Math. Prog. Comp. (2017) 9:135–202
DOI 10.1007/s12532-016-0110-1

FULL LENGTH PAPER

# Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm

**Stefan Hougardy**[1] · **Jannik Silvanus**[1] · **Jens Vygen**[1]

Received: 21 May 2015 / Accepted: 10 June 2016 / Published online: 14 July 2016
© Springer-Verlag Berlin Heidelberg and The Mathematical Programming Society 2016

**Abstract** We present a new exact algorithm for the Steiner tree problem in edge-weighted graphs. Our algorithm improves the classical dynamic programming approach by Dreyfus and Wagner. We achieve a significantly better practical performance via pruning and future costs, a generalization of a well-known concept to speed up shortest path computations. Our algorithm matches the best known worst-case run time and has a fast, often superior, practical performance: on some large instances originating from VLSI design, previous best run times are improved upon by orders of magnitudes. We are also able to solve larger instances of the $d$-dimensional rectilinear Steiner tree problem for $d \in \{3, 4, 5\}$, whose Hanan grids contain up to several millions of edges.

**Keywords** Graph algorithms · Steiner tree problem · Dynamic programming · Exact algorithm

**Mathematics Subject Classification** 05C85 · 68W35 · 90C27 · 90C39

## 1 Introduction

We consider the well-known Steiner tree problem (in graphs): Given an undirected graph $G$, costs $c : E(G) \to \mathbb{R}_{\geq 0}$ and a terminal set $R \subseteq V(G)$, find a tree $T$

✉ Jannik Silvanus
silvanus@or.uni-bonn.de

Stefan Hougardy
hougardy@or.uni-bonn.de

Jens Vygen
vygen@or.uni-bonn.de

[1] Research Institute for Discrete Mathematics, University of Bonn, Bonn, Germany

in $G$ such that $R \subseteq V(T)$ and $c(E(T))$ is minimum. The decision version of the Steiner tree problem is one of the classical NP-complete problems [22]; it is even NP-complete in the special case that $G$ is bipartite with $c \equiv 1$. Furthermore, it is NP-hard to approximate the Steiner tree problem within a factor of $\frac{96}{95}$ [6]. The currently best known approximation algorithm by Byrka et al. [5] uses polyhedral methods to achieve a 1.39-approximation. The Steiner tree problem has many applications, in particular in VLSI design [20], where electrical connections are realized by Steiner trees.

From now on, we will refer to $|V(G)|$ by $n$, $|E(G)|$ by $m$ and $|R|$ by $k$. Dreyfus and Wagner [9] applied dynamic programming to the Steiner tree problem to obtain an exact algorithm with a run time of $\mathcal{O}(n(n \log n + m) + 3^k n + 2^k n^2)$ if implemented using Fibonacci heaps [14]. In 1987, Erickson, Monma and Veinott [11] improved the run time to $\mathcal{O}(3^k n + 2^k(n \log n + m))$ using a very similar approach. In 2006, Fuchs et al. [15] proposed an algorithm with a run time of $\mathcal{O}((2+\delta)^k n^{(\ln(\frac{1}{\delta})/\delta)^\zeta})$ for every sufficiently small $\delta > 0$ and $\zeta > \frac{1}{2}$, improving the exponential dependence on $k$ from $3^k$ to $(2+\delta)^k$. Vygen [33] developed an algorithm with a worst-case run time of $\mathcal{O}(nk2^{k+\log_2(k)\log_2(n)})$, which is the fastest known algorithm if $f(n) < k < g(n)$ for some $f(n) = \text{polylog}(n)$ and $g(n) = \frac{n}{2} - \text{polylog}(n)$. However, for $k < 4 \log n$, the run time obtained by Erickson, Monma and Veinott [11] is still the best known. See [33] for a more detailed analysis of the run times mentioned above.

For graphs with treewidth $t$, one can solve the Steiner tree problem in time $\mathcal{O}(n2^{\mathcal{O}(t)})$ [4]. An implementation of this algorithm was evaluated in [12]. Polzin and Vahdati Daneshmand [26] proposed an algorithm with a worst-case run time of $\mathcal{O}(n2^{p \log p + 3p + \log p})$ where $p$ is a parameter closely related to the pathwidth of $G$. They use this algorithm as a subroutine in their successful reduction-based Steiner tree solver [24,32].

Except for the last mentioned algorithm, these results have played a very limited role in practice. Instead, empirically successful algorithms rely on preprocessing and reduction techniques, heuristics and branching: First, reductions [3,10,25,31] are applied to reduce the size of the graph and the number of terminals, guaranteeing that optimum solutions of the reduced instance correspond to optimum solutions of the original instance. These reductions are not limited to simple local edge elimination tests, but may also rely on linear programming formulations and optimum solutions of partial instances. Primal and dual [2,35] heuristics yield good upper and lower bounds, in many cases even resulting in a provably optimum solution. If these methods do not already solve the instance, enumerative algorithms are used. To this end, various authors [2,7,23] perform branch and cut. However, the solver by Polzin and Vahdati Daneshmand [24,32], which achieved the best results so far, uses a branch and bound approach, where high effort is put into single branching nodes.

We propose a dynamic programming based algorithm with a worst-case run time of $\mathcal{O}(3^k n + 2^k(n \log n + m))$, matching the best known bound for small $k$, and which is fast in practice. Good practical performance is achieved by effectively pruning partial solutions and using *future cost* estimates. The latter are motivated by the similarity of our algorithm with Dijkstra's algorithm [8] and the well-known speed-up technique for Dijkstra's algorithm (first described by Hart, Nilsson and Raphael [17]) which uses reduced edge costs. More precisely, given an instance $(G, c, s, t)$ of the shortest path

problem, where we assume $G$ to be a directed graph, we use a feasible potential $\pi$, which is a function $\pi : V(G) \to \mathbb{R}_{\geq 0}$ with

$$\pi(t) = 0 \tag{1}$$

and

$$\pi(v) \leq \pi(w) + c((v, w)) \tag{2}$$

for all $(v, w) \in E(G)$. Define reduced costs $c_\pi$ by

$$c_\pi(e) := c(e) + \pi(w) - \pi(v) \geq 0 \tag{3}$$

for every $e = (v, w) \in E(G)$. Then, run Dijkstra's algorithm on the instance $(G, c_\pi, s, t)$. The numbers $\pi(v)$ are lower bounds on the distance from $v$ to $t$ (we also say that $\pi(v)$ estimates the *future cost* at $v$). Moreover, the cost of every $s$-$t$-path changes by the same amount when going from $c$ to $c_\pi$, namely $-\pi(s)$, so any shortest $s$-$t$-path in $(G, c_\pi)$ is a shortest $s$-$t$-path in $(G, c)$. If the future costs are good lower bounds, only vertices close to a shortest path will be labeled by Dijkstra's algorithm before $t$ is labeled permanently (i.e., the distance to $t$ is known) and the algorithm can be stopped. This can lead to huge speedups.

The rest of this paper is organized as follows: In Sect. 2, we generalize this future cost idea from paths to Steiner trees and describe our algorithm. Examples of future cost estimates are given in Sect. 3. Section 4 introduces a pruning technique to further improve practical performance. Section 5 contains implementation details and computational results.

## 2 The algorithm

Now, let $(G, c, R)$ be an instance of the Steiner tree problem, where $G$ is an undirected graph, $c : E(G) \to \mathbb{R}_{\geq 0}$ is a cost function on the edges and $R$ is the set of terminals to be connected. As usual, for a set $X \subseteq V(G)$, we denote by $\mathrm{smt}(X)$, short for Steiner minimal tree, the cost of an optimum Steiner tree for the terminal set $X$. Our algorithm uses an arbitrary root terminal $r_0 \in R$. We will call the terminals in $R \backslash \{r_0\}$ source terminals. The algorithms by Dreyfus and Wagner [9] and Erickson et al. [11] as well as our algorithm use dynamic programming to compute $\mathrm{smt}(\{v\} \cup I)$ for $(v, I) \in V(G) \times 2^{R \backslash \{r_0\}}$. Then, at termination, $\mathrm{smt}(\{r_0\} \cup (R \backslash \{r_0\}))$ is the cost of an optimum Steiner tree.

The former two algorithms work as follows: They consider all $I \subseteq R \backslash \{r_0\}$ in order of non-decreasing cardinality and compute $\mathrm{smt}(\{v\} \cup I)$ for all $v \in V(G)$. This way, it is guaranteed that when computing $\mathrm{smt}(\{v\} \cup I)$, the values $\mathrm{smt}(\{w\} \cup I')$ for all $w \in V(G)$ and $I' \subset I$ are already known. However, this leads to an exponential best case run time and memory consumption of $\Omega(2^k n)$. In contrast, our new algorithm considers all subsets of source terminals simultaneously, using a labeling technique

similar to Dijkstra's algorithm. This way, we do not necessarily have to compute $\text{smt}(\{v\} \cup I)$ for all pairs $(v, I)$.

Our new algorithm labels from the source terminals towards the root $r_0$. More precisely, the algorithm labels elements of $V(G) \times 2^{R \setminus \{r_0\}}$. Each label $(v, I)$ represents the best Steiner tree for $\{v\} \cup I$ found so far. As in Dijkstra's algorithm, each iteration selects one label $(v, I)$ and declares it to be permanent. Each time a label $(v, I)$ becomes permanent, all neighbors $w$ of $v$ are checked and updated if the Steiner tree represented by $(v, I)$ plus the edge $\{v, w\}$ leads to a better solution for $(w, I)$ than previously known. This operation is well-known from Dijkstra's algorithm. In addition, for all sets $J \subseteq (R \setminus \{r_0\}) \setminus I$ it is checked whether the Steiner trees for $(v, I)$ and $(v, J)$ combined to a tree for $(v, I \cup J)$ lead to a better solution than previously known.

To allow a simpler presentation, we restrict ourselves to instances without edges of zero cost, as these can be contracted in a trivial preprocessing step.

Now, we introduce the notion of valid lower bounds, which are used by the algorithm to estimate the future cost of a label $(v, I)$.

**Definition 1** Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. A function $\mathcal{L} : V(G) \times 2^R \to \mathbb{R}_{\geq 0}$ is called a *valid lower bound* if

$$\mathcal{L}(r_0, \{r_0\}) = 0$$

and

$$\mathcal{L}(v, I) \leq \mathcal{L}(w, I') + \text{smt}((I \setminus I') \cup \{v, w\})$$

for all $v, w \in V(G)$ and $\{r_0\} \subseteq I' \subseteq I \subseteq R$.

Note that the values $\mathcal{L}(v, I)$ for $r_0 \notin I$ do not affect whether $\mathcal{L}$ is a valid lower bound. Also note that by choosing $I' = \{r_0\}$ and $w = r_0$, we have $\mathcal{L}(v, I) \leq \text{smt}(I \cup \{v\})$, so a valid lower bound by definition indeed is a lower bound on the cost of an optimum Steiner tree. Moreover, if $e = \{v, w\} \in E(G)$ is an edge, by choosing $I' = I$, we have

$$\mathcal{L}(v, I) \leq \mathcal{L}(w, I) + \text{smt}(\{v, w\}) \leq \mathcal{L}(w, I) + c(e).$$

This shows that valid lower bounds generalize feasible potentials as defined in (1) and (2). In fact, our algorithm applied to the case $|R| = 2$ is identical to Dijkstra's algorithm using future costs in the very same way.

Our new algorithm is described in Fig. 1. For each label $(v, I) \in V(G) \times 2^{R \setminus \{r_0\}}$, the algorithm stores the cost $l(v, I) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ of the cheapest Steiner tree for $\{v\} \cup I$ found so far as well as backtracking data $b(v, I) \subseteq V(G) \times 2^{R \setminus \{r_0\}}$ which is used to construct the Steiner tree represented by this label. If $b(v, I)$ is not empty, it will always either be of the form $b(v, I) = \{(w, I)\}$ where $w$ is a neighbor of $v$ or of the form $b(v, I) = \{(v, I_1), (v, I_2)\}$ where $I_1$ and $I_2$ form a partition of $I$ (into nonempty disjoint sets). In the first case, i.e., $b(v, I) = \{(w, I)\}$, the Steiner tree represented by the label $(v, I)$ contains exactly one edge incident to $v$, which is $\{v, w\}$. In the second case, i.e., $b(v, I) = \{(v, I_1), (v, I_2)\}$, the Steiner tree represented by $(v, I)$ can be

**Dijkstra-Steiner Algorithm**

**Input** : A connected undirected graph $G$, costs $c : E(G) \to \mathbb{R}_{>0}$, a terminal set $R \subseteq V(G)$, a root terminal $r_0 \in R$, and (an oracle to compute) a valid lower bound $\mathcal{L} : V(G) \times 2^R \to \mathbb{R}_{\geq 0}$.

**Output**: The edge set of an optimum Steiner tree for $R$ in $G$.

```
1  l(v, I)    := ∞ for all (v, I) ∈ V(G) × 2^{R\{r_0}};
2  l(s, {s})  := 0 for all s ∈ R \ {r_0};
3  l(v, ∅)    := 0 for all v ∈ V(G);
4  b(v, I)    := ∅ for all (v, I) ∈ V(G) × 2^{R\{r_0}};
5  N          := {(s, {s}) | s ∈ R \ {r_0}};
6  P          := V(G) × {∅};
7  while (r_0, R \ {r_0}) ∉ P do
8      Choose (v, I) ∈ N minimizing l(v, I) + L(v, R \ I);
9      N := N \ {(v, I)};
10     P := P ∪ {(v, I)};
11     for all edges e = {v, w} incident to v do
12         if l(v, I) + c(e) < l(w, I) and (w, I) ∉ P then
13             l(w, I) := l(v, I) + c(e);
14             b(w, I) := {(v, I)};
15             N       := N ∪ {(w, I)};
16         end
17     end
18     for all ∅ ≠ J ⊆ (R \ {r_0}) \ I with (v, J) ∈ P do
19         if l(v, I) + l(v, J) < l(v, I ∪ J) and (v, I ∪ J) ∉ P then
20             l(v, I ∪ J) := l(v, I) + l(v, J);
21             b(v, I ∪ J) := {(v, I), (v, J)};
22             N           := N ∪ {(v, I ∪ J)};
23         end
24     end
25 end
26 return backtrack(r_0, R \ {r_0});

   Procedure backtrack(v, I)
27     if b(v, I) = {(w, I)} then
28         return {{v, w}} ∪ backtrack(w, I);
29     else
30         return ⋃_{(w,I')∈b(v,I)} backtrack(w, I');
31     end
```

**Fig. 1** The Dijkstra–Steiner Algorithm

split into two Steiner trees for the terminal sets $I_1 \cup \{v\}$ and $I_2 \cup \{v\}$, where $I_1$ and $I_2$ form a partition of $I$. Of course, the Steiner trees for $I_1 \cup \{v\}$ and $I_2 \cup \{v\}$ may in turn be split recursively into further trees. To be precise, it may happen that the subgraph of $G$ corresponding to some label $(v, I)$ contains cycles. However, since we ruled out edges of zero cost, this can only be the case as long as the label is not permanent, because—as we will show—permanent labels correspond to optimal Steiner trees.

We incorporate the valid lower bound $\mathcal{L}$ into the algorithm as follows. First note that instead of running Dijkstra's algorithm on reduced edge costs $c_\pi$ as defined in (3), one can equivalently apply the following modification. For a vertex $v \in V(G)$, we denote by $l(v)$ the cost of a label $v$ in Dijkstra's algorithm, i.e., the cost of a shortest path connecting $s$ with $v$ found so far. Then, in each iteration, instead of choosing a non-

permanent vertex $v$ minimizing $l(v)$ to become permanent, choose a non-permanent vertex minimizing $l(v) + \pi(v)$. We generalize this approach by always choosing a non-permanent label minimizing $l(v, I) + \mathcal{L}(v, R \setminus I)$.

By $P \subseteq V(G) \times 2^{R \setminus \{r_0\}}$ we denote the set of permanently labeled elements. We also maintain a set $N$ of non-permanent labels which are candidates to be selected. The set $N$ contains exactly the labels $(v, I) \in (V(G) \times 2^{R \setminus \{r_0\}}) \setminus P$ with $l(v, I) < \infty$. Note that $\mathcal{L} \equiv 0$ is always a valid lower bound, which may serve as an example to help understanding the algorithm. Other examples of valid lower bounds will be discussed in Sect. 3.

**Theorem 2** *The Dijkstra–Steiner algorithm works correctly: Given an instance $(G, c, R)$ of the Steiner tree problem, $r_0 \in R$, and a valid lower bound $\mathcal{L}$, it always returns the edge set of an optimum Steiner tree for $R$ in $G$.*

*Proof* We will prove that the following invariants always hold when line 8 is executed:

(a) For each $(v, I) \in N \cup P$:

(a1) $l(v, I) = \begin{cases} c(\{v, w\}) + l(w, I) & \text{if } b(v, I) = \{(w, I)\}, \\ \sum_{(v,J) \in b(v,I)} l(v, J) & \text{otherwise,} \end{cases}$

(a2) $I \cup \{v\} = \{v\} \cup \bigcup_{(w,J) \in b(v,I)} J$,

(a3) $b(v, I) \subseteq P$, and `backtrack`$(v, I)$ returns the edge set $F$ of a connected subgraph of $G$ containing $\{v\} \cup I$ with $c(F) \leq l(v, I)$.

(b) For each $(v, I) \in P$:

$l(v, I) = \text{smt}(\{v\} \cup I)$.

(c) For each $(v, I) \in (V(G) \times 2^{R \setminus \{r_0\}}) \setminus P$:

(c1) $l(v, I) \geq \text{smt}(\{v\} \cup I)$,

(c2) If $I = \{v\}$, then $l(v, I) = 0$, otherwise

$l(v, I) \leq \min_{\{v,w\} \in E(G), (w,I) \in P} (l(w, I) + c(\{v, w\}))$ and

$l(v, I) \leq \min_{\emptyset \neq I' \subset I \text{ and } (v,I'), (v, I \setminus I') \in P} (l(v, I') + l(v, I \setminus I'))$.

(c3) If $l(v, I) = \text{smt}(\{v\} \cup I)$, then $(v, I) \in N$.

(d) $N$ is not empty and $N \cap P = \emptyset$.

Assuming (a)–(d), the correctness of the algorithm directly follows: Once we have $(r_0, R \setminus \{r_0\}) \in P$, (b) implies $l(r_0, R \setminus \{r_0\}) = \text{smt}(\{r_0\} \cup (R \setminus \{r_0\})) = \text{smt}(R)$. Furthermore, (a3) implies that the algorithm returns the edge set $F$ of a connected subgraph of $G$ containing $R$ with $c(F) \leq l(r_0, R \setminus \{r_0\}) = \text{smt}(R)$. Since there are no edges of zero cost, $F$ indeed is the edge set of a tree. Invariant (d) guarantees that in each iteration, a label $(v, I) \in N$ can be chosen and that the algorithm terminates, since $|P|$ is increased in each iteration.

Clearly, after line 6 these invariants hold. We have to prove that lines 8 to 24 preserve (a)–(d). Below, we first verify that (a) and (c) are preserved. Then, the main part of the proof shows that (b) is preserved. Finally, we will see that the latter argument also shows that (d) is preserved.

Let $(v, I)$ be the label chosen in line 8 in some iteration. Clearly, lines 8–24 preserve (a2), (a3), (c2) and (c3). If a label $l(w, I)$ or $l(v, I \cup J)$ is decreased, it cannot be permanent, so (a1) is maintained.

We now consider (c1). Since (c) held before the current iteration, we have $l(v, I) \geq \mathrm{smt}(\{v\} \cup I)$. This directly implies

$$\begin{aligned} l(v, I) + c(\{v, w\}) &\geq \mathrm{smt}(\{v\} \cup I) + c(\{v, w\}) \\ &\geq \mathrm{smt}(\{v, w\} \cup I) \\ &\geq \mathrm{smt}(\{w\} \cup I), \end{aligned}$$

so line 13 does not destroy (c1). Also, if $\emptyset \neq J \subseteq (R \backslash \{r_0\}) \backslash I$ is a set chosen in line 18 leading to the change of $l(v, I \cup J)$, we have

$$\begin{aligned} l(v, I \cup J) &= l(v, I) + l(v, J) \\ &\geq \mathrm{smt}(\{v\} \cup I) + \mathrm{smt}(\{v\} \cup J) \\ &\geq \mathrm{smt}(\{v\} \cup I \cup J), \end{aligned}$$

so (c1) indeed is preserved.

In order to prove that (b) is preserved, we now show that $l(v, I) \leq \mathrm{smt}(\{v\} \cup I)$, which together with (c1) yields the desired result. Since $l(v, \{v\}) = 0 = \mathrm{smt}(\{v\})$, we can assume $I \neq \{v\}$. Let $T$ be an optimum Steiner tree for $\{v\} \cup I$ in $G$. Then, all leaves of $T$ are contained in $\{v\} \cup I$. For a vertex $w \in V(T)$, let $T_w$ be the subtree of $T$ containing all vertices $x$ for which the unique $x$-$v$-path in $T$ contains $w$.

We will now show that there is a vertex $w \in V(T)$, a nonempty terminal set $I' \subseteq I \cap V(T_w)$ and a subtree $T'$ of $T_w$ such that

(I) $(w, I') \in N$,
(II) $l(w, I') = c(T') = \mathrm{smt}(\{w\} \cup I')$,
(III) $T'$ is a subtree of $T_w$ containing $I' \cup \{w\}$,
(IV) $T - T'$ is a tree containing $(I \backslash I') \cup \{v, w\}$.

Here, by $T - T'$, we refer to the graph $\big((V(T) \backslash V(T')) \cup \{w\}, E(T) \backslash E(T')\big)$. Assuming we have a triple $(w, I', T')$ satisfying (I)–(IV), $l(v, I) \leq \mathrm{smt}(\{v\} \cup I)$ can easily be proved: Since $\mathcal{L}$ is a valid lower bound, we have

$$\begin{aligned} \mathcal{L}(w, R \backslash I') &\leq \mathcal{L}(v, R \backslash I) + \mathrm{smt}((I \backslash I') \cup \{v, w\}) \\ &\leq \mathcal{L}(v, R \backslash I) + c(T - T') \\ &= \mathcal{L}(v, R \backslash I) + c(T) - c(T'). \end{aligned} \tag{4}$$

Adding (II) and (4) yields

$$l(w, I') + \mathcal{L}(w, R \backslash I') \leq c(T) + \mathcal{L}(v, R \backslash I).$$

By (I) and the choice of $(v, I)$ in line 8 we have

$$l(v, I) + \mathcal{L}(v, R \backslash I) \leq l(w, I') + \mathcal{L}(w, R \backslash I'),$$

so

$$l(v, I) \leq c(T).$$

It remains to be shown that there is such a triple $(w, I', T')$. We call $w \in V(T)$ *proper* if $(w, I \cap V(T_w)) \in P$ before the execution of line 24. If we have a leaf $w \in V(T)\backslash\{v\}$ which is not proper, we set $I' = \{w\}$ and $T' = (\{w\}, \emptyset)$, satisfying (I)–(IV). Note that here (I) follows from (c3). Otherwise, since $v$ is not proper, there is a vertex $w$ in $T$ which is not proper but all neighbors $w_i$ of $w$ in $T_w$ are proper. We define

$$\mathcal{J} = \begin{cases} \{I \cap V(T_{w_i}) | w_i \text{ is neighbor of } w \text{ in } T_w\} \cup \{\{w\}\} & \text{if } w \in I, \\ \{I \cap V(T_{w_i}) | w_i \text{ is neighbor of } w \text{ in } T_w\} & \text{if } w \notin I. \end{cases}$$

Note that $\mathcal{J}$ is a partition of $I \cap V(T_w)$. Let $\mathcal{J}'$ be an inclusion-wise minimal subset of $\mathcal{J}$ such that $\left(w, \bigcup_{J \in \mathcal{J}'} J\right) \notin P$. Since $w$ is not proper and $(w, \emptyset) \in P$, $\mathcal{J}'$ exists and $\mathcal{J}'$ is not empty. For $J \in \mathcal{J}'$, let $T_J$ be the unique minimum connected subgraph of $T$ such that $(\{w\} \cup J) \subseteq V(T_J)$. By optimality of $T$, $T_J$ is an optimum Steiner tree for $(\{w\} \cup J)$, as the only vertex in $T_J$ which is incident to edges in $E(T)\backslash E(T_J)$ is $w$. We define

(i) $I' = \bigcup_{J \in \mathcal{J}'} J$,
(ii) $V(T') = \bigcup_{J \in \mathcal{J}'} V(T_J)$ and
(iii) $E(T') = \bigcup_{J \in \mathcal{J}'} E(T_J)$.

See Fig. 2 for an illustration of this setting.

The conditions (III) and (IV) are satisfied by construction. We now show (II), which due do (c3) also implies (I). First, we deal with the case $|\mathcal{J}'| = 1$. If $\mathcal{J}' = \{\{w\}\}$, by (c2) clearly $l(w, I') = 0 = c(T')$. If $\mathcal{J}' = \{I \cap V(T_{w_i})\}$ for some neighbor $w_i$ of $w$ in $T_w$, by (c2), (b), and the fact that $w_i$ is proper, we have

$$\begin{aligned} l(w, I') &\leq l(w_i, I') + c(\{w_i, w\}) \\ &= \text{smt}(\{w_i\} \cup I') + c(\{w_i, w\}) \\ &= c(T_{w_i}) + c(\{w_i, w\}) \\ &= c(T'). \end{aligned}$$

Otherwise, i.e., $|\mathcal{J}'| > 1$, choose an element $J \in \mathcal{J}'$ and see again by (c2), (b) and the minimality of $\mathcal{J}'$ that

$$\begin{aligned} l(w, I') &\leq l(w, I'\backslash J) + l(w, J) \\ &= \text{smt}(\{w\} \cup (I'\backslash J)) + \text{smt}(\{w\} \cup J) \\ &= c(T' - T_J) + c(T_J) \\ &= c(T'). \end{aligned}$$

As $T$ is an optimum Steiner tree for $\{v\} \cup I$, by (IV) we get $c(T') = \text{smt}(\{w\} \cup I')$, together with (c1) showing (II). By (c3), $(w, I') \in N$, so (I) holds as well, completing the proof that (b) is preserved by lines 8 to 24.

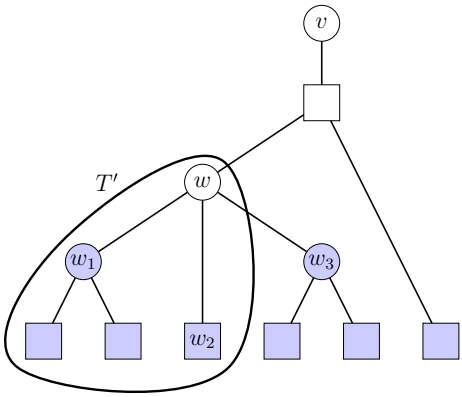


**Fig. 2** A possible configuration with $\mathcal{J}' = \{I \cap V(T_{w_1}), I \cap V(T_{w_2})\}$. Vertices in $I$ are drawn as *squares*, proper vertices are drawn in *blue*

To see that (d) is maintained, note that applying the previous argument to $(r_0, R\backslash\{r_0\})$ instead of $(v, I)$ always yields a label $(w, I') \in N$, so $N$ is not empty. Also, whenever a label $(v, I)$ is inserted to $P$, it is removed from $N$. By (b), from then on $l(v, I)$ cannot be changed, so $(v, I)$ is never inserted again into $N$. □

Note that in line 8, one can actually choose any label $(v, I) \in N$ with $l(v, I) + \mathcal{L}(v, R\backslash I) \leq l(w, J) + \mathcal{L}(w, R\backslash J)$ for all $(w, J) \in (V(G) \times 2^I) \cap N$. This is a generalization of the choice as specified in the algorithm. However, in our implementation, we always choose a label minimizing $l(v, I) + \mathcal{L}(v, R\backslash I)$.

**Theorem 3** *The Dijkstra–Steiner algorithm can be implemented to run in* $\mathcal{O}(3^k n + 2^k(n \log n + m) + 2^k n f_{\mathcal{L}})$ *time, where* $n = |V(G)|$, $m = |E(G)|$, $k = |R|$, *and* $f_{\mathcal{L}}$ *is an upper bound on the time required to evaluate* $\mathcal{L}$.

*Proof* Since $|P|$ increases in each iteration, we have at most $n2^{k-1}$ iterations. We use a Fibonacci heap [14] to store all labels $(v, I) \in N$, which allows updates in constant amortized time. Since the heap contains at most $2^{k-1}n$ elements, each execution of line 8 takes $\mathcal{O}(\log(2^{k-1}n)) = \mathcal{O}(k + \log(n))$ amortized time. Line 12 is executed at most $2^k m$ times and each execution takes $O(1)$ amortized time. Furthermore, there are exactly $3^{k-1}$ pairs $I, J \subseteq R\backslash\{r_0\}$ with $I \cap J = \emptyset$, since every element in $R\backslash\{r_0\}$ can either be contained in $I$, $J$ or $(R\backslash\{r_0\})\backslash(I \cup J)$, independently of the others. Thus, line 18–24 take $\mathcal{O}(3^{k-1}n)$ time. By caching values of $\mathcal{L}$, we can achieve that we query $\mathcal{L}$ at most once for each label, resulting in an additional run time of $\mathcal{O}(2^k n f_{\mathcal{L}})$. The run time of the backtracking clearly is dominated by the previous tasks, since `backtrack` is called at most $\mathcal{O}(kn)$ times and requires effort linear in the size of its output. We get a total run time of

$$\mathcal{O}\left(2^k n(\log n + k) + 2^k m + 3^k n + 2^k n f_{\mathcal{L}}\right) = \mathcal{O}\left(3^k n + 2^k(n \log n + m) + 2^k n f_{\mathcal{L}}\right),$$

since $2^k k = \mathcal{O}(3^k)$. □

In Sect. 3, we will see that there are non-trivial valid lower bounds $\mathcal{L}$ which can be used in the algorithm while still achieving a worst-case run time of $\mathcal{O}(3^k n + 2^k(n \log n + m))$, matching the bound of [11].

The memory consumption of our algorithm is obviously $\mathcal{O}(2^k n)$.

## 3 Lower bounds

Roughly speaking, the larger the valid lower bound $\mathcal{L}$ is, the faster our algorithm will be. Before we describe examples of valid lower bounds, we note:

**Proposition 4** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $\mathcal{L}$, $\mathcal{L}'$ be valid lower bounds. Furthermore, define* $\max(\mathcal{L}, \mathcal{L}') : V(G) \times 2^R \to \mathbb{R}_{\geq 0}$ *by*

$$\max(\mathcal{L}, \mathcal{L}')(v, I) := \max(\mathcal{L}(v, I), \mathcal{L}'(v, I))$$

*for $v \in V(G)$ and $\{r_0\} \subseteq I \subseteq R$.*

*Then,* $\max(\mathcal{L}, \mathcal{L}')$ *also is a valid lower bound.*

*Proof* Let $v, w \in V(G)$ and $I' \subseteq I \subseteq R$. Then

$$\begin{aligned}\mathcal{L}(v, I) &\leq \mathcal{L}(w, I') + \operatorname{smt}((I \backslash I') \cup \{v, w\}) \\ &\leq \max(\mathcal{L}(w, I'), \mathcal{L}'(w, I')) + \operatorname{smt}((I \backslash I') \cup \{v, w\})\end{aligned}$$

and

$$\begin{aligned}\mathcal{L}'(v, I) &\leq \mathcal{L}'(w, I') + \operatorname{smt}((I \backslash I') \cup \{v, w\}) \\ &\leq \max(\mathcal{L}(w, I'), \mathcal{L}'(w, I')) + \operatorname{smt}((I \backslash I') \cup \{v, w\}),\end{aligned}$$

so

$$\max(\mathcal{L}(v, I), \mathcal{L}'(v, I)) \leq \max(\mathcal{L}(w, I'), \mathcal{L}'(w, I')) + \operatorname{smt}((I \backslash I') \cup \{v, w\}).$$

□

Proposition 4 allows the combination of arbitrary valid lower bounds.

Now, we present three types of valid lower bounds. A simple valid lower bound can be obtained by considering terminal sets of bounded cardinality:

**Definition 5** Let $(G, c, R)$ be an instance of the Steiner tree problem, $r_0 \in R$ and let $j \geq 1$ be an integer. Then, $\mathcal{L}_j$ is defined as

$$\mathcal{L}_j(v, I) = \max_{\{r_0\} \subseteq J \subseteq I \cup \{v\}, |J| \leq j+1} \operatorname{smt}(J)$$

for $v \in V(G)$ and $\{r_0\} \subseteq I \subseteq R$. For $v \in V(G)$ and $I \subseteq R$ with $r_0 \notin I$, set $\mathcal{L}_j(v, I) = 0$.

**Lemma 6** *Let $(G, c, R)$ be an instance of the Steiner tree problem, $r_0 \in R$ and let $j \geq 1$ be an integer. Then, $\mathcal{L}_j$ is a valid lower bound. Furthermore, we can implement $\mathcal{L}_j$ such that after a preprocessing time of $\mathcal{O}(3^k + (2k)^{j-1} n + k^{j-1}(n \log n + m))$, we can evaluate $\mathcal{L}_j(v, I)$ for every $v \in V(G)$ and $\{r_0\} \subseteq I \subseteq R$ in time $\mathcal{O}(|I|^{j-1})$, where $n = |V(G)|$, $m = |E(G)|$ and $k = |R|$.*

*Proof* Let $v, w \in V(G)$ and $\{r_0\} \subseteq I' \subseteq I \subseteq R$. To prove that $\mathcal{L}_j$ is a valid lower bound, we have to show

$$\max_{\substack{\{r_0\} \subseteq J \subseteq I \cup \{v\} \\ |J| \leq j+1}} \operatorname{smt}(J) \leq \max_{\substack{\{r_0\} \subseteq J' \subseteq I' \cup \{w\} \\ |J'| \leq j+1}} \operatorname{smt}(J') + \operatorname{smt}((I \setminus I') \cup \{v, w\}).$$

Consider the map $f : I \cup \{v\} \to I' \cup \{w\}$ with $f(x) = w$ for $x \notin I'$ and $f(x) = x$ otherwise. Let $J$ be a set with $\{r_0\} \subseteq J \subseteq I \cup \{v\}$ and $|J| \leq j+1$. Set $J' = \{f(x) : x \in J\}$. Then, clearly $\{r_0\} \subseteq J' \subseteq I' \cup \{w\}$ and $|J'| \leq |J| \leq j+1$. Moreover,

$$\operatorname{smt}(J) \leq \operatorname{smt}(J') + \operatorname{smt}((I \setminus I') \cup \{v, w\}).$$

To achieve the given run time, we first compute $\operatorname{smt}(\{v\} \cup I)$ for all $v \in V(G)$ and $I \subseteq R$ with $|I \setminus \{r_0\}| \leq j-1$ in time $\mathcal{O}((2k)^{j-1}n + k^{j-1}(n \log n + m))$ using a modified variant of the Dijkstra–Steiner algorithm. More precisely, we do not use a lower bound, do not use a root terminal and consider terminal sets of increasing cardinality, very similar to [11]. There are $\mathcal{O}(k^{j-1})$ sets $I$ with $I \subseteq R, |I \setminus \{r_0\}| \leq j-1$. Since we consider terminal sets of increasing cardinality one after another, we always have at most $n$ labels in the Fibonacci heap. As a set of cardinality $j$ has $2^j$ subsets, there are $\mathcal{O}((2k)^{j-1}n)$ updates of supersets.

Then, to evaluate $\mathcal{L}_j(v, I)$, we exploit

$$\begin{aligned}\mathcal{L}_j(v, I) &= \max_{\{r_0\} \subseteq J \subseteq I \cup \{v\}, |J| \leq j+1} \operatorname{smt}(J) \\ &= \max\left( \max_{J \subseteq I, |J| \leq j-1} \operatorname{smt}(J \cup \{v, r_0\}), \max_{\{r_0\} \subseteq J \subseteq I, |J| \leq j+1} \operatorname{smt}(J) \right),\end{aligned}$$

where the first expression can be computed in $\mathcal{O}(|I|^{j-1})$ time using the precomputed values $\operatorname{smt}(\{v, r_0\} \cup J)$ and the second expression does not depend on $v$ and can be computed in advance for every $I \subseteq R$ in $\mathcal{O}(3^k)$ time. □

Thus, for small $j$, e.g., $j \leq 3$, $\mathcal{L}_j$ can be efficiently computed.

We now present a more effective lower bound. We will use the notion of *1-trees*, which have long been studied [19] as a lower bound for the traveling salesman problem. Given a complete graph $H$ with metric edge costs and a special vertex $v \in V(H)$, a 1-tree for $v$ and $H$ is a spanning tree on $H - v$ together with two additional edges connecting $v$ with the tree. A tour in $H$ is a connected 2-regular subgraph of $H$. Since every tour consists of a path, which is a spanning tree, and a vertex connected to the endpoints of the path, every tour is a 1-tree. Thus, a 1-tree of minimum cost is a lower bound on the cost of a tour of minimum cost. Since such a tour of minimum cost is at most twice as expensive as a minimum Steiner tree, we can use 1-trees to get a lower bound for the Steiner tree problem.

For $v, w \in V(G)$, we denote by $d(v, w)$ the cost of a shortest path connecting $v$ and $w$ in $G$. Furthermore, for a set of vertices $X \subseteq V$, we denote by $G_X$ the *distance graph* of $X$, which is the subgraph of the metric closure of $G$ induced by $X$. Note that since the edge costs in $G_X$ are the costs of shortest paths with respect to positive edge

costs in $G$, the edge costs in $G_X$ are always metric. Moreover, for $X \subseteq V(G)$, we denote by $\mathrm{mst}(X)$ the cost of a minimum spanning tree in $G_X$.

**Definition 7** Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, the *1-tree bound* $\mathcal{L}_{1-\mathrm{tree}}$ is defined as

$$\mathcal{L}_{1-\mathrm{tree}}(v, I) = \min_{i,j \in I : i \neq j \vee |I|=1} \frac{d(v,i) + d(v,j)}{2} + \frac{\mathrm{mst}(I)}{2}$$

for $v \in V(G)$ and $\{r_0\} \subseteq I \subseteq R$. For $v \in V(G)$ and $I \subseteq R \backslash \{r_0\}$, we set $\mathcal{L}_{1-\mathrm{tree}}(v, I) = 0$.

**Lemma 8** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, $\mathcal{L}_{1-\mathrm{tree}}$ is a valid lower bound.*

*Proof* Let $v, w \in V(G)$ and $\{r_0\} \subseteq I' \subseteq I \subseteq R$. We will show

$$2\mathcal{L}_{1-\mathrm{tree}}(v, I) \leq 2\mathcal{L}_{1-\mathrm{tree}}(w, I') + 2\,\mathrm{smt}((I \backslash I') \cup \{v, w\}),$$

which translates to

$$\begin{aligned} &\min_{i,j \in I : i \neq j \vee |I|=1} (d(v,i) + d(v,j)) + \mathrm{mst}(I) \\ &\quad \leq \min_{i,j \in I' : i \neq j \vee |I'|=1} (d(w,i) + d(w,j)) + \mathrm{mst}(I') + 2\,\mathrm{smt}((I \backslash I') \cup \{v, w\}). \end{aligned}$$

Consider a minimum spanning tree $T_1$ in $G_{I'}$ and a Steiner tree $T_2$ for $(I \backslash I') \cup \{v, w\}$. Furthermore, let $j_1, j_2 \in I'$ with $j_1 \neq j_2 \vee |I'| = 1$. This setting is illustrated in Fig. 3.

First, we construct a tour $C$ in $G_{(I \backslash I') \cup \{v,w\}}$ of cost at most $2c(T_2)$ using the standard double tree argument: If we double each edge in $T_2$, the graph becomes Eulerian and we can find a Eulerian cycle. If we visit the vertices in the order the Eulerian cycle visits them and skip already visited vertices, we obtain a tour of at most the same cost exploiting that the edge costs in the distance graph are metric.

We can decompose the tour into two paths $P_1$ and $P_2$ in $G_{I \cup \{v,w\}}$ with endpoints $v$ and $w$ such that $(I \backslash I') \cup \{v, w\} = V(P_1) \cup V(P_2)$ and $c(P_1) + c(P_2) = c(C)$. Now, for $i \in \{1, 2\}$, we define $P_i' = (V(P_i) \cup \{j_i\}, E(P_i) \cup \{\{w, j_i\}\})$, which is the path obtained by appending the edge $\{w, j_i\}$ to $P_i$.

If $I_1 := (I \backslash I') \cap V(P_1)$ is empty, let $j_1'$ be $j_1$, else, let $j_1'$ be the first terminal in $I_1$ when traversing $P_1$ from $v$ to $w$.

Similarly, if $I_2 := (I \backslash I') \backslash V(P_1) \subseteq V(P_2)$ is empty, set $j_2' = j_2$, else let $j_2'$ be the first terminal in $I_2$ when traversing $P_2$ from $v$ to $w$.

Since $I_1$ and $I_2$ are disjoint and do not contain $j_1, j_2 \in I'$, we have

$$j_1' = j_2' \implies (j_1 = j_1' = j_2' = j_2 \wedge I \backslash I' = \emptyset),$$

which together with

$$j_1 = j_2 \implies |I'| = 1$$

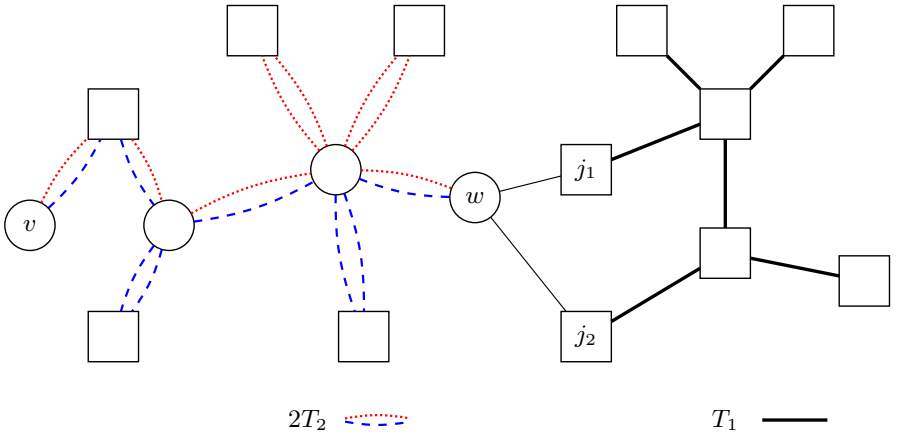


**Fig. 3** The minimum spanning tree $T_1$, the double Steiner tree $2T_2$ and the edges $\{w, j_1\}$ and $\{w, j_2\}$. Edges in $2T_2$ contributing to $P_1$ are colored *red* and edges contributing to $P_2$ are colored *blue*. Vertices in $I$ are drawn as *squares*

implies that

$$j_1' = j_2' \implies |I| = 1.$$

Then, let $Q_i$ be the path in $G_I$ obtained from the subpath of $P_i'$ from $j_i'$ to $j_i$ by skipping $w$. This is illustrated in Fig. 4. We have

$$d(v, j_1') + d(v, j_2') + d(Q_1) + d(Q_2) \leq d(w, j_1) + d(w, j_2) + 2c(T_2).$$

Also, we have $(I \setminus I') \cup \{j_1, j_2\} \subseteq V(Q_1) \cup V(Q_2)$ and clearly, we can find a spanning tree in $G_I$ with cost at most $d(Q_1) + d(Q_2) + d(T_1)$ by attaching $Q_1$ and $Q_2$ to $T_1$.

Therefore,

$$\begin{aligned}\min_{i,j \in I : i \neq j \vee |I|=1} (d(v, i) + d(v, j)) + \mathrm{mst}(I) &\leq d(v, j_1') + d(v, j_2') \\ + d(Q_1) + d(Q_2) + d(T_1) &\leq d(w, j_1) + d(w, j_2) + d(T_1) + 2c(T_2).\end{aligned}$$

□

Note that we could also use half the cost of a minimum spanning tree for $I \cup \{v\}$ to obtain a valid lower bound. However, the 1-tree lower bound is always larger and hence leads to better run times.

**Lemma 9** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, we can implement $\mathcal{L}_{1-\mathrm{tree}}$ such that after a preprocessing time of $\mathcal{O}(k(n \log n + m) + 2^k k^2)$, we can evaluate $\mathcal{L}_{1-\mathrm{tree}}$ for every $v \in V(G)$ and $\{r_0\} \subseteq I \subseteq R$ in time $\mathcal{O}(|I|)$, where $n = |V(G)|$, $m = |E(G)|$ and $k = |R|$.*

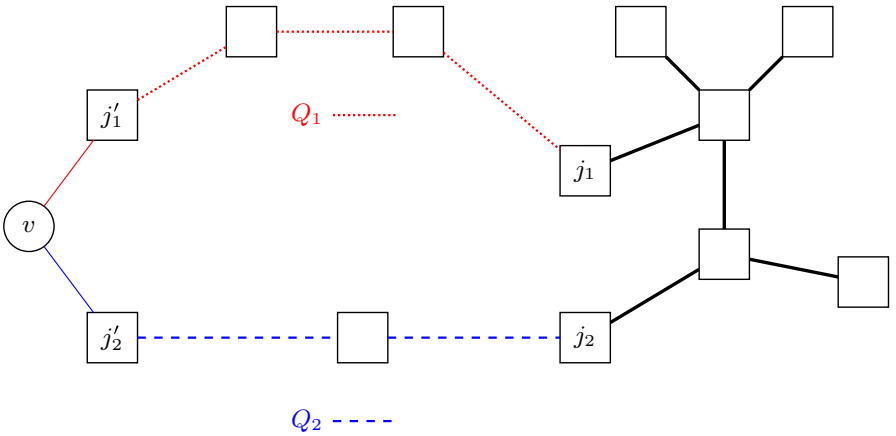


**Fig. 4** $j_1'$, $j_2'$, $Q_1$ and $Q_2$ in the same setting as in Fig. 3

*Proof* First, for every terminal $s \in R$, we compute $d(v, s)$ for all $v \in V(G)$ in $\mathcal{O}(n \log n + m)$ time using Dijkstra's algorithm implemented with a Fibonacci heap [14]. For every $I \subseteq R$, we compute $\text{mst}(I)$ in $\mathcal{O}(|I|^2)$ time using Prim's algorithm [28]. This results in a total preprocessing time of $\mathcal{O}(k(n \log n + m) + 2^k k^2)$. Clearly,

$$\min_{i,j \in I : i \neq j \vee |I|=1} (d(v, i) + d(v, j))$$

can be evaluated in $\mathcal{O}(|I|)$ time if $d(v, i)$ is known for all $v \in V(G)$ and $i \in I$. □

Of course, in practice we do not compute minimum spanning trees for all sets of terminals in advance, but compute them dynamically when needed.

**Theorem 10** *Let $j \in \mathbb{N}$ be a constant. Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, we can compute* $\text{smt}(R)$ *in time $\mathcal{O}(3^k n + 2^k(n \log n + m))$ using the Dijkstra–Steiner algorithm with $\mathcal{L} = \max(\mathcal{L}_j, \mathcal{L}_{1-\text{tree}})$.*

The 1-tree lower bound exploits the fact that 1-trees can be used to compute lower bounds on the minimum cost of a tour, which in turn is at most twice as expensive as a minimum cost Steiner tree. Using more preprocessing and evaluation time, we can eliminate the loss of approximating tours by 1-trees by using *optimum tours* to get lower bounds. While it may sound unreasonable to use optimum solutions for an NP-hard problem to speed up another algorithm, it turns out we can compute optimum tours for the union of sets of terminals and at most one vertex quite fast if there are only few terminals. This is due to the fact that the cost of an optimum tour in $G_{I \cup \{v\}}$ only depends on the distances from terminals to $v$ and shortest Hamiltonian paths with given endpoints in $G_I$, which can be computed in advance. For a set of vertices $X \subseteq V(G)$, we denote by $\text{tsp}(X)$ the minimum cost of a Hamiltonian circuit in $G_X$.

**Definition 11** Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, the *TSP bound* $\mathcal{L}_{\text{TSP}}$ is defined as

$$\mathcal{L}_{\text{TSP}}(v, I) = \frac{\text{tsp}(I \cup \{v\})}{2}$$

for $v \in V(G)$ and $I \subseteq R$.

**Lemma 12** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Then, $\mathcal{L}_{\text{TSP}}$ is a valid lower bound. Moreover, after a preprocessing time of $\mathcal{O}(k(n \log n + m) + 2^k k^3)$, we can evaluate $\mathcal{L}_{\text{TSP}}(v, I)$ in time $\mathcal{O}(|I|^2)$ for all $v \in V(G)$ and $I \subseteq R$.*

*Proof* Let $v, w \in V(G)$ and $\{r_0\} \subseteq I' \subseteq I \subseteq R$. We will show

$$2\mathcal{L}_{\text{TSP}}(v, I) \leq 2\mathcal{L}_{\text{TSP}}(w, I') + 2\,\text{smt}((I \setminus I') \cup \{v, w\}),$$

which is equivalent to

$$\text{tsp}(I \cup \{v\}) \leq \text{tsp}(I' \cup \{w\}) + 2\,\text{smt}((I \setminus I') \cup \{v, w\}).$$

First, we choose an optimal tour $C_1$ in $G_{I' \cup \{w\}}$. Then, we construct a tour $C_2$ in $G_{(I \setminus I') \cup \{v,w\}}$ of cost at most $2\,\text{smt}((I \setminus I') \cup \{v, w\})$ by doubling the edges of an optimum Steiner tree, finding a Eulerian walk and taking shortcuts. We have $I \cup \{v\} = V(C_1) \cup V(C_2)$ and $w \in V(C_1) \cap V(C_2)$, so we can construct a tour in $G_{I \cup \{v\}}$ by inserting $C_2$ into $C_1$ after $w$ and taking shortcuts, which results in a tour of cost of at most

$$\text{tsp}(I' \cup \{w\}) + 2\,\text{smt}((I \setminus I') \cup \{v, w\}).$$

We achieve the given run time using a dynamic programming approach very similar to the TSP algorithm by Held and Karp [18]. The idea is to compute shortest Hamiltonian paths in the distance graph of the terminals for all possible pairs of endpoints. Then, one can evaluate $\mathcal{L}_{\text{TSP}}(v, I)$ in $\mathcal{O}(|I|^2)$ time by enumerating all possible pairs of neighbors of $v$ in the tour. □

## 4 Pruning

In this section, we present techniques to speed up the algorithm further by discarding labels $(v, I)$ for which we can prove that they cannot contribute to an optimum solution. This affects the number of iterations, since these labels then are not chosen in line 8 of the algorithm. Also, it speeds up the execution of line 18, since we only have to consider existing labels in the merge step. First, we show how to identify labels that cannot contribute to an optimum solution. Then we show that we can indeed safely discard them in our algorithm.

**Definition 13** Let $(G, c, R)$ be an instance of the Steiner tree problem, $(v, I) \in V(G) \times 2^R$ and $T$ be a Steiner tree for $R$. A tree $T_1$ is said to be a $(v, I)$-*subtree of* $T$ if there exists a tree $T_2$ such that

(i) $V(T_1) \cup V(T_2) = V(T)$,
(ii) $V(T_1) \cap V(T_2) = \{v\}$,
(iii) $T_1$ is a subtree of $T$ containing $\{v\} \cup I$ and
(iv) $T_2$ is a subtree of $T$ containing $\{v\} \cup (R \backslash I)$.

For a tree $T$ and a $(v, I)$-subtree $T_1$ of $T$, we will also refer to the corresponding subtree $T_2$ by $T - T_1$.

**Lemma 14** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Let $\mathcal{L}$ be a valid lower bound and $U \geq \text{smt}(R)$. Furthermore, let $v \in V(G)$, $I \subseteq R \backslash \{r_0\}$ and $T_1$ be a tree in $G$ containing $\{v\} \cup I$ with*

$$c(T_1) + \mathcal{L}(v, R \backslash I) > U.$$

*Then, there is no optimum Steiner tree for $R$ containing $T_1$ as a $(v, I)$-subtree.*

*Proof* Let $T$ be a Steiner tree for $R$ such that $T_1$ is a $(v, I)$-subtree of $T$. Then,

$$\begin{aligned} c(T) &= c(T_1) + c(T - T_1) \\ &\geq c(T_1) + \text{smt}(\{v\} \cup (R \backslash I)) \\ &= c(T_1) + \text{smt}(\{v, r_0\} \cup ((R \backslash I) \setminus \{r_0\})) + \mathcal{L}(r_0, \{r_0\}) \\ &\geq c(T_1) + \mathcal{L}(v, R \backslash I) \\ &> U \\ &\geq \text{smt}(R). \end{aligned}$$

□

Lemma 14 is a trivial exploitation of the lower bound $\mathcal{L}$. Its effect on the run time of the algorithm is rather limited, since we only discard labels that would never have been labeled permanently anyway. In contrast, the following lemma allows significant run time improvements of our algorithm, in particular on geometric instances. An application is illustrated in Fig. 5.

**Lemma 15** *Let $(G, c, R)$ be an instance of the Steiner tree problem, $v \in V(G)$, $I \subset R$ and $\emptyset \neq S \subseteq R \backslash I$. Furthermore, let $T_1$ be a Steiner tree for $\{v\} \cup I$ and $H$ be a subgraph of $G$ with*

(i) $(I \cup S) \subseteq V(H)$,
(ii) *each connected component of $H$ contains a terminal in $S$ and*
(iii) $c(H) < c(T_1)$.

*Then, there is no optimum Steiner tree for $R$ in $G$ containing $T_1$ as a $(v, I)$-subtree.*

*Proof* Let $T$ be a Steiner tree for $R$ in $G$ containing $T_1$ as a $(v, I)$-subtree. Then, there exists a tree $T_2$ containing $\{v\} \cup (R \backslash I)$ with $c(T) = c(T_1) + c(T_2)$. We construct a subgraph $T'$ of $G$ containing $R$ by $T' = T_2 + H$. As $H$ contains a path from every

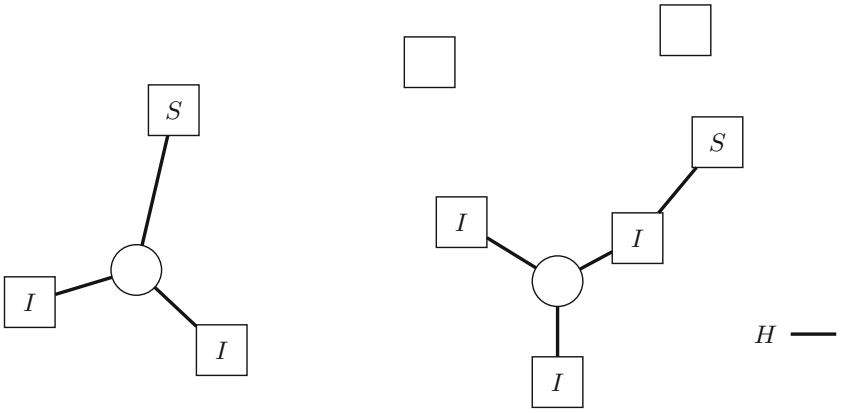


**Fig. 5** By Lemma 15, no label for the set $I$ with cost strictly larger than $c(H)$ can be part of an optimum solution. Terminals are drawn as *squares*, elements of $I$ and $S$ are labeled with the respective set

vertex in $H$ to some vertex in $S$, $T_2$ is connected and $S \subseteq R \backslash I \subseteq V(T_2)$, $T'$ is connected. Thus,

$$\begin{aligned} \mathrm{smt}(R) &\leq c(T') \\ &\leq c(T_2) + c(H) \\ &= c(T) - c(T_1) + c(H) \\ &< c(T). \end{aligned}$$

□

In Sect. 5, we will explain how suitable graphs $H$ can be found. Lemmata 14 and 15 allow us to identify labels that cannot contribute to an optimum solution. Theorem 17 shows that we can discard these labels without affecting the correctness of the algorithm. First, we prove an auxiliary lemma used in the proof of Theorem 17:

**Lemma 16** *Let $(G, c, R)$ be an instance of the Steiner tree problem and $r_0 \in R$. Let $O \subseteq V(G) \times 2^{R \backslash \{r_0\}}$ be the set of pairs $(v, I)$ with the property that there is an optimum Steiner tree for $R$ containing a $(v, I)$-subtree. Furthermore, let $(v, I) \in O$ and $T_1$ be a tree containing $\{v\} \cup I$ with $c(T_1) = \mathrm{smt}(\{v\} \cup I)$. Then, there is an optimum Steiner tree $T$ for $R$ containing $T_1$ as a $(v, I)$-subtree.*

*Proof* Since $(v, I) \in O$, there is an optimum Steiner tree $T'$ for $R$ and a tree $T_1'$ which is a $(v, I)$-subtree of $T$. Set $T = (T' - T_1') + T_1$. Then, since $T' - T_1'$ is a tree containing $\{v\} \cup (R \backslash I)$ and $T_1$ is a tree containing $\{v\} \cup I$, $T$ is a connected subgraph of $G$ containing $\{v\} \cup R$. Furthermore, we have

$$\begin{aligned} \mathrm{smt}(R) \leq c(T) &\leq c(T' - T_1') + c(T_1) \\ &= c(T' - T_1') + \mathrm{smt}(\{v\} \cup I) \end{aligned}$$

$$
\begin{aligned}
&= c(T') - c(T'_1) + \mathrm{smt}(\{v\} \cup I) \\
&\leq c(T') \\
&= \mathrm{smt}(R).
\end{aligned}
$$

Since we do not have edges of zero cost, this shows $T$ is an optimum Steiner tree and $T_1$ is a $(v, I)$-subtree of $T$. □

We now formalize a general method of pruning:

**Procedure** `prune`$(v, I)$

1 **if** *we can prove that there is no optimum Steiner tree $T$ for $R$ such that* `backtrack`$(v, I)$ *is the edge set of a $(v, I)$-subtree of $T$* **then**
2 | $N := N \backslash \{(v, I)\}$;
3 **end**

Note that when considering a not permanently labeled element $(v, I) \in N$, we cannot guarantee that `backtrack`$(v, I)$ is the edge set of a tree, since it may contain cycles. However, if it is not a tree, we can prune $(v, I)$ obviously.

**Theorem 17** *The Dijkstra–Steiner algorithm still works correctly if we modify it to execute* prune$(w, I)$ *after line 15 and* prune$(v, I \cup J)$ *after line 22.*

*Proof* Let $O \subseteq V(G) \times 2^{R \backslash \{r_0\}}$ be the set of pairs $(v, I)$ with the property that there is an optimum Steiner tree for $R$ containing a $(v, I)$-subtree. It suffices that the modified algorithm is correct on $O$, which we will now prove.

To this end, we modify invariants (a)–(d) as defined in the proof of Theorem 2 by restricting (a)–(c) to labels $(v, I) \in O$ and not changing (d). We call these new invariants (a')–(d').

Since $(r_0, R \backslash \{r_0\}) \in O$, the algorithm is correct assuming that these invariants hold. They clearly hold after the initialization.

Moreover, (a'), (c1') and (c2') are clearly preserved. To see that (c3') is preserved, recall that prune only removes labels $(v, I)$ from $N$ if there is no optimum Steiner tree $T$ for $R$ such that `backtrack`$(v, I)$ is the edge set of a $(v, I)$-subtree of $T$. However, if $(v, I) \in O$ and $l(v, I) = \mathrm{smt}(\{v\} \cup I)$, by (a3') and Lemma 16 the label $(v, I)$ cannot be pruned, so (c3') is preserved as well.

The argument showing that (b) is preserved remains unchanged: (c') can be applied to $(w, I')$, because if $(v, I) \in O$, then $(w, I')$ is in $O$ as well. This directly implies that the same argument as in the proof of Theorem 2 can be used to prove that (d') is preserved, since $(r_0, R \backslash \{r_0\}) \in O$. □

Of course, in practice we just avoid the creation of such labels instead of removing them immediately after creation. Moreover, whenever a label $(v, I)$ is selected in line 8, we also try to prune it. This is not redundant, since after $l(v, I)$ was updated the last time, bounds used to prune may have improved. Moreover, we could have pruned $(v, I)$ immediately after the last update of $l(v, I)$ with an equivalent impact on the execution of the algorithm, so the algorithm still works correctly.

## 5 Implementation and results

We implemented the algorithm using the C++ programming language. In our implementation, we use a binary heap instead of a Fibonacci heap, since binary heaps performed better in our experiments.

Our implementation is limited to instances with less than 64 terminals, allowing us to uniquely represent terminal sets by 64-bit words using the canonical bijection $2^R \to \{0, \ldots, 2^{|R|} - 1\}$. Basic operations on sets like union and intersection can be performed using a single bitwise instruction.

For each vertex $v \in V(G)$, we maintain an array containing the labels $(v, I)$ with $l(v, I) < \infty$ and a lookup table storing for each label its index in the array, if it exists. This enables us to access labels very quickly and traverse over the existing labels in linear time, which is important for an efficient implementation of the merge step:

To implement line 18, we have two options. Either we explicitly enumerate all sets $J \subseteq (R\backslash\{r_0\})\backslash I$ and check whether the label $(v, J)$ exists, or we traverse over all existing labels at $v$ and omit the labels $(v, J)$ with $J \cap I \neq \emptyset$. We always choose the option resulting in fewer sets to be considered.

We implement the pruning rule of Lemma 14 using a shortest-paths Steiner tree heuristic similar to Prim's algorithm [28], maintaining and extending one component at a time. This takes $\mathcal{O}(k(n \log n + m))$ time. Then, we use the cost of that Steiner tree as an upper bound and apply Lemma 14 each time we create a new label.

To implement Lemma 15, we maintain an upper bound $U(I)$ on the cost of labels for each set $I \subseteq R\backslash\{r_0\}$ of terminals, which is initially set to infinity. For each occurring set $I \subseteq R\backslash\{r_0\}$, we compute the distance $d(I, R\backslash I) = \min_{x \in I, y \in R\backslash I} d(x, y)$. Then, each time we extract a label $(v, I)$ from the heap, we update $U(I)$ by

$$U(I) := \min\left(U(I),\ l(v, I) + \min(d(I, R\backslash I), d(v, R\backslash I))\right).$$

Also, we keep track of the set $S(I)$ that was used to generate the currently best upper bound for the set $I$. In the routine described above, we always have $|S| = 1$. However, when merging two sets $I_1$ and $I_2$, we can use the sum of their upper bounds as an upper bound for the set $I_1 \cup I_2$ if $S(I_1) \cap I_2 = \emptyset$ or $S(I_2) \cap I_1 = \emptyset$, resulting in $S(I_1 \cup I_2) = (S(I_1) \cup S(I_2))\backslash(I_1 \cup I_2)$.

Lacking a good selection strategy for general instances, we always choose the last terminal of the instance w.r.t. the order in the instance file as root terminal.

The rest of this section is organized as follows: First, in Sect. 5.1, we demonstrate the impact of lower bounds and pruning on the practical run time of the algorithm and compare our algorithm with the dynamic programming algorithm by Erickson et al. [11] which achieves the same worst-case run time. Then, in Sect. 5.2, we compare our algorithm with the state-of-the-art algorithm by Polzin and Vahdati Daneshmand [27]. Finally, in Sect. 5.3, we consider the special case of the $d$-dimensional rectilinear Steiner tree problem for $d \in \{3, 4, 5\}$. All results of our algorithm were achieved single-threaded on a computer with 3.33 GHz Intel Xeon W5590 CPUs, which produced a score of 391 for the DIMACS benchmark.

### 5.1 Impact of lower bounds and pruning

Now, we compare the algorithm by Erickson et al. with our algorithm and discuss the impact of lower bounds and pruning. We reimplemented the algorithm by Erickson et al. using the same terminal set representation and the same heap as used in the implementation of our algorithm. Recall that this algorithm considers all sets $I \subseteq R \setminus \{r_0\}$ in order of non-decreasing cardinality and computes $\mathrm{smt}(\{v\} \cup I)$ for all $v \in V(G)$, leading to a run time of $\Theta(3^k n) + \mathcal{O}(2^k(n \log n + m))$ and memory consumption of $\Theta(2^k n)$.

For these experiments, we implemented the lookup table in our algorithm storing indices of labels $(v, I)$ for each $v \in V(G)$ using an array of length $2^{k-1}$. We compare three lower bound configurations for our algorithm: The trivial lower bound $\mathcal{L} \equiv 0$, the lower bound $\max(\mathcal{L}_{1-\mathrm{tree}}, \mathcal{L}_2)$ which can be evaluated in $\mathcal{O}(k)$ time and $\max(\mathcal{L}_{\mathrm{TSP}}, \mathcal{L}_3)$ which can be evaluated in $\mathcal{O}(k^2)$ time. For each of these configurations, we test the algorithm with and without pruning. Despite all edge costs being integral, the 1-tree bound and the TSP bound may be non-integral. Note that if all costs are integral, a valid lower bound stays valid when rounded up. Hence, we round $\mathcal{L}_{1-\mathrm{tree}}$ and $\mathcal{L}_{\mathrm{TSP}}$ up, leading to slightly improved bounds and allowing exact integer arithmetic.

Computational results on instances of the 11th DIMACS implementation challenge [1] are given in Table 1. For each instance, we give its name, the number of vertices, edges, terminals and its type. Then, we provide results for the algorithm by Erickson et al. [11] as well as the Dijkstra–Steiner algorithm, indicated by DS. For the latter, we further indicate which lower bound was used and whether pruning was enabled. For each result, we give the number of permanent labels $(v, I)$ at the termination of the algorithm. For the Dijkstra–Steiner algorithm, this coincides with the number of iterations of the algorithm. Furthermore, we give the number of encountered terminal sets and the run time in seconds.

Not surprisingly, running the Dijkstra–Steiner algorithm without lower bounds and pruning leads to significantly worse run times compared to the algorithm by Erickson et al. (except for es20fst10). This is caused by the number of permanent labels decreasing only slightly for lin24 and staying the same for wrp3-17 and cc3-5p, while the run time per label is increased in our algorithm. Note that our algorithm is more complicated than the algorithm by Erickson et al. Also, the implementation of our algorithm is tuned for the case that much less than $2^{k-1}$ labels are created per vertex: Instead of storing the labels directly in a table, we store them in a separate array for fast traversal and only store indices to that array in a lookup table, slowing down access to labels.

Except for the artificial instance cc3-5p, lower bounds significantly reduce the number of labels. As expected, the stronger lower bound leads to even less labels. In particular, the variant using the strongest lower bound $\max(\mathcal{L}_{\mathrm{TSP}}, \mathcal{L}_3)$ is faster than the algorithm by Erickson et al. Note that on the instance es20fst10, the preprocessing run time of $\Theta(2^k k^3)$ dominates the run time for the TSP bound.

On the geometric instances lin24 and es20fst10, pruning is extremely effective. Even without lower bounds, the number of labels and hence the run time is reduced by orders of magnitudes. In contrast, on the group Steiner tree instance wrp3-17, our pruning implementation has nearly no effect. This can be explained as follows. Although these instances are based on VLSI-derived grid graphs with holes as well, they have been modified to model the groups as terminals: For each group of the

**Table 1** Impact of lower bounds and pruning

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Solver | $\mathcal{L}$ | Prune | Labels | Sets | Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| | | | | [11] | | | 262,070,466 | 32,767 | 684.24 |
| lin24 | 7998 | 14734 | 16 | DS | 0 | No | 260,911,042 | 32,767 | 5241.77 |
| | | | | DS | 0 | Yes | 77,126 | 445 | 0.43 |
| Type: VLSI-derived grid | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | No | 31,447,839 | 32,767 | 1220.21 |
| graph with holes | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | Yes | 73,900 | 445 | 0.48 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | No | 4,781,074 | 31,690 | 115.91 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | Yes | 61,503 | 423 | 1.50 |
| | | | | [11] | | | 25,690,063 | 524,287 | 285.58 |
| es20fst10 | 49 | 67 | 20 | DS | 0 | No | 7,098,990 | 374,815 | 105.71 |
| | | | | DS | 0 | Yes | 1069 | 247 | 0.05 |
| Type: Rectilinear instance | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | No | 322,620 | 285,750 | 8.28 |
| after FST-preprocessing | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | Yes | 1029 | 247 | 0.05 |
| by GeoSteiner | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | No | 40,216 | 64,747 | 10.83 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | Yes | 754 | 237 | 10.32 |

**Table 1** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Solver | $\mathcal{L}$ | Prune | Labels | Sets | Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| | | | | [11] | | | 11,599,695 | 65,535 | 37.88 |
| wrp3-17 | 177 | 354 | 17 | DS | 0 | No | 11,599,695 | 65,535 | 125.12 |
| | | | | DS | 0 | Yes | 11,403,289 | 65,535 | 215.49 |
| Type: Group Steiner tree | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | No | 5,177,416 | 65,535 | 193.50 |
| instance | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | Yes | 5,174,919 | 65,535 | 284.01 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | No | 14,523 | 3180 | 0.86 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | Yes | 14,526 | 3179 | 0.89 |
| | | | | [11] | | | 511,875 | 4095 | 0.22 |
| cc3-5p | 125 | 750 | 13 | DS | 0 | No | 511,875 | 4095 | 2.33 |
| | | | | DS | 0 | Yes | 343,916 | 4095 | 1.18 |
| Type: Artificial instance | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | No | 506,462 | 4095 | 3.90 |
| from the hard PUC testset | | | | DS | $\max(\mathcal{L}_{1-\text{tree}}, \mathcal{L}_2)$ | Yes | 332,103 | 4095 | 1.87 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | No | 483,683 | 4095 | 4.46 |
| | | | | DS | $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ | Yes | 310,999 | 4095 | 2.27 |

group Steiner tree instance, a new terminal is added to the graph and connected to the elements of the group by edges of very high cost. By choosing the cost of these new edges sufficiently large, one can guarantee that an optimum Steiner tree in the new instance corresponds to an optimum group Steiner tree in the original instance and vice versa, since each terminal will be a leaf of the Steiner tree. To prune a label $(v, I) \in V(G) \times 2^{R \setminus \{r_0\}}$, our implementation needs to connect the terminal set $I$ to at least one additional terminal $s \in R \setminus I$ with cost strictly less than the cost of the label $(v, I)$. However, on these instances, connecting to an additional terminal is always much more expensive than any path in the original graph. Hence, no labels are pruned at non-terminals. On the artificial instance cc3-5p, pruning roughly halves run time, which still leads to a worse run time compared to the classical dynamic programming approach.

When combining lower bounds and pruning, albeit slightly improving the number of labels, run times are not improved compared to only using pruning on the geometric instances. However, on the group Steiner tree instance wrp3-17, where our pruning implementation has no effect, lower bounds still reduce the number of labels, with the stronger lower bound $\max(\mathcal{L}_{\text{TSP}}, \mathcal{L}_3)$ improving the run time significantly. Note that the 1-tree bound and the TSP bound completely incorporate the expensive edges incident to terminals.

In summary, on some instance classes including geometric instances, our algorithm significantly improves upon the classical dynamic programming algorithm by Erickson et al. This becomes even more evident in the results in Tables 2 and 3, where instances with more than 40 terminals are solved. For these instances, the Erickson et al. best-case run time of $\Omega(3^k n)$ clearly is impractical.

### 5.2 Comparison with the state of the art

Despite the TSP bound being significantly stronger on some instances, its required preprocessing leads to an exponential best-case run time. Hence, in the following experiments, we use the 1-tree bound as a lower bound. Also, in order to avoid the exponential best-case memory consumption, we replace the array of length $2^{k-1}$ by a hash table as lookup table for labels. This modification improves the run time on lin24, where much less than $2^{k-1}n$ labels are generated, but leads to worse run times on wrp3-17 and cc3-5p, where the used lower bound and pruning have little effect.

We now compare our results with those obtained by the state-of-the-art algorithm by Polzin and Vahdati Daneshmand [27] on instances of the 11th DIMACS implementation challenge [1]. Their results were obtained using one thread on a computer with a 2.66 GHz Intel i7 920 CPU which produced a score of 307 in the DIMACS benchmark. The algorithm by Polzin and Vahdati Daneshmand successively performs various optimality-preserving reductions combined with a branch and bound approach.

In Table 2, we show results on multiple instance classes (a)–(e). For each instance, we give its name, the number of vertices, edges and terminals. Then, we state the cost of an optimum solution as reported by our algorithm and the run time in seconds. Moreover, for each instance, we give the run time reported by Polzin and Vahdati Daneshmand. For the lin testset, Polzin and Vahdati Daneshmand improved run times

**Table 2** Results on various instance types

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) | Time PV (s) |
|---|---|---|---|---|---|---|
| (a) VLSI-derived grid graphs with holes | | | | | | |
| diw0779 | 11,821 | 22,516 | 50 | 4440 | 1.60 | 1.26 |
| diw0819 | 10,553 | 20,066 | 32 | 3399 | 0.29 | 0.52 |
| diw0820 | 11,749 | 22,384 | 37 | 4167 | 1.50 | 1.06 |
| lin23 | 3716 | 6750 | 52 | 17,560 | 11.08 | 0.54 |
| lin24 | 7998 | 14,734 | 16 | 15,076 | 0.10 | 1.73 |
| lin30 | 19,091 | 35,644 | 31 | 27,684 | 0.68 | 14.74 |
| lin32 | 19,112 | 35,665 | 53 | 39,832 | 150.20 | 816.51 |
| lin34 | 38,282 | 71,521 | 34 | 45,018 | 12.93 | 1848.24 |
| lin35 | 38,294 | 71,533 | 45 | 50,559 | 26.90 | 1911.09 |
| lin36 | 38,307 | 71,546 | 58 | 55,608 | 47.59 | 39,931.77 |
| (b) Rectilinear obstacle-avoiding instances preprocessed by ObSteiner | | | | | | |
| ind5 | 114 | 228 | 33 | 1341 | 0.02 | 0.01 |
| rc03 | 109 | 202 | 50 | 54,160 | 0.15 | 0.00 |
| rt02 | 788 | 1938 | 50 | 45,852 | 0.70 | 1.99 |
| (c) Group Steiner tree instances | | | | | | |
| wrp3-14 | 128 | 247 | 14 | 1,400,250 | 3.78 | 0.01 |
| wrp3-15 | 138 | 257 | 15 | 1,500,422 | 51.16 | 0.01 |
| wrp3-16 | 204 | 374 | 16 | 1,600,208 | 11.62 | 0.03 |
| wrp3-17 | 177 | 354 | 17 | 1,700,442 | 422.08 | 0.02 |
| wrp3-19 | 189 | 353 | 19 | 1,900,439 | 1765.72 | 0.03 |
| (d) Random graphs with so-called incidence costs | | | | | | |
| i160-141 | 160 | 2544 | 12 | 2549 | 3.40 | 0.01 |
| i320-111 | 320 | 1845 | 17 | 4273 | 1706.67 | 0.03 |
| i640-022 | 640 | 204,480 | 9 | 1756 | 4.03 | 0.52 |
| i640-031 | 640 | 1280 | 9 | 3278 | 0.07 | 0.00 |
| i640-043 | 640 | 40,896 | 9 | 1931 | 0.89 | 0.13 |
| (e) Artificial instances from the hard PUC testset | | | | | | |
| cc3-4p | 64 | 288 | 8 | 2338 | 0.01 | 1.99 |
| cc3-4u | 64 | 288 | 8 | 23 | 0.01 | 1.37 |
| cc3-5p | 125 | 750 | 13 | 3661 | 3.45 | 87.98 |
| cc3-5u | 125 | 750 | 13 | 36 | 4.64 | 115.83 |
| cc6-2p | 64 | 192 | 12 | 3271 | 0.17 | 0.40 |
| cc6-2u | 64 | 192 | 12 | 32 | 0.28 | 0.90 |

by modifying their algorithm to use stronger reductions. With default settings, their algorithm did not solve lin36 within a time limit of 24 h.

On instances (a) and (b), both arising from rectilinear VLSI problems, our algorithm is much faster than the worst-case bound tells. This is primarily caused by the high impact of our pruning method as seen in Table 1. In particular, on instances with large

underlying graphs, our algorithm performs very well, beating the reduction-based solver.

On group Steiner tree instances (c), our pruning implementation has no effect. Here, the reduction-based solver yields much better results. Note that the stronger TSP bound still improves run time significantly on these instances as seen in Table 1. On incidence cost instances (d), where edges incident to terminals are assigned larger costs, a similar effect can be observed.

Although neither pruning nor lower bounds do have a significant effect on instances from the hard PUC testset (e), our algorithm performs very well on instances with few terminals. This is caused by the strong worst-case run time guarantee, which, albeit exponential in the number of terminals, is quasilinear in the size of the graph. Note that on these instances, the classical algorithm by Erickson et al. [11] yields even better run times. Detailed further computational results can be found in Appendix 1.

### 5.3 $d$-dimensional rectilinear Steiner tree problem

We also applied our algorithm to the $d$-dimensional rectilinear Steiner tree problem for $d \in \{3, 4, 5\}$. For $d = 2$, instances with thousands of terminals can be solved by the GeoSteiner algorithm [34], which works by generating a candidate set of full Steiner trees for subsets of terminals and then concatenating a subset of these candidates to form an optimum Steiner tree. A full Steiner tree is a Steiner tree where each terminal has degree 1. The concatenation phase works by either solving a Steiner tree problem in graphs or a minimum spanning tree problem in hypergraphs. Exploiting a result by Hwang [21], the GeoSteiner algorithm only has to consider full Steiner trees following a special structure, which allows a significant reduction of the number of generated candidates. In contrast, for higher dimensions, eliminating possible full Steiner trees is much harder, as Hwang's result does not apply and complicated full Steiner trees have to be considered [36]. An implementation of the GeoSteiner algorithm for higher dimension is only able to solve instances with up to around 15 terminals in dimension 3 and up to around 10 terminals in dimensions 4, 5 and 6 [36]. The situation is similar in the Euclidean case, where huge instances can be solved for $d = 2$ using the GeoSteiner algorithm, but only instances with up to around 17 terminals for $d \geq 3$, using very different algorithms [13].

It is well-known that to compute an optimum rectilinear Steiner tree in dimension 2, it suffices to compute an optimum Steiner tree in the so-called Hanan grid [16]. The Hanan grid is the graph obtained by drawing axis-parallel lines through each terminal, taking intersections of these lines as vertex set and segments between intersections as edges. This result was later generalized to arbitrary dimension by Snyder [30], leading to a grid graph with $\mathcal{O}(k^d)$ vertices and $\mathcal{O}(dk^d)$ edges for a $d$-dimensional instance with $k$ terminals.

Using this reduction, we ran our algorithm on instances from the CARIOCA [1] testset, which contains randomly generated instances for $d \in \{3, 4, 5\}$ with between 11 and 20 terminals. To test our algorithm on larger instances, we generated new random instances (using the prefix "bonn"), as other available testsets do not contain sufficiently many instances which are neither too small nor too large. For these new

**Table 3** Results on $d$-dimensional rectilinear instances

| Instance | $d$ | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|---|
| carioca_3_11_01 | 3 | 1331 | 3630 | 11 | 311,221,222 | 0.02 |
| carioca_3_11_02 | 3 | 1331 | 3630 | 11 | 466,149,453 | 0.02 |
| carioca_3_20_01 | 3 | 8000 | 22,800 | 20 | 638,376,617 | 1.61 |
| carioca_3_20_02 | 3 | 8000 | 22,800 | 20 | 477,950,448 | 0.15 |
| bonn_3_40_1 | 3 | 64,000 | 187,200 | 40 | 9024 | 25.44 |
| bonn_3_40_2 | 3 | 57,798 | 168,909 | 40 | 9633 | 710.09 |
| bonn_3_55_3 | 3 | 154,548 | 455,004 | 55 | 12,138 | 6201.10 |
| carioca_4_11_01 | 4 | 14,641 | 53,240 | 11 | 627,022,001 | 0.43 |
| carioca_4_11_02 | 4 | 14,641 | 53,240 | 11 | 636,772,154 | 0.22 |
| carioca_4_20_01 | 4 | 160,000 | 608,000 | 20 | 889,180,827 | 82.72 |
| carioca_4_20_02 | 4 | 160,000 | 608,000 | 20 | 822,698,792 | 101.11 |
| carioca_5_11_01 | 5 | 161,051 | 732,050 | 11 | 925,163,690 | 34.55 |
| carioca_5_11_02 | 5 | 161,051 | 732,050 | 11 | 844,673,618 | 13.02 |
| carioca_5_15_01 | 5 | 759,375 | 3,543,750 | 15 | 1,011,895,745 | 1046.36 |
| carioca_5_15_02 | 5 | 759,375 | 3,543,750 | 15 | 1,067,623,193 | 888.81 |
| carioca_5_18_03 | 5 | 1,889,568 | 8,922,960 | 18 | 1,177,091,608 | 1081.01 |

instances, coordinates were chosen uniformly at random from $\{0, 1, \ldots, 999\}$. In particular for the larger instances, coordinates may appear multiple times, leading to grid graphs with slightly less than $k^d$ vertices. Coordinates of instances from the CARIOCA testset were scaled by $10^8$ to obtain integral coordinates. For these instances, we chose a terminal as close as possible to the center of gravity of all terminals as root terminal, improving results compared to a random choice.

Excerpts of experimental results are given in Table 3, the full results can be found in Appendix 1. In dimension 3, we are able to solve all tested instances with up to 34 terminals. Many of the larger instances with up to 40 terminals are solved as well, and additionally one instance with 55 terminals. In dimension 4, we solve all instances of the CARIOCA testset as well. Experiments with larger instances are not reported here, since we were only able to solve instances with up to 22 terminals. In dimension 5, all instances with up to 15 terminals are solved. The largest solved instance carioca_5_18_03 has 18 terminals and nearly 9 million edges. Solving it required approximately 20 GB of memory.

We also tried to exploit the geometric structure of these instances by using geometric lower bounds. For example, consider the bounding box bound $\mathcal{L}_{\text{BBox}}$ given by

$$\mathcal{L}_{\text{BBox}}(v, I) := \sum_{i=1}^{d} \left( \max_{x \in \{v\} \cup I} x_i - \min_{x \in \{v\} \cup I} x_i \right).$$

One can easily verify that $\mathcal{L}_{\text{BBox}}$ is a valid lower bound. Also note that $\mathcal{L}_{\text{BBox}} \leq \mathcal{L}_{\text{TSP}}$ and $\mathcal{L}_{\text{BBox}} \leq \mathcal{L}_{2d}$, but $\mathcal{L}_{\text{BBox}}$ can be larger than $\mathcal{L}_{1-\text{tree}}$. However, using $\mathcal{L}_{\text{BBox}}$ did not lead to better run times.

## 6 Discussion

Note that due to the dynamic programming nature of our algorithm, it can also be used to compute all optimum Steiner trees or even all Steiner trees up to a given cost. If we enumerate all Steiner trees up to a cost of Opt + $\Delta$, we have to relax the pruning implementations by $\Delta$ and continue labeling until all labels $(v, I)$ with $l(v, I) \leq \text{Opt} + \Delta$ are permanent. Also, we have to save all predecessors instead of only one optimum predecessor. Then, we can recursively combine Steiner trees for subsets of terminals. In practice, the additional effort is linear in the size of the output, allowing the enumeration of millions of near-optimum Steiner trees in seconds. See [29] for details.

The dynamic programming idea by Dreyfus and Wagner has been used extensively to obtain Steiner tree algorithms with fast theoretical worst-case behavior. However, in the field of practical solving, it has rather been disregarded prior to this work. Compared to other exact algorithms, our algorithm depends much less on effective preprocessing and performs well on large graphs. Our approach is very general and not limited to the lower bounds and pruning strategies proposed in this paper.

## Appendix 1: Results on graphic DIMACS instances

We present detailed computational results on DIMACS testsets. Our implementation is limited to instances with less than 64 terminals, so we exclude instances with more terminals.

The implementation of our algorithm is written in the C++ programming language and compiled using the GCC 4.8.2 compiler.

The experiments were performed single-threaded on a machine with 3.33 GHz Intel Xeon W5590 CPUs and 144 GB main memory which produced a score of 391.372724 using the DIMACS benchmark code.

For these experiments, we limited the memory consumption of the algorithm to 100 GB and set the time limit to 7200 s. The reported run times do not include the time to read the instance file from disk (Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46).

**Table 4** Results on the testset ALUE

Type: VLSI-derived grid graphs with holes

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| alue2087 | 1244 | 1971 | 34 | 1049 | 1.075 |
| alue2105 | 1220 | 1858 | 34 | 1032 | 0.204 |
| alue7066 | 6405 | 10,454 | 16 | 2256 | 0.068 |
| alue7229 | 940 | 1474 | 34 | 824 | 0.036 |

**Table 5** Results on the testset ALUT

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| alut0787 | 1160 | 2089 | 34 | 982 | 1.769 |
| alut0805 | 966 | 1666 | 34 | 958 | 0.258 |
| alut2764 | 387 | 626 | 34 | 640 | 0.042 |

Type: VLSI-derived grid graphs with holes

**Table 6** Results on the testset DIW

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| diw0234 | 5349 | 10,086 | 25 | 1996 | 0.096 |
| diw0250 | 353 | 608 | 11 | 350 | 0.002 |
| diw0260 | 539 | 985 | 12 | 468 | 0.003 |
| diw0313 | 468 | 822 | 14 | 397 | 0.003 |
| diw0393 | 212 | 381 | 11 | 302 | 0.002 |
| diw0445 | 1804 | 3311 | 33 | 1363 | 0.061 |
| diw0459 | 3636 | 6789 | 25 | 1362 | 0.042 |
| diw0460 | 339 | 579 | 13 | 345 | 0.004 |
| diw0473 | 2213 | 4135 | 25 | 1098 | 0.065 |
| diw0487 | 2414 | 4386 | 25 | 1424 | 0.373 |
| diw0495 | 938 | 1655 | 10 | 616 | 0.006 |
| diw0513 | 918 | 1684 | 10 | 604 | 0.006 |
| diw0523 | 1080 | 2015 | 10 | 561 | 0.005 |
| diw0540 | 286 | 465 | 10 | 374 | 0.002 |
| diw0559 | 3738 | 7013 | 18 | 1570 | 0.089 |
| diw0778 | 7231 | 13, 727 | 24 | 2173 | 0.139 |
| diw0779 | 11,821 | 22,516 | 50 | 4440 | 1.603 |
| diw0795 | 3221 | 5938 | 10 | 1550 | 0.024 |
| diw0801 | 3023 | 5575 | 10 | 1587 | 0.024 |
| diw0819 | 10,553 | 20,066 | 32 | 3399 | 0.291 |
| diw0820 | 11,749 | 22,384 | 37 | 4167 | 1.504 |

Type: VLSI-derived grid graphs with holes

**Table 7** Results on the testset DMXA

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| dmxa0296 | 233 | 386 | 12 | 344 | 0.002 |
| dmxa0368 | 2050 | 3676 | 18 | 1017 | 0.019 |
| dmxa0454 | 1848 | 3286 | 16 | 914 | 0.012 |
| dmxa0628 | 169 | 280 | 10 | 275 | 0.002 |
| dmxa0734 | 663 | 1154 | 11 | 506 | 0.006 |
| dmxa0848 | 499 | 861 | 16 | 594 | 0.041 |

**Table 7** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| dmxa0903 | 632 | 1087 | 10 | 580 | 0.008 |
| dmxa1010 | 3983 | 7108 | 23 | 1488 | 0.104 |
| dmxa1109 | 343 | 559 | 17 | 454 | 0.007 |
| dmxa1200 | 770 | 1383 | 21 | 750 | 0.069 |
| dmxa1304 | 298 | 503 | 10 | 311 | 0.002 |
| dmxa1516 | 720 | 1269 | 11 | 508 | 0.003 |
| dmxa1721 | 1005 | 1731 | 18 | 780 | 0.012 |
| dmxa1801 | 2333 | 4137 | 17 | 1365 | 0.049 |

Type: VLSI-derived grid graphs with holes

**Table 8** Results on the testset GAP

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| gap1307 | 342 | 552 | 17 | 549 | 0.057 |
| gap1413 | 541 | 906 | 10 | 457 | 0.007 |
| gap1500 | 220 | 374 | 17 | 254 | 0.003 |
| gap1810 | 429 | 702 | 17 | 482 | 0.013 |
| gap1904 | 735 | 1256 | 21 | 763 | 0.042 |
| gap2007 | 2039 | 3548 | 17 | 1104 | 0.051 |
| gap2119 | 1724 | 2975 | 29 | 1244 | 0.460 |
| gap2740 | 1196 | 2084 | 14 | 745 | 0.011 |
| gap2800 | 386 | 653 | 12 | 386 | 0.003 |
| gap2975 | 179 | 293 | 10 | 245 | 0.001 |
| gap3036 | 346 | 583 | 13 | 457 | 0.013 |
| gap3100 | 921 | 1558 | 11 | 640 | 0.007 |

Type: VLSI-derived grid graphs with holes

**Table 9** Results on the testset LIN

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| lin01 | 53 | 80 | 4 | 503 | 0.000 |
| lin02 | 55 | 82 | 6 | 557 | 0.000 |
| lin03 | 57 | 84 | 8 | 926 | 0.001 |
| lin04 | 157 | 266 | 6 | 1239 | 0.001 |
| lin05 | 160 | 269 | 9 | 1703 | 0.002 |
| lin06 | 165 | 274 | 14 | 1348 | 0.004 |
| lin07 | 307 | 526 | 6 | 1885 | 0.002 |
| lin08 | 311 | 530 | 10 | 2248 | 0.002 |
| lin09 | 313 | 532 | 12 | 2752 | 0.004 |
| lin10 | 321 | 540 | 20 | 4132 | 0.017 |
| lin11 | 816 | 1460 | 10 | 4280 | 0.013 |

**Table 9** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| lin12 | 818 | 1462 | 12 | 5250 | 0.017 |
| lin13 | 822 | 1466 | 16 | 4609 | 0.017 |
| lin14 | 828 | 1472 | 22 | 5824 | 0.023 |
| lin15 | 840 | 1484 | 34 | 7145 | 0.088 |
| lin16 | 1981 | 3633 | 12 | 6618 | 0.032 |
| lin17 | 1989 | 3641 | 20 | 8405 | 0.051 |
| lin18 | 1994 | 3646 | 25 | 9714 | 0.475 |
| lin19 | 2010 | 3662 | 41 | 13,268 | 10.966 |
| lin20 | 3675 | 6709 | 11 | 6673 | 0.029 |
| lin21 | 3683 | 6717 | 20 | 9143 | 0.062 |
| lin22 | 3692 | 6726 | 28 | 10,519 | 0.129 |
| lin23 | 3716 | 6750 | 52 | 17,560 | 11.080 |
| lin24 | 7998 | 14,734 | 16 | 15,076 | 0.104 |
| lin25 | 8007 | 14,743 | 24 | 17,803 | 0.291 |
| lin26 | 8013 | 14,749 | 30 | 21,757 | 0.319 |
| lin27 | 8017 | 14,753 | 36 | 20,678 | 1.577 |
| lin29 | 19,083 | 35,636 | 24 | 23,765 | 1.746 |
| lin30 | 19,091 | 35,644 | 31 | 27,684 | 0.681 |
| lin31 | 19,100 | 35,653 | 40 | 31,696 | 34.061 |
| lin32 | 19,112 | 35,665 | 53 | 39,832 | 150.201 |
| lin34 | 38,282 | 71,521 | 34 | 45,018 | 12.930 |
| lin35 | 38,294 | 71,533 | 45 | 50,559 | 26.904 |
| lin36 | 38,307 | 71,546 | 58 | 55,608 | 47.590 |

Type: VLSI-derived grid graphs with holes

**Table 10** Results on the testset MSM

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| msm0580 | 338 | 541 | 11 | 467 | 0.006 |
| msm0654 | 1290 | 2270 | 10 | 823 | 0.007 |
| msm0709 | 1442 | 2403 | 16 | 884 | 0.010 |
| msm0920 | 752 | 1264 | 26 | 806 | 0.041 |
| msm1008 | 402 | 695 | 11 | 494 | 0.008 |
| msm1234 | 933 | 1632 | 13 | 550 | 0.007 |
| msm1477 | 1199 | 2078 | 31 | 1068 | 0.259 |
| msm1707 | 278 | 478 | 11 | 564 | 0.001 |
| msm1844 | 90 | 135 | 10 | 188 | 0.001 |
| msm1931 | 875 | 1522 | 10 | 604 | 0.003 |
| msm2000 | 898 | 1562 | 10 | 594 | 0.004 |
| msm2152 | 2132 | 3702 | 37 | 1590 | 0.269 |
| msm2326 | 418 | 723 | 14 | 399 | 0.003 |

**Table 10** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| msm2492 | 4045 | 7094 | 12 | 1459 | 0.035 |
| msm2525 | 3031 | 5239 | 12 | 1290 | 0.016 |
| msm2601 | 2961 | 5100 | 16 | 1440 | 0.042 |
| msm2705 | 1359 | 2458 | 13 | 714 | 0.017 |
| msm2802 | 1709 | 2963 | 18 | 926 | 0.028 |
| msm3277 | 1704 | 2991 | 12 | 869 | 0.021 |
| msm3676 | 957 | 1554 | 10 | 607 | 0.006 |
| msm3727 | 4640 | 8255 | 21 | 1376 | 0.079 |
| msm3829 | 4221 | 7255 | 12 | 1571 | 0.042 |
| msm4038 | 237 | 390 | 11 | 353 | 0.003 |
| msm4114 | 402 | 690 | 16 | 393 | 0.004 |
| msm4190 | 391 | 666 | 16 | 381 | 0.005 |
| msm4224 | 191 | 302 | 11 | 311 | 0.002 |
| msm4312 | 5181 | 8893 | 10 | 2016 | 0.042 |
| msm4414 | 317 | 476 | 11 | 408 | 0.003 |
| msm4515 | 777 | 1358 | 13 | 630 | 0.013 |

Type: VLSI-derived grid graphs with holes

**Table 11** Results on the testset TAQ

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| taq0023 | 572 | 963 | 11 | 621 | 0.007 |
| taq0365 | 4186 | 7074 | 22 | 1914 | 0.079 |
| taq0431 | 1128 | 1905 | 13 | 897 | 0.017 |
| taq0631 | 609 | 932 | 10 | 581 | 0.008 |
| taq0739 | 837 | 1438 | 16 | 848 | 0.019 |
| taq0741 | 712 | 1217 | 16 | 847 | 0.024 |
| taq0751 | 1051 | 1791 | 16 | 939 | 0.032 |
| taq0891 | 331 | 560 | 10 | 319 | 0.003 |
| taq0910 | 310 | 514 | 17 | 370 | 0.010 |
| taq0920 | 122 | 194 | 17 | 210 | 0.003 |
| taq0978 | 777 | 1239 | 10 | 566 | 0.005 |

Type: VLSI-derived grid graphs with holes

**Table 12** Results on the testset 1R

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| 1r111 | 1250 | 4704 | 6 | 28,000 | 0.006 |
| 1r112 | 1250 | 4704 | 6 | 28,000 | 0.005 |
| 1r113 | 1250 | 4704 | 6 | 26,000 | 0.005 |
| 1r121 | 1250 | 4704 | 6 | 36,000 | 0.004 |
| 1r122 | 1250 | 4704 | 6 | 45,000 | 0.006 |
| 1r123 | 1250 | 4704 | 6 | 40,000 | 0.004 |
| 1r131 | 1250 | 4704 | 6 | 43,000 | 0.005 |
| 1r132 | 1250 | 4704 | 6 | 37,000 | 0.005 |
| 1r133 | 1250 | 4704 | 6 | 36,000 | 0.004 |
| 1r211 | 1250 | 4704 | 31 | 77,000 | 0.344 |
| 1r212 | 1250 | 4704 | 30 | 81,000 | 0.066 |
| 1r213 | 1250 | 4704 | 29 | 70,000 | 0.720 |
| 1r221 | 1250 | 4704 | 31 | 79,000 | 0.147 |
| 1r222 | 1250 | 4704 | 31 | 68,000 | 0.059 |
| 1r223 | 1250 | 4704 | 31 | 77,000 | 0.097 |
| 1r231 | 1250 | 4704 | 30 | 80,000 | 0.146 |
| 1r232 | 1250 | 4704 | 29 | 86,000 | 0.307 |
| 1r233 | 1250 | 4704 | 31 | 71,000 | 1.551 |
| 1r311 | 1250 | 4704 | 56 | – | Timeout |
| 1r312 | 1250 | 4704 | 60 | 113,000 | 1276.770 |
| 1r313 | 1250 | 4704 | 58 | 106,000 | 496.549 |
| 1r321 | 1250 | 4704 | 59 | – | Timeout |
| 1r322 | 1250 | 4704 | 60 | 113,000 | 1612.443 |
| 1r323 | 1250 | 4704 | 60 | – | Timeout |
| 1r331 | 1250 | 4704 | 58 | 103,000 | 1.107 |
| 1r332 | 1250 | 4704 | 58 | 109,000 | 50.351 |
| 1r333 | 1250 | 4704 | 58 | 113,000 | 1708.425 |

Type: 2D grid graphs

**Table 13** Results on the testset 2R

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| 2r111 | 2000 | 11,600 | 9 | 28,000 | 0.015 |
| 2r112 | 2000 | 11,600 | 9 | 32,000 | 0.013 |
| 2r113 | 2000 | 11,600 | 9 | 28,000 | 0.011 |
| 2r121 | 2000 | 11,600 | 9 | 28,000 | 0.011 |
| 2r122 | 2000 | 11,600 | 9 | 29,000 | 0.011 |
| 2r123 | 2000 | 11,600 | 9 | 25,000 | 0.009 |
| 2r131 | 2000 | 11,600 | 9 | 27,000 | 0.012 |
| 2r132 | 2000 | 11,600 | 9 | 33,000 | 0.016 |
| 2r133 | 2000 | 11,600 | 9 | 29,000 | 0.011 |

**Table 13** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| 2r211 | 2000 | 11,600 | 50 | – | Memout |
| 2r212 | 2000 | 11,600 | 49 | 80,000 | 2819.259 |
| 2r213 | 2000 | 11,600 | 48 | 76,000 | 3152.193 |
| 2r221 | 2000 | 11,600 | 50 | – | Timeout |
| 2r222 | 2000 | 11,600 | 50 | – | Timeout |
| 2r223 | 2000 | 11,600 | 49 | – | Timeout |
| 2r231 | 2000 | 11,600 | 50 | – | Timeout |
| 2r232 | 2000 | 11,600 | 49 | – | Timeout |
| 2r233 | 2000 | 11,600 | 47 | – | Timeout |

Type: 3D grid graphs

**Table 14** Results on the testset ES10FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es10fst01 | 18 | 20 | 10 | 22,920,745 | 0.000 |
| es10fst02 | 14 | 13 | 10 | 19,134,104 | 0.000 |
| es10fst03 | 17 | 20 | 10 | 26,003,678 | 0.000 |
| es10fst04 | 18 | 20 | 10 | 20,461,116 | 0.000 |
| es10fst05 | 12 | 11 | 10 | 18,818,916 | 0.000 |
| es10fst06 | 17 | 20 | 10 | 26,540,768 | 0.000 |
| es10fst07 | 14 | 13 | 10 | 26,025,072 | 0.000 |
| es10fst08 | 21 | 28 | 10 | 25,056,214 | 0.000 |
| es10fst09 | 21 | 29 | 10 | 22,062,355 | 0.000 |
| es10fst10 | 18 | 21 | 10 | 23,936,095 | 0.000 |
| es10fst11 | 14 | 13 | 10 | 22,239,535 | 0.000 |
| es10fst12 | 13 | 12 | 10 | 19,626,318 | 0.000 |
| es10fst13 | 18 | 21 | 10 | 19,483,914 | 0.000 |
| es10fst14 | 24 | 32 | 10 | 21,856,128 | 0.000 |
| es10fst15 | 16 | 18 | 10 | 18,641,924 | 0.000 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 15** Results on the testset ES20FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es20fst01 | 29 | 28 | 20 | 33,703,886 | 0.004 |
| es20fst02 | 29 | 28 | 20 | 32,639,486 | 0.001 |
| es20fst03 | 27 | 26 | 20 | 27,847,417 | 0.001 |
| es20fst04 | 57 | 83 | 20 | 27,624,394 | 0.003 |
| es20fst05 | 54 | 77 | 20 | 34,033,163 | 0.002 |
| es20fst06 | 29 | 28 | 20 | 36,014,241 | 0.001 |
| es20fst07 | 45 | 59 | 20 | 34,934,874 | 0.002 |
| es20fst08 | 52 | 74 | 20 | 38,016,346 | 0.014 |

**Table 15** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es20fst09 | 36 | 42 | 20 | 36,739,939 | 0.003 |
| es20fst10 | 49 | 67 | 20 | 34,024,740 | 0.002 |
| es20fst11 | 33 | 36 | 20 | 27,123908 | 0.001 |
| es20fst12 | 33 | 36 | 20 | 30,451,397 | 0.005 |
| es20fst13 | 35 | 40 | 20 | 34,438,673 | 0.003 |
| es20fst14 | 36 | 44 | 20 | 34,062,374 | 0.009 |
| es20fst15 | 37 | 43 | 20 | 32,303,746 | 0.003 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 16** Results on the testset ES30FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es30fst01 | 79 | 115 | 30 | 40,692,993 | 0.037 |
| es30fst02 | 71 | 97 | 30 | 40,900,061 | 0.029 |
| es30fst03 | 83 | 120 | 30 | 43,120,444 | 0.019 |
| es30fst04 | 80 | 115 | 30 | 42,150,958 | 0.010 |
| es30fst05 | 58 | 71 | 30 | 41,739,748 | 0.007 |
| es30fst06 | 83 | 119 | 30 | 39,955,139 | 0.053 |
| es30fst07 | 53 | 64 | 30 | 43,761,391 | 0.007 |
| es30fst08 | 69 | 93 | 30 | 41,691,217 | 0.008 |
| es30fst09 | 43 | 44 | 30 | 37,133,658 | 0.010 |
| es30fst10 | 48 | 52 | 30 | 42,686,610 | 0.009 |
| es30fst11 | 79 | 112 | 30 | 41,647,993 | 0.007 |
| es30fst12 | 46 | 48 | 30 | 38,416,720 | 0.013 |
| es30fst13 | 65 | 84 | 30 | 37,406,646 | 0.005 |
| es30fst14 | 53 | 58 | 30 | 42,897,025 | 0.021 |
| es30fst15 | 118 | 188 | 30 | 43,035,576 | 0.070 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 17** Results on the testset ES40FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es40fst01 | 93 | 127 | 40 | 44,841,522 | 0.036 |
| es40fst02 | 82 | 105 | 40 | 46,811,310 | 0.016 |
| es40fst03 | 87 | 116 | 40 | 49,974,157 | 0.093 |
| es40fst04 | 55 | 55 | 40 | 45,289,864 | 0.013 |
| es40fst05 | 121 | 180 | 40 | 51,940,413 | 0.133 |
| es40fst06 | 92 | 123 | 40 | 49,753,385 | 0.046 |
| es40fst07 | 77 | 95 | 40 | 45,639,009 | 0.109 |
| es40fst08 | 98 | 137 | 40 | 48,745,996 | 0.016 |
| es40fst09 | 107 | 153 | 40 | 51,761,789 | 0.040 |
| es40fst10 | 107 | 152 | 40 | 57,136,852 | 0.117 |

**Table 17** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es40fst11 | 97 | 135 | 40 | 46,734,214 | 0.027 |
| es40fst12 | 67 | 75 | 40 | 43,843,378 | 0.029 |
| es40fst13 | 78 | 95 | 40 | 51,884,545 | 0.030 |
| es40fst14 | 98 | 134 | 40 | 49,166,952 | 0.030 |
| es40fst15 | 93 | 129 | 40 | 50,828,067 | 0.050 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 18** Results on the testset ES50FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es50fst01 | 118 | 160 | 50 | 54,948,660 | 0.053 |
| es50fst02 | 125 | 177 | 50 | 55,484,245 | 0.474 |
| es50fst03 | 128 | 182 | 50 | 54,691,035 | 0.075 |
| es50fst04 | 106 | 138 | 50 | 51,535,766 | 0.348 |
| es50fst05 | 104 | 135 | 50 | 55,186,015 | 0.280 |
| es50fst06 | 126 | 182 | 50 | 55,804,287 | 0.539 |
| es50fst07 | 143 | 211 | 50 | 49,961,178 | 0.049 |
| es50fst08 | 83 | 96 | 50 | 53,754,708 | 0.089 |
| es50fst09 | 139 | 202 | 50 | 53,456,773 | 0.538 |
| es50fst10 | 139 | 207 | 50 | 54,037,963 | 3.006 |
| es50fst11 | 100 | 131 | 50 | 52,532,923 | 0.019 |
| es50fst12 | 110 | 149 | 50 | 53,409,291 | 0.172 |
| es50fst13 | 92 | 116 | 50 | 53,891,019 | 0.032 |
| es50fst14 | 120 | 167 | 50 | 53,551,419 | 0.082 |
| es50fst15 | 112 | 147 | 50 | 52,180,862 | 0.105 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 19** Results on the testset ES60FST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es60fst01 | 123 | 159 | 60 | 53,761,423 | 0.436 |
| es60fst02 | 186 | 280 | 60 | 55,367,804 | 3.033 |
| es60fst03 | 113 | 142 | 60 | 56,566,797 | 0.066 |
| es60fst04 | 162 | 238 | 60 | 55,371,042 | 0.118 |
| es60fst05 | 119 | 148 | 60 | 54,704,991 | 0.059 |
| es60fst06 | 130 | 174 | 60 | 60,421,961 | 1.064 |
| es60fst07 | 188 | 280 | 60 | 58,978,041 | 0.073 |
| es60fst08 | 109 | 133 | 60 | 58,138,178 | 0.366 |
| es60fst09 | 151 | 216 | 60 | 55,877,112 | 0.493 |
| es60fst10 | 133 | 177 | 60 | 57,624,488 | 0.039 |
| es60fst11 | 121 | 154 | 60 | 56,141,666 | 0.523 |

**Table 19** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| es60fst12 | 176 | 257 | 60 | 59,791,362 | 3.289 |
| es60fst13 | 157 | 226 | 60 | 61,213,533 | 0.091 |
| es60fst14 | 118 | 149 | 60 | 56,035,528 | 0.058 |
| es60fst15 | 117 | 151 | 60 | 56,622,581 | 0.104 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 20** Results on the testset TSPFST

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| att48fst | 139 | 202 | 48 | 30,236 | 0.453 |
| berlin52fst | 89 | 104 | 52 | 6760 | 33.108 |
| eil51fst | 181 | 289 | 51 | 409 | 513.690 |

Type: Rectilinear instances after FST-preprocessing by GeoSteiner

**Table 21** Results on the testset Copenhagen14

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| ind1 | 18 | 31 | 10 | 604 | 0.001 |
| ind2 | 31 | 57 | 10 | 9500 | 0.001 |
| ind3 | 16 | 23 | 10 | 600 | 0.000 |
| ind4 | 74 | 146 | 25 | 1086 | 0.023 |
| ind5 | 114 | 228 | 33 | 1341 | 0.021 |
| rc01 | 21 | 35 | 10 | 25,980 | 0.000 |
| rc02 | 87 | 176 | 30 | 41,350 | 0.029 |
| rc03 | 109 | 202 | 50 | 54,160 | 0.153 |
| rt01 | 262 | 740 | 10 | 2146 | 0.002 |
| rt02 | 788 | 1938 | 50 | 45,852 | 0.703 |

Type: Instances of the Obstacle-avoiding rectilinear Steiner tree problem after FSTpreprocessing by ObSteiner and merging the FSTs into a single graph

**Table 22** Results on the testset WRP3

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| wrp3-11 | 128 | 227 | 11 | 1,100,361 | 0.099 |
| wrp3-12 | 84 | 149 | 12 | 1,200,237 | 0.123 |
| wrp3-13 | 311 | 613 | 13 | 1,300,497 | 20.014 |
| wrp3-14 | 128 | 247 | 14 | 1,400,250 | 3.780 |
| wrp3-15 | 138 | 257 | 15 | 1,500,422 | 51.156 |
| wrp3-16 | 204 | 374 | 16 | 1,600,208 | 11.622 |
| wrp3-17 | 177 | 354 | 17 | 1,700,442 | 422.079 |
| wrp3-19 | 189 | 353 | 19 | 1,900,439 | 1765.723 |
| wrp3-20 | 245 | 454 | 20 | – | Timeout |
| wrp3-21 | 237 | 444 | 21 | – | Timeout |
| wrp3-22 | 233 | 431 | 22 | – | Timeout |

**Table 22** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| wrp3-23 | 132 | 230 | 23 | – | Timeout |
| wrp3-24 | 262 | 487 | 24 | – | Timeout |
| wrp3-25 | 246 | 468 | 25 | – | Memout |
| wrp3-26 | 402 | 780 | 26 | – | Memout |
| wrp3-27 | 370 | 721 | 27 | – | Timeout |
| wrp3-28 | 307 | 559 | 28 | – | Memout |
| wrp3-29 | 245 | 436 | 29 | – | Memout |
| wrp3-30 | 467 | 896 | 30 | – | Memout |
| wrp3-31 | 323 | 592 | 31 | – | Timeout |
| wrp3-33 | 437 | 838 | 33 | – | Memout |
| wrp3-34 | 1244 | 2474 | 34 | – | Memout |
| wrp3-36 | 435 | 818 | 36 | – | Memout |
| wrp3-37 | 1011 | 2010 | 37 | – | Memout |
| wrp3-38 | 603 | 1207 | 38 | – | Timeout |
| wrp3-39 | 703 | 1616 | 39 | – | Memout |
| wrp3-41 | 178 | 307 | 41 | – | Memout |
| wrp3-42 | 705 | 1373 | 42 | – | Memout |
| wrp3-43 | 173 | 298 | 43 | – | Memout |
| wrp3-45 | 1414 | 2813 | 45 | – | Memout |
| wrp3-48 | 925 | 1738 | 48 | – | Memout |
| wrp3-49 | 886 | 1800 | 49 | – | Memout |
| wrp3-50 | 1119 | 2251 | 50 | – | Memout |
| wrp3-52 | 701 | 1352 | 52 | – | Memout |
| wrp3-53 | 775 | 1471 | 53 | – | Memout |
| wrp3-55 | 1645 | 3186 | 55 | – | Memout |
| wrp3-56 | 853 | 1590 | 56 | – | Memout |
| wrp3-60 | 838 | 1763 | 60 | – | Memout |
| wrp3-62 | 670 | 1316 | 62 | – | Memout |

Type: Group Steiner tree instances arising from VLSI design modeled as Steiner tree instances by connecting each terminal to the vertices if its group by edges of very high cost

**Table 23** Results on the testset WRP4

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| wrp4-11 | 123 | 233 | 11 | 1,100,179 | 0.241 |
| wrp4-13 | 110 | 188 | 13 | 1,300,798 | 0.019 |
| wrp4-14 | 145 | 283 | 14 | 1,400,290 | 2.798 |
| wrp4-15 | 193 | 369 | 15 | 1,500,405 | 8.193 |
| wrp4-16 | 311 | 579 | 16 | 1,601,190 | 149.542 |
| wrp4-17 | 223 | 404 | 17 | 1,700,525 | 69.524 |

**Table 23** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| wrp4-18 | 211 | 380 | 18 | 1,801,464 | 1212.777 |
| wrp4-19 | 119 | 206 | 19 | 1,901,446 | 0.925 |
| wrp4-21 | 529 | 1032 | 21 | – | Timeout |
| wrp4-22 | 294 | 568 | 22 | – | Timeout |
| wrp4-23 | 257 | 515 | 23 | – | Memout |
| wrp4-24 | 493 | 963 | 24 | – | Memout |
| wrp4-25 | 422 | 808 | 25 | – | Memout |
| wrp4-26 | 396 | 781 | 26 | – | Memout |
| wrp4-27 | 243 | 497 | 27 | – | Memout |
| wrp4-28 | 272 | 545 | 28 | – | Memout |
| wrp4-29 | 247 | 505 | 29 | – | Memout |
| wrp4-30 | 361 | 724 | 30 | – | Memout |
| wrp4-31 | 390 | 786 | 31 | – | Memout |
| wrp4-32 | 311 | 632 | 32 | – | Memout |
| wrp4-33 | 304 | 571 | 33 | – | Timeout |
| wrp4-34 | 314 | 650 | 34 | – | Memout |
| wrp4-35 | 471 | 954 | 35 | – | Memout |
| wrp4-36 | 363 | 750 | 36 | – | Memout |
| wrp4-37 | 522 | 1054 | 37 | – | Memout |
| wrp4-38 | 294 | 618 | 38 | – | Memout |
| wrp4-39 | 802 | 1553 | 39 | – | Memout |
| wrp4-40 | 538 | 1088 | 40 | – | Memout |
| wrp4-41 | 465 | 955 | 41 | – | Memout |
| wrp4-42 | 552 | 1131 | 42 | – | Memout |
| wrp4-43 | 596 | 1148 | 43 | – | Timeout |
| wrp4-44 | 398 | 788 | 44 | – | Memout |
| wrp4-45 | 388 | 815 | 45 | – | Memout |
| wrp4-46 | 632 | 1287 | 46 | – | Memout |
| wrp4-47 | 555 | 1098 | 47 | – | Memout |
| wrp4-48 | 451 | 825 | 48 | – | Timeout |
| wrp4-49 | 557 | 1080 | 49 | – | Memout |
| wrp4-50 | 564 | 1112 | 50 | – | Memout |
| wrp4-51 | 668 | 1306 | 51 | – | Memout |
| wrp4-52 | 547 | 1115 | 52 | – | Memout |
| wrp4-53 | 615 | 1232 | 53 | – | Memout |
| wrp4-54 | 688 | 1388 | 54 | – | Memout |
| wrp4-55 | 610 | 1201 | 55 | – | Memout |

**Table 23** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| wrp4-56 | 839 | 1617 | 56 | – | Memout |
| wrp4-58 | 757 | 1493 | 58 | – | Memout |
| wrp4-59 | 904 | 1806 | 59 | – | Memout |
| wrp4-60 | 693 | 1370 | 60 | – | Memout |
| wrp4-61 | 775 | 1538 | 61 | – | Timeout |
| wrp4-62 | 1283 | 2493 | 62 | – | Memout |
| wrp4-63 | 1121 | 2227 | 63 | – | Memout |

Type: Group Steiner tree instances arising from VLSI design modeled as Steiner tree instances by connecting each terminal to the vertices if its group by edges of very high cost

**Table 24** Results on the testset vienna-i-advanced

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| I052a | 160 | 237 | 23 | 13,309,487 | 0.013 |
| I054a | 540 | 817 | 25 | 15,841,596 | 0.021 |
| I056a | 290 | 439 | 34 | 14,171,206 | 0.086 |

Type: Realworld telecommunication networks after an "advanced" preprocessing routine. We report the cost of an optimum solution in the original instance, computed as the sum of an optimum solution in the reduced instance and the fixed cost induced by the reductions

**Table 25** Results on the testset vienna-i-simple

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| I052 | 2363 | 3761 | 40 | – | Timeout |
| I054 | 3803 | 6213 | 38 | – | Timeout |
| I056 | 1991 | 3176 | 51 | – | Timeout |

Type: Real-world telecommunication networks after a "simple" preprocessing routine

**Table 26** Results on the testset X

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| berlin52 | 52 | 1326 | 16 | 1044 | 0.013 |
| brasil58 | 58 | 1653 | 25 | 13,655 | 0.005 |

Type: Complete graphs with Euclidean costs

**Table 27** Results on the testset P4E

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| p455 | 100 | 4950 | 5 | 1138 | 0.001 |
| p456 | 100 | 4950 | 5 | 1228 | 0.001 |
| p457 | 100 | 4950 | 10 | 1609 | 0.001 |
| p458 | 100 | 4950 | 10 | 1868 | 0.002 |
| p459 | 100 | 4950 | 20 | 2345 | 0.003 |
| p460 | 100 | 4950 | 20 | 2959 | 0.005 |
| p461 | 100 | 4950 | 50 | 4474 | 0.030 |
| p463 | 200 | 19,900 | 10 | 1510 | 0.005 |
| p464 | 200 | 19,900 | 20 | 2545 | 0.012 |
| p465 | 200 | 19,900 | 40 | 3853 | 0.066 |

Type: Complete graphs with Euclidean costs

**Table 28** Results on the testset P4Z

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| p401 | 100 | 4950 | 5 | 155 | 0.001 |
| p402 | 100 | 4950 | 5 | 116 | 0.001 |
| p403 | 100 | 4950 | 5 | 179 | 0.001 |
| p404 | 100 | 4950 | 10 | 270 | 0.001 |
| p405 | 100 | 4950 | 10 | 270 | 0.002 |
| p406 | 100 | 4950 | 10 | 290 | 0.003 |
| p407 | 100 | 4950 | 20 | 590 | 0.257 |
| p408 | 100 | 4950 | 20 | 542 | 0.154 |
| p409 | 100 | 4950 | 50 | 963 | 4350.849 |
| p410 | 100 | 4950 | 50 | 1010 | 245.304 |

Type: Complete graphs with random costs

**Table 29** Results on the testset P6E

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| p619 | 100 | 180 | 5 | 7485 | 0.001 |
| p620 | 100 | 180 | 5 | 8746 | 0.001 |
| p621 | 100 | 180 | 5 | 8688 | 0.001 |
| p622 | 100 | 180 | 10 | 15,972 | 0.001 |
| p623 | 100 | 180 | 10 | 19,496 | 0.002 |
| p624 | 100 | 180 | 20 | 20,246 | 0.003 |
| p625 | 100 | 180 | 20 | 23,078 | 0.003 |
| p626 | 100 | 180 | 20 | 22,346 | 0.011 |

**Table 29** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| p627 | 100 | 180 | 50 | 40,647 | 0.023 |
| p628 | 100 | 180 | 50 | 40,008 | 0.052 |
| p629 | 100 | 180 | 50 | 43,287 | 0.043 |
| p630 | 200 | 370 | 10 | 26,125 | 0.001 |
| p631 | 200 | 370 | 20 | 39,067 | 0.010 |
| p632 | 200 | 370 | 40 | 56,217 | 0.038 |

Type: Sparse graphs with Euclidean costs

**Table 30** Results on the testset P6Z

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| p602 | 100 | 180 | 5 | 8083 | 0.001 |
| p603 | 100 | 180 | 5 | 5022 | 0.001 |
| p604 | 100 | 180 | 10 | 11,397 | 0.001 |
| p605 | 100 | 180 | 10 | 10,355 | 0.001 |
| p606 | 100 | 180 | 11 | 13,048 | 0.001 |
| p607 | 100 | 180 | 21 | 15,358 | 0.003 |
| p608 | 100 | 180 | 21 | 14,439 | 0.002 |
| p609 | 100 | 180 | 20 | 18,263 | 0.004 |
| p610 | 100 | 180 | 50 | 30,161 | 0.034 |
| p611 | 100 | 180 | 50 | 26,903 | 0.090 |
| p612 | 100 | 180 | 50 | 30,258 | 0.107 |
| p613 | 200 | 370 | 10 | 18,429 | 0.001 |
| p614 | 200 | 370 | 20 | 27,276 | 0.017 |
| p615 | 200 | 370 | 40 | 42,474 | 0.049 |

Type: Sparse graphs with random costs

**Table 31** Results on the testset B

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| b01 | 50 | 63 | 9 | 82 | 0.001 |
| b02 | 50 | 63 | 13 | 83 | 0.003 |
| b03 | 50 | 63 | 25 | 138 | 0.136 |
| b04 | 50 | 100 | 9 | 59 | 0.001 |
| b05 | 50 | 100 | 13 | 61 | 0.008 |
| b06 | 50 | 100 | 25 | 122 | 0.256 |
| b07 | 75 | 94 | 13 | 111 | 0.005 |
| b08 | 75 | 94 | 19 | 104 | 0.004 |
| b09 | 75 | 94 | 38 | 220 | 1858.382 |
| b10 | 75 | 150 | 13 | 86 | 0.006 |
| b11 | 75 | 150 | 19 | 88 | 0.019 |
| b12 | 75 | 150 | 38 | 174 | 98.165 |

**Table 31** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| b13 | 100 | 125 | 17 | 165 | 0.111 |
| b14 | 100 | 125 | 25 | 235 | 1.704 |
| b15 | 100 | 125 | 50 | 318 | 176.192 |
| b16 | 100 | 200 | 17 | 127 | 0.008 |
| b17 | 100 | 200 | 25 | 131 | 266.814 |
| b18 | 100 | 200 | 50 | 218 | 3228.991 |

Type: Random sparse graphs with random costs

**Table 32** Results on the testset C

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| c01 | 500 | 625 | 5 | 85 | 0.002 |
| c02 | 500 | 625 | 10 | 144 | 0.005 |
| c06 | 500 | 1000 | 5 | 55 | 0.003 |
| c07 | 500 | 1000 | 10 | 102 | 0.024 |
| c11 | 500 | 2500 | 5 | 32 | 0.004 |
| c12 | 500 | 2500 | 10 | 46 | 0.012 |
| c16 | 500 | 12,500 | 5 | 11 | 0.005 |
| c17 | 500 | 12,500 | 10 | 18 | 0.011 |

Type: Random sparse graphs with random costs

**Table 33** Results on the testset D

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| d01 | 1000 | 1250 | 5 | 106 | 0.004 |
| d02 | 1000 | 1250 | 10 | 220 | 0.052 |
| d06 | 1000 | 2000 | 5 | 67 | 0.004 |
| d07 | 1000 | 2000 | 10 | 103 | 0.010 |
| d11 | 1000 | 5000 | 5 | 29 | 0.007 |
| d12 | 1000 | 5000 | 10 | 42 | 0.016 |
| d16 | 1000 | 25,000 | 5 | 13 | 0.011 |
| d17 | 1000 | 25,000 | 10 | 23 | 0.072 |

Type: Random sparse graphs with random costs

**Table 34** Results on the testset E

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| e01 | 2500 | 3125 | 5 | 111 | 0.007 |
| e02 | 2500 | 3125 | 10 | 214 | 0.045 |
| e06 | 2500 | 5000 | 5 | 73 | 0.008 |
| e07 | 2500 | 5000 | 10 | 145 | 0.230 |
| e11 | 2500 | 12,500 | 5 | 34 | 0.009 |
| e12 | 2500 | 12,500 | 10 | 67 | 0.161 |
| e16 | 2500 | 62,500 | 5 | 15 | 0.016 |
| e17 | 2500 | 62,500 | 10 | 25 | 0.124 |

Type: Random sparse graphs with random costs

**Table 35** Results on the testset I080

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i080-001 | 80 | 120 | 6 | 1787 | 0.001 |
| i080-002 | 80 | 120 | 6 | 1607 | 0.000 |
| i080-003 | 80 | 120 | 6 | 1713 | 0.000 |
| i080-004 | 80 | 120 | 6 | 1866 | 0.000 |
| i080-005 | 80 | 120 | 6 | 1790 | 0.001 |
| i080-011 | 80 | 350 | 6 | 1479 | 0.001 |
| i080-012 | 80 | 350 | 6 | 1484 | 0.001 |
| i080-013 | 80 | 350 | 6 | 1381 | 0.001 |
| i080-014 | 80 | 350 | 6 | 1397 | 0.001 |
| i080-015 | 80 | 350 | 6 | 1495 | 0.001 |
| i080-021 | 80 | 3160 | 6 | 1175 | 0.004 |
| i080-022 | 80 | 3160 | 6 | 1178 | 0.004 |
| i080-023 | 80 | 3160 | 6 | 1174 | 0.004 |
| i080-024 | 80 | 3160 | 6 | 1161 | 0.004 |
| i080-025 | 80 | 3160 | 6 | 1162 | 0.004 |
| i080-031 | 80 | 160 | 6 | 1570 | 0.001 |
| i080-032 | 80 | 160 | 6 | 2088 | 0.001 |
| i080-033 | 80 | 160 | 6 | 1794 | 0.001 |
| i080-034 | 80 | 160 | 6 | 1688 | 0.001 |
| i080-035 | 80 | 160 | 6 | 1862 | 0.001 |
| i080-041 | 80 | 632 | 6 | 1276 | 0.001 |
| i080-042 | 80 | 632 | 6 | 1287 | 0.001 |
| i080-043 | 80 | 632 | 6 | 1295 | 0.001 |
| i080-044 | 80 | 632 | 6 | 1366 | 0.001 |
| i080-045 | 80 | 632 | 6 | 1310 | 0.001 |
| i080-101 | 80 | 120 | 8 | 2608 | 0.001 |
| i080-102 | 80 | 120 | 8 | 2403 | 0.001 |
| i080-103 | 80 | 120 | 8 | 2603 | 0.002 |
| i080-104 | 80 | 120 | 8 | 2486 | 0.002 |
| i080-105 | 80 | 120 | 8 | 2203 | 0.001 |
| i080-111 | 80 | 350 | 8 | 2051 | 0.008 |
| i080-112 | 80 | 350 | 8 | 1885 | 0.006 |
| i080-113 | 80 | 350 | 8 | 1884 | 0.004 |
| i080-114 | 80 | 350 | 8 | 1895 | 0.005 |
| i080-115 | 80 | 350 | 8 | 1868 | 0.004 |
| i080-121 | 80 | 3160 | 8 | 1561 | 0.023 |
| i080-122 | 80 | 3160 | 8 | 1561 | 0.023 |
| i080-123 | 80 | 3160 | 8 | 1569 | 0.023 |

**Table 35** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i080-124 | 80 | 3160 | 8 | 1555 | 0.022 |
| i080-125 | 80 | 3160 | 8 | 1572 | 0.024 |
| i080-131 | 80 | 160 | 8 | 2284 | 0.001 |
| i080-132 | 80 | 160 | 8 | 2180 | 0.003 |
| i080-133 | 80 | 160 | 8 | 2261 | 0.002 |
| i080-134 | 80 | 160 | 8 | 2070 | 0.002 |
| i080-135 | 80 | 160 | 8 | 2102 | 0.001 |
| i080-141 | 80 | 632 | 8 | 1788 | 0.009 |
| i080-142 | 80 | 632 | 8 | 1708 | 0.009 |
| i080-143 | 80 | 632 | 8 | 1767 | 0.016 |
| i080-144 | 80 | 632 | 8 | 1772 | 0.015 |
| i080-145 | 80 | 632 | 8 | 1762 | 0.013 |
| i080-201 | 80 | 120 | 16 | 4760 | 0.128 |
| i080-202 | 80 | 120 | 16 | 4650 | 0.111 |
| i080-203 | 80 | 120 | 16 | 4599 | 0.891 |
| i080-204 | 80 | 120 | 16 | 4492 | 8.025 |
| i080-205 | 80 | 120 | 16 | 4564 | 0.615 |
| i080-211 | 80 | 350 | 16 | 3631 | 108.657 |
| i080-212 | 80 | 350 | 16 | 3677 | 104.799 |
| i080-213 | 80 | 350 | 16 | 3678 | 106.887 |
| i080-214 | 80 | 350 | 16 | 3734 | 107.154 |
| i080-215 | 80 | 350 | 16 | 3681 | 107.424 |
| i080-221 | 80 | 3160 | 16 | 3158 | 111.565 |
| i080-222 | 80 | 3160 | 16 | 3141 | 113.026 |
| i080-223 | 80 | 3160 | 16 | 3156 | 112.402 |
| i080-224 | 80 | 3160 | 16 | 3159 | 114.264 |
| i080-225 | 80 | 3160 | 16 | 3150 | 114.489 |
| i080-231 | 80 | 160 | 16 | 4354 | 32.829 |
| i080-232 | 80 | 160 | 16 | 4199 | 26.645 |
| i080-233 | 80 | 160 | 16 | 4118 | 16.691 |
| i080-234 | 80 | 160 | 16 | 4274 | 0.898 |
| i080-235 | 80 | 160 | 16 | 4487 | 2.326 |
| i080-241 | 80 | 632 | 16 | 3538 | 146.827 |
| i080-242 | 80 | 632 | 16 | 3458 | 141.589 |
| i080-243 | 80 | 632 | 16 | 3474 | 139.922 |
| i080-244 | 80 | 632 | 16 | 3466 | 143.147 |
| i080-245 | 80 | 632 | 16 | 3467 | 142.418 |
| i080-301 | 80 | 120 | 20 | 5519 | 38.991 |

**Table 35** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i080-302 | 80 | 120 | 20 | 5944 | 10.556 |
| i080-303 | 80 | 120 | 20 | 5777 | 19.155 |
| i080-304 | 80 | 120 | 20 | 5586 | 6.392 |
| i080-305 | 80 | 120 | 20 | 5932 | 201.235 |
| i080-311 | 80 | 350 | 20 | – | Timeout |
| i080-312 | 80 | 350 | 20 | – | Timeout |
| i080-313 | 80 | 350 | 20 | – | Timeout |
| i080-314 | 80 | 350 | 20 | – | Timeout |
| i080-315 | 80 | 350 | 20 | – | Timeout |
| i080-321 | 80 | 3160 | 20 | – | Timeout |
| i080-322 | 80 | 3160 | 20 | – | Timeout |
| i080-323 | 80 | 3160 | 20 | – | Timeout |
| i080-324 | 80 | 3160 | 20 | – | Timeout |
| i080-325 | 80 | 3160 | 20 | – | Timeout |
| i080-331 | 80 | 160 | 20 | 5226 | 814.502 |
| i080-332 | 80 | 160 | 20 | 5362 | 902.230 |
| i080-333 | 80 | 160 | 20 | 5381 | 541.448 |
| i080-334 | 80 | 160 | 20 | 5264 | 920.974 |
| i080-335 | 80 | 160 | 20 | 4953 | 1004.156 |
| i080-341 | 80 | 632 | 20 | – | Timeout |
| i080-342 | 80 | 632 | 20 | – | Timeout |
| i080-343 | 80 | 632 | 20 | – | Timeout |
| i080-344 | 80 | 632 | 20 | – | Timeout |
| i080-345 | 80 | 632 | 20 | – | Timeout |

Type: Random graphs with so-called incidence costs, designed to defy preprocessing

**Table 36** Results on the testset I160

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i160-001 | 160 | 240 | 7 | 2490 | 0.004 |
| i160-002 | 160 | 240 | 7 | 2158 | 0.001 |
| i160-003 | 160 | 240 | 7 | 2297 | 0.002 |
| i160-004 | 160 | 240 | 7 | 2370 | 0.002 |
| i160-005 | 160 | 240 | 7 | 2495 | 0.002 |
| i160-011 | 160 | 812 | 7 | 1677 | 0.007 |

**Table 36** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i160-012 | 160 | 812 | 7 | 1750 | 0.008 |
| i160-013 | 160 | 812 | 7 | 1661 | 0.005 |
| i160-014 | 160 | 812 | 7 | 1778 | 0.008 |
| i160-015 | 160 | 812 | 7 | 1768 | 0.009 |
| i160-021 | 160 | 12,720 | 7 | 1352 | 0.055 |
| i160-022 | 160 | 12,720 | 7 | 1365 | 0.057 |
| i160-023 | 160 | 12,720 | 7 | 1351 | 0.056 |
| i160-024 | 160 | 12,720 | 7 | 1371 | 0.058 |
| i160-025 | 160 | 12,720 | 7 | 1366 | 0.054 |
| i160-031 | 160 | 320 | 7 | 2170 | 0.003 |
| i160-032 | 160 | 320 | 7 | 2330 | 0.003 |
| i160-033 | 160 | 320 | 7 | 2101 | 0.005 |
| i160-034 | 160 | 320 | 7 | 2083 | 0.003 |
| i160-035 | 160 | 320 | 7 | 2103 | 0.004 |
| i160-041 | 160 | 2544 | 7 | 1494 | 0.013 |
| i160-042 | 160 | 2544 | 7 | 1486 | 0.013 |
| i160-043 | 160 | 2544 | 7 | 1549 | 0.016 |
| i160-044 | 160 | 2544 | 7 | 1478 | 0.011 |
| i160-045 | 160 | 2544 | 7 | 1554 | 0.015 |
| i160-101 | 160 | 240 | 12 | 3859 | 0.082 |
| i160-102 | 160 | 240 | 12 | 3747 | 0.205 |
| i160-103 | 160 | 240 | 12 | 3837 | 0.093 |
| i160-104 | 160 | 240 | 12 | 4063 | 0.014 |
| i160-105 | 160 | 240 | 12 | 3563 | 0.034 |
| i160-111 | 160 | 812 | 12 | 2869 | 1.430 |
| i160-112 | 160 | 812 | 12 | 2924 | 2.304 |
| i160-113 | 160 | 812 | 12 | 2866 | 1.575 |
| i160-114 | 160 | 812 | 12 | 2989 | 1.949 |
| i160-115 | 160 | 812 | 12 | 2937 | 1.436 |
| i160-121 | 160 | 12,720 | 12 | 2363 | 5.395 |
| i160-122 | 160 | 12,720 | 12 | 2348 | 5.355 |
| i160-123 | 160 | 12,720 | 12 | 2355 | 5.418 |
| i160-124 | 160 | 12,720 | 12 | 2352 | 5.330 |
| i160-125 | 160 | 12,720 | 12 | 2351 | 5.459 |
| i160-131 | 160 | 320 | 12 | 3356 | 0.306 |
| i160-132 | 160 | 320 | 12 | 3450 | 0.205 |
| i160-133 | 160 | 320 | 12 | 3585 | 0.345 |
| i160-134 | 160 | 320 | 12 | 3470 | 0.095 |

**Table 36** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i160-135 | 160 | 320 | 12 | 3716 | 0.286 |
| i160-141 | 160 | 2544 | 12 | 2549 | 3.398 |
| i160-142 | 160 | 2544 | 12 | 2562 | 3.434 |
| i160-143 | 160 | 2544 | 12 | 2557 | 2.955 |
| i160-144 | 160 | 2544 | 12 | 2607 | 3.238 |
| i160-145 | 160 | 2544 | 12 | 2578 | 3.451 |
| i160-201 | 160 | 240 | 24 | – | Timeout |
| i160-202 | 160 | 240 | 24 | – | Timeout |
| i160-203 | 160 | 240 | 24 | 7243 | 4173.679 |
| i160-204 | 160 | 240 | 24 | – | Timeout |
| i160-205 | 160 | 240 | 24 | – | Timeout |
| i160-211 | 160 | 812 | 24 | – | Timeout |
| i160-212 | 160 | 812 | 24 | – | Timeout |
| i160-213 | 160 | 812 | 24 | – | Timeout |
| i160-214 | 160 | 812 | 24 | – | Timeout |
| i160-215 | 160 | 812 | 24 | – | Timeout |
| i160-221 | 160 | 12,720 | 24 | – | Timeout |
| i160-222 | 160 | 12,720 | 24 | – | Timeout |
| i160-223 | 160 | 12,720 | 24 | – | Timeout |
| i160-224 | 160 | 12,720 | 24 | – | Timeout |
| i160-225 | 160 | 12,720 | 24 | – | Timeout |
| i160-231 | 160 | 320 | 24 | – | Timeout |
| i160-232 | 160 | 320 | 24 | – | Timeout |
| i160-233 | 160 | 320 | 24 | – | Timeout |
| i160-234 | 160 | 320 | 24 | – | Timeout |
| i160-235 | 160 | 320 | 24 | – | Timeout |
| i160-241 | 160 | 2544 | 24 | – | Timeout |
| i160-242 | 160 | 2544 | 24 | – | Timeout |
| i160-243 | 160 | 2544 | 24 | – | Timeout |
| i160-244 | 160 | 2544 | 24 | – | Timeout |
| i160-245 | 160 | 2544 | 24 | – | Timeout |
| i160-301 | 160 | 240 | 40 | – | Memout |
| i160-302 | 160 | 240 | 40 | – | Memout |
| i160-303 | 160 | 240 | 40 | – | Memout |
| i160-304 | 160 | 240 | 40 | – | Memout |
| i160-305 | 160 | 240 | 40 | – | Memout |

**Table 36** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i160-311 | 160 | 812 | 40 | – | Memout |
| i160-312 | 160 | 812 | 40 | – | Memout |
| i160-313 | 160 | 812 | 40 | – | Memout |
| i160-314 | 160 | 812 | 40 | – | Memout |
| i160-315 | 160 | 812 | 40 | – | Memout |
| i160-321 | 160 | 12,720 | 40 | – | Memout |
| i160-322 | 160 | 12,720 | 40 | – | Memout |
| i160-323 | 160 | 12,720 | 40 | – | Memout |
| i160-324 | 160 | 12,720 | 40 | – | Memout |
| i160-325 | 160 | 12,720 | 40 | – | Memout |
| i160-331 | 160 | 320 | 40 | – | Memout |
| i160-332 | 160 | 320 | 40 | – | Memout |
| i160-333 | 160 | 320 | 40 | – | Memout |
| i160-334 | 160 | 320 | 40 | – | Memout |
| i160-335 | 160 | 320 | 40 | – | Memout |
| i160-341 | 160 | 2544 | 40 | – | Memout |
| i160-342 | 160 | 2544 | 40 | – | Memout |
| i160-343 | 160 | 2544 | 40 | – | Memout |
| i160-344 | 160 | 2544 | 40 | – | Memout |
| i160-345 | 160 | 2544 | 40 | – | Memout |

Type: Random graphs with so-called incidence costs, designed to defy preprocessing

**Table 37** Results on the testset I320

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i320-001 | 320 | 480 | 8 | 2672 | 0.003 |
| i320-002 | 320 | 480 | 8 | 2847 | 0.004 |
| i320-003 | 320 | 480 | 8 | 2972 | 0.004 |
| i320-004 | 320 | 480 | 8 | 2905 | 0.007 |
| i320-005 | 320 | 480 | 8 | 2991 | 0.006 |
| i320-011 | 320 | 1845 | 8 | 2053 | 0.031 |
| i320-012 | 320 | 1845 | 8 | 1997 | 0.023 |
| i320-013 | 320 | 1845 | 8 | 2072 | 0.039 |
| i320-014 | 320 | 1845 | 8 | 2061 | 0.035 |
| i320-015 | 320 | 1845 | 8 | 2059 | 0.040 |
| i320-021 | 320 | 51,040 | 8 | 1553 | 0.366 |
| i320-022 | 320 | 51,040 | 8 | 1565 | 0.414 |
| i320-023 | 320 | 51,040 | 8 | 1549 | 0.373 |
| i320-024 | 320 | 51,040 | 8 | 1553 | 0.360 |
| i320-025 | 320 | 51,040 | 8 | 1550 | 0.360 |

**Table 37** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i320-031 | 320 | 640 | 8 | 2673 | 0.011 |
| i320-032 | 320 | 640 | 8 | 2770 | 0.010 |
| i320-033 | 320 | 640 | 8 | 2769 | 0.010 |
| i320-034 | 320 | 640 | 8 | 2521 | 0.005 |
| i320-035 | 320 | 640 | 8 | 2385 | 0.007 |
| i320-041 | 320 | 10,208 | 8 | 1707 | 0.061 |
| i320-042 | 320 | 10,208 | 8 | 1682 | 0.050 |
| i320-043 | 320 | 10,208 | 8 | 1723 | 0.059 |
| i320-044 | 320 | 10,208 | 8 | 1681 | 0.067 |
| i320-045 | 320 | 10,208 | 8 | 1686 | 0.056 |
| i320-101 | 320 | 480 | 17 | 5548 | 16.131 |
| i320-102 | 320 | 480 | 17 | 5556 | 10.289 |
| i320-103 | 320 | 480 | 17 | 6239 | 178.463 |
| i320-104 | 320 | 480 | 17 | 5703 | 108.739 |
| i320-105 | 320 | 480 | 17 | 5928 | 84.748 |
| i320-111 | 320 | 1845 | 17 | 4273 | 1706.674 |
| i320-112 | 320 | 1845 | 17 | 4213 | 1872.720 |
| i320-113 | 320 | 1845 | 17 | 4205 | 1464.609 |
| i320-114 | 320 | 1845 | 17 | 4104 | 1581.733 |
| i320-115 | 320 | 1845 | 17 | 4238 | 1747.827 |
| i320-121 | 320 | 51,040 | 17 | 3321 | 2381.939 |
| i320-122 | 320 | 51,040 | 17 | 3314 | 2336.603 |
| i320-123 | 320 | 51,040 | 17 | 3332 | 2379.991 |
| i320-124 | 320 | 51,040 | 17 | 3323 | 2342.527 |
| i320-125 | 320 | 51,040 | 17 | 3340 | 2353.655 |
| i320-131 | 320 | 640 | 17 | 5255 | 343.942 |
| i320-132 | 320 | 640 | 17 | 5052 | 33.234 |
| i320-133 | 320 | 640 | 17 | 5125 | 44.197 |
| i320-134 | 320 | 640 | 17 | 5272 | 374.938 |
| i320-135 | 320 | 640 | 17 | 5342 | 211.986 |
| i320-141 | 320 | 10,208 | 17 | 3606 | 2246.125 |
| i320-142 | 320 | 10,208 | 17 | 3567 | 2305.369 |
| i320-143 | 320 | 10,208 | 17 | 3561 | 2244.354 |
| i320-144 | 320 | 10,208 | 17 | 3512 | 2244.792 |
| i320-145 | 320 | 10,208 | 17 | 3601 | 2261.946 |
| i320-201 | 320 | 480 | 34 | – | Memout |
| i320-202 | 320 | 480 | 34 | – | Memout |
| i320-203 | 320 | 480 | 34 | – | Memout |

**Table 37** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i320-204 | 320 | 480 | 34 | – | Memout |
| i320-205 | 320 | 480 | 34 | – | Memout |
| i320-211 | 320 | 1845 | 34 | – | Memout |
| i320-212 | 320 | 1845 | 34 | – | Memout |
| i320-213 | 320 | 1845 | 34 | – | Memout |
| i320-214 | 320 | 1845 | 34 | – | Memout |
| i320-215 | 320 | 1845 | 34 | – | Memout |
| i320-221 | 320 | 51,040 | 34 | – | Memout |
| i320-222 | 320 | 51,040 | 34 | – | Memout |
| i320-223 | 320 | 51,040 | 34 | – | Memout |
| i320-224 | 320 | 51,040 | 34 | – | Memout |
| i320-225 | 320 | 51,040 | 34 | – | Memout |
| i320-231 | 320 | 640 | 34 | – | Memout |
| i320-232 | 320 | 640 | 34 | – | Memout |
| i320-233 | 320 | 640 | 34 | – | Memout |
| i320-234 | 320 | 640 | 34 | – | Memout |
| i320-235 | 320 | 640 | 34 | – | Memout |
| i320-241 | 320 | 10,208 | 34 | – | Memout |
| i320-242 | 320 | 10,208 | 34 | – | Memout |
| i320-243 | 320 | 10,208 | 34 | – | Memout |
| i320-244 | 320 | 10,208 | 34 | – | Memout |
| i320-245 | 320 | 10,208 | 34 | – | Memout |

Type: Random graphs with so-called incidence costs, designed to defy preprocessing

**Table 38** Results on the testset I640

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i640-001 | 640 | 960 | 9 | 4033 | 0.042 |
| i640-002 | 640 | 960 | 9 | 3588 | 0.030 |
| i640-003 | 640 | 960 | 9 | 3438 | 0.024 |
| i640-004 | 640 | 960 | 9 | 4000 | 0.070 |
| i640-005 | 640 | 960 | 9 | 4006 | 0.055 |
| i640-011 | 640 | 4135 | 9 | 2392 | 0.163 |
| i640-012 | 640 | 4135 | 9 | 2465 | 0.271 |
| i640-013 | 640 | 4135 | 9 | 2399 | 0.205 |
| i640-014 | 640 | 4135 | 9 | 2171 | 0.038 |
| i640-015 | 640 | 4135 | 9 | 2347 | 0.126 |
| i640-021 | 640 | 204,480 | 9 | 1749 | 4.426 |
| i640-022 | 640 | 204,480 | 9 | 1756 | 4.031 |
| i640-023 | 640 | 204,480 | 9 | 1754 | 4.499 |

**Table 38** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i640-024 | 640 | 204,480 | 9 | 1751 | 4.081 |
| i640-025 | 640 | 204,480 | 9 | 1745 | 4.452 |
| i640-031 | 640 | 1280 | 9 | 3278 | 0.065 |
| i640-032 | 640 | 1280 | 9 | 3187 | 0.056 |
| i640-033 | 640 | 1280 | 9 | 3260 | 0.062 |
| i640-034 | 640 | 1280 | 9 | 2953 | 0.015 |
| i640-035 | 640 | 1280 | 9 | 3292 | 0.033 |
| i640-041 | 640 | 40,896 | 9 | 1897 | 0.957 |
| i640-042 | 640 | 40,896 | 9 | 1934 | 0.811 |
| i640-043 | 640 | 40,896 | 9 | 1931 | 0.889 |
| i640-044 | 640 | 40,896 | 9 | 1938 | 0.975 |
| i640-045 | 640 | 40,896 | 9 | 1866 | 0.403 |
| i640-101 | 640 | 960 | 25 | – | Timeout |
| i640-102 | 640 | 960 | 25 | – | Timeout |
| i640-103 | 640 | 960 | 25 | – | Timeout |
| i640-104 | 640 | 960 | 25 | – | Timeout |
| i640-105 | 640 | 960 | 25 | – | Timeout |
| i640-111 | 640 | 4135 | 25 | – | Memout |
| i640-112 | 640 | 4135 | 25 | – | Memout |
| i640-113 | 640 | 4135 | 25 | – | Memout |
| i640-114 | 640 | 4135 | 25 | – | Memout |
| i640-115 | 640 | 4135 | 25 | – | Memout |
| i640-121 | 640 | 204,480 | 25 | – | Memout |
| i640-122 | 640 | 204,480 | 25 | – | Memout |
| i640-123 | 640 | 204,480 | 25 | – | Memout |
| i640-124 | 640 | 204,480 | 25 | – | Memout |
| i640-125 | 640 | 204,480 | 25 | – | Memout |
| i640-131 | 640 | 1280 | 25 | – | Timeout |
| i640-132 | 640 | 1280 | 25 | – | Timeout |
| i640-133 | 640 | 1280 | 25 | – | Timeout |
| i640-134 | 640 | 1280 | 25 | – | Timeout |
| i640-135 | 640 | 1280 | 25 | – | Timeout |
| i640-141 | 640 | 40,896 | 25 | – | Memout |
| i640-142 | 640 | 40,896 | 25 | – | Memout |
| i640-143 | 640 | 40,896 | 25 | – | Memout |
| i640-144 | 640 | 40,896 | 25 | – | Memout |
| i640-145 | 640 | 40,896 | 25 | – | Memout |

**Table 38** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| i640-201 | 640 | 960 | 50 | – | Memout |
| i640-202 | 640 | 960 | 50 | – | Memout |
| i640-203 | 640 | 960 | 50 | – | Memout |
| i640-204 | 640 | 960 | 50 | – | Memout |
| i640-205 | 640 | 960 | 50 | – | Memout |
| i640-211 | 640 | 4135 | 50 | – | Memout |
| i640-212 | 640 | 4135 | 50 | – | Memout |
| i640-213 | 640 | 4135 | 50 | – | Memout |
| i640-214 | 640 | 4135 | 50 | – | Memout |
| i640-215 | 640 | 4135 | 50 | – | Memout |
| i640-221 | 640 | 204,480 | 50 | – | Memout |
| i640-222 | 640 | 204,480 | 50 | – | Memout |
| i640-223 | 640 | 204,480 | 50 | – | Memout |
| i640-224 | 640 | 204,480 | 50 | – | Memout |
| i640-225 | 640 | 204,480 | 50 | – | Memout |
| i640-231 | 640 | 1280 | 50 | – | Memout |
| i640-232 | 640 | 1280 | 50 | – | Memout |
| i640-233 | 640 | 1280 | 50 | – | Memout |
| i640-234 | 640 | 1280 | 50 | – | Memout |
| i640-235 | 640 | 1280 | 50 | – | Memout |
| i640-241 | 640 | 40,896 | 50 | – | Memout |
| i640-242 | 640 | 40,896 | 50 | – | Memout |
| i640-243 | 640 | 40,896 | 50 | – | Memout |
| i640-244 | 640 | 40,896 | 50 | – | Memout |
| i640-245 | 640 | 40,896 | 50 | – | Memout |

Type: Random graphs with so-called incidence costs, designed to defy preprocessing

**Table 39** Results on the testset PUC

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| bipe2p | 550 | 5013 | 50 | – | Memout |
| bipe2u | 550 | 5013 | 50 | – | Memout |
| cc3-10p | 1000 | 13,500 | 50 | – | Memout |
| cc3-10u | 1000 | 13,500 | 50 | – | Memout |
| cc3-11p | 1331 | 19,965 | 61 | – | Memout |
| cc3-11u | 1331 | 19,965 | 61 | – | Memout |
| cc3-4p | 64 | 288 | 8 | 2338 | 0.006 |
| cc3-4u | 64 | 288 | 8 | 23 | 0.008 |
| cc3-5p | 125 | 750 | 13 | 3661 | 3.450 |
| cc3-5u | 125 | 750 | 13 | 36 | 4.637 |

**Table 39** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| cc5-3p | 243 | 1215 | 27 | – | Memout |
| cc5-3u | 243 | 1215 | 27 | – | Memout |
| cc6-2p | 64 | 192 | 12 | 3271 | 0.166 |
| cc6-2u | 64 | 192 | 12 | 32 | 0.279 |
| hc6p | 64 | 192 | 32 | – | Memout |
| hc6u | 64 | 192 | 32 | – | Memout |

Type: Artificial instances designed to be hard for existing solvers

**Table 40** Results on the testset SPG-PUCN

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| cc3-10n | 1000 | 13,500 | 50 | – | Memout |
| cc3-11n | 1331 | 19,965 | 61 | – | Memout |
| cc3-4n | 64 | 288 | 8 | 13 | 0.002 |
| cc3-5n | 125 | 750 | 13 | 20 | 0.563 |
| cc5-3n | 243 | 1215 | 27 | – | Memout |
| cc6-2n | 64 | 192 | 12 | 18 | 0.047 |

Type: Unweighted instances of the PUC testset, which contains artificial instances designed to be hard for existing solvers

**Table 41** Results on the testset SP

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| antiwheel5 | 10 | 15 | 5 | 7 | 0.000 |
| design432 | 8 | 20 | 4 | 9 | 0.000 |
| oddcycle3 | 6 | 9 | 3 | 4 | 0.000 |
| oddwheel3 | 7 | 9 | 4 | 5 | 0.000 |
| se03 | 13 | 21 | 4 | 12 | 0.000 |

Type: Artificial instances

**Table 42** Results on the testset MC

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| mc2 | 120 | 7140 | 60 | – | Memout |
| mc3 | 97 | 4656 | 45 | – | Timeout |

Type: Artificial instances

**Table 43** Results on the testset csd

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| csd02 | 3 | 2 | 1 | 0 | 0.000 |
| csd03 | 6 | 6 | 3 | 4 | 0.000 |
| csd04 | 10 | 12 | 6 | 8 | 0.000 |
| csd05 | 15 | 20 | 10 | 13 | 0.007 |
| csd06 | 21 | 30 | 15 | 19 | 2.724 |
| csd07 | 28 | 42 | 21 | 26 | 6138.678 |
| csd08 | 36 | 56 | 28 | – | Timeout |
| csd09 | 45 | 72 | 36 | – | Memout |
| csd10 | 55 | 90 | 45 | – | Memout |
| csd11 | 66 | 110 | 55 | – | Memout |

Type: Artificial instances arising from generalizations of Steiner tree LP gap examples

**Table 44** Results on the testset goemans

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g01-00 | 7 | 9 | 3 | 8 | 0.000 |
| g01-01 | 8 | 10 | 4 | 9 | 0.000 |
| g01-02 | 9 | 11 | 5 | 10 | 0.000 |
| g01-03 | 10 | 12 | 6 | 11 | 0.000 |
| g01-04 | 11 | 13 | 7 | 12 | 0.000 |
| g01-05 | 12 | 14 | 8 | 13 | 0.001 |
| g01-06 | 13 | 15 | 9 | 14 | 0.004 |
| g01-07 | 14 | 16 | 10 | 15 | 0.007 |
| g01-08 | 15 | 17 | 11 | 16 | 0.016 |
| g01-09 | 16 | 18 | 12 | 17 | 0.067 |
| g01-10 | 17 | 19 | 13 | 18 | 0.172 |
| g01-11 | 18 | 20 | 14 | 19 | 0.490 |
| g01-12 | 19 | 21 | 15 | 20 | 2.220 |
| g01-13 | 20 | 22 | 16 | 21 | 18.637 |
| g01-14 | 21 | 23 | 17 | 22 | 42.576 |
| g01-15 | 22 | 24 | 18 | 23 | 39.915 |
| g02-00 | 9 | 16 | 4 | 14 | 0.000 |
| g02-01 | 10 | 17 | 5 | 15 | 0.000 |
| g02-02 | 11 | 18 | 6 | 16 | 0.000 |
| g02-03 | 12 | 19 | 7 | 17 | 0.000 |
| g02-04 | 13 | 20 | 8 | 18 | 0.001 |
| g02-05 | 14 | 21 | 9 | 19 | 0.004 |
| g02-06 | 15 | 22 | 10 | 20 | 0.013 |
| g02-07 | 16 | 23 | 11 | 21 | 0.035 |
| g02-08 | 17 | 24 | 12 | 22 | 0.135 |
| g02-09 | 18 | 25 | 13 | 23 | 0.352 |
| g02-10 | 19 | 26 | 14 | 24 | 1.354 |
| g02-11 | 20 | 27 | 15 | 25 | 6.346 |
| g02-12 | 21 | 28 | 16 | 26 | 25.602 |
| g02-13 | 22 | 29 | 17 | 27 | 114.366 |
| g02-14 | 23 | 30 | 18 | 28 | 300.818 |
| g02-15 | 24 | 31 | 19 | 29 | 1171.010 |
| g03-00 | 11 | 25 | 5 | 22 | 0.000 |
| g03-01 | 12 | 26 | 6 | 23 | 0.000 |
| g03-02 | 13 | 27 | 7 | 24 | 0.000 |
| g03-03 | 14 | 28 | 8 | 25 | 0.001 |
| g03-04 | 15 | 29 | 9 | 26 | 0.004 |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g03-05 | 16 | 30 | 10 | 27 | 0.016 |
| g03-06 | 17 | 31 | 11 | 28 | 0.067 |
| g03-07 | 18 | 32 | 12 | 29 | 0.145 |
| g03-08 | 19 | 33 | 13 | 30 | 0.417 |
| g03-09 | 20 | 34 | 14 | 31 | 1.536 |
| g03-10 | 21 | 35 | 15 | 32 | 6.411 |
| g03-11 | 22 | 36 | 16 | 33 | 28.536 |
| g03-12 | 23 | 37 | 17 | 34 | 106.674 |
| g03-13 | 24 | 38 | 18 | 35 | 407.020 |
| g03-14 | 25 | 39 | 19 | 36 | 1414.010 |
| g03-15 | 26 | 40 | 20 | 37 | 4935.897 |
| g04-00 | 13 | 36 | 6 | 32 | 0.000 |
| g04-01 | 14 | 37 | 7 | 33 | 0.001 |
| g04-02 | 15 | 38 | 8 | 34 | 0.002 |
| g04-03 | 16 | 39 | 9 | 35 | 0.007 |
| g04-04 | 17 | 40 | 10 | 36 | 0.022 |
| g04-05 | 18 | 41 | 11 | 37 | 0.062 |
| g04-06 | 19 | 42 | 12 | 38 | 0.155 |
| g04-07 | 20 | 43 | 13 | 39 | 0.424 |
| g04-08 | 21 | 44 | 14 | 40 | 1.571 |
| g04-09 | 22 | 45 | 15 | 41 | 6.114 |
| g04-10 | 23 | 46 | 16 | 42 | 26.880 |
| g04-11 | 24 | 47 | 17 | 43 | 103.263 |
| g04-12 | 25 | 48 | 18 | 44 | 379.413 |
| g04-13 | 26 | 49 | 19 | 45 | 1380.378 |
| g04-14 | 27 | 50 | 20 | 46 | 4854.754 |
| g04-15 | 28 | 51 | 21 | – | Timeout |
| g05-00 | 15 | 49 | 7 | 44 | 0.001 |
| g05-01 | 16 | 50 | 8 | 45 | 0.002 |
| g05-02 | 17 | 51 | 9 | 46 | 0.007 |
| g05-03 | 18 | 52 | 10 | 47 | 0.023 |
| g05-04 | 19 | 53 | 11 | 48 | 0.065 |
| g05-05 | 20 | 54 | 12 | 49 | 0.164 |
| g05-06 | 21 | 55 | 13 | 50 | 0.417 |
| g05-07 | 22 | 56 | 14 | 51 | 1.502 |
| g05-08 | 23 | 57 | 15 | 52 | 5.839 |
| g05-09 | 24 | 58 | 16 | 53 | 22.368 |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g05-10 | 25 | 59 | 17 | 54 | 89.853 |
| g05-11 | 26 | 60 | 18 | 55 | 331.230 |
| g05-12 | 27 | 61 | 19 | 56 | 1183.867 |
| g05-13 | 28 | 62 | 20 | 57 | 4559.627 |
| g05-14 | 29 | 63 | 21 | – | Timeout |
| g05-15 | 30 | 64 | 22 | – | Timeout |
| g06-00 | 17 | 64 | 8 | 58 | 0.002 |
| g06-01 | 18 | 65 | 9 | 59 | 0.006 |
| g06-02 | 19 | 66 | 10 | 60 | 0.022 |
| g06-03 | 20 | 67 | 11 | 61 | 0.061 |
| g06-04 | 21 | 68 | 12 | 62 | 0.150 |
| g06-05 | 22 | 69 | 13 | 63 | 0.403 |
| g06-06 | 23 | 70 | 14 | 64 | 1.441 |
| g06-07 | 24 | 71 | 15 | 65 | 5.374 |
| g06-08 | 25 | 72 | 16 | 66 | 20.749 |
| g06-09 | 26 | 73 | 17 | 67 | 80.435 |
| g06-10 | 27 | 74 | 18 | 68 | 307.301 |
| g06-11 | 28 | 75 | 19 | 69 | 1129.370 |
| g06-12 | 29 | 76 | 20 | 70 | 4294.387 |
| g06-13 | 30 | 77 | 21 | – | Timeout |
| g06-14 | 31 | 78 | 22 | – | Timeout |
| g06-15 | 32 | 79 | 23 | – | Timeout |
| g07-00 | 19 | 81 | 9 | 74 | 0.003 |
| g07-01 | 20 | 82 | 10 | 75 | 0.011 |
| g07-02 | 21 | 83 | 11 | 76 | 0.037 |
| g07-03 | 22 | 84 | 12 | 77 | 0.112 |
| g07-04 | 23 | 85 | 13 | 78 | 0.390 |
| g07-05 | 24 | 86 | 14 | 79 | 1.370 |
| g07-06 | 25 | 87 | 15 | 80 | 5.032 |
| g07-07 | 26 | 88 | 16 | 81 | 19.383 |
| g07-08 | 27 | 89 | 17 | 82 | 76.224 |
| g07-09 | 28 | 90 | 18 | 83 | 276.931 |
| g07-10 | 29 | 91 | 19 | 84 | 1064.745 |
| g07-11 | 30 | 92 | 20 | 85 | 4173.155 |
| g07-12 | 31 | 93 | 21 | – | Timeout |
| g07-13 | 32 | 94 | 22 | – | Timeout |
| g07-14 | 33 | 95 | 23 | – | Timeout |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g07-15 | 34 | 96 | 24 | – | Timeout |
| g08-00 | 21 | 100 | 10 | 92 | 0.009 |
| g08-01 | 22 | 101 | 11 | 93 | 0.032 |
| g08-02 | 23 | 102 | 12 | 94 | 0.108 |
| g08-03 | 24 | 103 | 13 | 95 | 0.383 |
| g08-04 | 25 | 104 | 14 | 96 | 1.309 |
| g08-05 | 26 | 105 | 15 | 97 | 4.942 |
| g08-06 | 27 | 106 | 16 | 98 | 18.081 |
| g08-07 | 28 | 107 | 17 | 99 | 68.314 |
| g08-08 | 29 | 108 | 18 | 100 | 262.768 |
| g08-09 | 30 | 109 | 19 | 101 | 1019.464 |
| g08-10 | 31 | 110 | 20 | 102 | 3956.305 |
| g08-11 | 32 | 111 | 21 | – | Timeout |
| g08-12 | 33 | 112 | 22 | – | Timeout |
| g08-13 | 34 | 113 | 23 | – | Timeout |
| g08-14 | 35 | 114 | 24 | – | Timeout |
| g08-15 | 36 | 115 | 25 | – | Timeout |
| g09-00 | 23 | 121 | 11 | 112 | 0.028 |
| g09-01 | 24 | 122 | 12 | 113 | 0.150 |
| g09-02 | 25 | 123 | 13 | 114 | 0.383 |
| g09-03 | 26 | 124 | 14 | 115 | 1.291 |
| g09-04 | 27 | 125 | 15 | 116 | 4.625 |
| g09-05 | 28 | 126 | 16 | 117 | 17.479 |
| g09-06 | 29 | 127 | 17 | 118 | 66.582 |
| g09-07 | 30 | 128 | 18 | 119 | 240.706 |
| g09-08 | 31 | 129 | 19 | 120 | 939.553 |
| g09-09 | 32 | 130 | 20 | 121 | 3748.635 |
| g09-10 | 33 | 131 | 21 | – | Timeout |
| g09-11 | 34 | 132 | 22 | – | Timeout |
| g09-12 | 35 | 133 | 23 | – | Timeout |
| g09-13 | 36 | 134 | 24 | – | Timeout |
| g09-14 | 37 | 135 | 25 | – | Timeout |
| g09-15 | 38 | 136 | 26 | – | Timeout |
| g10-00 | 25 | 144 | 12 | 134 | 0.126 |
| g10-01 | 26 | 145 | 13 | 135 | 0.351 |
| g10-02 | 27 | 146 | 14 | 136 | 1.228 |
| g10-03 | 28 | 147 | 15 | 137 | 4.474 |
| g10-04 | 29 | 148 | 16 | 138 | 17.188 |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g10-05 | 30 | 149 | 17 | 139 | 63.640 |
| g10-06 | 31 | 150 | 18 | 140 | 223.035 |
| g10-07 | 32 | 151 | 19 | 141 | 878.752 |
| g10-08 | 33 | 152 | 20 | 142 | 3612.406 |
| g10-09 | 34 | 153 | 21 | – | Timeout |
| g10-10 | 35 | 154 | 22 | – | Timeout |
| g10-11 | 36 | 155 | 23 | – | Timeout |
| g10-12 | 37 | 156 | 24 | – | Timeout |
| g10-13 | 38 | 157 | 25 | – | Timeout |
| g10-14 | 39 | 158 | 26 | – | Timeout |
| g10-15 | 40 | 159 | 27 | – | Timeout |
| g11-00 | 27 | 169 | 13 | 158 | 0.369 |
| g11-01 | 28 | 170 | 14 | 159 | 1.147 |
| g11-02 | 29 | 171 | 15 | 160 | 4.529 |
| g11-03 | 30 | 172 | 16 | 161 | 16.525 |
| g11-04 | 31 | 173 | 17 | 162 | 61.300 |
| g11-05 | 32 | 174 | 18 | 163 | 216.074 |
| g11-06 | 33 | 175 | 19 | 164 | 845.014 |
| g11-07 | 34 | 176 | 20 | 165 | 3514.339 |
| g11-08 | 35 | 177 | 21 | – | Timeout |
| g11-09 | 36 | 178 | 22 | – | Timeout |
| g11-10 | 37 | 179 | 23 | – | Timeout |
| g11-11 | 38 | 180 | 24 | – | Timeout |
| g11-12 | 39 | 181 | 25 | – | Timeout |
| g11-13 | 40 | 182 | 26 | – | Timeout |
| g11-14 | 41 | 183 | 27 | – | Timeout |
| g11-15 | 42 | 184 | 28 | – | Timeout |
| g12-00 | 29 | 196 | 14 | 184 | 0.626 |
| g12-01 | 30 | 197 | 15 | 185 | 4.491 |
| g12-02 | 31 | 198 | 16 | 186 | 15.098 |
| g12-03 | 32 | 199 | 17 | 187 | 59.824 |
| g12-04 | 33 | 200 | 18 | 188 | 201.250 |
| g12-05 | 34 | 201 | 19 | 189 | 790.252 |
| g12-06 | 35 | 202 | 20 | 190 | 3361.075 |
| g12-07 | 36 | 203 | 21 | – | Timeout |
| g12-08 | 37 | 204 | 22 | – | Timeout |
| g12-09 | 38 | 205 | 23 | – | Timeout |
| g12-10 | 39 | 206 | 24 | – | Timeout |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g12-11 | 40 | 207 | 25 | – | Timeout |
| g12-12 | 41 | 208 | 26 | – | Timeout |
| g12-13 | 42 | 209 | 27 | – | Timeout |
| g12-14 | 43 | 210 | 28 | – | Timeout |
| g12-15 | 44 | 211 | 29 | – | Timeout |
| g13-00 | 31 | 225 | 15 | 212 | 1.815 |
| g13-01 | 32 | 226 | 16 | 213 | 14.468 |
| g13-02 | 33 | 227 | 17 | 214 | 56.674 |
| g13-03 | 34 | 228 | 18 | 215 | 195.572 |
| g13-04 | 35 | 229 | 19 | 216 | 728.650 |
| g13-05 | 36 | 230 | 20 | 217 | 3214.443 |
| g13-06 | 37 | 231 | 21 | – | Timeout |
| g13-07 | 38 | 232 | 22 | – | Timeout |
| g13-08 | 39 | 233 | 23 | – | Timeout |
| g13-09 | 40 | 234 | 24 | – | Timeout |
| g13-10 | 41 | 235 | 25 | – | Timeout |
| g13-11 | 42 | 236 | 26 | – | Timeout |
| g13-12 | 43 | 237 | 27 | – | Timeout |
| g13-13 | 44 | 238 | 28 | – | Timeout |
| g13-14 | 45 | 239 | 29 | – | Timeout |
| g13-15 | 46 | 240 | 30 | – | Timeout |
| g14-00 | 33 | 256 | 16 | 242 | 6.512 |
| g14-01 | 34 | 257 | 17 | 243 | 53.631 |
| g14-02 | 35 | 258 | 18 | 244 | 189.942 |
| g14-03 | 36 | 259 | 19 | 245 | 696.400 |
| g14-04 | 37 | 260 | 20 | 246 | 2918.275 |
| g14-05 | 38 | 261 | 21 | – | Timeout |
| g14-06 | 39 | 262 | 22 | – | Timeout |
| g14-07 | 40 | 263 | 23 | – | Timeout |
| g14-08 | 41 | 264 | 24 | – | Timeout |
| g14-09 | 42 | 265 | 25 | – | Timeout |
| g14-10 | 43 | 266 | 26 | – | Timeout |
| g14-11 | 44 | 267 | 27 | – | Timeout |
| g14-12 | 45 | 268 | 28 | – | Timeout |
| g14-13 | 46 | 269 | 29 | – | Timeout |
| g14-14 | 47 | 270 | 30 | – | Timeout |
| g14-15 | 48 | 271 | 31 | – | Timeout |
| g15-00 | 35 | 289 | 17 | 274 | 22.458 |

**Table 44** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| g15-01 | 36 | 290 | 18 | 275 | 181.218 |
| g15-02 | 37 | 291 | 19 | 276 | 653.425 |
| g15-03 | 38 | 292 | 20 | 277 | 2738.780 |
| g15-04 | 39 | 293 | 21 | – | Timeout |
| g15-05 | 40 | 294 | 22 | – | Timeout |
| g15-06 | 41 | 295 | 23 | – | Timeout |
| g15-07 | 42 | 296 | 24 | – | Timeout |
| g15-08 | 43 | 297 | 25 | – | Timeout |
| g15-09 | 44 | 298 | 26 | – | Timeout |
| g15-10 | 45 | 299 | 27 | – | Timeout |
| g15-11 | 46 | 300 | 28 | – | Timeout |
| g15-12 | 47 | 301 | 29 | – | Timeout |
| g15-13 | 48 | 302 | 30 | – | Timeout |
| g15-14 | 49 | 303 | 31 | – | Timeout |
| g15-15 | 50 | 304 | 32 | – | Timeout |

Type: Artificial instances arising from generalizations of Steiner tree LP gap examples

**Table 45** Results on the testset skutella

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| s1 | 15 | 35 | 8 | 10 | 0.003 |
| s2 | 106 | 399 | 50 | – | Memout |

Type: artificial instances arising from generalizations of Steiner tree LP gap examples

**Table 46** Results on the testset smc

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| smc01 | 2 | 1 | 1 | 0 | 0.000 |
| smc02 | 3 | 3 | 2 | 2 | 0.000 |
| smc03 | 4 | 6 | 3 | 3 | 0.000 |
| smc04 | 5 | 10 | 4 | 4 | 0.000 |
| smc05 | 6 | 15 | 5 | 5 | 0.000 |
| smc06 | 7 | 21 | 6 | 6 | 0.000 |
| smc07 | 8 | 28 | 7 | 7 | 0.000 |
| smc08 | 9 | 36 | 8 | 8 | 0.001 |
| smc09 | 10 | 45 | 9 | 9 | 0.001 |
| smc10 | 11 | 55 | 10 | 10 | 0.002 |
| smc11 | 12 | 66 | 11 | 11 | 0.003 |
| smc12 | 13 | 78 | 12 | 12 | 0.012 |
| smc13 | 14 | 91 | 13 | 13 | 0.046 |
| smc14 | 15 | 105 | 14 | 14 | 0.134 |
| smc15 | 16 | 120 | 15 | 15 | 0.313 |

Type: artificial instances arising from generalizations of Steiner tree LP gap examples

## Appendix 2: Results on rectilinear instances

For these experiments, the setup is identical to the one described in Appendix 1. For each instance, first, the Hanan grid was computed and written to an instance file describing an instance of the Steiner tree problem in graphs. The latter instance was then solved. The reported run times only include the time to solve the graphic instance, in particular, the time needed to read in the Hanan grid is not included (Tables 47, 48, 49, 50).

**Table 47** Results on the testset CARIOCA

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_3_11_01 | 1331 | 3630 | 11 | 311,221,222 | 0.020 |
| carioca_3_11_02 | 1331 | 3630 | 11 | 466,149,453 | 0.019 |
| carioca_3_11_03 | 1331 | 3630 | 11 | 439,391,117 | 0.058 |
| carioca_3_11_04 | 1331 | 3630 | 11 | 413,409,501 | 0.054 |
| carioca_3_11_05 | 1331 | 3630 | 11 | 387,407,782 | 0.009 |
| carioca_3_12_01 | 1728 | 4752 | 12 | 494,141,224 | 0.032 |
| carioca_3_12_02 | 1728 | 4752 | 12 | 443,366,694 | 0.024 |
| carioca_3_12_03 | 1728 | 4752 | 12 | 429,706,282 | 0.083 |
| carioca_3_12_04 | 1728 | 4752 | 12 | 486,545,112 | 0.031 |
| carioca_3_12_05 | 1728 | 4752 | 12 | 444,438,614 | 0.016 |
| carioca_3_13_01 | 2197 | 6084 | 13 | 452,770,450 | 0.037 |
| carioca_3_13_02 | 2197 | 6084 | 13 | 462,700,327 | 0.238 |
| carioca_3_13_03 | 2197 | 6084 | 13 | 474,263,794 | 0.153 |
| carioca_3_13_04 | 2197 | 6084 | 13 | 442,802,506 | 0.019 |
| carioca_3_13_05 | 2197 | 6084 | 13 | 547,158,862 | 0.032 |
| carioca_3_14_01 | 2744 | 7644 | 14 | 438,382,690 | 0.150 |
| carioca_3_14_02 | 2744 | 7644 | 14 | 495,879,854 | 0.136 |
| carioca_3_14_03 | 2744 | 7644 | 14 | 480,652,934 | 0.048 |
| carioca_3_14_04 | 2744 | 7644 | 14 | 473,370,979 | 0.097 |
| carioca_3_14_05 | 2744 | 7644 | 14 | 408,691,456 | 0.048 |
| carioca_3_15_01 | 3375 | 9450 | 15 | 603,071,413 | 0.352 |
| carioca_3_15_02 | 3375 | 9450 | 15 | 528,575,469 | 0.279 |
| carioca_3_15_03 | 3375 | 9450 | 15 | 490,905,559 | 0.154 |
| carioca_3_15_04 | 3375 | 9450 | 15 | 540,300,331 | 0.201 |
| carioca_3_15_05 | 3375 | 9450 | 15 | 535,330,648 | 0.126 |
| carioca_3_16_01 | 4096 | 11,520 | 16 | 527,653,658 | 0.181 |
| carioca_3_16_02 | 4096 | 11,520 | 16 | 500,606,262 | 0.213 |
| carioca_3_16_03 | 4096 | 11,520 | 16 | 468,414,684 | 0.130 |
| carioca_3_16_04 | 4096 | 11,520 | 16 | 539,655,260 | 0.239 |
| carioca_3_16_05 | 4096 | 11,520 | 16 | 540,126,099 | 0.140 |
| carioca_3_17_01 | 4913 | 13,872 | 17 | 555,032,119 | 0.236 |

**Table 47** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_3_17_02 | 4913 | 13,872 | 17 | 527,737,314 | 0.936 |
| carioca_3_17_03 | 4913 | 13,872 | 17 | 562,083,809 | 0.405 |
| carioca_3_17_04 | 4913 | 13,872 | 17 | 589,257,703 | 0.230 |
| carioca_3_17_05 | 4913 | 13,872 | 17 | 612,582,645 | 3.852 |
| carioca_3_18_01 | 5832 | 16,524 | 18 | 556,545,528 | 0.291 |
| carioca_3_18_02 | 5832 | 16,524 | 18 | 549,027,056 | 1.392 |
| carioca_3_18_03 | 5832 | 16,524 | 18 | 649,390,363 | 0.710 |
| carioca_3_18_04 | 5832 | 16,524 | 18 | 554,907,444 | 0.800 |
| carioca_3_18_05 | 5832 | 16,524 | 18 | 509,234,906 | 0.408 |
| carioca_3_19_01 | 6859 | 19,494 | 19 | 561,743,715 | 6.788 |
| carioca_3_19_02 | 6859 | 19,494 | 19 | 608,506,796 | 1.957 |
| carioca_3_19_03 | 6859 | 19,494 | 19 | 574,021,347 | 0.398 |
| carioca_3_19_04 | 6859 | 19,494 | 19 | 630,525,105 | 0.342 |
| carioca_3_19_05 | 6859 | 19,494 | 19 | 650,204,402 | 6.354 |
| carioca_3_20_01 | 8000 | 22,800 | 20 | 638,376,617 | 1.613 |
| carioca_3_20_02 | 8000 | 22,800 | 20 | 477,950,448 | 0.148 |
| carioca_3_20_03 | 8000 | 22,800 | 20 | 746,979,341 | 8.890 |
| carioca_3_20_04 | 8000 | 22,800 | 20 | 653,809,733 | 10.096 |
| carioca_3_20_05 | 8000 | 22,800 | 20 | 678,171,940 | 1.649 |

Type: Random 3-d rectilinear instances with coordinates scaled by $10^8$

**Table 48** Results on the testset CARIOCA

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_4_11_01 | 14,641 | 53,240 | 11 | 627,022,001 | 0.425 |
| carioca_4_11_02 | 14,641 | 53,240 | 11 | 636,772,154 | 0.223 |
| carioca_4_11_03 | 14,641 | 53,240 | 11 | 607,879,790 | 1.779 |
| carioca_4_11_04 | 14,641 | 53,240 | 11 | 638,743,359 | 1.166 |
| carioca_4_11_05 | 14,641 | 53,240 | 11 | 545,419,447 | 1.883 |
| carioca_4_12_01 | 20,736 | 76,032 | 12 | 641,297,479 | 0.350 |
| carioca_4_12_02 | 20,736 | 76,032 | 12 | 619,890,840 | 1.143 |
| carioca_4_12_03 | 20,736 | 76,032 | 12 | 618,169,838 | 1.152 |
| carioca_4_12_04 | 20,736 | 76,032 | 12 | 573,580,734 | 0.791 |
| carioca_4_12_05 | 20,736 | 76,032 | 12 | 690,707,456 | 0.907 |
| carioca_4_13_01 | 28,561 | 105,456 | 13 | 668,457,902 | 1.776 |
| carioca_4_13_02 | 28,561 | 105,456 | 13 | 732,093,506 | 5.151 |
| carioca_4_13_03 | 28,561 | 105,456 | 13 | 667,816,953 | 1.369 |
| carioca_4_13_04 | 28,561 | 105,456 | 13 | 618,023,300 | 1.465 |

**Table 48** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_4_13_05 | 28,561 | 105,456 | 13 | 816,676,045 | 9.274 |
| carioca_4_14_01 | 38,416 | 142,688 | 14 | 678,015,050 | 1.967 |
| carioca_4_14_02 | 38,416 | 142,688 | 14 | 770,189,931 | 32.560 |
| carioca_4_14_03 | 38,416 | 142,688 | 14 | 774,041,179 | 7.656 |
| carioca_4_14_04 | 38,416 | 142,688 | 14 | 753,394,327 | 15.256 |
| carioca_4_14_05 | 38,416 | 142,688 | 14 | 668,149,161 | 6.908 |
| carioca_4_15_01 | 50,625 | 189,000 | 15 | 867,047,412 | 3.505 |
| carioca_4_15_02 | 50,625 | 189,000 | 15 | 747,749,078 | 2.624 |
| carioca_4_15_03 | 50,625 | 189,000 | 15 | 797,161,324 | 41.344 |
| carioca_4_15_04 | 50,625 | 189,000 | 15 | 682,701,539 | 9.475 |
| carioca_4_15_05 | 50,625 | 189,000 | 15 | 816,563,142 | 5.407 |
| carioca_4_16_01 | 65,536 | 245,760 | 16 | 877,600,183 | 53.440 |
| carioca_4_16_02 | 65,536 | 245,760 | 16 | 840,957,543 | 11.185 |
| carioca_4_16_03 | 65,536 | 245,760 | 16 | 769,487,137 | 7.469 |
| carioca_4_16_04 | 65,536 | 245,760 | 16 | 883,810,994 | 40.603 |
| carioca_4_16_05 | 65,536 | 245,760 | 16 | 844,364,805 | 26.792 |
| carioca_4_17_01 | 83,521 | 314,432 | 17 | 778,812,798 | 46.322 |
| carioca_4_17_02 | 83,521 | 314,432 | 17 | 885,687,619 | 96.926 |
| carioca_4_17_03 | 83,521 | 314,432 | 17 | 866,496,021 | 345.275 |
| carioca_4_17_04 | 83,521 | 314,432 | 17 | 961,783,573 | 22.303 |
| carioca_4_17_05 | 83,521 | 314,432 | 17 | 926,268,906 | 107.808 |
| carioca_4_18_01 | 104,976 | 396,576 | 18 | 951,149,709 | 99.353 |
| carioca_4_18_02 | 104,976 | 396,576 | 18 | 842,404,966 | 98.72303 |
| carioca_4_18_03 | 104,976 | 396,576 | 18 | 848,447,422 | 1431.374 |
| carioca_4_18_04 | 104,976 | 396,576 | 18 | 956,109,307 | 212.732 |
| carioca_4_18_05 | 104,976 | 396,576 | 18 | 893,733,310 | 143.342 |
| carioca_4_19_01 | 130,321 | 493,848 | 19 | 917,950,857 | 186.383 |
| carioca_4_19_02 | 130,321 | 493,848 | 19 | 825,014,078 | 49.409 |
| carioca_4_19_03 | 130,321 | 493,848 | 19 | 945,521,812 | 25.660 |
| carioca_4_19_04 | 130,321 | 493,848 | 19 | 912,019,383 | 160.591 |
| carioca_4_19_05 | 130,321 | 493,848 | 19 | 963,415,391 | 1625.465 |
| carioca_4_20_01 | 160,000 | 608,000 | 20 | 889,180,827 | 82.724 |
| carioca_4_20_02 | 160,000 | 608,000 | 20 | 822,698,792 | 101.107 |
| carioca_4_20_03 | 160,000 | 608,000 | 20 | 884,633,836 | 125.995 |
| carioca_4_20_04 | 160,000 | 608,000 | 20 | 948,878,450 | 2618.045 |
| carioca_4_20_05 | 160,000 | 608,000 | 20 | 984,006,649 | 82.545 |

Type: Random 4-d rectilinear instances with coordinates scaled by $10^8$

**Table 49** Results on the testset CARIOCA

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_5_11_01 | 161,051 | 732,050 | 11 | 925,163,690 | 34.547 |
| carioca_5_11_02 | 161,051 | 732,050 | 11 | 844,673,618 | 13.019 |
| carioca_5_11_03 | 161,051 | 732,050 | 11 | 867,510,918 | 6.297 |
| carioca_5_11_04 | 161,051 | 732,050 | 11 | 906,103,201 | 14.043 |
| carioca_5_11_05 | 161,051 | 732,050 | 11 | 795,198,510 | 30.521 |
| carioca_5_12_01 | 248,832 | 1,140,480 | 12 | 953,491,398 | 59.408 |
| carioca_5_12_02 | 248,832 | 1,140,480 | 12 | 985,601,088 | 102.106 |
| carioca_5_12_03 | 248,832 | 1,140,480 | 12 | 844,385,082 | 57.499 |
| carioca_5_12_04 | 248,832 | 1,140,480 | 12 | 879,014,839 | 93.151 |
| carioca_5_12_05 | 248,832 | 1,140,480 | 12 | 815,604,529 | 15.595 |
| carioca_5_13_01 | 371,293 | 1,713,660 | 13 | 881,473,517 | 132.662 |
| carioca_5_13_02 | 371,293 | 1,713,660 | 13 | 873,559,091 | 177.287 |
| carioca_5_13_03 | 371,293 | 1,713,660 | 13 | 1,005,775,838 | 184.838 |
| carioca_5_13_04 | 371,293 | 1,713,660 | 13 | 922,258,018 | 237.882 |
| carioca_5_13_05 | 371,293 | 1,713,660 | 13 | 879,174,698 | 46.431 |
| carioca_5_14_01 | 537,824 | 2,497,040 | 14 | 1,080,307,930 | 1038.209 |
| carioca_5_14_02 | 537,824 | 2,497,040 | 14 | 1,082,279,116 | 150.913 |
| carioca_5_14_03 | 537,824 | 2,497,040 | 14 | 931,463,937 | 284.221 |
| carioca_5_14_04 | 537,824 | 2,497,040 | 14 | 1,037,634,219 | 821.468 |
| carioca_5_14_05 | 537,824 | 2,497,040 | 14 | 1,072,793,454 | 1224.013 |
| carioca_5_15_01 | 759,375 | 3,543,750 | 15 | 1,011,895,745 | 1046.361 |
| carioca_5_15_02 | 759,375 | 3,543,750 | 15 | 1,067,623,193 | 888.808 |
| carioca_5_15_03 | 759,375 | 3,543,750 | 15 | 1,093,631,593 | 1258.172 |
| carioca_5_15_04 | 759,375 | 3,543,750 | 15 | 890,715,927 | 66.472 |
| carioca_5_15_05 | 759,375 | 3,543,750 | 15 | 1,112,392,828 | 638.515 |
| carioca_5_16_01 | 1,048,576 | 4,915,200 | 16 | 1,140,155,635 | 3259.377 |
| carioca_5_16_02 | 1,048,576 | 4,915,200 | 16 | 1,114,675,222 | 2047.700 |
| carioca_5_16_03 | 1,048,576 | 4,915,200 | 16 | 1,097,447,396 | 464.254 |
| carioca_5_16_04 | 1,048,576 | 4,915,200 | 16 | – | Timeout |
| carioca_5_16_05 | 1,048,576 | 4,915,200 | 16 | 1,034,250,551 | 815.794 |
| carioca_5_17_01 | 1,419,857 | 6,681,680 | 17 | 1,084,906,998 | 803.613 |
| carioca_5_17_02 | 1,419,857 | 6,681,680 | 17 | – | Memout |
| carioca_5_17_03 | 1,419,857 | 6,681,680 | 17 | 1,030,965,254 | 845.502 |
| carioca_5_17_04 | 1,419,857 | 6,681,680 | 17 | 1,154,984,533 | 3195.596 |
| carioca_5_17_05 | 1,419,857 | 6,681,680 | 17 | – | Timeout |
| carioca_5_18_01 | 1,889,568 | 8,922,960 | 18 | – | Memout |

**Table 49** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| carioca_5_18_02 | 1,889,568 | 8,922,960 | 18 | – | Memout |
| carioca_5_18_03 | 1,889,568 | 8,922,960 | 18 | 1,177,091,608 | 1081.015 |
| carioca_5_18_04 | 1,889,568 | 8,922,960 | 18 | – | Timeout |
| carioca_5_18_05 | 1,889,568 | 8,922,960 | 18 | – | Memout |

Type: Random 5-d rectilinear instances with coordinates scaled by $10^8$. Instances with 19 and 20 terminals omitted

**Table 50** Results on the testset bonn-3d

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| bonn_3_21_1 | 8820 | 25,179 | 21 | 6217 | 1.542 |
| bonn_3_21_2 | 8820 | 25,179 | 21 | 6729 | 1.944 |
| bonn_3_21_3 | 8820 | 25,179 | 21 | 5738 | 0.948 |
| bonn_3_22_1 | 10,648 | 30,492 | 22 | 6681 | 842.792 |
| bonn_3_22_2 | 10,648 | 30,492 | 22 | 6797 | 3.242 |
| bonn_3_22_3 | 10,164 | 29,084 | 22 | 6941 | 4.239 |
| bonn_3_23_1 | 12,167 | 34,914 | 23 | 6195 | 1.530 |
| bonn_3_23_2 | 12,167 | 34,914 | 23 | 6094 | 2.275 |
| bonn_3_23_3 | 11,638 | 33,373 | 23 | 6398 | 12.394 |
| bonn_3_24_1 | 13,824 | 39,744 | 24 | 6622 | 4.948 |
| bonn_3_24_2 | 13,248 | 38,064 | 24 | 7136 | 14.769 |
| bonn_3_24_3 | 13,248 | 38,064 | 24 | 7014 | 7.010 |
| bonn_3_25_1 | 15,625 | 45,000 | 25 | 6928 | 12.241 |
| bonn_3_25_2 | 15,625 | 45,000 | 25 | 7249 | 9.195 |
| bonn_3_25_3 | 15,625 | 45,000 | 25 | 7504 | 36.022 |
| bonn_3_26_1 | 17,576 | 50,700 | 26 | 8184 | 11.849 |
| bonn_3_26_2 | 17,576 | 50,700 | 26 | 7172 | 6.536 |
| bonn_3_26_3 | 16,900 | 48,724 | 26 | 7732 | 7.608 |
| bonn_3_27_1 | 18,252 | 52,676 | 27 | 7504 | 12.941 |
| bonn_3_27_2 | 19,683 | 56,862 | 27 | 7032 | 20.365 |
| bonn_3_27_3 | 18,954 | 54,729 | 27 | 7688 | 21.094 |
| bonn_3_28_1 | 21,168 | 61,208 | 28 | 8106 | 1795.931 |
| bonn_3_28_2 | 21,168 | 61,208 | 28 | 7774 | 384.240 |
| bonn_3_28_3 | 21,952 | 63,504 | 28 | 7307 | 22.964 |
| bonn_3_29_1 | 22,736 | 65,800 | 29 | 8231 | 248.490 |
| bonn_3_29_2 | 23,548 | 68,179 | 29 | 8542 | 37.717 |
| bonn_3_29_3 | 22,736 | 65,800 | 29 | 8053 | 23.270 |

**Table 50** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| bonn_3_30_1 | 26,100 | 75,660 | 30 | 8141 | 181.245 |
| bonn_3_30_2 | 26,100 | 75,660 | 30 | 7897 | 25.814 |
| bonn_3_30_3 | 26,100 | 75,660 | 30 | 8527 | 63.451 |
| bonn_3_31_1 | 27,869 | 80,848 | 31 | 8664 | 16.913 |
| bonn_3_31_2 | 28,830 | 83,669 | 31 | 8409 | 328.879 |
| bonn_3_31_3 | 28,830 | 83,669 | 31 | 7583 | 45.304 |
| bonn_3_32_1 | 31,744 | 92,224 | 32 | 9483 | 59.047 |
| bonn_3_32_2 | 31,744 | 92,224 | 32 | 8828 | 2371.743 |
| bonn_3_32_3 | 31,744 | 92,224 | 32 | 8261 | 8.539 |
| bonn_3_33_1 | 33,792 | 98,240 | 33 | 9588 | 46.329 |
| bonn_3_33_2 | 35,937 | 104,544 | 33 | 9658 | 465.676 |
| bonn_3_33_3 | 31,744 | 92,224 | 33 | 8902 | 83.404 |
| bonn_3_34_1 | 37,026 | 107,745 | 34 | 9045 | 410.333 |
| bonn_3_34_2 | 38,148 | 111,044 | 34 | 9664 | 1822.087 |
| bonn_3_34_3 | 37,026 | 107,745 | 34 | 9105 | 959.836 |
| bonn_3_35_1 | 41,650 | 121,345 | 35 | – | Timeout |
| bonn_3_35_2 | 40,460 | 117,844 | 35 | 9372 | 513.599 |
| bonn_3_35_3 | 40,460 | 117,844 | 35 | 9803 | 528.187 |
| bonn_3_36_1 | 42,768 | 124,632 | 36 | 9353 | 2906.588 |
| bonn_3_36_2 | 45,360 | 132,264 | 36 | 9118 | 87.813 |
| bonn_3_36_3 | 44,100 | 128,555 | 36 | – | Timeout |
| bonn_3_37_1 | 49,284 | 143,819 | 37 | 9379 | 811.912 |
| bonn_3_37_2 | 47,915 | 139,786 | 37 | 9596 | 1602.806 |
| bonn_3_37_3 | 49,284 | 143,819 | 37 | 8768 | 113.704 |
| bonn_3_38_1 | 53,428 | 156,028 | 38 | – | Timeout |
| bonn_3_38_2 | 52,022 | 151,885 | 38 | 9895 | 317.967 |
| bonn_3_38_3 | 52,022 | 151,885 | 38 | – | Timeout |
| bonn_3_39_1 | 54,834 | 160,171 | 39 | 9757 | 581.362 |
| bonn_3_39_2 | 54,834 | 160,171 | 39 | – | Timeout |
| bonn_3_39_3 | 57,798 | 168,909 | 39 | – | Timeout |
| bonn_3_40_1 | 64,000 | 187,200 | 40 | 9024 | 25.440 |
| bonn_3_40_2 | 57,798 | 168,909 | 40 | 9633 | 710.092 |
| bonn_3_40_3 | 59,280 | 173,278 | 40 | 10,696 | 1183.496 |
| bonn_3_45_1 | 89,100 | 261,315 | 45 | – | Timeout |

**Table 50** continued

| Instance | $\|V\|$ | $\|E\|$ | $\|R\|$ | Opt | Time (s) |
|---|---|---|---|---|---|
| bonn_3_45_2 | 89,100 | 261,315 | 45 | – | Timeout |
| bonn_3_45_3 | 85,140 | 249,613 | 45 | – | Timeout |
| bonn_3_50_1 | 110,544 | 324,721 | 50 | – | Memout |
| bonn_3_50_2 | 117,600 | 345,598 | 50 | – | Timeout |
| bonn_3_50_3 | 112,800 | 331,394 | 50 | – | Memout |
| bonn_3_55_1 | 160,380 | 472,284 | 55 | – | Timeout |
| bonn_3_55_2 | 151,470 | 445,881 | 55 | – | Timeout |
| bonn_3_55_3 | 154,548 | 455,004 | 55 | 12,138 | 6201.104 |
| bonn_3_60_1 | 198,417 | 585,044 | 60 | – | Timeout |
| bonn_3_60_2 | 201,898 | 595,369 | 60 | – | Timeout |
| bonn_3_60_3 | 198,476 | 585,220 | 60 | – | Memout |

Type: Random 3-d rectilinear instances. Coordinates were chosen uniformly at random from $\{0, 1, \ldots, 999\}$

## References


1. 11th DIMACS implementation challenge (2014). http://dimacs11.zib.de/downloads.html. Accessed 5 July 2016
2. de Aragão, M.P., Uchoa, E., Werneck, R.F.: Dual heuristics on the exact solution of large Steiner problems. Electron. Notes Discrete Math. **7**, 150–153 (2001)
3. Beasley, J.: An algorithm for the Steiner problem in graphs. Networks **14**, 147–159 (1984)
4. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) Automata, Languages, and Programming, Lecture Notes in Computer Science, vol. 7965, pp. 196–207. Springer, Berlin, Heidelberg (2013)
5. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: Steiner tree approximation via iterative randomized rounding. J. ACM **60**(1), 6:1–6:33 (2013)
6. Chlebík, M., Chlebíková, J.: The Steiner tree problem on graphs: Inapproximability results. Theor. Comput. Sci. **406**(3), 207–214 (2008)
7. Chopra, S., Gorres, E., Rao, M.: Solving the Steiner tree problem on a graph using branch and cut. ORSA J. Comput. **4**, 320–335 (1992)
8. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
9. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**(3), 195–207 (1971)
10. Duin, C., Volgenant, A.: An edge elimination test for the Steiner problem in graphs. Oper. Res. Lett. **8**, 79–83 (1989)
11. Erickson, R.E., Monma, C.L., Veinott Jr., A.F.: Send-and-split method for minimum-concave-cost network flows. Math. Oper. Res. **12**(4), 634–664 (1987)
12. Fafianie, S., Bodlaender, H., Nederlof, J.: Speeding up dynamic programming with representative sets. In: Gutin, G., Szeider, S. (eds.) Parameterized and Exact Computation, Lecture Notes in Computer Science, vol. 8246, pp. 321–334. Springer International Publishing, New York (2013)
13. Fonseca, R., Brazil, M., Winter, P., Zachariasen, M.: Faster exact algorithms for computing Steiner trees in higher dimensional euclidean spaces. In: 11th DIMACS Implementation Challenge on Steiner Tree Problems (2014). http://dimacs11.zib.de/workshop/FonsecaBrazilWinterZachariasen.pdf. Accessed 5 July 2016
14. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987)
15. Fuchs, B., Kern, W., Mölle, D., Richter, S., Rossmanith, P., Wang, X.: Dynamic programming for minimum Steiner trees. Theory Comput. Syst. **41**(3), 493–500 (2007)
16. Hanan, M.: On Steiner's problem with rectilinear distance. SIAM J. Appl. Math. **14**(2), 255–265 (1966)
17. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **SSC–4**(2), 100–107 (1968)

18. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. J. Soc. Ind. Appl. Math. **10**(1), 196–210 (1962). doi:10.2307/2098806
19. Held, M., Karp, R.M.: The traveling salesman problem and minimum spanning trees. Oper. Res. **18**, 1138–1162 (1970)
20. Held, S., Korte, B., Rautenbach, D., Vygen, J.: Combinatorial optimization in VLSI design. In: Combinatorial Optimization—methods and applications, pp. 33–96. IOS Press, Amsterdam (2011)
21. Hwang, F.K.: On Steiner minimal trees with rectilinear distance. SIAM J. Appl. Math. **30**(1), 104–114 (1976)
22. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
23. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks **32**, 207–232 (1998)
24. Polzin, T.: Algorithms for the Steiner problem in networks. Ph.D. thesis (2004)
25. Polzin, T., Vahdati, S.: Extending reduction techniques for the Steiner tree problem: a combination of alternative- and bound-based approaches. Tech. Rep. MPI-I-2001-1-007, Max-Planck-Institut für Informatik (2001)
26. Polzin, T., Vahdati Daneshmand, S.: Practical partitioning-based methods for the Steiner problem. In: Àlvarez, C., Serna, M. (eds.) Experimental Algorithms, Lecture Notes in Computer Science, vol. 4007, pp. 241–252. Springer, Berlin, Heidelberg (2006)
27. Polzin, T., Vahdati Daneshmand, S.: The Steiner Tree Challenge: An updated Study (2014). http://dimacs11.zib.de/papers/PolzinVahdatiDIMACS.pdf. Accessed 5 July 2016
28. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Technol. J. **36**, 1389–1401 (1957)
29. Silvanus, J.: Fast exact Steiner tree generation using dynamic programming. Master's thesis, Research Institute for Discrete Mathematics, University of Bonn (2013)
30. Snyder, T.L.: On the exact location of Steiner points in general dimension. SIAM J. Comput. **21**(1), 163–180 (1992)
31. Uchoa, E., de Aragão, M., Ribeiro, C.: Preprocessing Steiner problems from VLSI layout. Tech. Rep. MCC 32/99, Catholic University of Rio de Janeiro, Rio de Janeiro (1999)
32. Vahdati Daneshmand, S.: Algorithmic approaches to the Steiner problem in networks. Ph.D. thesis (2004)
33. Vygen, J.: Faster algorithm for optimum Steiner trees. Inf. Process. Lett. **111**(21–22), 1075–1079 (2011)
34. Warme, D.M., Winter, P., Zachariasen, M.: Exact algorithms for plane Steiner tree problems: A computational study. Springer, New York (2000)
35. Wong, R.: A dual ascent approach for Steiner tree problems on a directed graph. Math. Program. **28**(3), 271–287 (1984)
36. Wulff-Nilsen, C.: Higher dimensional rectilinear Steiner minimal trees. Master's thesis, Department of Computer Science, University of Copenhagen (2006)