Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

THORSTEN KOCH, ALEXANDER MARTIN, STEFAN VOSS

SteinLib: An Updated Library on Steiner Tree Problems in Graphs

ZIB-Report 00-37 (November 2000)

SteinLib: An Updated Library on Steiner Tree Problems in Graphs[1]

Thorsten Koch
Konrad-Zuse-Zentrum für Informationstechnik Berlin
D-14195 Berlin, Germany
E-mail: `koch@zib.de`

Alexander Martin
Department of Mathematics
Darmstadt University of Technology, D-64289 Darmstadt, Germany
E-mail: `martin@mathematik.tu-darmstadt.de`

Stefan Voß
Department of Business Administration, Information Systems and Information Management
Braunschweig University of Technology, D-38106 Braunschweig, Germany
E-mail: `stefan.voss@tu-bs.de`

Contents

[1]to appear in: *D.-Z. Du and X. Cheng (Eds.)*, Steiner Trees in Industries

Abstract

In this paper we present the *SteinLib*, a library of data sets for the Steiner tree problem in graphs. This library extends former libraries on Steiner tree problems by many new interesting and difficult instances, most of them arising from real-world applications. We give a survey on the difficulty of these problem instances by stating references to state-of-the-art software packages that were the first or are currently among the best to solve these instances.

1 Introduction

The availability of computational test sets for combinatorial and integer programming problems is very important for the development and comparison of efficient implementations as well as robust and fast computer codes. For instance, the TSPLIB[2] [52] has very much influenced the development of very good software for the solution of the traveling salesman problem. The MIPLIB[3] [6], which is a library of real-world mixed integer programming problems, is *the* test set for MIP solvers and most comparisons of algorithmic developments in the literature are evaluated on the MIPLIB-instances. Important in this context is also the OR-Library[4] of Beasley [4], a collection of test data sets for a great variety of Operations Research (OR) problems including the Steiner tree problem in graphs.

In this paper we follow this line and present an updated and revised version of a library for the Steiner tree problem in graphs. Given an undirected graph $G = (V, E)$ and a node set $T \subseteq V$, a *Steiner tree for T in G* is a subset $S \subseteq E$ of the edges such that $(V(S), S)$ contains a path from s to t for all $s, t \in T$, where $V(S)$ denotes the set of nodes incident to an edge in S. In other words, a Steiner tree is an edge set S that spans the set of *terminals* or *basic nodes* T. The *Steiner tree problem in graphs* is to find a minimal Steiner tree with respect to some given edge costs $c_e, e \in E$. For a survey on the Steiner tree problem in graphs see, e.g., [28, 59, 12].

Among others, the mentioned OR-Library of Beasley contains test data for Steiner tree problems. In fact, the OR-Library-Instances B, C, D, and E are *the* source in the Steiner tree community for evaluating and testing new algorithmic developments for the Steiner tree problem, such as heuristics, cutting plane methods, preprocessing techniques and others. In the meantime, however, most of these instances are easy to solve for state-of-the-art software.

In this paper we give a summary of publically available test data for Steiner tree problems. We use or refer to state-of-the-art software packages and show which of these instances are easy to solve and which are still challenging. We will see that the difficulty of a problem instance does not only depend on the size of the problem (i. e., number of nodes, edges and terminals). There are relatively small sized instances that are still hard for current solvers.

[2] `http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95`
[3] `http://www.caam.rice.edu/~bixby/miplib/miplib.html`
[4] `http://mscmga.ms.ic.ac.uk/info.html`

All data sets mentioned in the paper are available through the newly updated library on Steiner tree problems *SteinLib*.[5] We encourage people to add hard problem instances to this library and to contribute in this way to obtain a valuable collection of test data that forms the basis for further improvements in the algorithmic developments for the solution of the Steiner tree problem.

In the following section we give a description of the data format that is used in *SteinLib*. Subsequently, we summarize the currently available problem instances within the Steiner tree library *SteinLib*. We distinguish those problem instances that are provided as graphs and those which have their background as instances for metrical Steiner tree problems but may be represented as graphs according to an appropriate conversion routine.

Besides randomly generated instances, data sets for the Steiner tree problem in graphs arise from various application areas such as biology and phylogeny [21, 48], biochemistry [55], mining [39], forestry [53], the design of sewer networks [65], telecommunication networks [73] or databases [64] as well as VLSI design [40, 38, 35]. Many of the problem instances given in those references describing these applications, however, turn out to be too small to be challenging for current state-of-the-art Steiner tree algorithms. Thus, we mainly restrict ourselves to instances that are still interesting from a computational point of view.

In the literature a great variety of papers provide theoretical results for polynomially solvable special cases of the Steiner tree problem (e.g., for series-parallel or Halin graphs [69]). However, as these instances turn out to be easy no specific data generation routines are provided.

We use the following notation throughout the paper. For a Steiner tree instance with a graph $G = (V, E)$ and a terminal set $T \subseteq V$, we denote by $n = |V|$ the number of nodes, by $m = |E|$ the number of edges, and by $t = |T|$ the number of terminals.

2 Description of the Data Format

In this section we describe the data format that is used in the *SteinLib*. For illustration we use an odd wheel depicted in Figure 1 with terminal nodes $1, 2, 3$ and 4. The following lines show this example in the *SteinLib* format.

```
33D32945 STP File, STP Format Version 1.0

SECTION Comment
Name    "Odd Wheel"
Creator "T. Koch, A. Martin and S. Voss"
Remark  "Example used to describe the STP data format"
END

SECTION Graph
Nodes 7
```

[5] http://elib.zib.de/steinlib

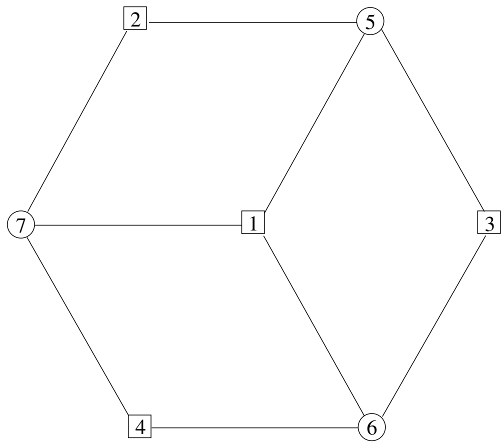

Figure 1: Data format example

```
Edges 9
E 1 5 1
E 1 6 1
E 1 7 1
E 2 5 1
E 2 7 1
E 3 5 1
E 3 6 1
E 4 6 1
E 4 7 1
END

SECTION Terminals
Terminals 4
T 1
T 2
T 3
T 4
END

SECTION Coordinates
DD 1  80 50
DD 2  55  5
DD 3 130 50
DD 4  55 95
DD 5 105  5
DD 6 105 95
DD 7  30 50
END
```

`Comment`	Gives general information about the problem instance, like name and creator.
`Graph`	Here the graph itself is specified.
`Terminals`	Lists the terminals for the problem instance.
`Coordinates`	This is an optional section giving coordinates for the nodes of the graph. This section is only necessary for drawing.
`Presolve`	This is an optional section stating that this problem is the result of some presolve processing.

Table 1: Section types

The format is line (or row) oriented. Each line is terminated with a line-feed. Everything on a line after (and including) a `#` is ignored. Blank lines are ignored. The first line of each data file is supposed to be

```
33D32945 STP File, STP Format Version 1.0
```

It contains a so-called magic number as an identification key. It provides an assertion that the data file is indeed a file in the *SteinLib* format.

Then, the file is divided into sections. A section starts with the keyword `SECTION` followed by the name of the section and ends with a line with the keyword `END`. The sections shown in Table 1 are possible and should appear in the given order.

Each line within a section starts with a keyword, indicating the type of the line. Depending on the section different keywords are allowed. Each keyword follows a number of fields in the line. Fields can be either a string, i. e., an arbitrary string enclosed in double quotes, or a number, where integer numbers are allowed only.

The following Figure 2 lists the keywords for each section. The Column *Fields* shows how many strings (S) or numbers (N) are required.

The sections `Graph` and `Terminals` need to be given. Within the section `Graph` we consider either directed or undirected graphs. For directed graphs the keyword `Arcs` must appear and each line of an arc must start with an `A`. In addition, the section `Terminals` must specify a root node. For undirected graphs the keyword `Edges` gives the number of edges and `E` lines in the file. The format does not allow mixed graphs as any undirected edge may easily be represented by two anti-parallel arcs with the same end-nodes.

The sections `Comment`, `Presolve`, and `Coordinates` are optional. If these sections appear to be in the data file, each of their keywords is optional itself. Within the section `Coordinates` all entries must be of the same type. Furthermore, whenever coordinates are given, they must be given for all nodes.

There are further options of the *SteinLib* format like an *include* mechanism for a compact representation of instances that share certain sections. Special keywords for generalized Steiner trees are also available. These options are described in detail on the web-page[6] of the library.

[6] `http://elib.zib.de/steinlib`

Name	*Fields*	*Description*
		Section `Comment`
`Name`	1 S	The name of the instance
`Date`	1 S	The date of the creation of the instance
`Creator`	1 S	Who did it
`Remark`	1 S	Some other information
		Section `Graph`
`Nodes`	1 N	Number of nodes in the graph
`Edges`	1 N	Number of edges in the graph
`E`	3 N	Specification of one edge. The number of `E` lines must match the number given in the `Edges` line. The three values are Node1, Node2 and Weight. The nodes are numbered from one to the number given in the `Nodes` line.
`Arcs`	1 N	Number of arcs in the graph
`A`	3 N	Specification of one arc. The number of `A` lines must match the number given in the `Arcs` line. The three values are Tail-Node, Head-Node and Weight. The nodes are numbered from one to the number given in the `Nodes` line.
		Section `Terminals`
`Terminals`	1 N	Number of terminals. Must be between one and the number of Nodes given in the Graph section.
`Root`	1 N	Node number of the Root-Node for directed Steiner tree problems.
`T`	1 N	Specification of one terminal. The number of `T` lines must match the number given in the `Terminals` line. The field specifies the node that is a terminal. The nodes are numbered from one to the number given in the `Nodes` line.
		Section `Coordinates`
`DD`	3 N	2D-Coordinates. The three values are Node-Number, X- and Y-Coordinate.
`DDD`	4 N	The four values correspond to Node-Number, X-, Y- and Z-Coordinate.
		Section `Presolve`
`Date`	1 S	Date of the preprocessing.
`Fixed`	1 N	The value that must be added to the optimal value of this preprocessed instance to get the optimal value of the original instance.
`lower`	1 N	A lower bound.
`upper`	1 N	An upper bound.
`time`	1 N	Used processing time in seconds.

Figure 2: Description of the keywords

3 The Steiner Tree Problem in Graphs

In this section we summarize existing data sets for the Steiner tree problem in graphs as they have been proposed in the literature. We provide tables where the most important characteristics are given, i.e., the name of the instance, the number of nodes, the number of edges (or arcs) and the number of basic nodes. Column "Opt" shows the objective function value of the best known solution. If this is proven to be optimal, it is written in bold-face.

In Column "D" we provide a *difficulty classification*. In capital letters it is indicated if the instance may be solved to optimality by applying local reduction techniques L. One of the best known mathematical programming formulations for the Steiner tree problem in graphs is a multicommodity flow formulation due to Wong [72]. Theoretically, the linear programming relaxations of this formulation and some other formulations are equivalent [24]. We classify an instance with the letter P when the solution to the linear programming relaxation is non-fractional. All other cases are classified as NP indicating that no polynomial time algorithm is known to date for this instance. Finally, a small s, m, h, d or w indicates that an instance may be solved to optimality within seconds, minutes, hours, days or weeks with state-of-the-art algorithms (based on up-to-date computers). A ? on either of the two values states that the classification is not known. Our difficulty classification relies on numerical experiments we have performed based on [35] and [56] for the Steiner tree problem in graphs. For the metrical Steiner tree problems discussed in Section 4 we additionally refer to [66, 67].

The most influencing data generation scheme in the literature was proposed by Aneja [1] in 1980. Each problem instance corresponds to a randomly generated connected graph with a specified set of n nodes and m edges. To generate instances that guarantee the existence of a feasible solution, connectivity needs to be assured. Therefore, first a random spanning tree of the entire set V of nodes is generated. Additional edges are then added randomly over the graph. Random weights between 1 and 10 are then assigned to all edges. In [1] the number of basic nodes in the set T was chosen to be $\frac{n}{2}$ but may be modified arbitrarily, and problem instances generated along these lines had between 10 and 50 nodes.

This data generation scheme became quite popular in the community and was applied by various authors. Beasley [2] generated a set of 18 instances according to this scheme with 50, 75, and 100 nodes. For each number of nodes the number of basic nodes was chosen to be $\frac{1}{6}n$, $\frac{1}{4}n$, and $\frac{1}{2}n$ and for each graph the number of edges was specified to achieve an average node degree of 2.5 and 4. This set of 18 instances was chosen as a benchmark by several authors later on (e.g., [51, 3]) and became the data set B within the OR-Library [4], see Table 2. Independently, Wong [72] generated similar instances with up to 60 nodes, 120 edges and 30 basic nodes, however, with edge weights being real values from the interval $(0, 1)$. Later on these instances were also used as a benchmark (see, e.g., [3, 10]).

Beasley [3] extended the size of his instances when he generated new data sets (see Tables 3 - 5). As simple reduction techniques were able to solve the

B-data to optimality (see, e.g., [15, 16, 17, 13, 59]) the new problem instances had up to 2500 nodes and 62500 edges.

Another collection of randomly generated examples is described in [10] (see Tables 6 and 7). p401 through p410 are complete graphs with random edge weights from the interval $[1, 1500]$. p455 through p466 are complete graphs with Euclidean weights. For these graphs coordinates between 1 and 900 were assigned to the nodes and edge weights were computed according to Euclidean distances rounded to the nearest integer. Furthermore, instances p601 through p616 are grid graphs with random weights and p619 through p633 with Euclidean weights.

Tables 8 through 11 show the so-called *incidence* instances. These problem instances, described in [13] and [18, 19], are randomly generated and have the following sizes. There are four choices of the node set cardinality $n = 80, 160, 320$, and 640, for each of them twenty variants are generated combining four sizes of the terminal set $|T| = \log n$, $\sqrt{n}$, $2\sqrt{n}$, and $\frac{n}{4}$ with five different densities $m = \frac{3n}{2}$, $n \ln n$, $\frac{n(n-1)}{2}$, $2n$, and $\frac{n(n-1)}{10}$, all values are rounded down to the next integer. Every variant was drawn five times. The problem names have the pattern $n.tei$, where $n = 80$, 160, 320, and 640 gives the number of nodes of the instance, $t = 0, 1, 2, 3$ indicates which of the four alternatives (in the above sequence) of the sizes for the terminal sets have been chosen, $e = 0, 1, 2, 3, 4$ stands for the five densities, and $i = 1, 2, 3, 4, 5$ distinguishes the five instances drawn for each variant. To give an example, problem *160.411* is the first out of five instances with 160 nodes, $\frac{n(n-1)}{10} = \frac{160 \cdot 159}{10} = 2544$ edges, and $\lfloor \sqrt{n} \rfloor = \lfloor \sqrt{160} \rfloor = 12$ terminals.

In the incidence instances the weight on each edge (i, j) is defined with a sample r from a normal distribution, rounded to the closest integer value with a minimum value of 1 and a maximum value of 500, i.e., $c_{ij} = \min\{500, \max\{1, \mathrm{round}(r)\}\}$. To obtain a graph that is much harder to reduce by preprocessing techniques such as given in [15, 16, 17, 13, 59], three distributions with a different mean value are used. Any edge (i, j) is incident to none, to one or to two basic nodes. The mean of r is 100 for edges (i, j) with $i, j \notin T$ (no incidence with T), 200 on edges (i, j) with one basic node and 300 on edges (i, j) with both end-nodes $i, j \notin T$. The standard deviation for each of the three normal distributions is 5.

One of the challenging problems in the design of electronic circuits is the routing problem which is, roughly speaking, the task to connect terminal sets via wires on a predefined area. Depending on the underlying technology and the design rules subproblems arise that can be formulated as the problem of packing Steiner trees in certain graphs (see [40] for an excellent treatment of this subject). In Tables 12 through 18 we give corresponding real-world VLSI instances. They result from seven different circuits described in [32]. The underlying graphs are grid graphs that contain holes. The holes result from so-called cells that block certain areas of the grid. The sets of terminals are located on the border of these holes. For each of the seven circuits and for each terminal set T_i (where index i runs from 1 to the number of terminal sets of the circuit) we constructed an instance of the Steiner tree problem. For the

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
b01	50	63	9	Ls	**82**
b02	50	63	13	Ls	**83**
b03	50	63	25	Ls	**138**
b04	50	100	9	Ls	**59**
b05	50	100	13	Ls	**61**
b06	50	100	25	Ps	**122**
b07	75	94	13	Ls	**111**
b08	75	94	19	Ls	**104**
b09	75	94	38	Ls	**220**
b10	75	150	13	Ps	**86**
b11	75	150	19	Ls	**88**
b12	75	150	38	Ls	**174**
b13	100	125	17	Ps	**165**
b14	100	125	25	Ps	**235**
b15	100	125	50	Ps	**318**
b16	100	200	17	Ps	**127**
b17	100	200	25	Ps	**131**
b18	100	200	50	Ps	**218**

Table 2: B-Instances from [2]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
c01	500	625	5	Ps	**85**
c02	500	625	10	Ps	**144**
c03	500	625	83	Ps	**754**
c04	500	625	125	Ps	**1079**
c05	500	625	250	Ls	**1579**
c06	500	1000	5	Ps	**55**
c07	500	1000	10	Ps	**102**
c08	500	1000	83	Ps	**509**
c09	500	1000	125	Ps	**707**
c10	500	1000	250	Ps	**1093**
c11	500	2500	5	Ps	**32**
c12	500	2500	10	Ps	**46**
c13	500	2500	83	Ps	**258**
c14	500	2500	125	Ps	**323**
c15	500	2500	250	Ls	**556**
c16	500	12500	5	Ps	**11**
c17	500	12500	10	Ps	**18**
c18	500	12500	83	Ps	**113**
c19	500	12500	125	Ps	**146**
c20	500	12500	250	Ls	**267**

Table 3: C-Instances from [3]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
d01	1000	1250	5	Ps	**106**
d02	1000	1250	10	Ps	**220**
d03	1000	1250	167	Ps	**1565**
d04	1000	1250	250	Ps	**1935**
d05	1000	1250	500	Ps	**3250**
d06	1000	2000	5	Ps	**67**
d07	1000	2000	10	Ps	**103**
d08	1000	2000	167	Ps	**1072**
d09	1000	2000	250	Ps	**1448**
d10	1000	2000	500	Ps	**2110**
d11	1000	5000	5	Pm	**29**
d12	1000	5000	10	Pm	**42**
d13	1000	5000	167	Ps	**500**
d14	1000	5000	250	Ps	**667**
d15	1000	5000	500	Ps	**1116**
d16	1000	25000	5	Pm	**13**
d17	1000	25000	10	Pm	**23**
d18	1000	25000	167	Ps	**223**
d19	1000	25000	250	Ps	**310**
d20	1000	25000	500	Ls	**537**

Table 4: D-Instances from [3]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
e01	2500	3125	5	Ps	**111**
e02	2500	3125	10	Ps	**214**
e03	2500	3125	417	Ps	**4013**
e04	2500	3125	625	Ps	**5101**
e05	2500	3125	1250	Ps	**8128**
e06	2500	5000	5	Ps	**73**
e07	2500	5000	10	Pm	**145**
e08	2500	5000	417	Pm	**2640**
e09	2500	5000	625	Pm	**3604**
e10	2500	5000	1250	Pm	**5600**
e11	2500	12500	5	Pm	**34**
e12	2500	12500	10	Pm	**67**
e13	2500	12500	417	Pm	**1280**
e14	2500	12500	625	Pm	**1732**
e15	2500	12500	1250	Ps	**2784**
e16	2500	62500	5	Ph	**15**
e17	2500	62500	10	Ph	**25**
e18	2500	62500	417	NPh	**564**
e19	2500	62500	625	Pm	**758**
e20	2500	62500	1250	Ls	**1342**

Table 5: E-Instances from [3]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
P4E	100	4950	5	Ps	**1138**
P4E	100	4950	5	Ps	**1228**
P4E	100	4950	10	Ps	**1609**
P4E	100	4950	10	Ps	**1868**
P4E	100	4950	20	Ps	**2345**
P4E	100	4950	20	Ps	**2959**
P4E	100	4950	50	Ps	**4474**
P4E	200	19900	10	Ps	**1510**
P4E	200	19900	20	Ps	**2545**
P4E	200	19900	40	Ps	**3853**
P4E	200	19900	100	Ps	**6234**

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
P4Z	100	4950	5	Ps	**155**
P4Z	100	4950	5	Ps	**116**
P4Z	100	4950	5	Ps	**179**
P4Z	100	4950	10	Ls	**270**
P4Z	100	4950	10	Ls	**270**
P4Z	100	4950	10	Ps	**290**
P4Z	100	4950	20	Ps	**590**
P4Z	100	4950	20	Ls	**542**
P4Z	100	4950	50	Ps	**963**
P4Z	100	4950	50	Ls	**1010**

Table 6: Instances from [10]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
P6E	100	180	5	Ls	**7485**
P6E	100	180	5	Ps	**8746**
P6E	100	180	5	Ls	**8688**
P6E	100	180	10	Ps	**15972**
P6E	100	180	10	Ps	**19496**
P6E	100	180	20	Ls	**20246**
P6E	100	180	20	Ps	**23078**
P6E	100	180	20	Ls	**22346**
P6E	100	180	50	Ps	**40647**
P6E	100	180	50	Ls	**40008**
P6E	100	180	50	Ls	**43287**
P6E	200	370	10	Ps	**26125**
P6E	200	370	20	Ps	**39067**
P6E	200	370	40	Ps	**56217**
P6E	200	370	100	Ls	**86268**

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
P6Z	100	180	5	Ps	**8083**
P6Z	100	180	5	Ps	**5022**
P6Z	100	180	10	Ps	**11397**
P6Z	100	180	10	Ps	**10355**
P6Z	100	180	10	Ps	**13048**
P6Z	100	180	20	Ls	**15358**
P6Z	100	180	20	Ls	**14439**
P6Z	100	180	20	Ps	**18263**
P6Z	100	180	50	Ps	**30161**
P6Z	100	180	50	Ls	**26903**
P6Z	100	180	50	Ps	**30258**
P6Z	200	370	10	Ps	**18429**
P6Z	200	370	20	Ps	**27276**
P6Z	200	370	40	Ps	**42474**
P6Z	200	370	100	Ps	**62263**

Table 7: Instances from [10]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
i080-001	80	120	6	Ps	**1787**
i080-002	80	120	6	Ps	**1607**
i080-003	80	120	6	Ps	**1713**
i080-004	80	120	6	Ps	**1866**
i080-005	80	120	6	Ps	**1790**
i080-011	80	350	6	Ps	**1479**
i080-012	80	350	6	Ps	**1484**
i080-013	80	350	6	Ps	**1381**
i080-014	80	350	6	Ps	**1397**
i080-015	80	350	6	Ps	**1495**
i080-021	80	3160	6	Ps	**1175**
i080-022	80	3160	6	Ps	**1178**
i080-023	80	3160	6	Ps	**1174**
i080-024	80	3160	6	Ps	**1161**
i080-025	80	3160	6	Ps	**1162**
i080-031	80	160	6	Ps	**1570**
i080-032	80	160	6	Ps	**2088**
i080-033	80	160	6	Ps	**1794**
i080-034	80	160	6	Ps	**1688**
i080-035	80	160	6	Ps	**1862**
i080-041	80	632	6	Ps	**1276**
i080-042	80	632	6	Ps	**1287**
i080-043	80	632	6	Ps	**1295**
i080-044	80	632	6	NPs	**1366**
i080-045	80	632	6	Ps	**1310**
i080-101	80	120	8	Ps	**2608**
i080-102	80	120	8	Ps	**2403**
i080-103	80	120	8	Ps	**2603**
i080-104	80	120	8	Ps	**2486**
i080-105	80	120	8	Ps	**2203**
i080-111	80	350	8	NPs	**2051**
i080-112	80	350	8	Ps	**1885**
i080-113	80	350	8	Ps	**1884**
i080-114	80	350	8	Ps	**1895**
i080-115	80	350	8	Ps	**1868**
i080-121	80	3160	8	Ps	**1561**
i080-122	80	3160	8	Ps	**1561**
i080-123	80	3160	8	Ps	**1569**
i080-124	80	3160	8	Ls	**1555**
i080-125	80	3160	8	Ps	**1572**
i080-131	80	160	8	Ps	**2284**
i080-132	80	160	8	Ps	**2180**
i080-133	80	160	8	Ps	**2261**
i080-134	80	160	8	Ps	**2070**
i080-135	80	160	8	Ps	**2102**
i080-141	80	632	8	Ps	**1788**
i080-142	80	632	8	Ps	**1708**
i080-143	80	632	8	NPs	**1767**
i080-144	80	632	8	Ps	**1772**
i080-145	80	632	8	Ps	**1762**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
i080-201	80	120	16	Ps	**4760**
i080-202	80	120	16	Ps	**4650**
i080-203	80	120	16	Ps	**4599**
i080-204	80	120	16	Ps	**4492**
i080-205	80	120	16	Ps	**4564**
i080-211	80	350	16	Ps	**3631**
i080-212	80	350	16	NPs	**3677**
i080-213	80	350	16	NPs	**3678**
i080-214	80	350	16	NPs	**3734**
i080-215	80	350	16	NPs	**3681**
i080-221	80	3160	16	Ps	**3158**
i080-222	80	3160	16	Ps	**3141**
i080-223	80	3160	16	Ps	**3156**
i080-224	80	3160	16	Ps	**3159**
i080-225	80	3160	16	Ps	**3150**
i080-231	80	160	16	Ps	**4354**
i080-232	80	160	16	Ps	**4199**
i080-233	80	160	16	Ps	**4118**
i080-234	80	160	16	Ps	**4274**
i080-235	80	160	16	NPs	**4487**
i080-241	80	632	16	NPm	**3538**
i080-242	80	632	16	Pm	**3458**
i080-243	80	632	16	NPm	**3474**
i080-244	80	632	16	NPs	**3466**
i080-245	80	632	16	NPs	**3467**
i080-301	80	120	20	Ps	**5519**
i080-302	80	120	20	Ps	**5944**
i080-303	80	120	20	Ps	**5777**
i080-304	80	120	20	Ps	**5586**
i080-305	80	120	20	NPs	**5932**
i080-311	80	350	20	Ps	**4554**
i080-312	80	350	20	NPs	**4534**
i080-313	80	350	20	Ps	**4509**
i080-314	80	350	20	NPs	**4515**
i080-315	80	350	20	NPs	**4459**
i080-321	80	3160	20	Ps	**3932**
i080-322	80	3160	20	Ps	**3937**
i080-323	80	3160	20	Ps	**3946**
i080-324	80	3160	20	Ps	**3932**
i080-325	80	3160	20	Ps	**3924**
i080-331	80	160	20	NPs	**5226**
i080-332	80	160	20	NPs	**5362**
i080-333	80	160	20	Ps	**5381**
i080-334	80	160	20	Ps	**5264**
i080-335	80	160	20	Ps	**4953**
i080-341	80	632	20	Ps	**4236**
i080-342	80	632	20	NPm	**4337**
i080-343	80	632	20	NPs	**4246**
i080-344	80	632	20	NPs	**4310**
i080-345	80	632	20	NPm	**4341**

Table 8: Incidence instances from [13, 18]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
i160-001	160	240	7	Ps	**2490**
i160-002	160	240	7	Ps	**2158**
i160-003	160	240	7	Ps	**2297**
i160-004	160	240	7	Ps	**2370**
i160-005	160	240	7	Ps	**2495**
i160-011	160	812	7	Ps	**1677**
i160-012	160	812	7	Ps	**1750**
i160-013	160	812	7	Ps	**1661**
i160-014	160	812	7	Ps	**1778**
i160-015	160	812	7	Ps	**1768**
i160-021	160	12720	7	Ps	**1352**
i160-022	160	12720	7	Ps	**1365**
i160-023	160	12720	7	Ps	**1351**
i160-024	160	12720	7	Ps	**1371**
i160-025	160	12720	7	Ps	**1366**
i160-031	160	320	7	Ps	**2170**
i160-032	160	320	7	Ps	**2330**
i160-033	160	320	7	NPs	**2101**
i160-034	160	320	7	Ps	**2083**
i160-035	160	320	7	Ps	**2103**
i160-041	160	2544	7	Ps	**1494**
i160-042	160	2544	7	Ps	**1486**
i160-043	160	2544	7	NPs	**1549**
i160-044	160	2544	7	Ps	**1478**
i160-045	160	2544	7	NPs	**1554**
i160-101	160	240	12	Ps	**3859**
i160-102	160	240	12	Ps	**3747**
i160-103	160	240	12	Ps	**3837**
i160-104	160	240	12	Ps	**4063**
i160-105	160	240	12	Ps	**3563**
i160-111	160	812	12	Ps	**2869**
i160-112	160	812	12	NPs	**2924**
i160-113	160	812	12	Ps	**2866**
i160-114	160	812	12	Ps	**2989**
i160-115	160	812	12	NPm	**2937**
i160-121	160	12720	12	Pm	**2363**
i160-122	160	12720	12	Ps	**2348**
i160-123	160	12720	12	Ps	**2355**
i160-124	160	12720	12	Ps	**2352**
i160-125	160	12720	12	Ps	**2351**
i160-131	160	320	12	Ps	**3356**
i160-132	160	320	12	Ps	**3450**
i160-133	160	320	12	Ps	**3585**
i160-134	160	320	12	Ps	**3470**
i160-135	160	320	12	Ps	**3716**
i160-141	160	2544	12	Ps	**2549**
i160-142	160	2544	12	NPm	**2562**
i160-143	160	2544	12	Ps	**2557**
i160-144	160	2544	12	NPm	**2607**
i160-145	160	2544	12	Ps	**2578**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
i160-201	160	240	24	NPs	**6923**
i160-202	160	240	24	Ps	**6930**
i160-203	160	240	24	Ps	**7243**
i160-204	160	240	24	Ps	**7068**
i160-205	160	240	24	Ps	**7122**
i160-211	160	812	24	NPm	**5583**
i160-212	160	812	24	NPm	**5643**
i160-213	160	812	24	NPm	**5647**
i160-214	160	812	24	NPm	**5720**
i160-215	160	812	24	NPm	**5518**
i160-221	160	12720	24	Pm	**4729**
i160-222	160	12720	24	Pm	**4697**
i160-223	160	12720	24	Pm	**4730**
i160-224	160	12720	24	Pm	**4721**
i160-225	160	12720	24	Pm	**4728**
i160-231	160	320	24	Ps	**6662**
i160-232	160	320	24	NPs	**6558**
i160-233	160	320	24	Ps	**6339**
i160-234	160	320	24	Ps	**6594**
i160-235	160	320	24	Ps	**6764**
i160-241	160	2544	24	NPh	**5086**
i160-242	160	2544	24	NPh	**5106**
i160-243	160	2544	24	NPm	**5050**
i160-244	160	2544	24	NPm	**5076**
i160-245	160	2544	24	NPm	**5084**
i160-301	160	240	40	Ps	**11816**
i160-302	160	240	40	Ps	**11497**
i160-303	160	240	40	Ps	**11445**
i160-304	160	240	40	Ps	**11448**
i160-305	160	240	40	NPs	**11423**
i160-311	160	812	40	NPm	**9135**
i160-312	160	812	40	NPm	**9052**
i160-313	160	812	40	NPh	**9159**
i160-314	160	812	40	NPm	**8941**
i160-315	160	812	40	NPm	**9086**
i160-321	160	12720	40	Pm	**7876**
i160-322	160	12720	40	NPm	**7859**
i160-323	160	12720	40	Pm	**7876**
i160-324	160	12720	40	NPm	**7884**
i160-325	160	12720	40	NPm	**7862**
i160-331	160	320	40	Ps	**10414**
i160-332	160	320	40	NPs	**10806**
i160-333	160	320	40	Ps	**10561**
i160-334	160	320	40	Ps	**10327**
i160-335	160	320	40	Ps	**10589**
i160-341	160	2544	40	NPh	**8331**
i160-342	160	2544	40	NPd	**8348**
i160-343	160	2544	40	NPh	**8275**
i160-344	160	2544	40	NPh	**8307**
i160-345	160	2544	40	NPh	**8327**

Table 9: Incidence instances from [13, 18]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
i320-001	320	480	8	Ps	**2672**
i320-002	320	480	8	Ps	**2847**
i320-003	320	480	8	Ps	**2972**
i320-004	320	480	8	Ps	**2905**
i320-005	320	480	8	Ps	**2991**
i320-011	320	1845	8	NPs	**2053**
i320-012	320	1845	8	Ps	**1997**
i320-013	320	1845	8	Ps	**2072**
i320-014	320	1845	8	NPs	**2061**
i320-015	320	1845	8	NPs	**2059**
i320-021	320	51040	8	Lm	**1553**
i320-022	320	51040	8	Pm	**1565**
i320-023	320	51040	8	Pm	**1549**
i320-024	320	51040	8	Pm	**1553**
i320-025	320	51040	8	Pm	**1550**
i320-031	320	640	8	NPs	**2673**
i320-032	320	640	8	NPs	**2770**
i320-033	320	640	8	Ps	**2769**
i320-034	320	640	8	Ps	**2521**
i320-035	320	640	8	Ps	**2385**
i320-041	320	10208	8	Ps	**1707**
i320-042	320	10208	8	Ps	**1682**
i320-043	320	10208	8	NPm	**1723**
i320-044	320	10208	8	Ps	**1681**
i320-045	320	10208	8	Ps	**1686**
i320-101	320	480	17	Ps	**5548**
i320-102	320	480	17	Ps	**5556**
i320-103	320	480	17	Ps	**6239**
i320-104	320	480	17	Ps	**5703**
i320-105	320	480	17	Ps	**5928**
i320-111	320	1845	17	NPm	**4273**
i320-112	320	1845	17	NPm	**4213**
i320-113	320	1845	17	NPm	**4205**
i320-114	320	1845	17	NPm	**4104**
i320-115	320	1845	17	NPs	**4238**
i320-121	320	51040	17	Pm	**3321**
i320-122	320	51040	17	Pm	**3314**
i320-123	320	51040	17	Pm	**3332**
i320-124	320	51040	17	Pm	**3323**
i320-125	320	51040	17	Pm	**3340**
i320-131	320	640	17	Ps	**5255**
i320-132	320	640	17	Ps	**5052**
i320-133	320	640	17	Ps	**5125**
i320-134	320	640	17	Ps	**5272**
i320-135	320	640	17	NPs	**5342**
i320-141	320	10208	17	NPh	**3606**
i320-142	320	10208	17	Pm	**3567**
i320-143	320	10208	17	Pm	**3561**
i320-144	320	10208	17	Ps	**3512**
i320-145	320	10208	17	NPm	**3601**

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
i320-201	320	480	34	Ps	**10044**
i320-202	320	480	34	Ps	**11223**
i320-203	320	480	34	Ps	**10148**
i320-204	320	480	34	Ps	**10275**
i320-205	320	480	34	NPs	**10573**
i320-211	320	1845	34	NPm	**8039**
i320-212	320	1845	34	NPm	**8044**
i320-213	320	1845	34	NPm	**7984**
i320-214	320	1845	34	NPm	**8046**
i320-215	320	1845	34	NPh	**8015**
i320-221	320	51040	34	NPh	**6679**
i320-222	320	51040	34	NPh	**6686**
i320-223	320	51040	34	NPm	**6695**
i320-224	320	51040	34	NPm	**6694**
i320-225	320	51040	34	NPm	**6691**
i320-231	320	640	34	NPs	**9862**
i320-232	320	640	34	NPs	**9933**
i320-233	320	640	34	Ps	**9787**
i320-234	320	640	34	Ps	**9517**
i320-235	320	640	34	Ps	**9945**
i320-241	320	10208	34	NPh	**7027**
i320-242	320	10208	34	NPh	**7072**
i320-243	320	10208	34	NPh	**7044**
i320-244	320	10208	34	NPh	**7078**
i320-245	320	10208	34	NPh	**7046**
i320-301	320	480	80	Ps	**23279**
i320-302	320	480	80	Ps	**23387**
i320-303	320	480	80	Ps	**22693**
i320-304	320	480	80	Ps	**23451**
i320-305	320	480	80	NPs	**22547**
i320-311	320	1845	80	NPd	**17945**
i320-312	320	1845	80	NPd	**18122**
i320-313	320	1845	80	NPd	**17991**
i320-314	320	1845	80	NPd	**18088**
i320-315	320	1845	80	NPd	**17987**
i320-321	320	51040	80	NPh	**15648**
i320-322	320	51040	80	NPh	**15646**
i320-323	320	51040	80	NPh	**15654**
i320-324	320	51040	80	NPh	**15667**
i320-325	320	51040	80	NPh	**15649**
i320-331	320	640	80	NPs	**21517**
i320-332	320	640	80	NPs	**21674**
i320-333	320	640	80	NPs	**21339**
i320-334	320	640	80	Ps	**21415**
i320-335	320	640	80	NPs	**21378**
i320-341	320	10208	80	NPh	**16296**
i320-342	320	10208	80	NPh	**16228**
i320-343	320	10208	80	NPh	**16281**
i320-344	320	10208	80	NPh	**16295**
i320-345	320	10208	80	NPh	**16289**

Table 10: Incidence instances from [13, 18]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
i640-001	640	960	9	Ps	**4033**
i640-002	640	960	9	Ps	**3588**
i640-003	640	960	9	Ps	**3438**
i640-004	640	960	9	Ps	**4000**
i640-005	640	960	9	Ps	**4006**
i640-011	640	4135	9	Ps	**2392**
i640-012	640	4135	9	Ps	**2465**
i640-013	640	4135	9	Ps	**2399**
i640-014	640	4135	9	Ps	**2171**
i640-015	640	4135	9	NPs	**2347**
i640-021	640	204480	9	Ph	**1749**
i640-022	640	204480	9	??	*1671*
i640-023	640	204480	9	Pm	**1754**
i640-024	640	204480	9	??	*1768*
i640-025	640	204480	9	Pm	**1745**
i640-031	640	1280	9	Ps	**3278**
i640-032	640	1280	9	Ps	**3187**
i640-033	640	1280	9	Ps	**3260**
i640-034	640	1280	9	Ps	**2953**
i640-035	640	1280	9	Ps	**3292**
i640-041	640	40896	9	Pm	**1897**
i640-042	640	40896	9	NPm	**1934**
i640-043	640	40896	9	NPm	**1931**
i640-044	640	40896	9	NPm	**1938**
i640-045	640	40896	9	Pm	**1866**
i640-101	640	960	25	Ps	**8764**
i640-102	640	960	25	Ps	**9109**
i640-103	640	960	25	Ps	**8819**
i640-104	640	960	25	Ps	**9040**
i640-105	640	960	25	NPs	**9623**
i640-111	640	4135	25	NPm	**6167**
i640-112	640	4135	25	NPm	**6304**
i640-113	640	4135	25	NP?	*6298*
i640-114	640	4135	25	NPm	**6308**
i640-115	640	4135	25	NPh	**6217**
i640-121	640	204480	25	?m	**4906**
i640-122	640	204480	25	??	*4976*
i640-123	640	204480	25	??	*4942*
i640-124	640	204480	25	??	*4955*
i640-125	640	204480	25	??	*4958*
i640-131	640	1280	25	Ps	**8097**
i640-132	640	1280	25	NPs	**8154**
i640-133	640	1280	25	Ps	**8021**
i640-134	640	1280	25	Ps	**7754**
i640-135	640	1280	25	NPs	**7696**
i640-141	640	40896	25	NP?	*5203*
i640-142	640	40896	25	NP?	*5193*
i640-143	640	40896	25	NP?	*5194*
i640-144	640	40896	25	NP?	*5218*
i640-145	640	40896	25	NP?	*5218*

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
i640-201	640	960	50	NPs	**16079**
i640-202	640	960	50	Ps	**16324**
i640-203	640	960	50	Ps	**16124**
i640-204	640	960	50	Ps	**16239**
i640-205	640	960	50	NPs	**16616**
i640-211	640	4135	50	NP?	*12025*
i640-212	640	4135	50	NP?	*11847*
i640-213	640	4135	50	NP?	*11910*
i640-214	640	4135	50	NP?	*11898*
i640-215	640	4135	50	NP?	*12141*
i640-221	640	204480	50	??	*9917*
i640-222	640	204480	50	??	*9957*
i640-223	640	204480	50	??	*9927*
i640-224	640	204480	50	??	*9938*
i640-225	640	204480	50	??	*9933*
i640-231	640	1280	50	NPm	**15014**
i640-232	640	1280	50	NPs	**14630**
i640-233	640	1280	50	NPm	**14797**
i640-234	640	1280	50	Ps	**15203**
i640-235	640	1280	50	NPm	**14803**
i640-241	640	40896	50	NP?	*10230*
i640-242	640	40896	50	NP?	*10197*
i640-243	640	40896	50	NP?	*10228*
i640-244	640	40896	50	NP?	*10263*
i640-245	640	40896	50	NP?	*10234*
i640-301	640	960	160	Ps	**45005**
i640-302	640	960	160	Ps	**45736**
i640-303	640	960	160	Ps	**44922**
i640-304	640	960	160	Ps	**46233**
i640-305	640	960	160	Ps	**45902**
i640-311	640	4135	160	NP?	*35999*
i640-312	640	4135	160	NP?	*36057*
i640-313	640	4135	160	NP?	*35654*
i640-314	640	4135	160	NP?	*35699*
i640-315	640	4135	160	NP?	*36003*
i640-321	640	204480	160	??	*31394*
i640-322	640	204480	160	??	*31353*
i640-323	640	204480	160	??	*31349*
i640-324	640	204480	160	??	*31613*
i640-325	640	204480	160	??	*31380*
i640-331	640	1280	160	NPm	**42796**
i640-332	640	1280	160	NPm	**42548**
i640-333	640	1280	160	NPm	**42345**
i640-334	640	1280	160	NP?	*36960*
i640-335	640	1280	160	NPm	**43035**
i640-341	640	40896	160	NP?	*32128*
i640-342	640	40896	160	NP?	*32065*
i640-343	640	40896	160	NP?	*32068*
i640-344	640	40896	160	NP?	*32097*
i640-345	640	40896	160	NP?	*32074*

Table 11: Incidence instances from [13, 18]

graph G we have chosen the underlying grid graph restricted to the minimal enclosing rectangle of the terminal set. The distance of two neighbored grid points in horizontal and vertical direction differ for these circuits. This results in different edge costs for horizontal and vertical edges in G.

In the library *SteinLib* we put instances with terminal sets whose cardinality is at least 10. The examples are distinguished by the name of the circuit followed by the index of the terminal set. For example *msm1234* means that the instance is defined by terminal set 1234 of circuit *msm*. As test problem instances we have chosen for each circuit all instances whose two leading non-zeros of the index of the terminal set differ from the two leading non-zeros of all other indices. If there is more than one index with the same two leading non-zeros we have chosen the instance with the smallest index (for instance among examples *msm3727, msm3731, msm3761, msm3786* we have chosen *msm3727*). In addition, we added an instance with the smallest and largest number of terminals for each circuit. Finally, we extended the original test set as introduced in [35] by ten additional instances from the large circuits *alue* and *alut* that contain a large number of terminals. All together we obtain 126 different VLSI test instances.

In Table 19 some instances are shown that we recently obtained from Lin [41]. These instances have their origin in VLSI design and are derived from the placement of rectangular blocks in a 2D plane.

The instances starting with 'mc' in Table 20 are generated by Margot [45]. They were randomly generated, either on a grid graph or a complete graph, and were selected as they turned out to be difficult for the branch-and-cut algorithm written by Margot at that time. Table 21 shows some instances on a complete graph with Euclidean weights. Instance *brasil58* was introduced in [22], whereas *berlin52* and *world666* are taken from the TSP library, where some nodes are randomly defined as terminals. Modifying data from the TSP library has been done by Verhoeven [57, 58], too. He uses k-th order Delaunay graphs to derive Steiner tree problem instances from TSP instances. Two data sets are developed called F data (with $n = 783, 1000, m \leq 184597$, and $t = \frac{3n}{20}, \frac{5n}{20}, \frac{7n}{20}$) and G data (with $n \leq 18512, m \leq 325093$, and $t = \frac{3n}{20}, \frac{5n}{20}, \frac{7n}{20}$) with 24 and 18 instances, respectively. More instances from the TSP library appear in Table 27 in connection with metrical Steiner tree problems.

A data generation scheme used quite frequently for producing generalized Steiner problems, see Section 5, goes back to [68]. For the construction of a graph with n nodes he proceeds as follows. The n nodes are randomly distributed over a rectangular grid with integer coordinates (the size of the grid may be specified by the user). Then the distance $d(i,j)$ between any two nodes i and j may be either the Euclidean metric or chosen from the interval $(0, maxL]$ from a uniform random distribution (of course other ideas might be investigated as well) where $maxL \leq \sqrt{2}n$ is the maximum possible Euclidean distance between any two nodes in the grid.

In both models for defining the distance an edge probability

$$P[(i,j)] = \beta \cdot exp \frac{-d(i,j)}{maxL \cdot \alpha}$$

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
alue2087	1244	1971	34	Ps	**1049**
alue2105	1220	1858	34	Ps	**1032**
alue3146	3626	5869	64	NPs	**2240**
alue5067	3524	5560	68	Nm	**2586**
alue5345	5179	8165	68	NPm	**3507**
alue5623	4472	6938	68	NPm	**3413**
alue5901	11543	18429	68	Pm	**3912**
alue6179	3372	5213	67	NPm	**2452**
alue6457	3932	6137	68	Pm	**3057**
alue6735	4119	6696	68	Pm	**2696**
alue6951	2818	4419	67	Pm	**2386**
alue7065	34046	54841	544	Pd	**23881**
alue7066	6405	10454	16	Ph	**2256**
alue7080	34479	55494	2344	NPd	**62449**
alue7229	940	1474	34	Ps	**824**

Table 12: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
alut0787	1160	2089	34	Ps	**982**
alut0805	966	1666	34	Ps	**958**
alut1181	3041	5693	64	Pm	**2353**
alut2010	6104	11011	68	Pm	**3307**
alut2288	9070	16595	68	NPm	**3843**
alut2566	5021	9055	68	NPm	**3073**
alut2610	33901	62816	204	Pd	**12239**
alut2625	36711	68117	879	NPw	**35459**
alut2764	387	626	34	Ls	**640**

Table 13: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
diw0234	5349	10086	25	Pm	**1996**
diw0250	353	608	11	Ls	**350**
diw0260	539	985	12	Ls	**468**
diw0313	468	822	14	Ls	**397**
diw0393	212	381	11	Ls	**302**
diw0445	1804	3311	33	Ps	**1363**
diw0459	3636	6789	25	Ls	**1362**
diw0460	339	579	13	Ls	**345**
diw0473	2213	4135	25	Ps	**1098**
diw0487	2414	4386	25	Ps	**1424**
diw0495	938	1655	10	Ls	**616**
diw0513	918	1684	10	Ls	**604**
diw0523	1080	2015	10	Ls	**561**
diw0540	286	465	10	Ls	**374**
diw0559	3738	7013	18	Ps	**1570**
diw0778	7231	13727	24	Ps	**2173**
diw0779	11821	22516	50	NPh	**4440**
diw0795	3221	5938	10	Pm	**1550**
diw0801	3023	5575	10	Pm	**1587**
diw0819	10553	20066	32	Ph	**3399**
diw0820	11749	22384	37	Ph	**4167**

Table 14: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
dmxa0296	233	386	12	Ls	**344**
dmxa0368	2050	3676	18	Ps	**1017**
dmxa0454	1848	3286	16	Ls	**914**
dmxa0628	169	280	10	Ls	**275**
dmxa0734	663	1154	11	Ls	**506**
dmxa0848	499	861	16	Ps	**594**
dmxa0903	632	1087	10	Ps	**580**
dmxa1010	3983	7108	23	Ls	**1488**
dmxa1109	343	559	17	Ps	**454**
dmxa1200	770	1383	21	Ps	**750**
dmxa1304	298	503	10	Ls	**311**
dmxa1516	720	1269	11	Ls	**508**
dmxa1721	1005	1731	18	Ps	**780**
dmxa1801	2333	4137	17	Pm	**1365**

Table 15: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
msm0580	338	541	11	Ps	**467**
msm0654	1290	2270	10	Ls	**823**
msm0709	1442	2403	16	Ls	**884**
msm0920	752	1264	26	Ps	**806**
msm1008	402	695	11	Ps	**494**
msm1234	933	1632	13	Ps	**550**
msm1477	1199	2078	31	Ps	**1068**
msm1707	278	478	11	Ls	**564**
msm1844	90	135	10	Ps	**188**
msm1931	875	1522	10	Ps	**604**
msm2000	898	1562	10	Ls	**594**
msm2152	2132	3702	37	Ps	**1590**
msm2326	418	723	14	Ps	**399**
msm2492	4045	7094	12	Ps	**1459**
msm2525	3031	5239	12	Ps	**1290**
msm2601	2961	5100	16	Ps	**1440**
msm2705	1359	2458	13	Ps	**714**
msm2802	1709	2963	18	Ps	**926**
msm2846	3263	5783	89	NPm	**3135**
msm3277	1704	2991	12	Ls	**869**
msm3676	957	1554	10	Ls	**607**
msm3727	4640	8255	21	Ls	**1376**
msm3829	4221	7255	12	Pm	**1571**
msm4038	237	390	11	Ls	**353**
msm4114	402	690	16	Ls	**393**
msm4190	391	666	16	Ls	**381**
msm4224	191	302	11	Ls	**311**
msm4312	5181	8893	10	Pm	**2016**
msm4414	317	476	11	Ls	**408**
msm4515	777	1358	13	Ps	**630**

Table 16: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
gap1307	342	552	17	Ls	**549**
gap1413	541	906	10	Ls	**457**
gap1500	220	374	17	Ls	**254**
gap1810	429	702	17	Ls	**482**
gap1904	735	1256	21	Ps	**763**
gap2007	2039	3548	17	NPs	**1104**
gap2119	1724	2975	29	Ls	**1244**
gap2740	1196	2084	14	Ps	**745**
gap2800	386	653	12	Ls	**386**
gap2975	179	293	10	Ls	**245**
gap3036	346	583	13	Ps	**457**
gap3100	921	1558	11	Ps	**640**
gap3128	10393	18043	104	Pm	**4292**

Table 17: VLSI instances from [35]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
taq0014	6466	11046	128	Ph	**5326**
taq0023	572	963	11	Ps	**621**
taq0365	4186	7074	22	Pm	**1914**
taq0377	6836	11715	136	NPh	**6393**
taq0431	1128	1905	13	Ps	**897**
taq0631	609	932	10	Ps	**581**
taq0739	837	1438	16	NPs	**848**
taq0741	712	1217	16	Ps	**847**
taq0751	1051	1791	16	Ps	**939**
taq0891	331	560	10	Ls	**319**
taq0903	6163	10490	130	NPh	**5099**
taq0910	310	514	17	Ls	**370**
taq0920	122	194	17	Ls	**210**
taq0978	777	1239	10	Ls	**566**

Table 18: VLSI instances from [35]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
lin01	53	80	4	Ls	**503**
lin02	55	82	6	Ls	**557**
lin03	57	84	8	Ls	**926**
lin04	157	266	6	Ps	**1239**
lin05	160	269	9	Ls	**1703**
lin06	165	274	14	Ls	**1348**
lin07	307	526	6	Ps	**1885**
lin08	311	530	10	Ps	**2248**
lin09	313	532	12	Ps	**2752**
lin10	321	540	20	Ps	**4132**
lin11	816	1460	10	Ps	**4280**
lin12	818	1462	12	Ps	**5250**
lin13	822	1466	16	Ps	**4609**
lin14	828	1472	22	Ps	**5824**
lin15	840	1484	34	Ps	**7145**
lin16	1981	3633	12	Pm	**6618**
lin17	1989	3641	20	Pm	**8405**
lin18	1994	3646	25	NPm	**9714**
lin19	2010	3662	41	Pm	**13268**
lin20	3675	6709	11	Pm	**6673**
lin21	3683	6717	20	Pm	**9143**
lin22	3692	6726	28	Pm	**10519**
lin23	3716	6750	52	Ph	**17560**
lin24	7998	14734	16	Pd	**15076**
lin25	8007	14743	24	Pd	**17803**
lin26	8013	14749	30	Pm	**21757**
lin27	8017	14753	36	NPd	**20678**
lin28	8062	14798	81	Nd	**32584**
lin29	19083	35636	24	Nd	**23765**
lin30	19091	35644	31	Ph	**27684**
lin31	19100	35653	40	??	*33435*
lin32	19112	35665	53	??	*41456*
lin33	19177	35730	117	??	*58313*
lin34	38282	71521	34	??	*47234*
lin35	38294	71533	45	??	*52793*
lin36	38307	71546	58	??	*58366*
lin37	38418	71657	172	??	*102874*

Table 19: VLSI instances from [41]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
mc11	400	760	213	Ps	**11689**
mc13	150	11175	80	NPm	**92**
mc2	120	7140	60	NPs	**71**
mc3	97	4656	45	NPs	**47**
mc7	400	760	170	Ps	**3417**
mc8	400	760	188	Ps	**1566**

Table 20: Instances of F. Margot

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
berlin52	52	1326	16	Ps	**1044**
brasil58	58	1653	25	Ps	**13655**
world666	666	221445	174	Ps	**122467**

Table 21: Complete Instances

is given indicating the probability that edge (i, j) is included in the graph. The probability uses parameters α and β from $(0, 1]$ to control the density of the graph. That is, β is used for the overall density while α is used to control the proportion of edges with a smaller distance and those with a longer distance. Finally, edge weights for those edges generated are chosen according to one of the above distance models. With this procedure Steiner tree problem instances are generated.

Other sources for Steiner tree problem instances in graphs are, e.g., [11, 46, 51, 60, 71]. These instances are, like the TSP instances of Verhoeven [57, 58], no longer available or are just too simple.

The test data of [11] were randomly generated graphs with 100 nodes, real valued edge weights, a density (i.e., ratio of actual edges to the maximum number of possible edges) of $\frac{1}{2}$, $\frac{3}{4}$, and 1 as well as $t \in \{30, 50, 80\}$. Unfortunately, these instances are no longer available.

The problem instances of [46] were randomly generated with up to 110 nodes. Edges were generated with Euclidean or rectilinear distances so that they fulfill a so-called non-adjacency condition, i.e., no two Steiner nodes are connected with each other.

In [51] random graphs with pre-specified numbers of nodes n have been generated as follows. Two probabilities are fixed, p_e for the probability that an edge exists between any two nodes, and p_t for the probability that a node is specified as a terminal. Under the assumption that only connected graphs will be considered, real edge weights are assigned either uniformly distributed in the range $(0, 1]$ or by a normal distribution with mean 0.5 and standard deviation 0.125. This data generation routine was used and extended by [60, 71]. Furthermore, in both papers randomly generated instances with Euclidean and rectilinear distances were considered.

4 Metrical Steiner Tree Problems Representable in Graphs

Many applications of the Steiner tree problem require the connection of a given set of basic nodes within a metric space. Examples are Euclidean distances, rectilinear distances (building design) or the Hamming metric (phylogeny).

According to Hanan [27] rectilinear Steiner tree problems may be represented as (grid) graphs according to a simple conversion routine. That is, optimal solutions for both ways of representing the data are identical. Based on the x- and y-coordinates of pre-specified or given basic nodes a grid graph is constructed as follows. The graph is induced by the set of basic nodes by running a vertical line and a horizontal line through each basic node and retaining the finite segments between interconnections of these lines with rectilinear distances as weights.

There is a nice way to represent metrical Steiner tree problems in graphs via the concept of full Steiner trees (or sets). A *full Steiner tree (FST)* for some terminal set T is a Steiner tree for T such that the leaves of the tree are exactly the terminals. One can show that for any Steiner tree instance there is an optimal Steiner tree that can be decomposed into FSTs. Though the computation of all FSTs is difficult in general, it turns out that it often can be done very efficiently for metrical Steiner tree problems. In addition, the number of FSTs is frequently almost linear for many practical instances. Having this set of FSTs at hand we can easily set up a Steiner tree problem in a graph. We introduce nodes for the (original) terminals and for the Steiner nodes appearing in the FSTs, and we introduce edges representing the edges in the FSTs. In fact, the use of FSTs is *the* key to solve large rectilinear and Euclidean Steiner tree problems [66, 67], because in this way the size of the instances can be reduced drastically, see also Tables 23 through 27.

The series of problem instances denoted by R is taken from [54], see Table 22. These are randomly generated instances on grid graphs. Over the years various authors have contributed new best solutions for various of these instances [5, 33, 61]. In the meantime all optimal solutions are known and most instances have to be considered as easy.

5 Generalized Steiner Tree Problems and Modifications

In this section we consider some generalized Steiner tree problems and data generation schemes in that sense that they might lead to interesting instances for the original problem as well. As the number of generalized Steiner tree problems is too large to be considered comprehensively (see, e.g., [28, 59] for some references), we restrict ourselves to only very few in the following subsections.

Modifications of the Steiner tree problem in graphs that are still hard to solve from a theoretical point of view may become easy from an algorithmic point of view. For instance, the cardinality Steiner tree problem arises when all edge weights are equal to 1. We have tested the algorithm of [35] on various problem instances and they always turn out to be easy.

5.1 Directed Steiner Tree Problems

Steiner tree problems may also be considered in directed graphs. We are given a directed graph $D = (V, A)$, a terminal set $T \subseteq V$ with one specific root node $r \in T$ and arc costs $c_a, a \in A$, and we look for an arborescence rooted at r that spans all terminals in T. It is well known that the undirected Steiner tree problem can be modeled as a directed Steiner problem by bidirecting the edges. However, there are directed instances that do not come from undirected graphs.

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
r01	10	12	5	Ls	**187**
r02	8	9	6	Ls	**164**
r03	16	21	7	Ls	**236**
r04	32	48	8	Ps	**254**
r05	8	9	6	Ls	**226**
r06	20	30	12	Ls	**242**
r07	21	31	12	Ls	**248**
r08	20	29	12	Ps	**236**
r09	11	14	7	Ls	**164**
r10	17	22	6	Ls	**177**
r11	11	11	6	Ls	**144**
r12	21	30	9	Ls	**180**
r13	27	41	9	Ps	**150**
r14	36	60	12	Ls	**260**
r15	50	80	14	Ps	**148**
r16	5	4	3	Ls	**160**
r17	28	42	10	Ps	**200**
r18	180	333	62	Ps	**404**
r19	61	96	14	Ps	**188**
r20	4	3	3	Ls	**112**
r21	9	10	5	Ls	**192**
r22	8	8	4	Ls	**63**
r23	8	8	4	Ls	**65**
r24	8	8	4	Ls	**30**
r25	5	4	3	Ls	**23**
r26	5	4	3	Ls	**15**
r27	8	8	4	Ls	**133**
r28	7	7	4	Ls	**24**
r29	5	4	3	Ls	**200**
r30	20	29	12	Ps	**110**
r31	79	135	14	Ps	**259**
r32	148	267	19	Ps	**313**
r33	132	241	18	Ps	**268**
r34	194	355	19	Ps	**241**
r35	142	253	18	Ps	**151**
r36	4	3	4	Ls	**90**
r37	19	24	8	Ls	**90**
r38	64	108	14	Ps	**166**
r39	64	108	14	Ps	**166**
r40	42	68	10	Ps	**155**
r41	118	211	20	Ps	**224**
r42	40	62	15	Ps	**153**
r43	129	230	16	Ps	**255**
r44	140	252	17	Ps	**252**
r45	200	367	19	Ps	**220**
r46	16	24	16	Ls	**150**

Table 22: Instances from [54]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es10fst01	18	20	10	?s	**22920745**
es10fst02	14	13	10	?s	**19134104**
es10fst03	17	20	10	?s	**26003678**
es10fst04	18	20	10	?s	**20461116**
es10fst05	12	11	10	?s	**18818916**
es10fst06	17	20	10	?s	**26540768**
es10fst07	14	13	10	?s	**26025072**
es10fst08	21	28	10	?s	**25056214**
es10fst09	21	29	10	?s	**22062355**
es10fst10	18	21	10	?s	**23936095**
es10fst11	14	13	10	?s	**22239535**
es10fst12	13	12	10	?s	**19626318**
es10fst13	18	21	10	?s	**19483914**
es10fst14	24	32	10	?s	**21856128**
es10fst15	16	18	10	?s	**18641924**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es20fst01	29	28	20	?s	**33703886**
es20fst02	29	28	20	?s	**32639486**
es20fst03	27	26	20	?s	**27847417**
es20fst04	57	83	20	?s	**27624394**
es20fst05	54	77	20	?s	**34033163**
es20fst06	29	28	20	?s	**36014241**
es20fst07	45	59	20	?s	**34934874**
es20fst08	52	74	20	?s	**38016346**
es20fst09	36	42	20	?s	**36739939**
es20fst10	49	67	20	?s	**34024740**
es20fst11	33	36	20	?s	**27123908**
es20fst12	33	36	20	?s	**30451397**
es20fst13	35	40	20	?s	**34438673**
es20fst14	36	44	20	?s	**34062374**
es20fst15	37	43	20	?s	**32303746**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es30fst01	79	115	30	?s	**40692993**
es30fst02	71	97	30	?s	**40900061**
es30fst03	83	120	30	?s	**43120444**
es30fst04	80	115	30	?s	**42150958**
es30fst05	58	71	30	?s	**41739748**
es30fst06	83	119	30	?s	**39955139**
es30fst07	53	64	30	?s	**43761391**
es30fst08	69	93	30	?s	**41691217**
es30fst09	43	44	30	?s	**37133658**
es30fst10	48	52	30	?s	**42686610**
es30fst11	79	112	30	?s	**41647993**
es30fst12	46	48	30	?s	**38416720**
es30fst13	65	84	30	?s	**37406646**
es30fst14	53	58	30	?s	**42897025**
es30fst15	118	188	30	?s	**43035576**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es40fst01	93	127	40	?s	**44841522**
es40fst02	82	105	40	?s	**46811310**
es40fst03	87	116	40	?s	**49974157**
es40fst04	55	55	40	?s	**45289864**
es40fst05	121	180	40	?s	**51940413**
es40fst06	92	123	40	?s	**49753385**
es40fst07	77	95	40	?s	**45639009**
es40fst08	98	137	40	?s	**48745996**
es40fst09	107	153	40	?s	**51761789**
es40fst10	107	152	40	?s	**57136852**
es40fst11	97	135	40	?s	**46734214**
es40fst12	67	75	40	?s	**43843378**
es40fst13	78	95	40	?s	**51884545**
es40fst14	98	134	40	?s	**49166952**
es40fst15	93	129	40	?s	**50828067**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es50fst01	118	160	50	?s	**54948660**
es50fst02	125	177	50	?s	**55484245**
es50fst03	128	182	50	?s	**54691035**
es50fst04	106	138	50	?s	**51535766**
es50fst05	104	135	50	?s	**55186015**
es50fst06	126	182	50	?s	**55804287**
es50fst07	143	211	50	?s	**49961178**
es50fst08	83	96	50	?s	**53754708**
es50fst09	139	202	50	?s	**53456773**
es50fst10	139	207	50	?s	**54037963**
es50fst11	100	131	50	?s	**52532923**
es50fst12	110	149	50	?s	**53409291**
es50fst13	92	116	50	?s	**53891019**
es50fst14	120	167	50	?s	**53551419**
es50fst15	112	147	50	?s	**52180862**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es60fst01	123	159	60	?s	**53761423**
es60fst02	186	280	60	?s	**55367804**
es60fst03	113	142	60	?s	**56566797**
es60fst04	162	238	60	?s	**55371042**
es60fst05	119	148	60	?s	**54704991**
es60fst06	130	174	60	?s	**60421961**
es60fst07	188	280	60	?s	**58978041**
es60fst08	109	133	60	?s	**58138178**
es60fst09	151	216	60	?s	**55877112**
es60fst10	133	177	60	?s	**57624488**
es60fst11	121	154	60	?s	**56141666**
es60fst12	176	257	60	?s	**59791362**
es60fst13	157	226	60	?s	**61213533**
es60fst14	118	149	60	?s	**56035528**
es60fst15	117	151	60	?s	**56622581**

Table 23: FST preprocessed instances by [67]

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es70fst01	154	209	70	?s	**62058863**
es70fst02	147	197	70	?s	**60928488**
es70fst03	181	264	70	?s	**61934664**
es70fst04	167	231	70	?s	**62938583**
es70fst05	169	231	70	?s	**62256993**
es70fst06	187	268	70	?s	**62124528**
es70fst07	167	230	70	?s	**62223666**
es70fst08	209	314	70	?s	**61872849**
es70fst09	161	220	70	?s	**62986133**
es70fst10	165	225	70	?s	**62511830**
es70fst11	177	254	70	?s	**66455760**
es70fst12	142	181	70	?s	**63047132**
es70fst13	160	219	70	?s	**62912258**
es70fst14	143	184	70	?s	**60411124**
es70fst15	178	251	70	?s	**62318458**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es80fst01	187	255	80	?s	**70927442**
es80fst02	183	249	80	?s	**65273810**
es80fst03	189	261	80	?s	**65332546**
es80fst04	198	280	80	?s	**64193446**
es80fst05	172	228	80	?s	**66350529**
es80fst06	172	224	80	?s	**71007444**
es80fst07	193	271	80	?s	**68228475**
es80fst08	217	306	80	?s	**67452377**
es80fst09	236	343	80	?s	**69825651**
es80fst10	156	197	80	?s	**65497988**
es80fst11	209	295	80	?s	**66283099**
es80fst12	147	180	80	?s	**65070089**
es80fst13	164	211	80	?s	**68022647**
es80fst14	209	297	80	?s	**70077902**
es80fst15	197	282	80	?s	**69939071**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es90fst01	181	231	90	?s	**68350357**
es90fst02	221	313	90	?s	**71294845**
es90fst03	284	430	90	?s	**74817473**
es90fst04	217	299	90	?s	**70910063**
es90fst05	190	254	90	?s	**71831224**
es90fst06	215	290	90	?s	**68640346**
es90fst07	175	221	90	?s	**72036885**
es90fst08	234	332	90	?s	**72341668**
es90fst09	234	331	90	?s	**67856007**
es90fst10	246	356	90	?s	**72310409**
es90fst11	225	323	90	?s	**72310039**
es90fst12	207	284	90	?s	**69367257**
es90fst13	240	349	90	?s	**72810663**
es90fst14	185	243	90	?s	**69188992**
es90fst15	207	286	90	?s	**71778294**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es100fst01	250	354	100	?s	**72522165**
es100fst02	339	522	100	?s	**75176630**
es100fst03	189	233	100	?s	**72746006**
es100fst04	188	235	100	?s	**74342392**
es100fst05	188	238	100	?s	**75670198**
es100fst06	301	452	100	?s	**74414990**
es100fst07	276	401	100	?s	**77740576**
es100fst08	210	276	100	?s	**73033178**
es100fst09	248	342	100	?s	**77952027**
es100fst10	229	312	100	?s	**75952202**
es100fst11	253	362	100	?s	**78674859**
es100fst12	266	385	100	?s	**76131099**
es100fst13	254	361	100	?s	**74604990**
es100fst14	198	253	100	?s	**78632795**
es100fst15	231	319	100	?s	**70446493**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es250fst01	623	876	250	?s	**116609813**
es250fst02	542	719	250	?s	**115150079**
es250fst03	543	727	250	?s	**114650399**
es250fst04	604	842	250	?s	**117819530**
es250fst05	596	832	250	?s	**116927089**
es250fst06	596	824	250	?s	**116256250**
es250fst07	585	799	250	?s	**115277351**
es250fst08	657	947	250	?s	**116833323**
es250fst09	570	770	250	?s	**116821988**
es250fst10	662	951	250	?s	**116857628**
es250fst11	661	952	250	?s	**112889613**
es250fst12	619	872	250	?s	**119035256**
es250fst13	684	993	250	?s	**116049496**
es250fst14	710	1046	250	?s	**116188791**
es250fst15	713	1053	250	?s	**115558198**

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
es500fst01	1250	1763	500	?s	**162978810**
es500fst02	1408	2056	500	?s	**160756854**
es500fst03	1337	1933	500	?s	**162664661**
es500fst04	1296	1879	500	?s	**164110997**
es500fst05	1172	1627	500	?s	**160586161**
es500fst06	1335	1932	500	?s	**164685074**
es500fst07	1214	1700	500	?s	**160124233**
es500fst08	1349	1972	500	?s	**161248138**
es500fst09	1294	1853	500	?s	**162100435**
es500fst10	1203	1679	500	?s	**155581203**
es500fst11	1274	1808	500	?s	**161674316**
es500fst12	1322	1918	500	?s	**164009591**
es500fst13	1273	1814	500	?s	**161324201**
es500fst14	1477	2204	500	?s	**165984329**
es500fst15	1334	1927	500	?s	**160758467**

Table 24: FST preprocessed instances by [67]

Name	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert T \rvert$	D	Opt
es1000fst01	2865	4267	1000	?m	**230535806**
es1000fst02	2629	3793	1000	?s	**227886471**
es1000fst03	2762	4047	1000	?s	**227807756**
es1000fst04	2778	4083	1000	?s	**230200846**
es1000fst05	2676	3894	1000	?s	**228330602**
es1000fst06	2815	4162	1000	?m	**231028456**
es1000fst07	2604	3756	1000	?s	**230945623**
es1000fst08	2834	4207	1000	?m	**230639115**
es1000fst09	2846	4187	1000	?s	**227745838**
es1000fst10	2546	3620	1000	?s	**229267101**
es1000fst11	2763	4038	1000	?s	**231605619**
es1000fst12	2984	4484	1000	?s	**230904712**
es1000fst13	2532	3615	1000	?s	**228031092**
es1000fst14	2840	4200	1000	?m	**234318491**
es1000fst15	2733	3997	1000	?s	**229965775**

Table 25: FST preprocessed instances by [67]

Name	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert T \rvert$	D	Opt
es10000fst01	27019	39407	10000	NPw	**716174280**

Table 26: FST preprocessed instances by [67]

Name	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert T \rvert$	D	Opt
a280fst	314	328	280	?s	**2502**
att48fst	139	202	48	?s	**30236**
att532fst	1468	2152	532	?m	**84009**
berlin52fst	89	104	52	?s	**6760**
bier127fst	258	357	127	?s	**104284**
d1291fst	1365	1456	1291	?s	**481421**
d1655fst	1906	2083	1655	?s	**584948**
d198fst	232	256	198	?s	**129175**
d2103fst	2206	2272	2103	?s	**769797**
d493fst	1055	1473	493	?m	**320137**
d657fst	1416	1978	657	?m	**471589**
dsj1000fst	2562	3655	1000	?s	**17564659**
eil101fst	330	538	101	?s	**605**
eil51fst	181	289	51	?s	**409**
eil76fst	237	378	76	?s	**513**
fl1400fst	2694	4546	1400	NP?	*18004178*
fl1577fst	2413	3412	1577	?m	**19825626**
fl3795fst	4859	6539	3795	NP?	*12704393*
fl417fst	732	1084	417	?s	**10883190**
fnl4461fst	17127	27352	4461	??	*154038*
gil262fst	537	723	262	?s	**2306**
kroA100fst	197	250	100	?s	**20401**
kroA150fst	389	562	150	?s	**25700**
kroA200fst	500	714	200	?s	**28652**
kroB100fst	230	313	100	?s	**21211**
kroB150fst	420	619	150	?s	**25217**
kroB200fst	480	670	200	?s	**28803**
kroC100fst	244	337	100	?s	**20492**
kroD100fst	216	288	100	?s	**20437**
kroE100fst	226	306	100	?s	**21245**
lin105fst	216	323	105	?s	**13429**
lin318fst	678	1030	318	?s	**39335**
linhp318fst	678	1030	318	?s	**39335**
nrw1379fst	5096	8105	1379	?h	**56207**
p654fst	777	867	654	?s	**314925**
pcb1173fst	1912	2223	1173	?s	**53301**
pcb3038fst	5829	7552	3038	?m	**131895**
pcb442fst	503	531	442	?s	**47675**
pla7397fst	8790	9815	7397	?m	**22481625**
pr1002fst	1473	1715	1002	?s	**243176**
pr107fst	111	110	107	?s	**34850**
pr124fst	154	165	124	?s	**52759**
pr136fst	196	250	136	?s	**86811**
pr144fst	221	285	144	?s	**52925**
pr152fst	308	431	152	?s	**64323**
pr226fst	255	269	226	?s	**70700**
pr2392fst	3398	3966	2392	?s	**358989**
pr264fst	280	287	264	?s	**41400**
pr299fst	420	500	299	?s	**44671**
pr439fst	572	662	439	?s	**97400**
pr76fst	168	247	76	?s	**95908**
rat195fst	560	870	195	?s	**2386**
rat575fst	1986	3176	575	?s	**6808**
rat783fst	2397	3715	783	?m	**8883**
rat99fst	269	399	99	?s	**1225**
rd100fst	201	253	100	?s	**764269099**
rd400fst	1001	1419	400	?s	**1490972006**
rl11849fst	13963	15315	11849	?h	**8779590**
rl1304fst	1562	1694	1304	?s	**236649**
rl1323fst	1598	1750	1323	?s	**253620**
rl1889fst	2382	2674	1889	?s	**295208**
rl5915fst	6569	6980	5915	?m	**533226**
rl5934fst	6827	7365	5934	?m	**529890**
st70fst	133	169	70	?s	**626**
ts225fst	225	224	225	?s	**1120**
tsp225fst	242	252	225	?s	**356850**
u1060fst	1835	2429	1060	?m	**21265372**
u1432fst	1432	1431	1432	?s	**1465**
u159fst	184	186	159	?s	**390**
u1817fst	1831	1846	1817	?s	**5513053**
u2152fst	2167	2184	2152	?s	**6253305**
u2319fst	2319	2318	2319	?s	**2322**
u574fst	990	1258	574	?s	**3509275**
u724fst	1180	1537	724	?s	**4069628**
vm1084fst	1679	2058	1084	?s	**2248390**
vm1748fst	2856	3641	1748	?s	**3194670**

Table 27: FST preprocessed TSPLIB instances by [67]

Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	D	Opt
gene181	267	355	25	Ps	**88**
gene1	267	355	25	Ps	**88**
gene425	425	554	86	Ps	**214**
gene42	335	456	43	Ps	**126**
gene442	442	594	86	Ps	**207**
gene575	575	824	86	Ps	**207**
gene602	602	858	86	Ps	**209**
gene61a	395	512	82	Ps	**205**
gene61b	570	808	82	Ps	**199**
gene61c	549	790	82	Ps	**196**
gene61f	412	552	82	Ps	**198**

Table 28: Genetic instances from [31]

Table 28 shows such instances arising from problems in genetics [31]. Other examples result from modeling set covering problems as Steiner tree problems.

5.2 Multicast Routing

A great variety of generalizations of the Steiner tree problem may be found in the telecommunications industry such as multicast routing or multi-point problems. For instance, the underlying graph may represent a network for the transport of information between a single source and some destinations such that the (weighted) length of the paths used for this transport is somehow limited (see, e.g., multicast routing or the Steiner tree problem with hop constraints [62, 25]).

Of course additional values need to be defined regarding multicast routing or delay constraints (see, e.g., [68, 9, 23, 42]).

5.3 The Group Steiner Tree Problem

The group Steiner tree problem in graphs is a generalization of the Steiner tree problem where the assumption to connect given fixed basic nodes is relaxed. Instead of basic nodes some regions are considered and the objective is to find a tree of minimum weight connecting at least one point of every region.

Given a connected undirected graph $G = (V, E)$ and g so-called basic sets $T_1, \ldots, T_g \subseteq V$, the problem is to find a connected minimum cost subgraph of G such that the node set of this subgraph contains at least one node from each basic set T_i for all $1 \leq i \leq g$. This *group Steiner tree problem* can be found under different names in the literature such as *class Steiner tree problem* or *Steiner problem with basic sets*. It is considered, e.g., in [29, 47].

The data sets considered in [29] are as follows. For a given number of nodes ($n = 10, 50, 250$) a number of groups ($g = 2, \log n, \sqrt{n}, \frac{n}{2}, n$) is specified (for $n = 250$ the authors also use $g = \log^2 n$). The nodes are assigned to groups such that each group has about the same number $\frac{n}{g}$ of nodes. The number of edges is chosen to derive average node degrees of $4, \log n, \sqrt{n}, n-1$. Starting

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
wrp3-11	128	227	11	Ps	**1100361**
wrp3-12	84	149	12	Ps	**1200237**
wrp3-13	311	613	13	Ps	**1300497**
wrp3-14	128	247	14	Ps	**1400250**
wrp3-15	138	257	15	Ps	**1500422**
wrp3-16	204	374	16	Ps	**1600208**
wrp3-17	177	354	17	Ps	**1700442**
wrp3-19	189	353	19	Ps	**1900439**
wrp3-20	245	454	20	Ps	**2000271**
wrp3-21	237	444	21	Ps	**2100522**
wrp3-22	233	431	22	Ps	**2200557**
wrp3-23	132	230	23	Ps	**2300245**
wrp3-24	262	487	24	Ps	**2400623**
wrp3-25	246	468	25	Ps	**2500540**
wrp3-26	402	780	26	Ps	**2600484**
wrp3-27	370	721	27	Ps	**2700502**
wrp3-28	307	559	28	Ps	**2800379**
wrp3-29	245	436	29	Ps	**2900479**
wrp3-30	467	896	30	Ps	**3000569**
wrp3-31	323	592	31	Ps	**3100635**
wrp3-33	437	838	33	Ps	**3300513**
wrp3-34	1244	2474	34	Pm	**3400646**
wrp3-36	435	818	36	Ps	**3600610**
wrp3-37	1011	2010	37	Pm	**3700485**
wrp3-38	603	1207	38	Pm	**3800656**
wrp3-39	703	1616	39	Ph	**3900450**
wrp3-41	178	307	41	Ps	**4100466**
wrp3-42	705	1373	42	Pm	**4200598**
wrp3-43	173	298	43	Ps	**4300457**
wrp3-45	1414	2813	45	Pm	**4500860**
wrp3-48	925	1738	48	Pm	**4800552**
wrp3-49	886	1800	49	Pm	**4900882**
wrp3-50	1119	2251	50	NPh	**5000673**
wrp3-52	701	1352	52	NPm	**5200825**
wrp3-53	775	1471	53	Ps	**5300847**
wrp3-55	1645	3186	55	NPh	**5500888**
wrp3-56	853	1590	56	Pm	**5600872**
wrp3-60	838	1763	60	Ph	**6001164**
wrp3-62	670	1316	62	Pm	**6201016**
wrp3-64	1822	3610	64	Ph	**6400931**
wrp3-66	2521	4858	66	Ph	**6600922**
wrp3-67	987	1923	67	Pm	**6700776**
wrp3-69	856	1621	69	Pm	**6900841**
wrp3-70	1468	2931	70	Pm	**7000890**
wrp3-71	1221	2414	71	Pm	**7101028**
wrp3-73	1890	3613	73	Ph	**7301207**
wrp3-74	1019	1941	74	Pm	**7400759**
wrp3-75	729	1395	75	Pm	**7501020**
wrp3-76	1761	3370	76	Ph	**7601028**
wrp3-78	2346	4656	78	??	*7801107*
wrp3-79	833	1595	79	Pm	**7900444**
wrp3-80	1491	2831	80	Pm	**8000849**
wrp3-83	3168	6220	83	??	*8300941*
wrp3-84	2356	4547	84	??	*8401115*
wrp3-85	528	1017	85	NPm	**8500739**
wrp3-86	1360	2607	86	Pm	**86000746**
wrp3-88	743	1409	88	NPm	**88001175**
wrp3-91	1343	2594	91	Ph	**91000866**
wrp3-92	1765	3613	92	Ph	**92000764**
wrp3-94	1976	3836	94	NPh	**94001181**
wrp3-96	2518	4985	96	??	*96001202*
wrp3-98	2265	4545	98	??	*98001294*
wrp3-99	2076	4072	99	??	*99001131*

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
wrp4-11	123	233	11	Ps	**1100179**
wrp4-13	110	188	13	Ps	**1300798**
wrp4-14	145	283	14	Ps	**1400290**
wrp4-15	193	369	15	Ps	**1500405**
wrp4-16	311	579	16	Ps	**1601190**
wrp4-17	223	404	17	Ps	**1700525**
wrp4-18	211	380	18	Ps	**1801464**
wrp4-19	119	206	19	Ps	**1901446**
wrp4-21	529	1032	21	Ps	**2103283**
wrp4-22	294	568	22	Ps	**2200394**
wrp4-23	257	515	23	Ps	**2300376**
wrp4-24	493	963	24	Ps	**2403332**
wrp4-25	422	808	25	Ps	**2500828**
wrp4-26	396	781	26	Pm	**2600443**
wrp4-27	243	497	27	Ps	**2700441**
wrp4-28	272	545	28	Ps	**2800466**
wrp4-29	247	505	29	Ps	**2900484**
wrp4-30	361	724	30	Pm	**3000526**
wrp4-31	390	786	31	Pm	**3100526**
wrp4-32	311	632	32	Ps	**3200554**
wrp4-33	304	571	33	Ps	**3300655**
wrp4-34	314	650	34	Ps	**3400525**
wrp4-35	471	954	35	Pm	**3500601**
wrp4-36	363	750	36	Pm	**3600596**
wrp4-37	522	1054	37	Pm	**3700647**
wrp4-38	294	618	38	Ps	**3800606**
wrp4-39	802	1553	39	Pm	**3903734**
wrp4-40	538	1088	40	Pm	**4000758**
wrp4-41	465	955	41	Pm	**4100695**
wrp4-42	552	1131	42	Pm	**4200701**
wrp4-43	596	1148	43	Ps	**4301508**
wrp4-44	398	788	44	NPs	**4401504**
wrp4-45	388	815	45	Ps	**4500728**
wrp4-46	632	1287	46	Pm	**4600756**
wrp4-47	555	1098	47	Ps	**4701318**
wrp4-48	451	825	48	Ps	**4802220**
wrp4-49	557	1080	49	Ps	**4901968**
wrp4-50	564	1112	50	Pm	**5001625**
wrp4-51	668	1306	51	Pm	**5101616**
wrp4-52	547	1115	52	Pm	**5201081**
wrp4-53	615	1232	53	Pm	**5301351**
wrp4-54	688	1388	54	NPm	**5401534**
wrp4-55	610	1201	55	Pm	**5501952**
wrp4-56	839	1617	56	Pm	**5602299**
wrp4-58	757	1493	58	Pm	**5801466**
wrp4-59	904	1806	59	NPm	**5901592**
wrp4-60	693	1370	60	Pm	**6001782**
wrp4-61	775	1538	61	Ps	**6102210**
wrp4-62	1283	2493	62	Pm	**6202100**
wrp4-63	1121	2227	63	Ph	**6301479**
wrp4-64	632	1281	64	Pm	**6401996**
wrp4-66	844	1691	66	Pm	**6602931**
wrp4-67	1518	3060	67	Pm	**6702800**
wrp4-68	917	1850	68	Pm	**6801753**
wrp4-69	574	1165	69	Ps	**6902328**
wrp4-70	637	1269	70	Ps	**7003022**
wrp4-71	802	1609	71	Pm	**7102320**
wrp4-72	1151	2274	72	NPm	**7202807**
wrp4-73	1898	3616	73	Ph	**7302643**
wrp4-74	802	1620	74	Pm	**7402046**
wrp4-75	938	1869	75	Pm	**7501712**
wrp4-76	766	1535	76	Pm	**7602040**

Table 29: VLSI Wire-Routing instances from industry

from a complete graph, edges are randomly deleted such that the graph remains connected and obtains the desired number of edges. Edge weights are either chosen to be identical or integer values randomly selected from the interval $[1, 100]$.

As the group Steiner problem is motivated by the wire routing phase in physical VLSI design, we add some real-world instances to our library. These are rectilinear instances with 30 to 84 terminal groups (see Table 29). The instances are already converted to a Steiner tree problem in graphs by introducing a pseudo-terminal for each group and connecting the terminals of the group to this pseudo-terminal by an edge with high cost. The original terminals of the group get non-terminals and the pseudo-terminals become the terminals of the Steiner tree instance.

5.4 The Prize Collecting Steiner Problem

The Steiner tree problem in graphs seeks a minimum cost tree connecting a given set of basic nodes or terminals. In various settings, however, even the basic nodes are not known beforehand. The prize-collecting Steiner problem assumes revenues for the inclusion of nodes into a solution. That is, the objective is to maximize the revenues minus the cost of the included edges. This problem has important applications in the telecommunications industry.

Problem instances in the literature are, e.g., modifications from the C and D instances from the OR-Library as well as data representing real-world networks [8, 30].

5.5 The Steiner Tree Packing Problem

As mentioned on page 8 the routing problem in VLSI design can be modeled as the problem of packing Steiner trees in certain graphs. That is we are given a capacitated graph $G = (V, E)$ with edge capacities c_e and edge weights $w_e, e \in E$, and a list of terminal sets $T_1, \ldots, T_N \subseteq V$. The task is to find edge sets $S_1, \ldots, S_N$ such that each S_i is a Steiner tree for $T_i \, (i = 1, \ldots, N)$, the capacity constraints are satisfied, i.e., the number of Steiner trees using edge e is at most c_e, and the total weight of the Steiner trees, i.e., $\sum_{i=1}^{N} w(S_i)$, is minimized. We have added some small Steiner tree packing problems to the *SteinLib*. These instances are defined on complete rectangular grid graphs, where all terminal sets are located on the outer face. These instances have been used as benchmark test sets in the VLSI literature for the evaluation of heuristics for so-called switchbox routing problems. More information on this subject can be found in [40, 26].

6 Evaluation of the Data Set

In the last decade significant progress has been made regarding the optimal solution of the Steiner tree problem in graphs. Besides heuristics (for surveys see, e.g., [17, 63]) the most important contributions came from clever reduction techniques (for an update of research see [14]) incorporated into branch-and-bound or branch-and-cut techniques. The currently most successful algorithms have been devised by various authors [10, 13, 35, 43, 44, 50, 56, 70]. With those algorithms most of the data described above may be solved to optimality within computation times deemed practical on up-to-date computers. (See also our difficulty classification given in the above tables.)

A remarkable horse-race happened to be undertaken on one of Beasley's E-data, i.e., E-18. This instance turned out to be the most prominent one to be "nailed down" throughout the years [35]. Unfortunately, however, this does not allow any deep conclusions regarding the difficulty of various instances. That is, no single formula may be given to indicate whether an instance is difficult and it is not expected to find such a formula.

For metrical Steiner tree problems the key for success is the reduction of the instances by computation of full Steiner trees. Based on these reductions rectilinear instances with thousands of nodes are now tractable [66, 36, 67].

Some obvious observations are as follows. If the number of basic nodes is very small or very large then the Steiner tree problem in graphs becomes somewhat easy as the shortest path problem ($t = 2$) and the minimum spanning tree problem ($t = n$) are polynomially solvable special cases. Furthermore, for t being very small or $t \geq \frac{n}{2}$ some powerful reduction techniques are at hand to solve most of those instances to optimality, cf. [15, 16, 17, 13, 59].

7 Conclusions

In this paper we have described *SteinLib*, a library of problem instances for the Steiner tree problem in graphs. While some instances turned out to be hard for some time most of them have been solved over the years and are no longer challenging for state-of-the-art algorithms and software packages. Therefore, readers are invited to add to the library new instances, especially those that are relevant in practice and turn out to be somewhat hard to solve.

Motivated from different application areas we may obtain difficult instances that may model hard instances from other areas. Examples in this respect are set covering problem instances (see, e.g., [7]), warehouse location problem instances (see, e.g., [37]), or certain lot sizing problems (see, e.g., [20, 62]). In addition it may be of interest to construct instances that turn out to be difficult for current state-of-the-art algorithms. Ideas in this respect may consider to make problem reduction unlikely to work (see, e.g., the incidence instances) and to combine parts of instances that are difficult for one or the other algorithm, such as combining worst case instances for some heuristics (e.g., [49]) with parts that are difficult for primal or dual approaches.

As a possible idea we propose to start, e.g., from simple wheels of some size w as the one given in Figure 1 with size $w = 3$. Now we consider a cycle of size c and replace each node of the cycle by one such wheel. Two neighboring wheels are connected as follows. The non-terminal nodes of one wheel and the terminal nodes of the other build again a wheel replacing the connecting edge of the cycle. To avoid the immediate success of simple reduction techniques all

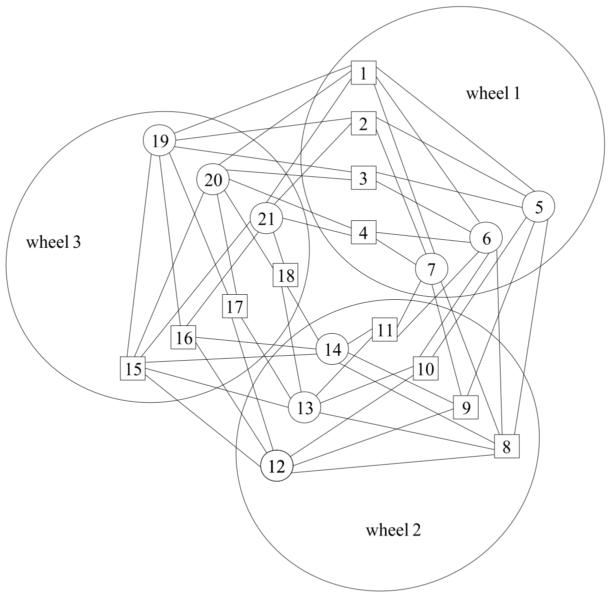

Figure 3: Cycle of wheels with $w = 3$ and $c = 3$

Name	$\|V\|$	$\|E\|$	$\|T\|$	D	Opt
w13c29	783	2262	406	NP?	*510*
w23c23	1081	3174	552	NP?	*694*
w3c571	3997	10278	2284	NP?	*3050*

Table 30: Cycle of wheels instances

edge weights are one. In Figure 3 we provide an example for $w = 3$ and $c = 3$. In Table 30 three instances of moderate size are given as example.

Future research should consider the development of measures and criteria that may be used to determine whether problem instances are hard. Of course those instances (e.g., some of the incidence data) that turned out to be somewhat more difficult to solve should replace the by now well understood B-E data which served as an excellent benchmark for more than a decade.

Another research question refers to the development of problem generators. While different data generation schemes have been described above, optimal solutions to the resulting instances are only known once someone has nailed them down with an appropriate procedure. To avoid these so-called horse-races, it may be of interest to develop data generators that provide instances with known optimal solutions, as these might serve as benchmark instances. The only known generator in this respect [34] provides problem instances based on a mathematical programming formulation of the Steiner tree problem in graphs and the so-called Karush-Kuhn-Tucker optimality conditions. However, this generator provides instances that are easily solvable with currently known state-of-the-art software so that the development of generators providing more versatile instances is still a challenge.

References

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

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

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

[4] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.

[5] J.E. Beasley. A heuristic for Euclidean and rectilinear Steiner problems. *European Journal of Operational Research*, 58:284–292, 1992.

[6] R.E. Bixby, S. Ceria, C. McZeal, and M.W.P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. Paper and Problems available at WWW Page: http://www.caam.rice.edu/~bixby/miplib/miplib.html, 1998.

[7] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, Technische Universität Berlin, 1998.

[8] S.A. Canuto, C.C. Ribeiro, and M.G.C. Resende. Local search with perturbations for the prize collecting Steiner tree problem. In *Extended Abstracts of the third Metaheuristics International Conference*, pages 115–119. Catholic University of Rio de Janeiro, 1999.

[9] J. Cho and J. Breen. Analysis of the performance of dynamic multicast routing algorithms. *Computer Communications*, 22:667–674, 1999.

[10] S. Chopra, E.R. Gorres, and M.R. Rao. Solving the Steiner tree problem on a graph using branch and cut. *ORSA Journal on Computing*, 4:320–335, 1992.

[11] K.A. Dowsland. Hill-climbing, simulated annealing and the Steiner problem in graphs. *Engineering Optimization*, 17:91–107, 1991.

[12] D.-Z. Du, J.M. Smith, and J.H. Rubinstein, editors. *Advances in Steiner Trees*. Kluwer, Boston, 2000.

[13] C. Duin. *Steiner Problems in Graphs*. PhD thesis, University of Amsterdam, 1993.

[14] C. Duin. Preprocessing the Steiner problem in graphs. In D.-Z. Du, J.M. Smith, and J.H. Rubinstein, editors, *Advances in Steiner Trees*, pages 175–233. Kluwer, 2000.

[15] C. Duin and A. Volgenant. An edge elimination test for the Steiner problem in graphs. *Operations Research Letters*, 8:79–83, 1989.

[16] C. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19:549–567, 1989.

[17] C. Duin and S. Voß. Steiner tree heuristics — a survey. In H. Dyckhoff, U. Derigs, M. Salomon, and H. C. Tijms, editors, *Operations Research Proceedings*, pages 485–496, Berlin, 1994. Springer.

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

[19] C. Duin and S. Voß. The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks*, 34:181–191, 1999.

[20] R.E. Erickson, C.L. Monma, and A.F. Veinott. Send-and-split method for minimum concave-cost network flows. *Mathematics of Operations Research*, 12:634–664, 1987.

[21] J.S. Farris. Inferring phylogenetic trees from chromosome inversion data. *Systematic Zoology*, 27:275–284, 1987.

[22] C.E. Ferreira. O problema de Steiner em grafos: uma abordagem Poliédrica. Master's thesis, Universidade de São Paulo, 1989.

[23] A. Ghanwani. Neural and delay based heuristics for the Steiner problem in networks. *European Journal of Operational Research*, pages 241–265, 1998.

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

[25] L. Gouveia. Using variable redefinition for computing minimum spanning and Steiner trees with hop constraints. Technical report, Faculdade de Ciencias, Universidade de Lisboa, 1997.

[26] M. Grötschel, A. Martin, and R. Weismantel. The Steiner tree packing problem in VLSI-design. *Mathematical Programming*, 78:265–281, 1997.

[27] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal of Applied Mathematics*, 14:255–265, 1966.

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

[29] E. Ihler, G. Reich, and P. Widmayer. Class Steiner trees and VLSI-design. *Discrete Applied Mathematics*, 90:173–194, 1999.

[30] D.S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: theory and practice. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 760–769. SIAM, 2000.

[31] J.J. Johnston, R.I. Kelley, T.O. Crawford, D.H. Morton, R. Agarwala, T. Koch, A.A. Schäffer, C.A. Francomano, and L.G. Biesecker. A novel nemaline myopathy in the Amish caused by a mutation in troponin T1. *American Journal of Human Genetics*, pages 814–821, October 2000.

[32] M. Jünger, A. Martin, G. Reinelt, and R. Weismantel. Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. *Mathematical Programming*, 63:257–279, 1994.

[33] B.N. Khoury and P.M. Pardalos. A heuristic for the Steiner problem in graphs. *Computational Optimization and Applications*, 6:5–14, 1996.

[34] B.N. Khoury, P.M. Pardalos, and D.-Z. Du. A test problem generator for the Steiner problem in graphs. *ACM Transactions on Mathematical Software*, 19:509–522, 1993.

[35] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

[36] T. Koch, A. Martin, and M. Zachariasen. Computations based on [35] and [66], 1999.

[37] M. Körkel. *Effiziente Verfahren zur Lösung unkapazitierter Standort-Probleme.* vwf, Berlin, 1999.

[38] B. Korte, H.J. Prömel, and A. Steger. Steiner trees in VLSI-layout. In B. Korte, L. Lovász, H.J. Prömel, and A. Schrijver, editors, *Paths, Flows, and VLSI-Layout*, pages 185–214. Springer, Berlin, 1990.

[39] D. Lee. Some industrial case studies of Steiner trees. Paper presented at the NATO Advanced Research Workshop Topological Network Design Analysis and Synthesis, Copenhagen, 1989.

[40] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley, New York, 1990.

[41] A. Lin. Personal communication, 2001.

[42] C.P. Low. Loop-free multicast routing with end-to-end delay constraints. *Computer Communications*, 22:181–192, 1999.

[43] A. Lucena. Steiner problem in graphs: Lagrangean relaxation and cutting-planes. *Bulletin of the Committee on Algorithms*, 21:2–7, 1992.

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

[45] F. Margot. Personal communication, 1994.

[46] M. Minoux. Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity. *INFOR*, 28:221–233, 1990.

[47] Y.-S. Myung, C.-H. Lee, and D.-W. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26:231–242, 1995.

[48] D. Penny and M.D. Hendy. Turbotree: A fast algorithm for minimal trees. *Computer Applications in the Biosciences*, 3:183–187, 1987.

[49] J. Plesnik. The Steiner tree problem in graphs: Worst case examples for insertion heuristics. *International Journal of Mathematical Algorithms*, 1:21–34, 1999.

[50] T. Polzin and S.V. Daneshmand. Improved algorithms for the Steiner problem in networks. Technical report, University of Mannheim, 1998.

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

[52] G. Reinelt. TSPLIB — a traveling salesman problem library. *ORSA Journal on Computing*, 3:376 – 384, 1991.

[53] J. Sessions. Solving for habitat connections as a Steiner network problem. *Forest Science*, 38:203–207, 1992.

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

[55] C. Stanton and J. MacGregor Smith. Steiner trees and 3d macromolecular conformation. Technical report, University of Massachusets, Amherst, 2000.

[56] E. Uchoa, M.P. de Aragão, and C. Ribeiro. Preprocessing Steiner problems from vlsi layout. Technical Report MCC 32/99, PUC-Rio, 1999.

[57] M.G.A. Verhoeven. *Parallel Local Search.* PhD thesis, Eindhoven University of Technology, 1996.

[58] M.G.A. Verhoeven, M.E.M. Severens, and E.H.L. Aarts. Local search for Steiner trees in graphs. In V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, *Modern Heuristic Search Methods*, pages 117–129. Wiley, Chichester, 1996.

[59] S. Voß. *Steiner-Probleme in Graphen*. Hain, Frankfurt/Main, 1990.

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

[61] S. Voß. Observing logical interdependencies in tabu search — methods and results. In V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, *Modern Heuristic Search Methods*, pages 41–59. Wiley, Chichester, 1996.

[62] S. Voß. The Steiner tree problem with hop constraints. *Annals of Operations Research*, 86:321–345, 1999.

[63] S. Voß. Modern heuristic search methods for the Steiner tree problem in graphs. In D.-Z. Du, J.M. Smith, and J.H. Rubinstein, editors, *Advances in Steiner Trees*, pages 283–323. Kluwer, 2000.

[64] J.A. Wald and P.G. Sorensen. Resolving the query inference problem using Steiner trees. *ACM Transactions on Database Systems*, 9:348–368, 1984.

[65] G.A. Walters. The design of the optimal layout for a sewer network. *Engineering Optimization*, 9:37–50, 1985.

[66] D.M. Warme, P. Winter, and M. Zachariasen. Exact algorithms for plane Steiner tree problems: A computational study. In D.-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 81–116. Kluwer, 2000.

[67] D.M. Warme and M. Zachariasen. Personal communication, 2000.

[68] B.M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6:1617–1622, 1988.

[69] P. Winter. Steiner problem in Halin networks. *Discrete Applied Mathematics*, 17:281–294, 1987.

[70] P. Winter. Reductions for the rectilinear Steiner tree problem. Technical Report 11-95, Rutcors University, 1995.

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

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

[73] J. Xu, S.Y. Chiu, and F. Glover. A probabilistic tabu search for the telecommunications network design. *Combinatorial Optimization: Theory and Practice*, 1:69–94, 1996.