining representatives for classes of isomorphic FSTs

An observation about Algorithm 3 should be made. Let's consider, for example, the FST represented in Figure 4, where terminals are numbered and Steiner points are not. For this example, we have $w = (1,2,5,3,4,6,7,8,9)$ defined in step 3 of Algorithm 3. Each permutation of the components in w, computed in step 4, represents a FST. We are interested in saving one representative of each class of isomorphic FSTs. Clearly, we can already discard all the permutations of w, where only the two terminals connected to the same Steiner point are exchanged. For example, $\bar{w} = (1,2,5,4,3,6,7,8,9)$ should be immediately discarded, while $\bar{w} = (1,5,2,3,4,6,7,8,9)$ should not, because terminals 2 and 5 are not connected to the same Steiner point.

In Figure 4 we also show the labels of each node of the given FST assigned by Algorithm 4. For this FST, the vectors v_i defined in step 7 of the algorithm would be given by:

$$\begin{aligned}
v_5 &= (8,9),\\
v_4 &= (3,4,5,6,7,(8,9)),\\
v_3 &= (2,(3,4),(5,(8,9)),(6,7)),\\
v_2 &= ((2,(3,4)),((5,(8,9)),(6,7))),\\
v_1 &= (((2,(3,4)),((5,(8,9)),(6,7)))).
\end{aligned}$$

Vector v_1 constructed by Algorithm 4 completely describes the input FST. Two given FSTs are isomorphic if their corresponding vectors v_1 are the same (see [2]). The comparison between the vectors is the procedure used in steps 6-11 of Algorithm 3 to detect isomorphism among the FSTs.

```
  Input: A FST $T$
  Output: A vector $v_i$ corresponding to each level $i$ of $T$, where terminal 1 is
          considered the root node.
1 Assign to terminal $j$ the label $j$, for $j = 1, \ldots, p$ ;
2 Consider terminal 1 as the root of $T$ ;
3 Let $\rho$ be the largest level number of a node in this rooted tree (the level number of a
  node is the number of edges between the root and the node) ;
4 for $i = \rho - 1, \ldots, 1$ do
5     for each Steiner point $s$ with level number $i$ do
6         Label $s$ with a vector of two components. Each component is equal to the
          label of each son of $s$ . The first component should be lexicographically
          smaller than the second one ;
7     Create a vector $v_i$ with dimension given by the number of nodes in level $i$. Each
      component of $v_i$ is the label of a node in level $i$ and the components satisfy a
      lexicographically increasing order ;
```

Algorithm 4: Labeling on FSTs

4 Representative topologies and the b&b algorithm

In this section we consider that the set of representative FSTs for p terminals, constructed by Algorithm 3, is given, and we discuss different ways to apply it to improve the efficiency of the b&b algorithm. It is very important to note that this set does not depend on the instance data of the ESTP. Therefore once it is generated and saved, it may be applied to the solution of any instance with p terminals. Furthermore, the representatives for spanning trees on the $p-2$ Steiner points constructed in the first step of the procedure described in the previous section could be used to initiate the computation of the representatives for a bigger p, significantly reducing the computational work, as compared to computing them from scratch. Some other improvements to the b&b algorithm, not related to the construction of the set of representative FSTs, are also presented in Subsections 4.3 and 4.4, and they could be applied to more general convex MINLP problems as well.

In what follows, let Ψ be the set of representative FSTs for p terminals.

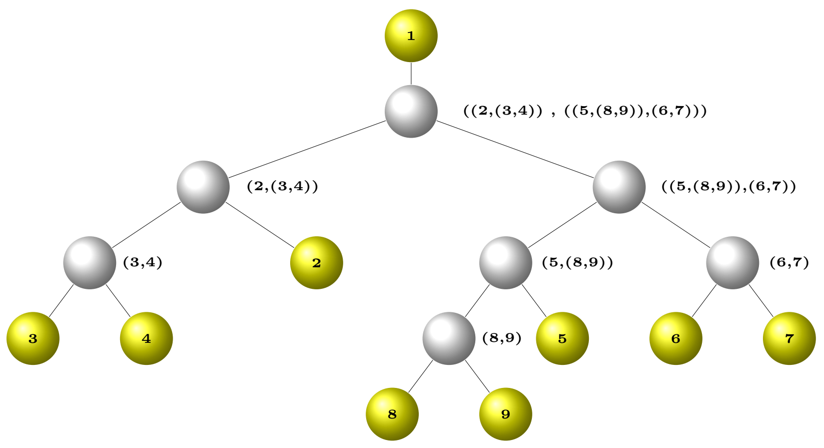

Fig. 4 Labels assigned by Algorithm 4 to the nodes of a FST

4.1 The preprocessing phase

4.1.1 Eliminating FSTs based on geometric properties of the SMT

Some geometric conditions that are satisfied by the SMT were presented in [21] and used to eliminate FSTs in the implicit enumeration scheme proposed by Smith [19]. These same conditions may be used to eliminate topologies from the list of representative FSTs in a preprocessing phase of our b&b algorithm. Unlike the generation of representative FSTs, when isomorphic topologies are discarded, this procedure depends on the instance data, and therefore it should be applied independently to each instance.

Let η_i be the distance from terminal a^i to the nearest other terminal, i.e., $\eta_i := \min_{j\in P,\, j\neq i}\{\|a^i - a^j\|\}$, for all $i \in P$. Let T be a minimum-length spanning tree on terminals a^i, $i \in P$, and let β_{ij} be the length on the longest edge on the unique path between a^i and a^j in T, for $i, j \in P$.

Two conditions on SMTs based on these parameters are known (see [21]): (i) two terminals a^i and a^j may be connected to a common Steiner point only if $\|a^i - a^j\| \leq \eta_i + \eta_j$; (ii) two terminals a^i and a^j may be connected by two or fewer Steiner points only if $\|a^i - a^j\| \leq \eta_i + \eta_j + \beta_{ij}$.

Considering these two conditions, in a preprocessing phase of our b&b algorithm, we eliminate from Ψ all topologies that do not satisfy them.

4.1.2 Fixing variables

We can use the set Ψ to fix variables, still in the preprocessing phase of the b&b algorithm.

Let $E^+ \subset E$ be the set of edges of G that belong to every topology in Ψ, and let $E^- \subset E$ be the set of edges of G that do not belong to any topology in Ψ. If $[i, j] \in E^+$, we set $y_{ij} = 1$, and if $[i, j] \in E^-$, we set $y_{ij} = 0$.

4.1.3 Generating valid inequalities

Representative topologies also allow the generation of valid inequalities, as described in the following procedure.

```
For each nonempty subset $E' \subset E$:
    For each edge $[i,j] \in E \setminus E'$:
        If all FST in $\Psi$ that contain all edges in $E'$ do not contain edge
        $[i,j]$, then add the cut

            $y_{ij} \leq |E'| - \sum_{[p,q]\in E'} y_{pq}.$        (11)

        If all FST in $\Psi$ that contain all edges in $E'$ also contain edge $[i,j]$,
        then add the cut

            $y_{ij} \geq \sum_{[p,q]\in E'} y_{pq} - |E'| + 1.$        (12)
```

We note that alternatively to the addition of cuts (11) and (12) to the original formulation of the problem, it is also possible to create a pool containing those cuts in the preprocessing phase of our method, and apply them inside the b&b scheme. In this case, during the b&b execution, each time a node of the enumeration tree is created, it is possible to check if the cuts in the pool allow us to fix variables for the descendants of the node, instead of actually including the cuts in the model. We have created specific pools of cuts for each subset of representative FSTs with a common subtree connecting the Steiner points, which increases the total number of cuts generated and allows more variable fixing.

4.2 Branching, pruning, and fixing variables in the b&b algorithm

In contrast to a standard b&b algorithm, which is initiated with only one root node, where all variables are unfixed, our b&b algorithm is initiated with several root nodes. In each node all variables y_{ij}, such that $[i,j] \in E_1$, are already fixed at 0 or 1, and each partial integer solution corresponds to one of the non-isomorphic representatives of spanning trees on the $p-2$ Steiner points, computed by Algorithm 1. This procedure avoids the solution of subproblems corresponding to isomorphic spanning trees on Steiner points. We note that in our numerical experiments we also implemented a standard b&b, and because no pruning occurred in the first levels of the enumeration tree, we only gained with the non standard version.

At each level of our b&b enumeration trees, a different terminal is selected to be connected to the partially constructed Steiner tree. The complete enumeration trees have $p-1$ levels where the FSTs are built (once $p-1$ terminals are connected, there is only one possibility to connect the last one).

At each node of the b&b trees, the branching variable is selected among $y_{ij} \in (0,1)$ in the solution of the subproblem continuous relaxation, such that $[i,j]$ belongs to E_2. The selected branching variable y_{ij} corresponds to the farthest terminal a^i to the other terminals already connected to the Steiner tree, attempting to increase the lower bounds rapidly. Once the branching variable y_{ij} is selected, the descendants of the node represent different possibilities for connecting the terminal a^i to the Steiner tree, dividing the feasible region of (FM) according to constraint (4). The maximum number of descendants of each node is $p-2$, and in each one of them, a^i is connected to a different Steiner point x^k. Consequently, for descendant k of the node, we have $y_{ik} = 1$ and $y_{il} = 0$, for every $l \neq k$, $p+1 \leq l \leq 2p-2$. The terminals can only be connected to Steiner points still with degree less than 3, which constantly decreases the number of descendants as we go deeper in the enumeration trees.

Finally, the set Ψ is considered to prune all descendant nodes where the variables fixed at 1 do not correspond to edges in any representative topology. This is the procedure that avoids the solution of subproblems corresponding to isomorphic FSTs in our b&b algorithm.

In case we choose to construct pools with the cuts (11) and (12) in the preprocessing phase, as described in the previous subsection, these pools are checked, and variables are possibly fixed in the descendants according to them. Also, when terminal a^i is added to the Steiner tree, i.e., when y_{ij} is fixed at one, for some j, we consider the two geometric conditions described in Subsection 4.1.1 to fix y_{wk} at zero, if a^w is a terminal not yet included in the Steiner tree, such that its connection to Steiner point x^k would violate one of the geometric conditions.

4.3 Dynamic Constraint Set (DCS)

Constraints (2) and (3) are a typical way of modeling the requirement of considering the constraints

$$d_{ij} \geq \|x^i - x^j\| \text{ and } d_{ij} \geq \|a^i - x^j\|$$

only when the corresponding variables y_{ij} are equal to 1. The Big-M parameters are specifically chosen to make the constraints redundant when $y_{ij} = 0$, i.e.,

$$d_{ij} \geq \|a^i - x^j\| - M_i \text{ and } d_{ij} \geq \|a^i - x^j\| - M$$

are dominated by $d_{ij} \geq 0$.

Although these redundant nonlinear inequalities do not modify the solution of the b&b subproblems, they increase considerably the overall running time. Therefore, we have added to our b&b algorithm, a strategy named "Dynamic Constraint Set" (DCS) – the idea is to dynamically change the set of constraints of the subproblems, giving to the b&b algorithm the ability to consider a different set of constraints at each node of the enumeration tree, eliminating the redundant constraints from the subproblems. All constraints in (2) and (3) corresponding to variables y_{ij} fixed at 0 are then removed from the model.

4.4 Improving upper bounds and generating more valid inequalities

We propose a heuristic procedure to improve the upper bound on the optimal value of the problem, which can be applied at any node of the b&b enumeration tree.

Let $\bar{y}_{ij}$ be the value of the binary variable y_{ij} at the optimal solution of the relaxation solved at a node of the b&b tree. If the solution does not satisfy the integrality constraints, we consider the graph $G = (V, E)$ defined in Section 2 and assign to each edge $[i, j]$ of E, the weight given by $w_{ij} := 1 - \bar{y}_{ij}$. Considering these weights, we then search for a minimum spanning tree of G having a FST.

For the computation of this tree, we propose a simple greedy heuristic that starts considering a tree with the first two Steiner points $p + 1$ and $p + 2$ and then iteratively connects the next Steiner point to a node already in the tree, such that the new edge added to the tree has minimum possible weight and the degree condition on a FST is not violated, i.e., no Steiner point receives more than three neighbors. When the Steiner points are all in the tree, the process continues with the terminal nodes, taking into account that terminal nodes should be connected to Steiner points and can be connected to any one of them. The detailed heuristic is presented in Algorithm 5.

```
   Input: $G = (V, E_1 \cup E_2), p$.
   Output: $T = (V_T, E_T)$.
1  $V_T := \{p+1, p+2\}$; $E_T := \{[p+1, p+2]\}$ ;
2  degree$(p+1)$ =degree$(p+2) = 1$ ;
3  degree$(j) = 0$, for $j = p+3, \dots, 2p-2$ ;
4  for $j = p+3, \dots, 2p-2$ do
5      $V_T := V_T \cup \{j\}$ ;
6      $\bar{E} := \{[i,j] \in E_1 | \text{degree}(i) < 3\}$ ;
7      $E_T = E_T \cup \{[\bar{\imath}, j]\}$, where $\bar{\imath} := \text{argmin}_i \{w_{ij} | [i,j] \in \bar{E}\}$ ;
8      degree$(\bar{\imath})$:= degree$(\bar{\imath}) + 1$ ;
9      degree$(j) := 1$ ;
10 $\bar{P} := \{1, \dots, p\}$ ;
11 while $\bar{P} \neq \emptyset$ do
12     $\bar{E} := \{[i,j] \in E_2 | i \in \bar{P}, \text{degree}(j) < 3\}$ ;
13     $E_T = E_T \cup \{[\bar{\imath}, \bar{\jmath}]\}$, where $[\bar{\imath}, \bar{\jmath}] := \text{argmin}_{ij} \{w_{ij} | [i,j] \in \bar{E}\}$ ;
14     $V_T := V_T \cup \{\bar{\imath}\}$ ;
15     $\bar{P} := \bar{P} \setminus \{\bar{\imath}\}$ ;
16     degree$(j)$:=degree$(j) + 1$ ;
```

Algorithm 5: Heuristic for computing a FST for the SMT

Once the FST is obtained by the heuristic, we locate the Steiner points by solving the convex problem obtained by fixing all integer variables in (FM).

The solution obtained is feasible for the ESTP, and therefore the upper bound on its optimal value is possibly updated. Furthermore, since this is a best-possible solution for the given topology, the following cut can be added to all subproblems still open in the b&b enumeration tree:

$$\sum_{[i,j]\in E} \hat{y}_{ij} y_{ij} \leq 2p - 4, \tag{13}$$

where $\hat{y}$ is the characteristic vector of the tree, i.e. $\hat{y}_{ij} = 1$ if edge $[i, j]$ belongs to the tree, and $\hat{y}_{ij} = 0$, otherwise.

5 Dealing with the non-differentiability

The continuous relaxation of (FM), as well as the problems obtained when we fix the values of all binary variables y_{ij} in (FM), are convex NLP problems that we can solve to optimality by general-purpose NLP solvers (e.g. `Ipopt`, `Mosek`). Most NLP solvers, however, require all functions in the problem to be twice continuously differentiable, which is not the case for the problems mentioned above, due to the non-differentiability of the Euclidean norm at points where the solution degenerates. There are different ways that we can deal with the particular non-differentiability that we face.

A simple fix is to approximate $\sqrt{w}$ by $h(w) := \sqrt{w+\delta} - \sqrt{\delta}$ for some small $\delta > 0$. A nice feature of this approach is that it correctly calculates zero distances. Furthermore, it underestimates *all* positive distances (via the triangle inequality), and because the objective function is increasing in distances, it leads to a relaxation of (FM). Let (FM_δ) be (FM) with all distances functions (in (2-3) shifted by δ as above. The minimum objective value of (FM_δ) is then a lower bound on the minimum objective value of (FM). Furthermore any lower bound on the objective value for (FM_δ), as would be calculated and improved in the process of solving (FM_δ) by a branch-and-bound algorithm, is a lower bound for the optimal objective value of (FM). Finally, taking an optimal solution (y, x, d) of (FM_δ) and evaluating it with the distances adjusted appropriately for (FM) — we can think of this as fixing (y, x) in (FM) and then just minimizing over d — gives an upper bound on the optimal objective value of (FM). So we can expect a very good solution to (FM) and a nearby rigorous lower bound on the exact optimal objective value of (FM).

6 Numerical results

In this section, we analyze the impact on the b&b algorithm of the specialized procedures for the ESTP that we propose, and we compare the performance of the b&b algorithm of `Muriqui` with the specialized b&b algorithm implemented as `SAMBA`. Both algorithms are implemented in `C++`, and all runs were conducted on a 3.60 GHz core i7-4790 CPU, 8 MB, 16 GB, running under

Table 1 Average results

Dimension	Muriqui b&b		SAMBA	
	gap (%)	cpu time (sec)	gap (%)	cpu time (sec)
$n = 3$	84	14400.38	0	240.96
$n = 4$	84	14401.12	0	740.87
$n = 5$	82	14400.34	0	1301.54

Linux, with a time limit of 4 hours. `MOSEK` [1] was used to solve the subproblems relaxations in b&b, and also to solve the convex problem obtained by fixing all integer variables in (FM), in the heuristic procedure described in Subsection 4.4. Instances considered were the same used in [7,21]: 30 instances with 10 terminals randomly distributed in the hypercube $[0, 10]^n$, 10 in each dimension $n = 3, 4, 5$.

In Figure 5, we show the impact on the running time of the b&b algorithm of the more effective procedures proposed in this work, which were incrementally added to the code. The first chart shows how the use of the set of representative FSTs affects the algorithm. More specifically, we show a final reduction in the running time of 40% on average after adding to the algorithm the strategy of fixing variables introduced in Subsection 4.1.2 (fix. var.) and the isomorphism pruning strategy introduced in Subsection 4.2 (isom. prune), both using the representative topologies. The second chart in Figure 5 shows an additional reduction in the running time of 54% on average, due to the application of the geometric condition (i) presented in Subsection 4.1.1 (rule 1). We note that for the instances considered in our tests, the geometric condition (ii) (rule 2) was not effective to eliminate candidate representative FSTs. Finally, the third chart in Figure 5 shows another reduction of 58% in the running time due to the DCS strategy described in Subsection 4.3. This strategy was very effective in our experiments and can be applied to other MINLPs having disjunctive constraints modeled with Big-M parameters.

In Table 1, we compare the performance of the b&b algorithm of `Muriqui` with the final version of our specialized b&b algorithm `SAMBA`. We show average results for each dimension n. The columns present the gap (100%×(upper bound - lower bound)/upper bound), and the cpu time in seconds for both `Muriqui` and `SAMBA`. No instances could be solved by `Muriqui` within the time limit, and the average duality gap was of 83%, endorsing the well-known difficulty of the ESTP. We also applied the MINLP solver `Bonmin` [5,6], with similar results to those of `Muriqui`; no instance could be solved in the time limit. In contrast, we solved all 30 instances to optimality with `SAMBA` in an average time of about 4, 12 and 22 minutes, for $n = 3, 4, 5$, respectively. More detailed results for these experiments are given in Table 2, where lb and ub stand for lower and upper bound, respectively. Instance inst$(p \times n)_k$ corresponds to the k-th instance with p terminals in dimension n. The results indicate significant gains obtained by the procedures we propose, making the application of b&b to the convex MINLP formulation of the ESTP more practical.

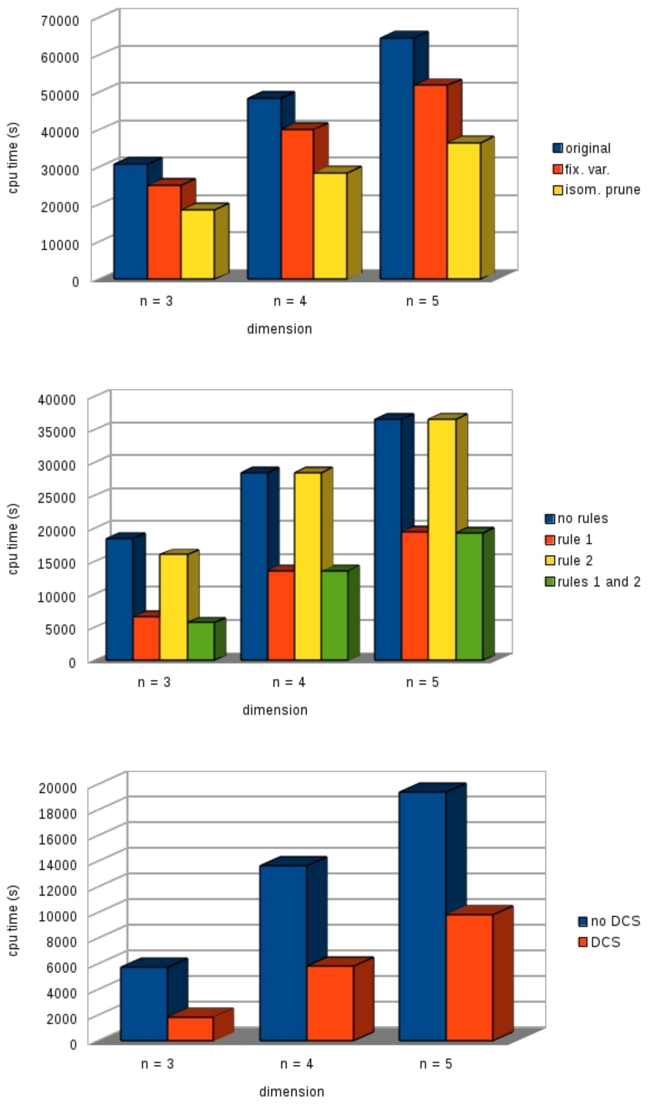

Fig. 5 Impact of the specialized procedures on b&b

7 Conclusion

Although two MINLP formulations were presented in the literature for the ESTP in $\mathbb{R}^n$, no numerical results were previously presented using them. Even with the considerable advances in MINLP in the last decade, in particular in the solution of convex MINLP problems, the application of the well-known solver `Bonmin` to the convex MINLP formulation of the ESTP proposed by

Table 2 Numerical Results

Instance	Muriqui b&b			SAMBA	
	lb	ub	cpu time (sec)	ub	cpu time (sec)
inst10x3_01	1.888	37.358	14,400.421	28.515	226.499
inst10x3_02	5.851	35.579	14,400.407	27.804	46.680
inst10x3_03	6.998	41.148	14,400.317	29.809	71.467
inst10x3_04	5.130	37.808	14,400.444	34.619	281.375
inst10x3_05	6.733	38.420	14,400.331	30.943	243.685
inst10x3_06	4.534	30.938	14,400.317	21.851	43.035
inst10x3_07	6.572	37.893	14,400.385	30.277	252.736
inst10x3_08	6.082	33.864	14,400.429	30.903	344.361
inst10x3_09	7.616	36.804	14,400.486	31.799	438.627
inst10x3_10	6.542	31.139	14,400.296	28.080	461.109
inst10x4_01	8.562	46.974	14,400.655	42.871	860.945
inst10x4_02	7.171	38.885	14,400.374	32.677	651.011
inst10x4_03	3.732	44.700	14,405.476	40.933	953.050
inst10x4_04	8.815	50.255	14,400.292	46.021	1,858.901
inst10x4_05	9.057	46.853	14,400.421	40.975	614.825
inst10x4_06	6.553	45.603	14,400.448	39.211	1,015.922
inst10x4_07	6.879	41.467	14,400.363	33.578	321.173
inst10x4_08	9.569	45.139	14,402.475	39.036	582.041
inst10x4_09	6.693	42.018	14,400.298	35.135	227.584
inst10x4_10	3.937	39.743	14,400.403	32.800	323.262
inst10x5_01	9.092	51.877	14,400.272	43.747	861.452
inst10x5_02	8.051	49.709	14,400.365	42.660	2,105.376
inst10x5_03	9.972	52.144	14,400.213	44.553	1,113.525
inst10x5_04	8.167	46.633	14,400.256	41.265	856.387
inst10x5_05	7.618	44.268	14,400.351	42.682	896.618
inst10x5_06	11.711	58.400	14,400.261	52.949	844.907
inst10x5_07	8.905	51.467	14,400.378	48.498	3,378.936
inst10x5_08	11.532	57.697	14,400.432	49.456	534.453
inst10x5_09	8.123	50.300	14,400.298	47.153	1,638.311
inst10x5_10	8.886	47.218	14,400.536	39.309	785.414

Fampa and Maculan (FM), failed to solve all instances considered in this paper in a time limit of four hours, and computed in this time feasible solutions of very poor quality. We presented procedures to be included in a b&b algorithm for convex MINLP problems, aiming at the development of a specialized b&b for the ESTP. Our main concern was with the well-known inefficiency of the b&b algorithm when dealing with a large number of isomorphic or symmetric subproblems. This has been a subject of intense research and was considered, for example, in the work of Margot [17], where the construction of a list of representatives for non-isomorphic subproblems during the b&b execution is suggested, in order to avoid the solution of isomorphic subproblems. We have observed that the isomorphism among the subproblems in the solution of the ESTP is derived from the isomorphism among Full Steiner Topologies (FSTs), and those topologies depend only on the number of terminals, but not on their location. This, together with the possibility of checking if two given topologies are isomorphic in polynomial time, allows us to construct a list of non-isomorphic topologies (or subproblems) in advance, saving them in our framework. For the case $p = 10$ considered in our experiments, the total number of FSTs in the feasible set of the formulation (FM) is 428,652,000, while the number of representative non-isomorphic FSTs is 2,027,025, representing a reduction of more than 99% in the number of possible subproblems to solve. Although this reduction is very impressive, the number of representatives grows extremely quickly, and it would not be possible to save all representative FSTs for a significantly larger number of terminals. We note, however, that in case we are solving problems with p' terminals, and we have only the list of non-isomorphic FSTs with p terminals, where $p < p'$, it is trivial to construct the list of non-isomorphic spanning trees on the $p' - 2$ Steiner points, taking as input the list of representatives for the spanning trees on the $p - 2$ Steiner points already constructed. This pre-saved list may still be used to initiate the b&b algorithm, and then, during the execution of the b&b, when the terminals are added to the Steiner trees, we can construct the list of representative FSTs as they are iteratively generated by the algorithm, as suggested in [17].

Besides introducing to the b&b algorithm the use of representative non-isomorphic FSTs, we also propose the use of a heuristic to be solved in the nodes of the b&b tree to improve the upper bounds and generate combinatorial cuts. Finally, after noticing the increase in the running time of `MOSEK`, when solving the subproblems relaxations, due to the nonlinear redundant Big-M constraints when the corresponding binary variables are already fixed in zero, we introduced to our code a procedure named Dynamic Constraint Set, that eliminates from the subproblems the corresponding constraints. The procedure was very effective and can be applied to other MINLP problems with disjunctive constraints.

Numerical results obtained with the final version of our specialized b&b algorithm, coded as `SAMBA`, demonstrate substantial improvements in the total running time required to solve the ESTP, compared to the original general b&b algorithm to solve convex MINLP problems.

References

1. Andersen, E., Andersen, K.: The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm. in: H. Frenk et al., eds., High Performance Optimization; Appl. Opt. **33**, Kluwer (1999).
2. Aho, A. V., Hopcroft, J. E., and Ullman, J. D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974).
3. Beasley, J.E.: OR-Library: Distributing test problems by electronic mail. J. Operational Research Society **41**, 1069–1072 (1990). `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.
4. Belotti, P., Lee, J., Liberti, L., Margot, M., and Wächter, A.: Branching and Bounds Tightening Techniques for Nonconvex MINLPs. Optimization Methods and Software **24**, 597–634 (2009).
5. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., and Wächter, A.: An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs. Discrete Optimization **5(2)**, 186–204 (2008).

6. Bonmin (Basic Open-source Mixed INteger programming). http://projects.coin-or.org/Bonmin.
7. Fampa, M., and Anstreicher, K.M.: An improved algorithm for computing Steiner minimal trees in Euclidean *d*-space. Discrete Optimization **5(2)**, 530–540 (2008).
8. Fampa, M., and Maculan, N.: Using a conic formulation for finding Steiner minimal trees. Numerical Algorithms **35**, 315–330 (2004).
9. Garey, M.R., Graham, R.L. and Johnson, D.S.: The complexity of computing Steiner minimal trees. SIAM J. Applied Mathematics **32(4)**, 835–859 (1977).
10. Gentilini, I., Margot, F., Shimada, K.: The Traveling Salesman Problem with Neighborhoods: MINLP Solution. Optimization Methods and Software **28**, 364–378 (2013).
11. Gilbert, E.N. and Pollack, H.O.: Steiner minimal trees. SIAM J. Applied Math. **16**, 1–29 (1968).
12. Hwang, F.K., Richards, D.S. and Winter, W.: The Steiner tree problem. Annals of Discrete Mathematics **53**, Elsevier, Amsterdan (1992).
13. Kelly, P.J.: A congruence theorem for trees. Pacific J. Math. **7**, 961–968 (1957).
14. Köber, J., Schöning, U., and Torán, J.: The graph isomorphism problem: its structural complexity (1993).
15. Maculan, N., Michelon, P. and Xavier, A.E.: The Euclidean Steiner problem in $\mathbb{R}^n$: a mathematical programming formulation. Annals of Operations Research **96**, 209–220 (2000).
16. Margot, F.: Symmetry in Integer Linear Programming. 50 Years of Integer Programming, Juenger et al. (Eds), Springer (2009).
17. Margot, F.: Pruning by Isomorphism in Branch-and-Cut. Mathematical Programming **94**, 71–90 (2002).
18. Melo, W, Fampa, M., and Raupp F., Integrating nonlinear branch-and-bound and outer approximation for convex Mixed Integer Nonlinear Programming. Journal of Global Optimization **60**, 373–389 (2014).
19. Smith, W.D.: How to find Steiner minimal trees in Euclidean *d*-space. Algorithmica **7**, 137–177 (1992).
20. Soukup, J., Chow, W.F.: Set of test problems for minimum length connection networks, ACM/SIGMAP Newslett. **15**, 48–51 (1973).
21. Van Laarhoven, J.W., and Anstreicher, K.M.: Geometric conditions for Euclidean Steiner trees in R^d, Computational Geometry. Theory and Applications **46(5)**, 520–531 (2013).
22. Warme, D.M., Winter, P. and Zachariasen, M.: Exact algorithms for plane Steiner tree problems: a computational study. In Advances in Steiner Trees, D.Z. Du, J.M. Smith and J.H. Rubinstein eds., Kluwer Academic Publishers, 81–116 (2000).
23. Winter, P., Zachariasen, M.: Euclidean Steiner minimum trees: An improved exact algorithm. Networks **30**, 149–166 (1997).