New Real-world Instances for the Steiner Tree Problem in Graphs

Markus Leitner[1], Ivana Ljubić[1], Martin Luipersbeck[2], Markus Prossegger[3], Max Resch[2]

January 13, 2014

Abstract

This work deals with the creation and optimization of real-world instances in the design of telecommunication networks. We propose two new sets of benchmark instances for the Steiner tree problem in graphs, which is one of the fundamental network optimization problems. Our instances are large, sparse graphs that contain over 100 000 nodes and originate from real-world applications of deploying last-mile fiber-optic networks. To obtain a rough estimate on the hardness of the new instances, we measured the performance of preprocessing techniques and of an exact algorithm based on branch-and-cut. This work shall establish a missing link between the real world and the mathematical modeling and optimization of telecommunication networks.

1 Introduction

One of the fundamental problems in the design of telecom networks is the Steiner tree problem in graphs (STP) which searches for a subtree in an edge-weighted graph that connects a subset of nodes (called *terminals*) at minimum cost. The performance of algorithms for the STP is usually evaluated through practical experiments on publicly available sets of benchmark instances (e.g., SteinLib [10]). However, at the time of this paper, only a relatively small number of these instances include real-world graphs with more than 10 000 nodes. The main purpose of this article is to establish a missing link between the real-world input data and the mathematical modeling and optimization of telecommunication networks. To this end, we propose *two new sets of benchmark instances* that contain a huge number of nodes (up to 100 000) and that represent edge-weighted graphs based on spatial data.

[1]Department of Statistics and Operations Research, University of Vienna, Austria
markus.leitner@univie.ac.at, ivana.ljubic@univie.ac.at
[2]Vienna University of Technology, Austria
martin.luipersbeck@alumni.tuwien.ac.at, max.resch@alumni.tuwien.ac.at
[3]Carinthia University of Applied Sciences, Klagenfurt, Austria
m.prossegger@cuas.at

The presented instances should be of broader interest for the network optimization community since they originate from real-world problems that appear in practical network design. To ensure that the instances pose a challenge to state-of-the-art exact methods for solving the STP, we applied preprocessing techniques and a branch-and-cut algorithm. The remainder of this paper is structured as follows: In Section 2, we provide information about the set of new benchmark instances. In Sections 3 and 4, we describe the applied preprocessing techniques and the branch-and-cut algorithm, respectively. In Section 5, we present the computational results. Concluding remarks are made in Section 6.

2 New Benchmark Instances

The STP is known to be $\mathcal{NP}$-hard [8] and can be formally defined as follows: Let $G = (V, E, c)$ be an edge-weighted undirected graph with a weight function $c : E \to \mathbb{Z}^+$ and a set of terminals $T \subseteq V$. The goal is to find a connected subgraph $S = (V_S, E_S)$ of G for which $T \subseteq V_S$ and the sum of edge weights $\sum_{e \in E_S} c_e$ is minimal. Nodes from $V \setminus T$ are also called *Steiner nodes*.

In this section we provide some details on the generation of instances originating from spatial data (more detailed information can be found in Prossegger [12]). The instances were created using land use data covering part of a city and its surrounding rural environment. More precisely, the following information is used:

1. spatial polygon data describing the land use of an area, and
2. point-objects, describing customer locations.

Spatial data originates from the Austrian digital cadastral map that contains information on parcel's boundaries of all public and private properties. It also documents the type of land use of each parcel as well as buildings. *Edges* of the graph represent the *contours* and emphcrossings of spatial polygon-objects. *Steiner nodes* are spatial point-objects on edge crossings. *Terminals* are spatial point-objects representing centroids of buildings. To ensure data privacy, terminals in our instances are chosen as random subsets of artificial customer locations.

Since we are dealing with graphs modeling the deployment of fiber optic telecommunication networks, *edge weights* are obtained as *averaged construction costs*, including costs for the underground work and costs for building cable poles in a correct proportion. The type of the land use (e.g., building land, forest, highway, street,...) determines the costs for the underground work.

The instances are divided into two sets:

`GEO`-**Instances**: This set contains 23 instances originating from an Austrian city, with different deployment areas and different density concerning the number of terminals. The graphs contain between 42 481 and 235 686 nodes, 52 552 and 366 093 edges, and between 88 and 6313 terminals. These basic instance properties, namely $|V|$, $|E|$ and $|T|$, are listed in Table 2. Figures 1 to 3 show `GEO` instances and their optimal solutions, plotted with coordinates included in the instance files. The *simple preprocessing step* (see below) was skipped for the GEO instances, it deemed unnecessary, hence no comparison data is available for this step.

I-Instances: This set contains 85 instances representing deployment areas from various Austrian cities, but they also include rural areas with smaller population density and very sparse infrastructure. The coordinates and construction data of the set I cannot be disclosed to the public. The instances we publish are modified in a way that does not allow inference of the original data. This is the reason why only *simple preprocessed data* (see below) is available for the I-instances. The underlying graphs contain between 7886 and 178 810 nodes, 9265 and 239 552 edges, and between 38 and 4991 terminals. Table 3 provides $|V|$, $|E|$ and $|T|$ for each single instance of this set.

3 Preprocessing

The aim of preprocessing is to simplify all instances prior to using a time-consuming exact algorithm. Table 1 lists all used tests. For a description of the tests, the reader is referred to the works of Uchoa, Aragão, and Ribeiro [15] and Duin [6].

We first apply a *simple preprocessing* procedure in which degree reduction tests (NTD1 and NTD2) are executed exhaustively. Then we continue with an *advanced preprocessing* procedure as proposed by Ribeiro, Uchoa, and Werneck [13], using their publicly available implementation Bossa [**bossa**]. Algorithm 1 presents a short description of the preprocessing procedure's structure as implemented in Bossa [**bossa**]. The procedure includes a number of reduction tests, which are referred to by their acronym. If not further specified, a test in the algorithm refers to calling it once for each node/edge in the graph.

Table 1: Applied reduction tests.

Acronym	Reduction test
NTD1	Non-Terminal of Degree 1
NTD2	Non-Terminal of Degree 2
TD1	Terminal of Degree 1
NSV	Nearest Special Vertex (also known as Terminal Distance test)
SDE	Special Distance with Equality
SDExp	SDE with Expansion

4 Exact Solution Approach

The implemented branch-and-cut procedure is based on the algorithm proposed by Koch and Martin [9]. In this paper we will only cover the differences between our and their implementation. For a complete description the reader is referred to the original paper.

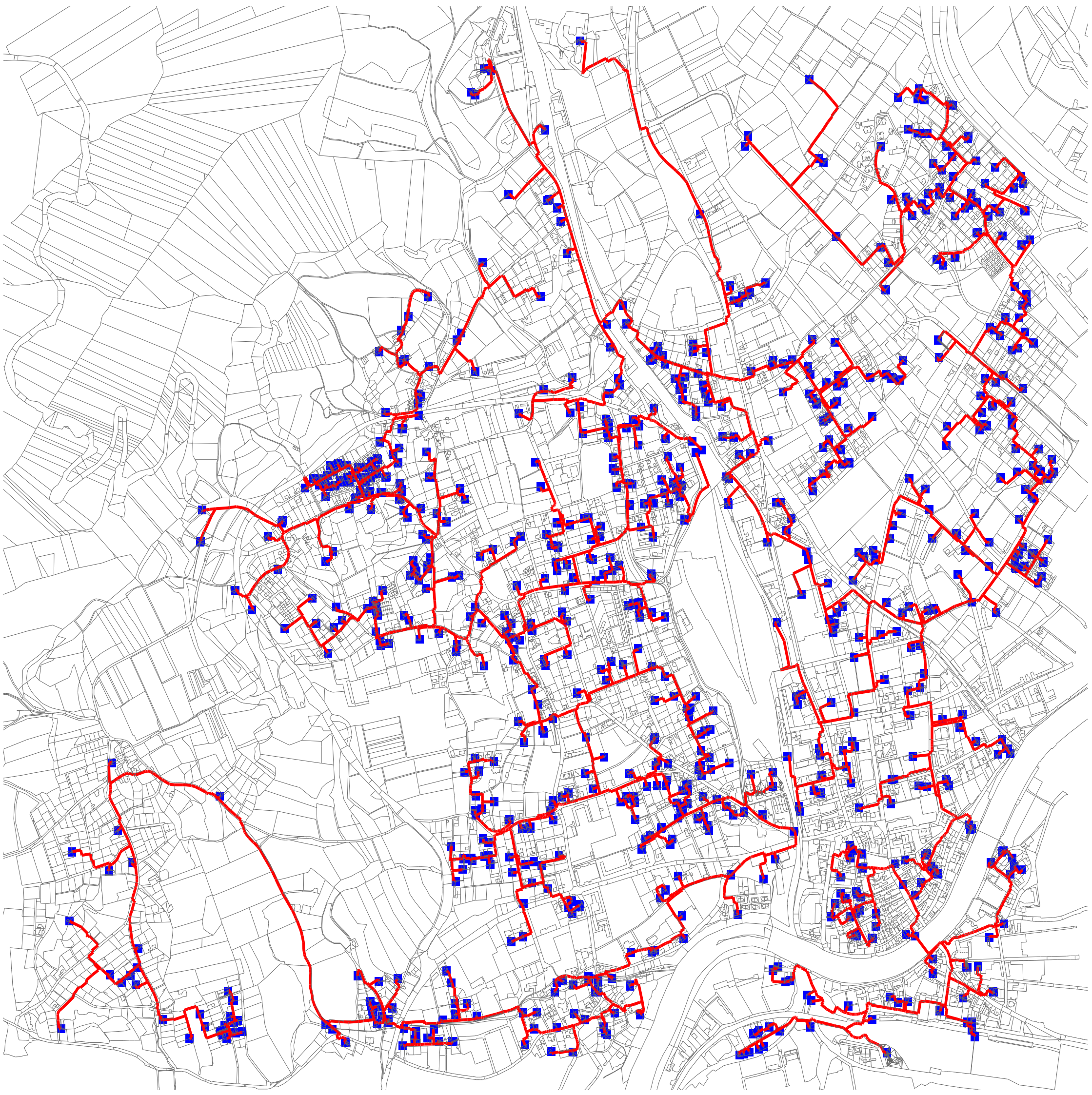


Figure 1: Instance G107

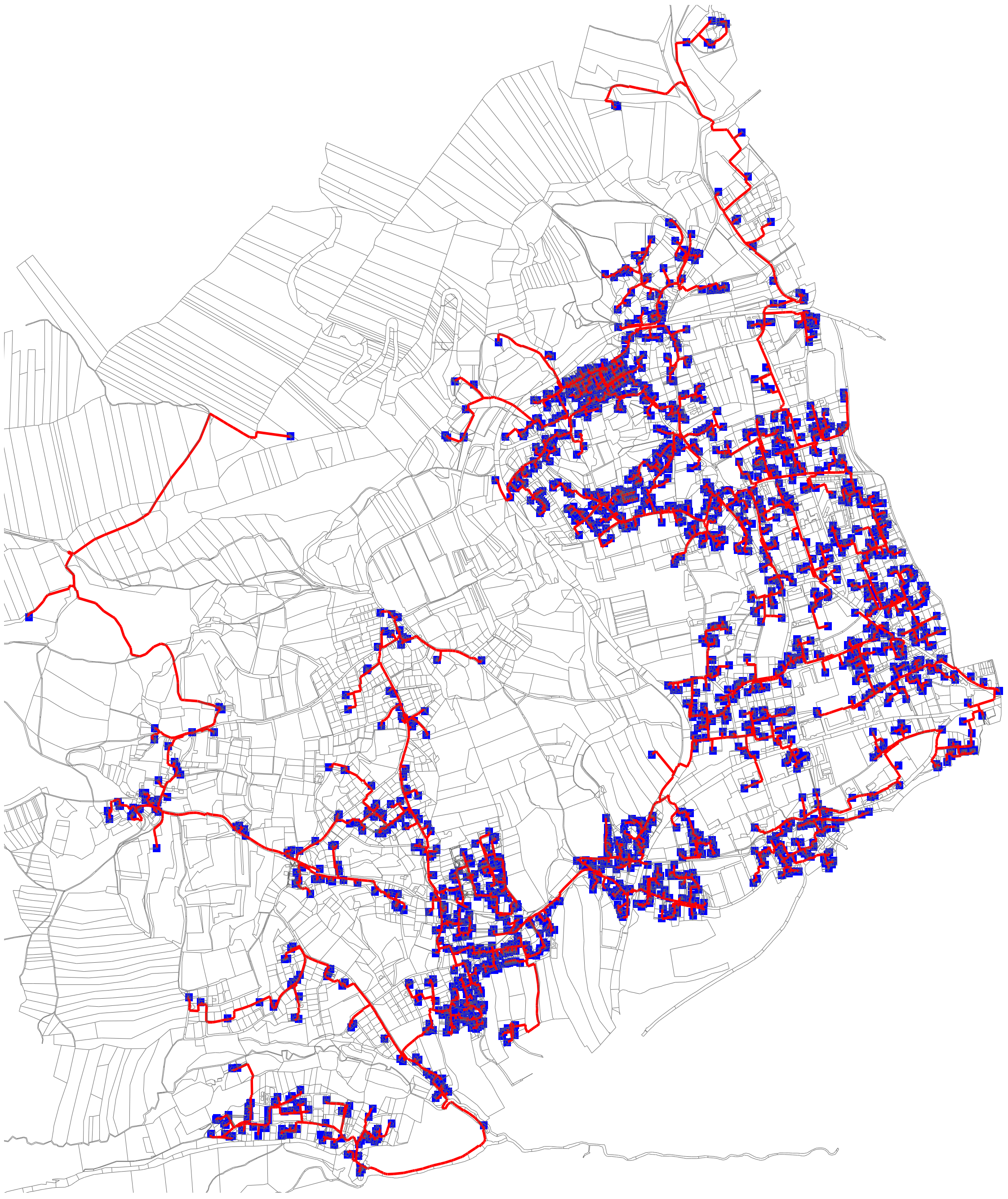


Figure 2: Instance G203

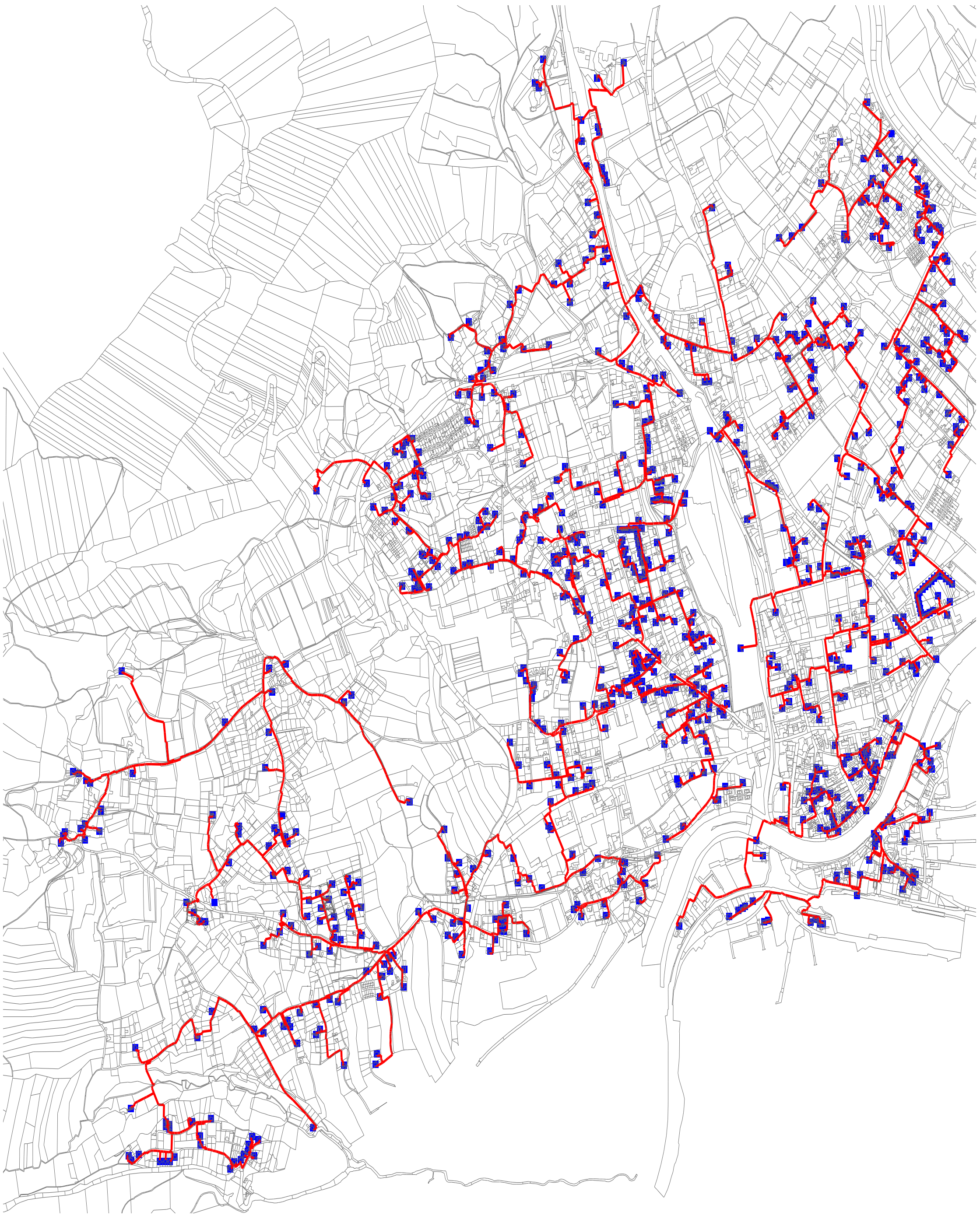


Figure 3: Instance `G309`

Algorithm 1: Preprocessing Procedure, cf. [**bossa**, 13]

```
Data: A weighted graph $G = (V, E, c)$ with terminals $T \subseteq V$.
Result: A reduced instance.
1  repeat
2      NSV
3  until no further reductions possible
4  SDE
5  repeat
6      NTD1, NTD2, TD1, NSV, SDE
7  until no further reductions possible
8  repeat
9      NTD1, NTD2, TD1, SDE, NSV, SDExp
10 until no further reductions possible
```

Koch and Martin's branch-and-cut procedure is based on the well-known directed cut formulation [16], which they extended through so-called flow-balance inequalities (see below). These inequalities were originally considered by Duin [6] and may improve the LP relaxation in some cases. In our approach we use a similar formulation with the exception that we incorporate additional variables for Steiner nodes to facilitate node-oriented branching, which is generally more effective than branching on arc variables [4].

We refer to this *ILP model* as extended directed cut formulation (EDCF). To apply a directed formulation to an undirected problem instance, the original graph $G = (V, E, c)$ has to be transformed into an equivalent directed version $G_D = (V, A, c)$. The arc set A contains two antiparallel arcs for all edges in E and each arc is assigned the same weight as its corresponding edge.

For each arc $(i, j) \in A$, an arc variable x_{ij} denotes membership of the corresponding arc to the Steiner tree ($x_{ij} = 1$) or not ($x_{ij} = 0$). Similarly, additional node variables y_i for $i \in (V \setminus T)$ denote if i is spanned by the Steiner tree ($y_i = 1$) or not ($y_i = 0$). An arbitrary terminal is chosen as root node r. For brevity, we use the following notations: Given a set $W \subset V$, we define $\delta^+(W) = \{(i, j) \in A \mid i \in W \wedge j \in V \setminus W\}$ as the set of all arcs with tail inside W and the head in its complement. Conversely, $\delta^-(W)$ denotes the set of arcs pointing into W from its complement set. For short, if W contains only a single element v, we write $\delta^+(\{v\})$ as $\delta^+(v)$ and $\delta^-(\{v\})$ as $\delta^-(v)$, respectively.

$$\text{(EDCF)} \qquad \min \quad \Big\{ \sum_{(ij)\in A} c_{ij} \cdot x_{ij} \mid (x,y) \in \{0,1\}^{|A|+|V|-|T|}$$

$$x(\delta^-(i)) = 1, \forall i \in T \setminus \{r\}, \quad x(\delta^-(i)) = y_i, \forall i \in V \setminus T \tag{1}$$

$$x(\delta^-(W)) \geq 1, \forall\, W \subset V, r \notin W,\; W \cap T \neq \emptyset \Big\} \tag{2}$$

The objective function minimizes the weight of the selected arcs. Degree constraints (1) ensure that each terminal except the root and all Steiner nodes that are part of the solution have in-degree exactly one. Constraints (2) are directed cut constraints that ensure that there is a directed path between the root and any other terminal node.

The following inequalities are additionally used to initialize the branch-and-cut procedure:

$$x(\delta^+(i)) \geq y_i \qquad \forall i \in V \setminus T \tag{3}$$

$$x_{ij} + x_{ji} \leq y_i \qquad \forall (i,j) \in A, i \in V \setminus T \tag{4}$$

Constraints (3) ensure that Steiner nodes that are part of the solution have at least one outgoing arc (they were referred to as “flow-balance” constraints in the literature). Constraints (4) express that each arc in the solution tree can only be oriented in one way. We also add root in- and out-degree constraints: $x(\delta^+(r)) \geq 1$ and $x(\delta^-(r)) = 0$ (notice that one can alternatively remove root-incoming arcs from the input graph).

The size of the formulation is exponential due to the directed cut constraints (2). Such a formulation can be solved efficiently through the application of the well-known *cutting-plane method*. For description of the general approach the reader is referred to Grötschel, Monma, and Stoer [7]. The cutting-plane method requires a separation method, which decides which inequalities are added to the LP. We implement the separation method in the same manner as in Koch and Martin [9]. The *push-relabel maximum flow* algorithm [3] is applied, which runs in $O(|V|^2 \cdot \sqrt{|E|})$. Nested cuts, back cuts and creep-flow (facilitates the separation of maximum cardinality cuts) are used by default to improve the number and strength of separated inequalities per call. In their approach Koch and Martin restrict themselves to the use of creep-flow, because experiments suggested that nested and back cuts do not provide significant additional improvements when already using creep-fow and flow-balance inequalities. We chose to include nested and back cuts anyway, since for several of the proposed instances a performance improvement could be achieved. This may be attributable to the fact that without nested and back cuts, the solution of the LP relaxation does not change much in each cutting-plane iteration.

Further differences between our and the implementation of Koch and Martin are in the creation of fesible solutions and in the initialization of the cutting plane procedure. To further increase the branch-and-cut procedure’s performance, an initial set of directed cut inequalities is computed heuristically through Wong’s *dual ascent* algorithm [16]. This method is very effective for decreasing runtime, since less cutting-plane iterations are generally necessary to find the optimal solution [1, 11]. Dual ascent also calculates a feasible solution which is used to initialize the upper bounds. During the branch-and-cut procedure feasible solutions are additionally computed using a *primal heuristic*. We apply the improved implementation of the well-known shortest path heuristic [14] as proposed by Aragão and Werneck [2], which achieves a much better average-case runtime than the classic implementation. The heuristic is called after each cutting-plane iteration (instead of every five iterations like in Koch and Martin [9]), since the running time for a single separation iteration can be quite high for large instances. The heuristic is applied to the original undirected graph with adapted edge weights c'_{ij}, which are computed from the current LP solution $(\tilde{x}, \tilde{y})$ as follows:

$$c'_{ij} = c_{ij} \cdot (1 - \max(\tilde{x}_{ij}, \tilde{x}_{ji})) \qquad \forall \{i,j\} \in E$$

5 Results

In this section we first analyze the influence of the described preprocessing procedures on the proposed benchmark sets. Afterwards, computational results of the branch-and-cut procedure as described in Section 4 are given. For the latter we compare the performance of solving instances after simple and advanced preprocessing. The branch-and-cut procedure was implemented in C++ using ILOG CPLEX and Concert Technology 12.5. The code was compiled using gcc 4.8.1 with the `-O4` flag (full optimization). All algorithms were executed on a Sun Grid Engine cluster with 14 Intel Xeon E5540 2.53 GHz with 24 GB RAM and 2 Intel Xeon E5649 2.53 GHz with 60 GB RAM. All given runtimes were measured as real CPU runtime (walltime).

Since in most instances the edge weights range from very small to quite large numbers, CPLEX was configured to solve instances to a gap of 0% (default is 0.01%). Additionally the preprocessing reduction (`CPX_PARAM_REDUCE`) switch was set to primal only, the generation of general purpose cuts was deactivated by setting the `CPX_PARAM_EACHCUTLIM` parameter to `0`. `CPX_PARAM_DIVETYPE` was set to probing, as CPLEX documentation indicated faster results for integer problems. Variable selection (`CPX_PARAM_VARSEL`) was set to strong branching, and branching priorities were set to prefer node variables (y_i). The CPLEX time limit was set to 86 400 seconds (24 hours). All other CPLEX parameters were left at their respective defaults.

5.1 Preprocessing

Tables 2 and 3 list the preprocessing results on the set of `GEO` and `I` instances, respectively. Each entry contains the number of edges, nodes and terminals before and after preprocessing. Additionally, we also show the percentage of remaining edges compared to the original instances after each preprocessing step (denoted by $R[\%]$). Recall that, in contrast to `I` instances, the simple preprocessing had no effect on `GEO` instances, and therefore only the advanced preprocessing was applied to them. We notice that the advanced preprocessing applied to the group `GEO` removes 60 to 80% of all edges. In case of `I` instances, the advanced preprocessing was applied after the graphs were already reduced by the simple preprocessing. We note that the simple preprocessing procedure already manages to eliminate at least ~30% of all edges for each instance of group `I`. The application of advanced preprocessing manages to remove another ~30% of all edges. Regarding the running time, we report it only for the advanced preprocessing – it can be found in Tables 4 and 5. The runtime of simple preprocessing was insignificant in comparison and was thus not documented. A cumulative chart of the advanced preprocessing runtimes is depicted in Figure 4. The time axis is drawn in a logarithmic scale. The line marks the last instance that finished preprocessing within 24 hours. We note that the advanced preprocessing of the larger instances took quite long, e.g. instance I024 finished preprocessing after 5.8 days).

Figure 5 plots the distribution of both terminal percentage and the ratio between nodes and edges before and after advanced preprocessing. The structure plot of the original instances was omitted, since the changes introduced by simple preprocessing are minimal. On the contrary, after advanced preprocessing the distribution indicates that most instances have become more sparse than before. For the `GEO` instances the step of simple preprocessing was skipped there for, no data was available. In case of advanced preprocessing, a clear divergence for the `GEO` instances from the `I` instances can be seen.

Table 2: Results of the preprocessing procedure for the `GEO` instances.

ID	original			advanced preprocessing				
	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	R [%]	time [s]
G101	67 966	82 485	100	10 734	16 345	96	20	13
G102	111 707	160 504	2052	27 896	43 925	2003	27	1003
G103	135 543	201 803	3033	36 270	57 370	2930	28	2580
G104	158 212	240 022	3914	44 251	70 029	3776	29	4307
G105	79 244	101 189	550	14 586	22 450	525	22	72
G106	204 621	318 136	5556	62 618	100 067	5373	31	7401
G107	85 568	114 113	938	15 536	23 858	893	21	359
G201	44 624	56 205	190	8286	12 617	188	22	17
G202	62 174	87 562	1015	14 028	21 610	985	25	933
G203	88 728	133 625	2041	25 651	40 610	1999	30	1784
G204	50 002	65 203	386	9939	15 249	376	23	30
G205	120 866	187 312	3224	37 398	59 323	3146	32	3458
G206	60 446	82 940	803	13 688	21 197	789	26	87
G207	42 481	52 552	97	7565	11 521	98	22	11
G301	80 736	98 750	191	13 291	20 261	181	21	24
G302	117 756	165 153	1879	24 951	38 647	1797	23	2668
G303	147 718	214 176	2992	37 085	57 711	2915	27	2793
G304	86 413	108 872	419	15 213	23 329	403	21	162
G305	172 687	255 825	3902	47 016	73 861	3809	29	4011
G307	235 686	366 093	6313	71 184	113 616	6107	31	21 691
G308	78 834	95 732	88	13 298	20 351	86	21	32
G309	97 928	128 632	902	18 704	28 851	868	22	287

Table 3: Results of the preprocessing procedures for the `I` instances. The column R denotes the percentage of $|E|$ remaining compared to the original instance.

ID	original			simple preprocessing				advanced preprocessing				
	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	R [%]	$\|V\|$	$\|E\|$	$\|T\|$	R [%]	time [s]
`I001`	46 051	64 083	1184	30 190	47 748	1184	75	14 675	22 055	941	34	5851
`I002`	86 009	115 002	1665	49 920	77 871	1665	68	23 800	35 758	1282	31	165 874
`I003`	79 177	109 757	3222	44 482	73 419	3222	67	16 270	23 919	2336	22	197 038
`I004`	9128	12 409	570	5556	8552	570	69	867	1238	263	10	183
`I005`	16 914	22 958	1017	10 284	15 980	1017	70	1677	2430	491	11	484
`I006`	48 804	70 254	2202	31 754	52 875	2202	75	13 339	19 532	1842	28	49 796
`I007`	23 332	32 772	737	15 122	24 371	737	74	6873	10 299	599	31	2979
`I008`	25 130	35 244	871	15 714	25 567	871	73	6522	9629	708	27	2885
`I009`	52 316	71 775	1262	33 188	52 007	1262	72	14 977	22 435	1053	31	29 516
`I010`	65 533	83 865	943	29 905	47 457	943	57	13 041	19 545	782	23	3113
`I011`	45 510	62 188	1428	25 195	41 298	1428	66	9298	13 685	1202	22	2491
`I012`	32 326	40 562	503	12 355	19 962	503	49	3500	5214	387	13	186
`I013`	30 754	41 753	891	18 242	28 976	891	69	7147	10 608	670	25	5512
`I014`	36 097	44 609	475	12 715	20 632	475	46	3577	5311	364	12	35
`I015`	92 217	124 613	2493	48 833	79 987	2493	64	20 573	30 541	2119	25	59 274
`I016`	143 463	187 841	4391	72 038	115 055	4391	61	27 214	39 824	3434	21	472 546
`I017`	25 393	34 679	478	15 095	24 091	478	69	7571	11 571	386	33	484
`I018`	52 889	73 439	1898	31 121	51 113	1898	70	12 258	18 014	1549	25	31 068
`I019`	58 078	74 770	866	25 946	41 645	866	56	11 693	17 624	732	24	1133
`I020`	68 626	83 380	594	21 808	34 921	594	42	6405	9564	508	11	485
`I021`	45 459	55 846	392	16 013	25 269	392	45	5195	7861	295	14	101
`I022`	31 703	41 466	437	16 224	25 691	437	62	8869	13 551	356	33	1100
`I023`	33 382	46 156	582	22 805	35 307	582	76	13 724	20 863	403	45	1628
`I024`	113 054	154 736	3001	68 464	108 732	3001	70	32 357	48 250	2511	31	503 528
`I025`	50 126	65 383	945	23 412	37 952	945	58	10 055	14 961	833	23	1328
`I026`	79 487	111 878	3334	47 429	79 307	3334	71	18 155	26 568	2661	24	291 744
`I027`	169 438	224 904	3954	85 085	138 888	3954	62	40 772	60 555	3490	27	246 569
`I028`	119 785	163 543	1790	72 701	115 430	1790	71	43 690	66 461	1597	41	39 934
`I029`	128 122	171 369	2162	69 988	111 804	2162	65	32 979	49 627	1946	29	62 948
`I030`	80 126	101 802	1263	33 188	53 680	1263	53	12 941	19 279	1093	19	3862
`I031`	110 930	146 104	2182	54 351	88 211	2182	60	21 054	31 410	1832	22	11 549
`I032`	119 110	155 597	3017	56 023	91 399	3017	59	21 345	31 353	2454	20	50 222
`I033`	33 309	44 769	636	18 555	29 730	636	66	8500	12 700	548	28	1511
`I034`	49 017	62 922	735	22 311	35 516	735	56	9128	13 668	606	22	1187
`I035`	72 466	92 967	1704	30 585	50 454	1704	54	13 129	19 420	1428	21	23 569
`I036`	92 336	116 626	1411	37 208	60 356	1411	52	17 036	25 482	1258	22	6664
`I037`	33 711	42 651	427	13 694	22 126	427	52	5886	8869	392	21	460
`I038`	38 081	50 417	967	18 747	30 639	967	61	7733	11 478	798	23	2326
`I039`	18 250	24 156	347	8755	14 449	347	60	3719	5533	306	23	96
`I040`	78 351	104 573	1762	40 389	65 820	1762	63	18 837	28 156	1501	27	35 413
`I041`	107 893	137 798	1193	47 197	75 307	1193	55	22 466	33 868	1014	25	6411
`I042`	98 374	133 196	2171	51 896	85 550	2171	64	23 925	35 806	1923	27	15 693
`I043`	24 460	31 168	367	10 398	16 787	367	54	4511	6740	335	22	214
`I044`	130 289	176 526	3358	68 905	113 889	3358	65	31 500	46 757	2954	26	122 199
`I045`	32 420	41 763	421	14 685	23 466	421	56	6775	10 227	378	24	94
`I046`	144 745	192 528	3598	70 843	117 209	3598	61	32 376	48 054	3154	25	124 694
`I047`	46 509	64 573	2354	28 524	46 251	2354	72	10 622	15 440	1791	24	61 349
`I048`	39 363	48 182	358	13 189	21 219	358	44	4920	7356	320	15	272

Table 3: Results of the preprocessing procedures for the I instances. The column R denotes the percentage of $|E|$ remaining compared to the original instance.

ID	original			simple preprocessing				advanced preprocessing				
	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	R [%]	$\|V\|$	$\|E\|$	$\|T\|$	R [%]	time [s]
I049	79 338	99 310	990	30 857	49 591	990	50	15 045	22 713	821	23	8417
I050	71 355	99 944	2868	43 073	71 276	2868	71	17 787	26 176	2232	26	253 915
I051	48 764	67 543	1524	27 028	45 406	1524	67	12 130	17 892	1337	26	3368
I052	9257	10 789	40	2363	3761	40	35	160	237	23	2	<1
I053	8604	10 807	126	3224	5285	126	49	693	1023	102	9	1
I054	32 788	35 681	38	3803	6213	38	17	540	817	25	2	<1
I055	27 519	36 175	570	13 332	21 580	570	60	4701	6979	483	19	301
I056	7886	9265	51	1991	3176	51	34	290	439	34	5	<1
I057	68 134	90 703	1569	33 231	55 149	1569	61	13 078	19 368	1346	21	14 587
I058	54 221	71 062	1256	23 527	39 628	1256	56	7877	11 657	997	16	852
I059	21 746	27 716	363	9287	14 975	363	54	2800	4157	286	15	86
I060	137 451	165 937	1242	42 008	67 572	1242	41	18 991	28 536	1158	17	19 905
I061	70 170	95 284	1458	39 160	63 659	1458	67	20 958	31 465	1337	33	32 477
I062	155 326	202 462	3343	66 048	110 491	3343	55	23 714	35 305	2812	17	9369
I063	52 176	69 576	1645	26 840	43 661	1645	63	9600	14 042	1291	20	55 795
I064	94 336	138 745	3458	63 158	107 345	3458	77	31 712	46 711	3182	34	332 814
I065	10 200	12 807	144	3898	6356	144	50	1185	1756	119	14	4
I066	59 872	70 249	551	15 038	24 596	551	35	4551	6821	417	10	38
I067	43 552	56 846	627	20 547	33 230	627	58	10 318	15 588	579	27	972
I068	68 863	91 586	1553	33 118	55 127	1553	60	12 191	18 023	1302	20	5992
I069	18 855	25 626	543	9574	16 208	543	63	3508	5156	452	20	843
I070	37 489	47 602	550	15 079	24 608	550	52	6739	10 064	511	21	874
I071	79 580	102 014	1494	33 203	54 427	1494	53	12 772	18 886	1281	19	3843
I072	80 184	98 679	993	26 948	44 194	993	45	11 628	17 411	851	18	721
I073	35 009	48 757	1847	21 653	35 171	1847	72	7510	10 873	1337	22	30 970
I074	29 623	38 611	653	13 316	22 033	653	57	4441	6562	548	17	123
I075	110 782	149 620	2973	57 551	95 381	2973	64	23 195	34 362	2498	23	20 143
I076	31 738	41 032	598	14 023	22 895	598	56	4909	7268	498	18	187
I077	31 318	44 908	1787	20 856	34 237	1787	76	9153	13 363	1490	30	133 537
I078	23 220	32 034	835	13 294	21 948	835	69	5864	8662	692	27	6513
I079	57 402	70 047	565	19 867	31 271	565	45	7933	11 807	497	17	533
I080	47 422	59 412	548	18 695	29 708	548	50	7589	11 256	499	19	712
I081	56 718	73 051	888	25 081	40 739	888	56	10 747	16 029	751	22	1217
I082	41 475	51 351	515	15 592	24 788	515	48	5850	8693	435	17	728
I083	178 810	239 522	4991	89 596	148 583	4991	62	34 221	50 301	4138	21	77 571
I084	96 899	126 877	2319	44 934	73 727	2319	58	17 050	25 201	1918	20	33 793
I085	26 002	31 896	301	9113	14 491	301	45	2780	4123	243	13	80

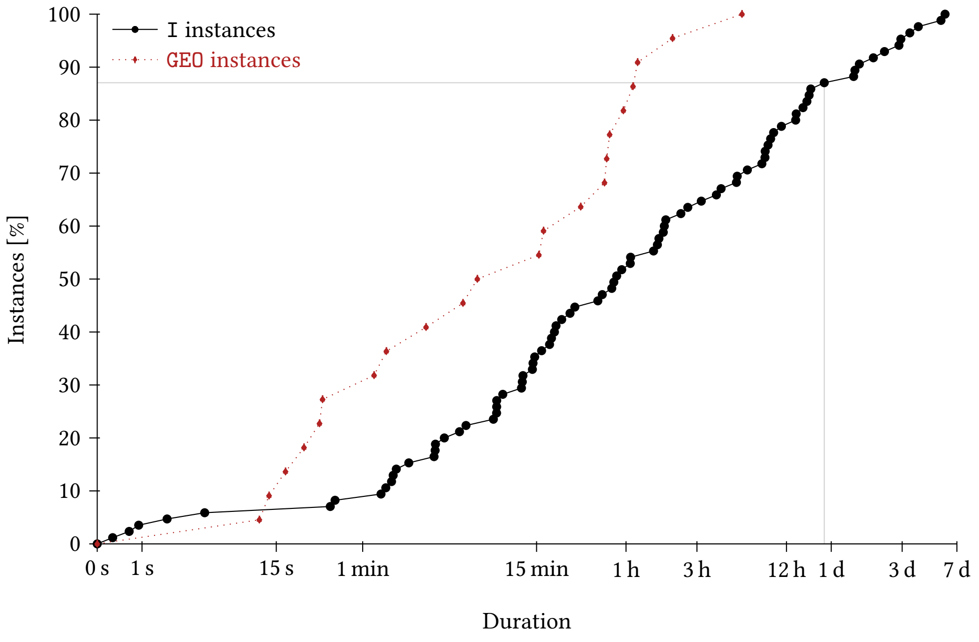


Figure 4: Runtimes of advanced preprocessing as depicted in Table 5.

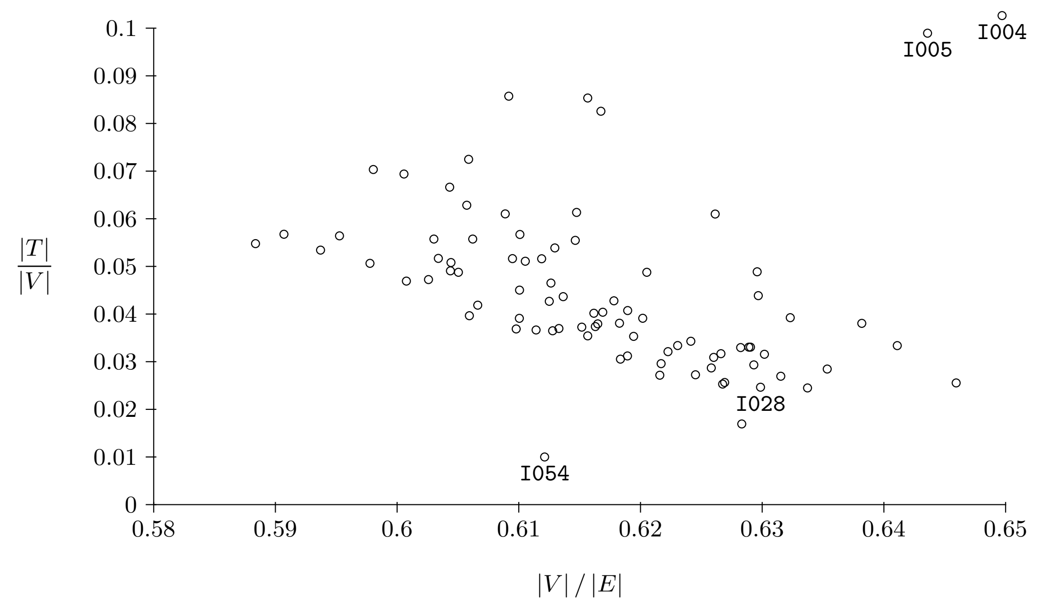
Figure 6 indicates a correlation between the number of nodes and the number of edges in an instance. We note that preprocessing techniques (simple and advanced) increase this correlation. Figure 6(b) shows a more detailed view of this graphic for the advanced preprocessing. In this series of plots no significant difference between the `I` and `GEO` instances can be seen.

5.2 Exact Solution

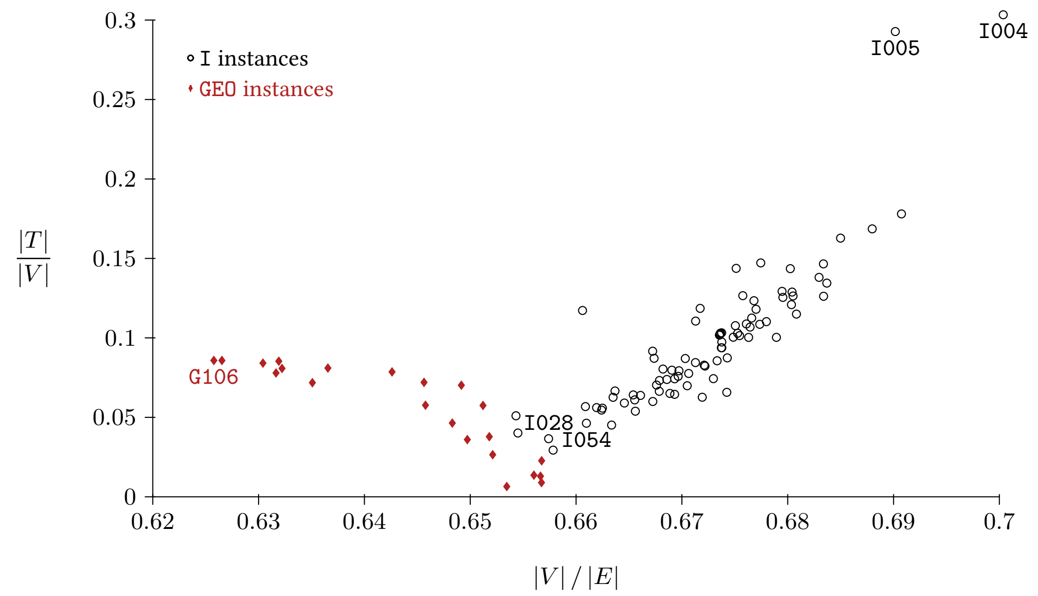
Table 4 provides results for solving `GEO` instances without preprocessing and after advanced preprocessing. Table 5 lists the results for solving `I` instances after simple and advanced preprocessing. The following values are reported: The column "DAgap[%]" gives the gaps obtained by the dual ascent algorithm, which can be seen as a starting point for the ILP (recall that the ILP is initialized by the cutting planes found in the dual ascent and the the first ILP primal solution is the dual ascent feasible solution). The "gap" column lists the relative ILP optimality gaps, which specify the difference between the best found solution (shown in the "objective" column) and the best lower bound (shown in the "lower bound" column).

$$\text{gap} = \frac{\text{objective value} - \text{lower bound}}{\text{lower bound}}$$

For the instances with advanced preprocessing the gaps have been scaled to include the preprocessing offset so that they can be compared more easily with the simple preprocessed ones. Finally, the column "time[s]" shows the running time (in seconds) of the exact approach (TL stands for "the time limit reached", which was set to 24 hours per instance).

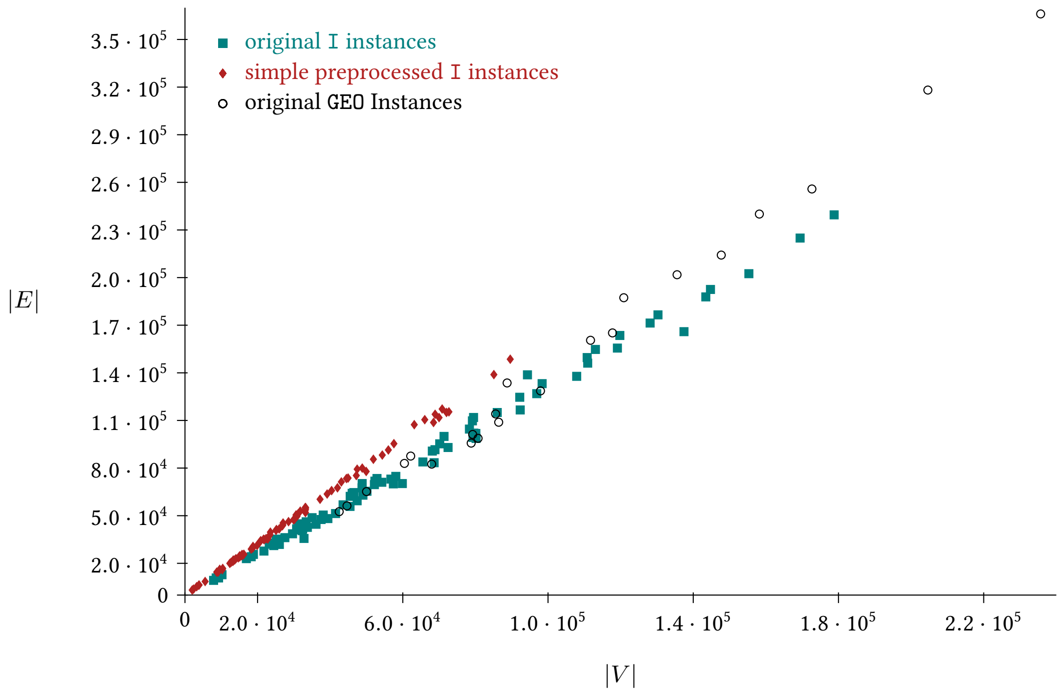


(a) After simple preprocessing for `I` instances

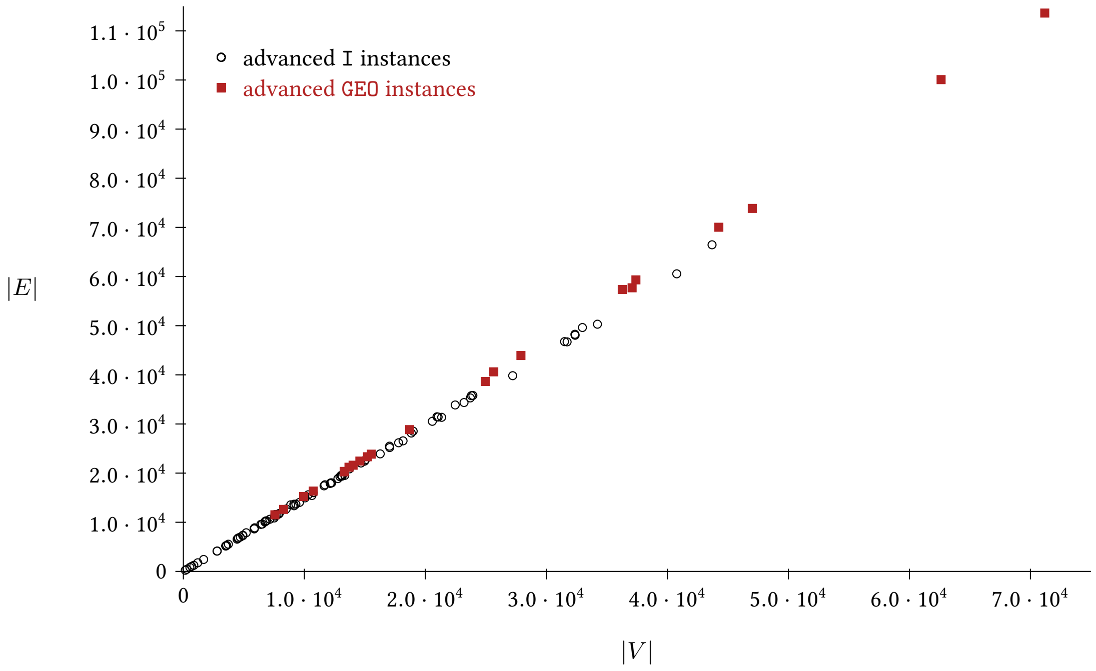


(b) After advanced preprocessing, for `GEO` and `I` instances

Figure 5: Scatter plot of the node-edge ratio and the terminal-node ratio. Some outlier instances from (a) are marked to illustrate the changes after preprocessing.



(a) Comparison of original and simple preprocessed `I` instances and `GEO` instances



(b) Comparison of after advanced preprocessing of `GEO` and `I` instances

Figure 6: Scatter plots showing the relation of nodes to edges.

In Figure 7 the cumulative runtimes from Table 5 are plotted. The graph reveals that the runtime improvement gained from preprocessing for branch-and-cut increases heavily with instance size. Almost all instances could be solved to optimality within one day after advanced preprocessing. If the preprocessing duration is added, the set of solved instances is still ~85% within this time limit. On the contrary, only ~65% of all instances could be solved in one day without advanced preprocessing. Table 4 reveals that only two unpreprocessed instances from the `GEO` group could be solved to optimality within 24 hours, and therefore, we do not show a similar chart for this group.

In general, without advanced preprocessing, a large number of instances could not be solved to optimality within one day. Figure 8 shows the relative ILP gaps (as reported by CPLEX) of the unsolved instances. Note that we did not include data for the `I` instances after advanced preprocessing since the unsolved instances in this case where few and the gaps small.

Table 4: Results of the branch-and-cut procedure for the `GEO` instances.

ID	original instances					instances with advanced preprocessing				
	DAgap[%]	gap [%]	lower bound	objective	time [s]	DAgap[%]	gap [%]	lower bound	objective	time [s]
G101	4.8084	4.8084	3 439 226	3 604 599	TL	3.2171	0	3 492 405	3 492 405	11 887
G102	4.6487	1.1388	15 132 879	15 305 206	TL	3.9544	0.1421	15 184 047	15 205 628	TL
G103	4.4554	1.0418	19 857 612	20 064 484	TL	3.7510	0.0806	19 927 745	19 943 807	TL
G104	4.3463	4.3463	25 847 585	26 971 006	TL	3.6252	0.1606	26 155 589	26 197 583	TL
G105	4.1508	4.1508	12 362 889	12 876 053	TL	3.4270	0	12 507 877	12 507 877	73 229
G106	3.8048	3.8048	44 062 993	45 739 520	TL	3.2867	0.6150	44 458 569	44 731 984	TL
G107	4.7539	0.6993	7 309 295	7 360 406	TL	3.6783	0	7 325 530	7 325 530	7815
G201	3.8601	0.2380	3 481 975	3 490 260	TL	4.3274	0	3 484 028	3 484 028	1369
G202	4.2548	0.0433	6 849 281	6 852 245	TL	3.1465	0	6 849 423	6 849 423	1191
G203	4.2008	0.9392	13 107 861	13 230 972	TL	3.6252	0	13 155 210	13 155 210	23 703
G204	4.1588	0	5 313 548	5 313 548	34 256	3.2254	0	5 313 548	5 313 548	1057
G205	3.8414	3.8414	24 534 820	25 477 296	TL	3.1937	0.3579	24 792 524	24 881 257	TL
G206	4.2856	0.3279	9 166 968	9 197 029	TL	3.8149	0	9 175 622	9 175 622	2296
G207	3.0872	0	2 265 834	2 265 834	50 754	1.9700	0	2 265 834	2 265 834	1028
G301	4.4834	4.4834	4 736 298	4 948 643	TL	3.9353	0	4 797 441	4 797 441	13 252
G302	4.7381	1.1628	13 243 377	13 397 374	TL	3.4634	0	13 300 990	13 300 990	31 291
G303	3.7310	3.7310	27 645 432	28 676 881	TL	3.2469	0.0027	27 941 035	27 941 801	TL
G304	4.0303	4.0303	6 629 770	6 896 969	TL	3.7322	0	6 721 180	6 721 180	30 326
G305	3.6847	3.6847	40 198 331	41 679 517	TL	3.1814	0.0777	40 615 001	40 646 576	TL
G307	3.8089	3.8089	50 652 541	52 581 831	TL	3.2236	0.5967	51 117 755	51 422 797	TL
G308	4.3545	4.3545	4 634 667	4 836 484	TL	4.2217	0	4 699 474	4 699 474	55 315
G309	3.6466	3.6466	11 143 170	11 549 514	TL	3.4066	0	11 256 303	11 256 303	36 578

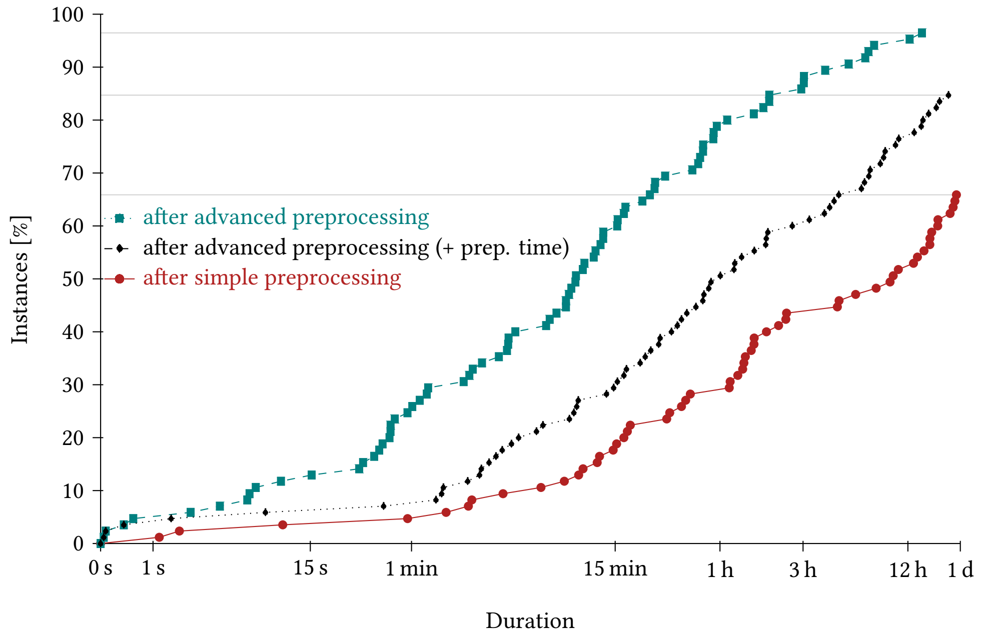


Figure 7: Runtimes of `I` Instances as depicted in Table 5.

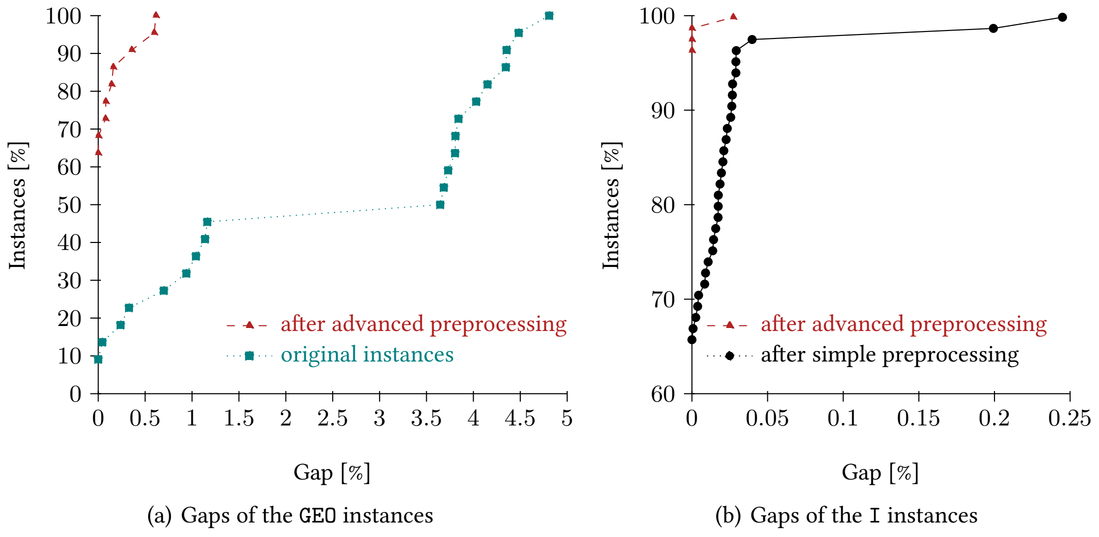


Figure 8: ILP gaps before and after preprocessing for `GEO` and `I` instances as depicted in Tables 4 and 5 (time limit of 24 hrs)

Table 5: Results of the branch-and-cut procedure, for the `I` instances

ID	instances with simple preprocessing					instances with advanced preprocessing				
	DAgap[%]	gap [%]	lower bound	objective	time [s]	DAgap[%]	gap [%]	lower bound	objective	time [s]
I001	0.1654	0	253 921 201	253 921 201	8474	0.0593	0	253 921 201	253 921 201	909
I002	0.1993	0.1993	399 673 678	400 470 203	TL	0.0643	0	399 809 303	399 809 303	6885
I003	0.3985	0.0038	788 767 152	788 796 834	TL	0.0484	0	788 774 494	788 774 494	3295
I004	0.5017	0	279 512 692	279 512 692	125	0.0980	0	279 512 692	279 512 692	2
I005	0.4606	0	390 876 350	390 876 350	577	0.1839	0	390 876 350	390 876 350	6
I006	0.1228	0	504 526 035	504 526 035	59 133	0.0534	0	504 526 035	504 526 035	1739
I007	0.1627	0	177 909 660	177 909 660	1830	0.0552	0	177 909 660	177 909 660	217
I008	0.1748	0	201 788 202	201 788 202	2411	0.0484	0	201 788 202	201 788 202	218
I009	0.2069	0	275 558 727	275 558 727	46 354	0.0953	0	275 558 727	275 558 727	480
I010	0.1965	0	207 889 674	207 889 674	64 207	0.0787	0	207 889 674	207 889 674	501
I011	0.1876	0	317 589 880	317 589 880	4837	0.0655	0	317 589 880	317 589 880	377
I012	0.3891	0	118 893 243	118 893 243	547	0.0968	0	118 893 243	118 893 243	14
I013	0.1600	0	193 190 339	193 190 339	7758	0.0560	0	193 190 339	193 190 339	153
I014	0.3561	0	105 173 465	105 173 465	454	0.0917	0	105 173 465	105 173 465	6
I015	0.2404	0.0195	592 199 698	592 315 122	TL	0.0925	0	592 240 832	592 240 832	6383
I016	0.2017	0.0174	1 110 829 727	1 111 023 215	TL	0.0786	0	1 110 914 623	1 110 914 623	25 647
I017	0.2099	0	109 739 695	109 739 695	703	0.0677	0	109 739 695	109 739 695	53
I018	0.1639	0	463 887 832	463 887 832	28 352	0.0548	0	463 887 832	463 887 832	923
I019	0.3357	0.0137	217 631 791	217 661 665	TL	0.1095	0	217 647 693	217 647 693	1026
I020	0.3100	0	146 515 460	146 515 460	5022	0.1027	0	146 515 460	146 515 460	66
I021	0.2945	0	106 470 644	106 470 644	4109	0.1036	0	106 470 644	106 470 644	47
I022	0.2251	0	106 799 980	106 799 980	2270	0.0687	0	106 799 980	106 799 980	233
I023	0.2519	0	131 044 872	131 044 872	5578	0.0565	0	131 044 872	131 044 872	677
I024	0.1690	0.0090	758 461 284	758 529 425	TL	0.0676	0	758 483 415	758 483 415	44 124
I025	0.4678	0	232 790 758	232 790 758	48 985	0.2180	0	232 790 758	232 790 758	1289
I026	0.2070	0.0084	927 995 642	928 073 794	TL	0.0565	0	928 032 223	928 032 223	3955
I027	0.2471	0.0256	976 718 597	976 968 933	TL	0.0938	0.0000	976 811 902	976 814 293	TL
I028	0.2450	0.2450	383 904 288	384 844 706	TL	0.0963	0.0273	374 103 584	384 324 703	TL
I029	0.3059	0.0173	492 167 713	492 252 699	TL	0.1247	0	492 193 565	492 193 565	10 876
I030	0.4950	0	321 646 787	321 646 787	63 989	0.1585	0	321 646 787	321 646 787	467
I031	0.3144	0.0234	578 237 328	578 372 425	TL	0.1719	0	578 284 709	578 284 709	2759
I032	0.1316	0.0107	773 064 822	773 147 670	TL	0.0561	0	773 096 651	773 096 651	2493
I033	0.1711	0	134 461 857	134 461 857	1735	0.0780	0	134 461 857	134 461 857	118
I034	0.3018	0	165 115 148	165 115 148	80 365	0.0894	0	165 115 148	165 115 148	410
I035	0.1813	0	414 440 370	414 440 370	34 056	0.0704	0	414 440 370	414 440 370	767
I036	0.3184	0.0211	375 236 129	375 315 190	TL	0.1328	0	375 260 864	375 260 864	2696
I037	0.4207	0	105 720 727	105 720 727	5427	0.1292	0	105 720 727	105 720 727	59
I038	0.2291	0	255 767 543	255 767 543	8669	0.0746	0	255 767 543	255 767 543	584
I039	0.2572	0	85 566 290	85 566 290	331	0.0726	0	85 566 290	85 566 290	30
I040	0.3016	0.0263	431 468 812	431 582 451	TL	0.0925	0	431 498 867	431 498 867	2874
I041	0.3686	0.0291	301 879 284	301 967 000	TL	0.1350	0	301 914 840	301 914 840	10 892
I042	0.2338	0.0267	532 079 428	532 221 597	TL	0.0985	0	532 131 412	532 131 412	10 532
I043	0.2555	0	95 722 094	95 722 094	1002	0.1358	0	95 722 094	95 722 094	35
I044	0.2372	0.0143	804 487 839	804 602 796	TL	0.1013	0	804 532 332	804 532 332	24 578
I045	0.2889	0	105 944 062	105 944 062	4047	0.1175	0	105 944 062	105 944 062	73
I046	0.2455	0.0293	925 400 944	925 672 015	TL	0.1415	0	925 470 052	925 470 052	52 020
I047	0.1963	0.0007	695 159 075	695 164 265	TL	0.0597	0	695 163 406	695 163 406	1524
I048	0.2728	0	91 509 264	91 509 264	4547	0.1486	0	91 509 264	91 509 264	44
I049	0.3290	0.0398	294 771 429	294 888 690	TL	0.1327	0	294 811 505	294 811 505	3302
I050	0.1689	0.0174	792 559 129	792 696 807	TL	0.0661	0	792 599 114	792 599 114	19 726
I051	0.1463	0	357 230 839	357 230 839	75 456	0.0582	0	357 230 839	357 230 839	1420
I052	0.1188	0	13 309 487	13 309 487	1	0.0197	0	13 309 487	13 309 487	<1
I053	0.1544	0	30 854 904	30 854 904	10	0.0863	0	30 854 904	30 854 904	<1
I054	1.1682	0	15 841 596	15 841 596	94	0.6760	0	15 841 596	15 841 596	<1
I055	0.1552	0	144 164 924	144 164 924	912	0.0498	0	144 164 924	144 164 924	44
I056	0.2039	0	14 171 206	14 171 206	1	0.0442	0	14 171 206	14 171 206	<1

Table 5: Results of the branch-and-cut procedure, for the I instances

ID	instances with simple preprocessing					instances with advanced preprocessing				
	DAgap[%]	gap [%]	lower bound	objective	time [s]	DAgap[%]	gap [%]	lower bound	objective	time [s]
I057	0.2014	0	412 746 415	412 746 415	78 499	0.0646	0	412 746 415	412 746 415	764
I058	0.2829	0	305 024 188	305 024 188	6638	0.0788	0	305 024 188	305 024 188	191
I059	0.1968	0	107 617 854	107 617 854	130	0.0397	0	107 617 854	107 617 854	6
I060	0.4419	0.0044	337 277 249	337 292 055	TL	0.3193	0	337 290 460	337 290 460	3441
I061	0.1894	0.0269	363 005 993	363 103 538	TL	0.0831	0	363 042 722	363 042 722	14 454
I062	0.3359	0.0226	792 875 218	793 054 349	TL	0.1416	0	792 941 137	792 941 137	6898
I063	0.2691	0	459 801 704	459 801 704	57 789	0.1252	0	459 801 704	459 801 704	1008
I064	0.1575	0.0205	863 036 549	863 213 284	TL	0.0701	0.0000	863 103 171	863 104 019	TL
I065	0.2297	0	32 965 718	32 965 718	55	0.0712	0	32 965 718	32 965 718	4
I066	0.2602	0	174 219 813	174 219 813	2153	0.1377	0	174 219 813	174 219 813	45
I067	0.2214	0	175 540 750	175 540 750	57 694	0.1142	0	175 540 750	175 540 750	462
I068	0.2179	0	420 730 046	420 730 046	16 961	0.0751	0	420 730 046	420 730 046	597
I069	0.1882	0	135 161 583	135 161 583	866	0.0487	0	135 161 583	135 161 583	38
I070	0.2802	0	136 700 139	136 700 139	5639	0.1017	0	136 700 139	136 700 139	135
I071	0.1891	0	382 539 099	382 539 099	38 045	0.0741	0	382 539 099	382 539 099	533
I072	0.2683	0	289 019 226	289 019 226	82 039	0.0909	0	289 019 226	289 019 226	528
I073	0.2313	0	663 004 987	663 004 987	17 297	0.0486	0	663 004 987	663 004 987	359
I074	0.3288	0	165 573 383	165 573 383	725	0.1134	0	165 573 383	165 573 383	29
I075	0.2546	0.0186	815 360 214	815 511 714	TL	0.0814	0	815 404 026	815 404 026	5622
I076	0.3925	0	166 249 692	166 249 692	1091	0.0801	0	166 249 692	166 249 692	40
I077	0.1719	0	472 503 150	472 503 150	21 580	0.1124	0	472 503 150	472 503 150	740
I078	0.1544	0	185 525 490	185 525 490	1049	0.0562	0	185 525 490	185 525 490	129
I079	0.4900	0.0025	150 506 371	150 510 132	TL	0.2471	0	150 506 933	150 506 933	1514
I080	0.4339	0	164 299 652	164 299 652	53 423	0.1349	0	164 299 652	164 299 652	213
I081	0.3561	0	247 527 679	247 527 679	35 565	0.1270	0	247 527 679	247 527 679	674
I082	0.3735	0	147 407 632	147 407 632	4923	0.1481	0	147 407 632	147 407 632	73
I083	0.2787	0.0291	1 405 421 745	1 405 830 098	TL	0.1217	0	1 405 593 856	1 405 593 856	27 622
I084	0.3082	0.0157	627 148 556	627 246 982	TL	0.1065	0	627 187 559	627 187 559	2876
I085	0.2585	0	80 628 079	80 628 079	199	0.1053	0	80 628 079	80 628 079	10

6 Conclusion

We presented two new sets of STP benchmark instances that were created from real-world fiber optic network design problems. Our experiments indicate that the application of preprocessing requires a large amount of time on such huge graphs. However, results suggest that the effort pays off and the application of preprocessing prior to exact methods is recommendable. It remains an open question how other solution approaches would perform on this instance set. We expect that the results presented in this work can further be improved by applying more sophisticated and effective methods as e.g. the framework by Polzin [11] and Daneshmand [5].

References

[1] Aragão, M. P. de, Uchoa, E. and Werneck, R. F. F. Dual Heuristics on the Exact Solution of Large Steiner Problems. *Electronic Notes in Discrete Mathematics* 7 (2001), 150–153. doi: 10.1016/S1571-0653(04)00247-1.

[2] Aragão, M. P. de and Werneck, R. F. F. On the implementation of MST-based heuristics for the Steiner problem in graphs. In: *ALENEX*. Ed. by Mount, D. M. and Stein, C. Vol. 2409. Lecture Notes in Computer Science. Springer, 2002, 1–15. isbn: 3-540-43977-3. doi: 10.1007/3-540-45643-0_1.

[3] Cherkassky, B. V. and Goldberg, A. V. On implementing push-relabel method for the maximum flow problem. Lecture Notes in Computer Science 920 (1995). Ed. by Balas, E. and Clausen, J., 157–171. doi: 10.1007/3-540-59408-6_49.

[4] Cronholm, W., Ajili, F. and Panagiotidi, S. On the minimal Steiner tree subproblem and its application in branch-and-price. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, 2005, 125–139.

[5] Daneshmand, S. V. Algorithmic Approaches to the Steiner Problem in Networks. PhD thesis. University of Mannheim, 2003.

[6] Duin, C. W. Steiner's problem in graphs. PhD thesis. University of Amsterdam, 1993.

[7] Grötschel, M., Monma, C. L. and Stoer, M. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research* 40, 2 (1992), 309–330. doi: 10.1287/opre.40.2.309.

[8] Karp, R. M. Reducibility among combinatorial problems. In: *Complexity of Computer Computations*. Ed. by Miller, R. E. and Thatcher, J. W. Plenum Press, New York, 1972, 85–103.

[9] Koch, T. and Martin, A. Solving Steiner tree problems in graphs to optimality. *Networks* 32, 3 (1998), 207–232. doi: 10.1002/(SICI)1097-0037(199810)32:3<207::AID-NET5>3.0.CO;2-O.

[10] Koch, T., Martin, A. and Voß, S. *SteinLib: An updated library on Steiner tree problems in graphs*. Tech. rep. 00-37. Takustr.7, 14195 Berlin: ZIB, 2000.

[11] Polzin, T. Algorithms for the Steiner problem in networks. PhD thesis. Saarland University, 2004.

[12] Prossegger, M. Generation of a Weighted Network Graph based-on Hybrid Spatial Data. In: *Proceedings of The Fifth International Conference on Advanced Geographic Information Systems, Applications, and Services, GEOProcessing, Nice, France*. ISBN: 978-1-61208-251-6. 2013, 120–124.

[13] Ribeiro, C. C., Uchoa, E. and Werneck, R. F. F. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing* 14, 3 (2002), 228–246. doi: 10.1287/ijoc.14.3.228.116.

[14] Takahashi, H. and Matsuyama, A. An approximate solution for the Steiner problem in graphs. *Math. Japonica* 24, 6 (1980), 573–577.

[15] Uchoa, E., Aragão, M. P. de and Ribeiro, C. C. Preprocessing Steiner problems from VLSI layout. *Networks* 40, 1 (2002), 38–50. doi: 10.1002/net.10035.

[16] Wong, R. T. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming* 28, 3 (1984), 271–287. issn: 0025-5610. doi: 10.1007/BF02612335.