Noname manuscript No.
(will be inserted by the editor)

SCIP-Jack – A solver for STP and variants with parallelization extensions

**Gerald Gamrath · Thorsten Koch ·
Stephen J. Maher · Daniel Rehfeldt ·
Yuji Shinano**

the date of receipt and acceptance should be inserted later

Abstract The Steiner tree problem in graphs is a classical problem that commonly arises in practical applications as one of many variants. While often a strong relationship between different Steiner tree problem variants can be observed, solution approaches employed so far have been prevalently problem-specific. In contrast, this paper introduces a general-purpose solver that can be used to solve both the classical Steiner tree problem and many of its variants without modification. This versatility is achieved by transforming various problem variants into a general form and solving them by using a state-of-the-art MIP-framework. The result is a high-performance solver that can be employed in massively parallel environments and is capable of solving previously unsolved instances.

1 Introduction

The *Steiner tree problem in graphs* (STP) is one of the classical $\mathcal{NP}$-hard problems [1]. Given an undirected connected graph $G = (V, E)$, costs $c : E \rightarrow \mathbb{Q}_{\geq 0}$ and a set $T \subseteq V$ of *terminals*, the problem is to find a tree $S \subseteq G$ of minimum cost that spans T.

Practical applications of the STP can be found for instance in the design of fiber-optic networks [2]. However, it is more common that practical applications are formulated as a particular variant of the STP [3,4,5,6].

The announcement of the 11th DIMACS Challenge initiated our work with an investigation into the STP solver Jack-III, described in [7]. The model and code of Jack-III provided a base for the development of a general STP solver—being able to solve many of the problem variants. However, Jack-III is more than 15 years old. As such, many modern developments regarding STP solution methods and MIP solving techniques are not available. Our approach

Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany,
E-mail: {gamrath, koch, maher, rehfeldt, shinano}@zib.de

to address this limitation of Jack-III includes the combination of the model used in [7] with the start-of-the-art MIP-framework SCIP [8,9]. Employing SCIP naturally facilitated the incorporation of many algorithm developments from the past two decades and provided a platform for the development of new methods.

A major contribution of this paper is the development of a general Steiner tree problem solver. This achievement stands in contrast to the many problem-specific solvers observed within the literature. Furthermore, SCIP provides a massively parallel MIP-framework that is employed with this general solver. Thereupon, bolstered by algorithmic improvements, the developed solver is able to solve several previously unsolved benchmark instances. Detailing the approach delineated above, the remainder of this paper will be structured as follows:

- Section 2 demonstrates the impact of transitioning from a simple, ad hoc created branch-and-cut code to the use of a full fledged, state-of-the-art MIP-framework.
- Section 3 shows how to employ the versatility of MIP models to not only solve a whole class of related problem variants, but—in combination with further algorithmic advances—be competitive with or even superior to problem-specific state-of-the-art solvers.
- Finally, in Section 4 the potential from using hundreds of CPU cores to solve a single problem is illustrated.

The results achieved in this paper demonstrate the value of revisiting topics after some time. In our case this occurred in two steps: First, prior to the DIMACS Challenge, with the developments delineated above, and second, after the completion of the Challenge, when further algorithmic methods were devised and implemented to considerably enhance the performance of SCIP-Jack. Further examples of revisiting research topics can be found in [10,11].

In general, it can be stated that a branch-and-cut based Steiner tree solver has three major components. First, preprocessing is extremely important. Apart from some instances either specifically constructed or insightfully hand-picked to defy presolving techniques, such as the PUC [12] and I640 [13] test sets, preprocessing is often able to significantly reduce instances. Results presented in the PhD theses of Polzin [14] and Daneshmand [15] report an average reduction in the number of edges of 78 %, with many instances being solved completely by presolving. In computational experiments performed for this paper, reduction rates of more than 90 % for some Steiner problem variants (e.g., for the maximum-weight connected subgraph problem) are obtained.

Second, heuristics are needed to find good or even optimal solutions and help find strong upper and lower bounds quickly. In our experiments, for more than 90 % of the instances that were not already solved during preprocessing the final solution was found by a heuristic. Furthermore, heuristics can be especially important for hard instances, for which the dual bound often stays substantially below the optimum for a long time.

Finally, the core of the approach is constituted by the branch-and-cut procedure used to compute lower bounds and prove optimality. The results of [14] show that many STP instances can already be solved by reduction- and heuristic-based approaches [14]. However, the failure of the state-of-the-art solver described in [14] to solve a number of hard instances that defy preprocessing highlights the importance of strong branch-and-cut procedures.

2 From simple hand tailored to off-the-shelf state-of-the-art

The model employed in the solver SCIP-Jack uses the *flow-balance directed cut formulation* described in [7]. This formulation provides a tight linear programming (LP) relaxation. It is built upon the directed equivalent of the STP, the *Steiner arborescence problem* (*SAP*): Given a directed graph $D = (V, A)$, a root $r \in V$, costs $c : A \to \mathbb{Q}_{\geq 0}$ and a set $T \subseteq V$ of terminals, a directed tree $(V_S, A_S) \subseteq D$ of minimum cost is required such that for all $t \in T$, (V_S, A_S) contains exactly one directed path from r to t. Each STP can be transformed to an SAP by replacing each edge with two anti-parallel arcs of the same cost and distinguishing an arbitrary terminal as the root. This procedure results in a one-to-one correspondence between the respective solution sets, see [16] for a proof.

An integer program for the SAP can be obtained by introducing a variable y_a for each arc $a \in A$ with the interpretation $y_a = 1$ if a is in the Steiner arborescence, and $y_a = 0$ otherwise. These considerations set the stage for the following formulation:

Formulation 1 *Flow Balance Directed Cut Formulation*

$$\begin{aligned}
&\min c^T y && && (1)\\
&y(\delta^+(W)) &\geq\ & 1, && \textit{for all } W \subset V, r \in W, (V \setminus W) \cap T \neq \emptyset (2)\\
&y(\delta^-(v)) &&\begin{cases} = & 0 \ \textit{if } v = r, \\ = & 1 \ \textit{if } v \in T \setminus r, \textit{for all } v \in V \\ \leq & 1 \ \textit{if } v \in N, \end{cases} && (3)\\
&y(\delta^-(v)) &\leq\ & y(\delta^+(v)), && \textit{for all } v \in N \quad (4)\\
&y(\delta^-(v)) &\geq\ & y_a, && \textit{for all } a \in \delta^+(v), v \in N \quad (5)\\
&0 \leq y_a &\leq\ & 1, && \textit{for all } a \in A \quad (6)\\
&y_a &\in\ & \{0, 1\}, && \textit{for all } a \in A \quad (7)
\end{aligned}$$

where $N = V \backslash T$, $\delta^+(X) := \{(u, v) \in A | u \in X, v \in V \backslash X\}$, $\delta^-(X) := \delta^+(V \backslash X)$ for $X \subseteq V$; i.e., $\delta^+(X)$ is the set of all arcs going out of, and $\delta^-(X)$ the set of all arcs going into X.

Constraints (4) strengthen the LP-relaxation of Formulation 1, see [13]. However, the remaining additional constraints (3), (5), and (6) do not improve the value of the LP-relaxation; although they nevertheless lead to an empirical speed-up in practical solving [14]. Further details of Formulation 1 are given in [7].

Since the model potentially contains an exponential number of constraints a separation routine is employed. Violated constraints, are separated during the execution of the branch-and-cut algorithm. Jack-III employed this problem formulation along with a model-specific branch-and-bound search. Strong branching [17] was used with a depth-first search node selection.

The implementation of SCIP-Jack is based on the academic MIP solver SCIP [8,9]. Besides being one of the fastest non-commercial MIP solvers [18], SCIP is a general branch-and-cut framework. The plugin-based design of SCIP provides a simple method of extension to handle a variety of specific problem classes.

In the case of SCIP-Jack, the first plugins implemented were a *reader* to read problem instances and *problem data* to store the graph and build the model within SCIP. Within these plugins it was possible to re-use the reading methods and data structures of Jack-III. However, each of these had to be extended as part of the implementation in SCIP-Jack. The heart of the new implementation is a *constraint handler* that checks solutions for feasibility and separates any violated model constraints. Again, separation methods of the more than 15-year old code are re-used in SCIP-Jack, while SCIP provides a filtering of cuts to improve numerical stability and dynamic aging of the generated cuts. Additionally, the general-purpose separation methods that exist within SCIP are used, which include Gomory and mixed-integer rounding cuts.

Jack-III includes many STP-specific preprocessing techniques, as described in [7]. However, for SCIP-Jack only the Degree-Test (DT) [19] method has been reused. All other tests were replaced by more efficient variants, which have emerged in the decade following the release of Jack-III, cf. [14]. Moreover, after the DIMACS Challenge work on reduction techniques continued and various new reduction methods were developed for several of the Steiner problem variants described in this paper. They are a pivotal factor in the improved performance of SCIP-Jack as compared to its predecessor participating in the Challenge—the main motivation behind the development of the new methods was to enhance SCIP-Jack [16]. Due to the large number of presolving techniques and their complexity it is not possible to provide individual descriptions within the frame of this paper. The reader is referred to [16] for detailed information. The preprocessing techniques implemented in SCIP-Jack are listed in Table 2 according to the abbreviations used in [16]; the full names of the preprocessing techniques can be found in Section B of the appendix. Supplementary to the presolving techniques, a Steiner problem specific propagator is implemented that fixes edges during the branch-and-cut according to the same criteria used in the dual-ascent (DA) reduction method [13,14].

For the branch-and-bound search a straightforward STP-specific branching-rule has been implemented. Specifically, instead of branching on variables, i.e., in the case of the STP on arcs, vertex branching [20] is employed. This has been identified empirically by the authors of this publication to be stronger than the generic branching rules natively implemented in SCIP. During the branch-and-bound procedure, vertex branching selects a Steiner vertex to be rendered a terminal in one child node and excluded in the second child.

Determining such a Steiner vertex is achieved by means of the following criterion. Let $y \in [0,1]^A$ be an LP solution at the current node during branch-and-cut. Select a vertex $v_i \in V \setminus T$ to branch upon, such that

$$\left| \sum_{a \in \delta^-(v_i)} y_a - 0.5 \right| \tag{8}$$

is minimal among all Steiner vertices.

The node selection is organized by SCIP and is performed with respect to a best estimate criterion—interleaved with best bound and depth-first search phases [21].

One dual and several primal STP-specific heuristics have been implemented in SCIP-Jack—the dual-ascent heuristic (DA), the repetitive shortest path heuristic (RSPH), in the form proposed in [22], an improvement heuristic (VQ) [23], the reduction-based heuristics prune (P) and ascend-and-prune (AP) [14], and a new recombination heuristic (RC).

The dual-ascent algorithm was introduced in [24]. It exhibits a time complexity of $O(|E| \min\{|V||T|, |E|\})$, see [14], but is usually faster than this bound might suggest; efficient implementations can be found in [13] and [25]. In SCIP-Jack the implementation of [25] is used. At termination, dual-ascent provides a dual solution to a reduced version of Formulation 1 that contains only the constraints (2) and (6). This solution involves directed paths along arcs of reduced cost 0 from the root to each other terminal. The heuristic is executed prior to the branch-and-cut procedure and includes all cuts corresponding to the dual solution found by DA. Due to strong duality, the objective value of the first LP solved during branch-and-cut corresponds to the objective value of the dual solution found by DA.

On the primal side, SCIP-Jack includes the well-known repetitive shortest paths heuristic. Starting with a single vertex, the heuristic iteratively connects the current subtree to a nearest terminal by a shortest path. This procedure is reiterated until all terminals are spanned. The heuristic is implemented in Jack-III, but in its original form detailed by [26]. In SCIP-Jack an empirically faster version based on Dijkstra's algorithm [22] is implemented. In addition to being used as an initial heuristic, the RSPH is also employed, with altered costs, during the branch-and-cut. Specifically, given an LP optimal solution $y \in \mathbb{Q}^A$, the heuristic is called with the costs $(1 - y_a) \cdot c_a$ for all $a \in A$. Thus, a stimulus for the heuristic to choose arcs contained in the LP solution is provided. Moreover, the heuristic is started from several distinct vertices, making it empirically much stronger (by default 100 start vertices for the initial call at the root node, 50 at the beginning of the processing of each other branch-and-bound node and 15 for calls within the cut loop). Terminals are preferred as start points, but vertices that exhibit a high (fractional) outdegree in the incumbent LP solution are also selected. The heuristic is called before and after the processing of a (branch-and-bound) node, after each cut loop and after each LP solving during a cut loop.

The improvement heuristic VQ is a combination of the three local search heuristics vertex insertion, key-path exchange, and key-vertex elimination as described in [23]. The basic idea of vertex insertion (denoted by V) is to connect further vertices to an existing Steiner tree in such a way that expensive edges can be removed. Key-vertices with respect to a tree S are either terminals or vertices of degree at least three in S. Correspondingly, a key-path is a path in S with a key-vertex at both endpoints, but without any intermediary key-vertices. A key-path exchange attempts to replace existing key-paths by others that are less costly. Similarly, for key-vertex elimination in each step a non-terminal key-vertex and all adjoining key-paths (except for the key-vertices at their respective ends) are extracted and an attempt is made to reconnect the disconnected subtrees at a lower cost. As in [23], the combination of key-path exchange and key-vertex elimination is denoted by Q. VQ is called for a newly found solution whenever the latter is among the five best known solutions.

The prune heuristic comes with a less customary approach obtained by building upon bound-based reductions introduced in [14] that were afterwards slightly improved in [16]. While for the original bound-based reductions an upper bound is provided by the weight of a given Steiner tree, in the prune heuristic the bound is reduced such that in each iteration a certain proportion of edges and vertices is eliminated. Thereupon, all exact reductions methods are executed on the reduced graph, motivated by the assumption that the (possibly inexact) eliminations performed by the bound-based method will allow for further (exact) reductions. To avoid infeasibility, a Steiner tree is initially computed by using RSPH and afterwards the elimination of any of its vertices by the bound-based method is being prohibited. Within SCIP-Jack the heuristic is called whenever a new best solution has been found.

Another powerful heuristic approach is borne by the combination of the prune heuristic and dual-ascent: the ascend-and-prune [14] method. Ascend-and-prune is motivated by the assumption that certain similarities exist between an optimal Steiner tree and the LP solution that is identified by the reduced costs provided by dual-ascent. Thereupon, the heuristic attempts to find an optimal solution on the graph constituted by the undirected edges corresponding to zero-reduced-cost paths from the root to all additional terminals. On this subgraph a solution is computed by first employing an (exact) reduction package and then using the prune heuristic. Within SCIP-Jack, ascend-and-prune is performed after each execution of dual-ascent, in particular prior to the initiation of the branch-and-cut procedure.

Finally, the recombination of given solutions to find improved primal bounds is performed by the RC heuristic. In the following, RC is described in the context of an STP, but it can be naturally extended to cover all Steiner tree problem variants discussed in this paper. First, the set of solutions to be considered for recombination is defined by $\mathcal{L}$; in the case of SCIP-Jack $\mathcal{L}$ comprises the best found solutions and its cardinality is bounded from above by 50.

The heart of RC is the *n-merging* ($n \geq 2$) operation subsequently defined for a given solution S_0 to an STP $P = (V, E, T, c)$: S_0 is merged with pseudo-randomly selected $n-1$ solutions $S_1, ..., S_n$ out of $\mathcal{L} \setminus \{S_0\}$ to form a new STP $\tilde{P}$ consisting of all edges and vertices that are part of at least one of the n solutions. By applying the reduction techniques provided by SCIP-Jack to $\tilde{P}$, a reduced problem $\tilde{P}'$ is obtained. Thereupon, a solution to $\tilde{P}'$ is computed in several steps. First, it is observed that each edge e in $\tilde{P}'$ corresponds to a set of ancestor edges $E^e \subseteq E$. Denoting the edges of a solution S_i by E_{S_i} gives the definition:

$$\alpha(e) = \frac{\sum_{i=0}^{n} |E^e \cap E_{S_i}|}{|E^e|}.$$

Next, the cost of each edge e in $\tilde{P}'$ is multiplied by a pseudo-randomized number that is anti-proportional to $\alpha(e)$ (i.e., the number increases as $\alpha(e)$ decreases). This edge cost multiplication approach is a more general variant of a procedure suggested in [27]. The latter approach recombines two solutions without employing reduction techniques. Using the new edge cost, RSPH is employed to obtain a solution $\tilde{S}'$ to $\tilde{P}'$. For the starting points of RSPH, vertices v_i are used such that $\sum_{e \in \delta_{\tilde{P}'}(v_i)} \alpha(e)$ is maximized. Next, after retrieving the original arc costs, VQ is applied on $\tilde{S}'$. Finally, $\tilde{S}'$ is retransformed to the original solution space.

The RC heuristic is clustered around the n-merging operation: Given a new solution S, in one *run* consecutively six 2-, two 3- and one 4–merge operations are performed. When a solution S' is generated during an n-merging with a smaller cost than S, the solution S is replaced by S', which is attempted to be added to $\mathcal{L}$. Moreover, in this case the n-merging is performed again in a new run that is started after the conclusion of the current run. The total number of runs is limited to ten. RC is called whenever r new solutions have been found compared to its last execution. Initially, r is set to 4 and modified throughout the solution process, setting $r := 0$ if a solution has been improved during the execution of RC and $r := \min\{r + 1, 4\}$ otherwise.

By the combination of the previously described heuristics the ability to generate good primal solutions quickly is considerably improved, as compared to employing SCIP-Jack without Steiner problem specific heuristics. Furthermore, this combination is able to eventually find optimal solutions to most problems.

2.1 Computational experiments

Several thousand instances of 15 Steiner tree problem variants were collected as part of the DIMACS Challenge. To show the performance of the developed general Steiner tree problem solver, computational experiments on ten variants of the STP will be presented.

All computational experiments described were performed on a cluster of Intel Xeon X5672 CPUs with 3.20 GHz and 48 GB RAM, running Kubuntu

14.04. A development version of SCIP 3.2.1 was used and SoPlex [28] version 2.2.1 was employed as the underlying LP solver. Moreover, the overall run time for each instance was limited by two hours. If an instance was not solved to optimality within the time limit, the gap is reported, which is defined as $\frac{|\mathrm{pb}-\mathrm{db}|}{\max\{|\mathrm{pb}|,|\mathrm{db}|\}}$ for final primal bound (pb) and dual bound (db). The average gap is obtained as an arithmetic mean. The averages of the number of nodes and the solving time are computed by taking the shifted geometric mean [21] with a shift of 10.0 and 1.0, respectively.

Prior to the discussion of the different STP variants solved by SCIP-Jack in the following section, the solver performance will be demonstrated on pure STP instances. To this end, six STP test sets have been selected for computational experiments. Five of them, X [7] E [29], I640 [13], PUC [12], and ALUE [7], are test sets from SteinLib. First, the three X instances include complete graphs with Euclidean distances corresponding to geographical locations (in Berlin, Brazil, and worldwide). In contrast, the E and I640 test sets contain randomly generated instances. The (sparse) E test set has proved to be solvable within short time limits by state-of-the-art solvers [14]. However, the I640 set—whose instances were selected to defy preprocessing—contains several problems that have remained unsolved until today. Similarly, many unsolved instances still remain in the PUC test set, which contains artificially designed problems such as instances composed of combinations of odd wheels and odd circles. As opposed to the previous three test sets, the ALUE instances are not artificially designed, but derive from a VLSI application and contain grid graphs with rectangular holes. The final test set is vienna-i-simple [2], which contains real-world instances generated from telecommunication networks that have already been preprocessed by the Degree-Test described in [19].

A summary of the computational performance of SCIP-Jack on the five STP test sets is presented in Table 1. Each line in the table shows aggregated results for the test set specified in the first column. The second column, labeled #, lists the number of instances in the test set, the third column states how many of them were solved to optimality within the time limit. The average number of branch-and-bound nodes and the average running time in seconds of these instances are presented in the next two columns, named optimal. The last two columns, labeled timeout, show the average number of branch-and-bound nodes and the average gap for the remaining instances, i.e., all

Table 1: Computational results for STP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
X	3	3	1.0	0.3	–	–
E	20	20	1.4	1.0	–	–
ALUE	15	12	1.7	12.4	1.0	1.7
I640	100	78	17.7	9.3	85.3	0.8
PUC	50	8	406.8	36.9	121.9	3.5
vienna-i-simple	85	69	2.7	149.2	1.8	0.0

instances that hit the time limit. In the next section, similar tables will be presented for different STP variants. If all instances of a particular variant are solved to optimality within the time limit, the timeout columns are omitted. Detailed instance-wise computational results of all experiments can be found in Appendix C.

The instances from the X test set is solved without any branching and in very short run times; even the largest instance, consisting of more than 200 000 edges, requires only one second. Similarly, SCIP-Jack solves the entire, mostly sparse, E test set to optimality within an average time of 1.0 seconds. The only instance requiring branching is *e18*, which also exhibits a run time much longer than any other of that test set, 104.5 seconds. The, similarly sparse, VLSI-derived ALUE instances are harder to solve for SCIP-Jack: three problems remain unsolved with gaps of 1.4, $1, 5$, and 2.3 percent, while the solved instances require an average run time of 12.4 seconds. For only three of the instances branching is performed, and none requires more than five nodes.

SCIP-Jack exhibits a distinctively disparate behavior on the I640 test set: While more than half of the instances are solved within a few seconds, 22 problems remain unsolved after two hours. As compared with the previous instances the number of branch-and-bound nodes is much higher—up to 4481.

The PUC test set proves to be much more difficult for SCIP-Jack. This is unsurprising since more than half of the instances in this set still remain unsolved. SCIP-Jack only solves eight of 50 instances and none at the root node. More than half of the unsolved instance are still processing the root node when terminated. Finally, 80 percent of the vienna instances set are solved by SCIP-Jack within the time limit, with none of the remainder exhibiting an optimality gap of more than 0.01%. As compared to the PUC test set, the number of branch-and-bound nodes is much smaller and more than half of the instances can be solved in the root.

The results demonstrate an improvement of SCIP-Jack in two dimensions. First, compared to Jack-III, SCIP-Jack empirically yields significantly better results, as exemplified by the E test set. While Jack-III and SCIP-Jack both manage to solve all instances in relatively short times, there is a large difference in the run times achieved by each. Jack-III needs more than one second for all but one instance. The hardest instance, *e18*, requires more than 11 minutes. In contrast, SCIP-Jack solves all but three instances in less than a second, with a maximum runtime (for *e18*) of about two minutes. On average, SCIP-Jack is more than two orders of magnitude faster on the E test set than Jack-III.

The drastically stronger performance of SCIP-Jack in comparison with Jack-III comes hardly as a surprise. While the main component of Jack-III, the separation algorithm, is being reused within SCIP-Jack, a variety of powerful new methods, as described heretofore, is clustered around it. The new reduction techniques alone, for instance, are more than ten times faster than those implemented in Jack-III and nevertheless notably stronger.

The second dimension is seen when comparing the performance of the current version of SCIP-Jack with its predecessor participating in the DIMACS Challenge, cf. [30]. In particular, computational experiments demonstrate that the current version is significantly stronger. Taking the example of the I640 test set, one sees additional 13 instances being solved to optimality within two hours. Also, the run time for several instances (such as i640-043) is reduced by a factor of more than a hundred.

The improvement of the current version of SCIP-Jack as compared with the previous version that competed in the DIMACS Challenge can be put down to several major algorithmic improvements. First, the implementation of new or enhanced problem-specific preprocessing methods, such as the dual-ascent reductions [31]. Second, the implementation of the new heuristics dual-ascent, and, on the primal side, prune, ascend-and-prune, and an improved version of the recombination heuristic. Third, the implementation of a problem-specific propagator and branching rule.

However, while the current version of SCIP-Jack proves to be competitive with the best exact results obtained at the DIMACS Challenge, it falls short of matching the fastest STP solver described in the literature [14,31]. Apart from very easy instances, such as in X, and the hard PUC test set, for which SCIP-Jack yields comparable results, the solver described in [14] (which uses the commercial CPLEX 12.6[1]) is more than an order of magnitude faster for most instances [31].

3 From single problem to class solver

SCIP-Jack is developed as a general STP solver—being able to solve many problem variants. An overview of the problem variants solved by SCIP-Jack is given in Table 2. This table also presents the heuristics (see Section 2) and presolving techniques (see Table 14) that are applied to each of the problem variants. Specific transformation approaches have been employed in order to solve each variant by SCIP-Jack. Each of these transformations will be described in detail. Throughout this section the weights of an (undirected) edge e and an (directed) arc a are denoted by c_e and c_a respectively and the weight of a vertex v by p_v.

3.1 The Steiner arborescence problem

As SCIP-Jack transforms each Steiner tree problem to a Steiner arborescence problem (SAP), the branch-and-cut framework can be used for general SAPs with only minor modifications. Slightly modified forms of the RSPH, AP, and RC heuristics can also be used for SAP instances. However, due to the missing bi-direction with equal cost, the VQ heuristic cannot be applied.

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Table 2: Problem variants solved by SCIP-Jack

Variant	Abbreviation	Preprocessing	Heuristics
Steiner Tree in Graphs	STP	DT, NV/SL, SD/SDC, NTD$_{3,4}$, BND, DA	P, AP, RSPH, VQ, RC
Steiner Arborescence	SAP	BR, RPT, CT, SDC, PNT, BND, DA	AP, RSPH, RC
Rectilinear Steiner Minimum Tree	RSMTP	DT, NV/SL, SD/SDC, NTD$_{3,4}$, BND, DA	P, AP, RSPH, VQ, RC
Node-weighted Steiner Tree	NWSTP	DA	AP, RSPH, RC
Prize-collecting Steiner Tree	PCSTP	UNT/DT, SD/SDC, NTD$_3$, NV/SL, BND, DA	P, AP, RSPH, VQ, RC
Rooted Prize-collecting Steiner Tree	RPCSTP	UNT/DT, SD/SDC, NTD$_3$, NV/SL, BND, DA	P, AP, RSPH, VQ, RC
Maximum-weight Connected Subgraph	MWCSP	UNPV/BT, CNS, NNP, NPV$_{2,3,4,5}$, DA	AP, RSPH, RC
Degree-constrained Steiner Tree	DCSTP	None	RSPH, RC
Group Steiner Tree	GSTP	DT, NV/SL, SD/SDC, NTD$_{3,4}$, BND, DA	P, AP, RSPH, VQ, RC
Hop-constrained directed Steiner Tree	HCDSTP	CBND, HBND	RSPH, RC

As to presolving techniques, besides DA, specific SAP reduction methods have been implemented—as described in [16].

Computational results Computational experiments have been performed on two test sets of Steiner arborescence problems. These instances are derived from a genetic application [32]. The results are summarized in Table 3. The test sets contain small SAP instance, with the largest consisting of 602 nodes, 1716 edges and 86 terminals. Because of their size, SCIP-Jack solves all instances within fractions of a second without requiring any branching. Furthermore, the reduction techniques eliminate more than 90 percent of the arcs on average.

Table 3: Computational results for SAP instances

test set	#	solved	∅ nodes	∅ time [s]
gene	10	10	1.0	0.2
gene2002	9	9	1.0	0.1

3.2 The rectilinear Steiner minimum tree problem

The *rectilinear Steiner minimum tree problem* (*RSMTP*) can be described as follows: Given a number of $n \in N$ points in the plane, find a shortest tree consisting only of vertical and horizontal line segments, containing all n points. The *RSMTP* is $\mathcal{NP}$-hard, as proved in [33], and has been the subject of various publications, see [34,35,36]. In addition to this two-dimensional variant, a generalization of the problem to the d-dimensional case, with $d \geq 3$, will be considered. The presented computational experiments include instances that derive from a cancer research application [3] and exhibit up to eight dimensions.

Hanan [37] reduced the RSMTP to the *Hanan-grid* obtained by constructing vertical and horizontal lines through each given point of the RSMTP. It is proved in [37] that there is at least one optimal solution to an RSMTP that is a subgraph of the grid. Hence, the RSMTP can be reduced to an STP. Subsequently, this construction and its multi-dimensional generalization [38] is exploited in order to adapt the RSMTP to be solved by SCIP-Jack. Given a d-dimensional, $d \in \mathbb{N} \setminus \{1\}$, RSMTP represented by a set of $n \in N$ points in $\mathbb{Q}^d$, the first step involves building a d-dimensional Hanan-grid. By using the resulting Hanan-grid an STP $P = (V, E, T, c)$ can be constructed, which is handled equivalently to a usual STP problem by SCIP-Jack.

It certainly bears mentioning that this simple Hanan-grid based approach is not expected to be competitive with highly specialized solvers such as GeoSteiner [34] in the case $d = 2$. However, a motivation for the implementation in SCIP-Jack is to address the obvious lack of solvers—specialized or general—that can provide solutions to RSMTP instances in dimensions $d \geq 3$. Still, it is not practical to apply the grid transformation for large instances in high dimension, as the number of both vertices and edges increases exponentially with the dimension.

A variant of the RSMTP is the *obstacle-avoiding rectilinear Steiner minimum tree problem* (*OARSMTP*). This problem requires that the minimum-length rectilinear tree does not pass through the interior of any specified axis-aligned rectangles, denoted as *obstacles*. SCIP-Jack is easily extended to solve the OARSMTP with a simple modification to the Hanan grid approach applied to the RSMTP. This modification involves removing all vertices that are located in the interior of an obstacle together with their incident edges as well as all edges crossing an obstacle. There was no competition for this variant in the DIMACS Challenge and for the OARSMTP, unlike the RSMTP, optimal solutions to all instances submitted to the Challenge have already been published. While SCIP-Jack is capable of solving all instances submitted to the DIMACS Challenge, computational experiments for this problem variant have been omitted.

Computational results The experiments on the RSMTP involve solving five of the test sets submitted to the DIMACS Challenge. These test sets contain instances ranging from less than 10 to 10 000 points and from two to eight dimensions. Specifically, the test sets used in the presented experiments include the two-dimensional estein instances with up to 60 nodes, the solids test set with three-dimensional instances whose terminals are the vertices of the five platonic solids, and the real-world derived cancer instances in up to eight dimensions. Computational results are summarized in Table 4 with the detailed results listed in the appendix.

The vast majority of the estein40 and estein50 instances can be solved to optimality, 14 and 13 out of 15 respectively. For all but one of these in-

Table 4: Computational results for RSMTP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
estein40	15	14	1.0	255.9	144.0	0.2
estein50	15	13	1.4	1868.0	31.8	0.4
estein60	15	6	1.0	5396.6	3.9	0.7
solids	5	4	4.9	0.2	8435.0	0.5
cancer	14	13	1.0	3.0	1.0	100.0

stances, the optimal solution was found at the root node. Also, none of the unsolved instances exhibits an optimality gap above 0.7 percent at the time limit. However, as the number of terminals increases, so does the run time and the number of unsolved instances: Only six of the estein60 instances can be solved within two hours, requiring more than twice as much time on average than the estein50 problems. The optimality gap of the unsolved instances ranges from 0.1 to 1.6 percent. Only one of the estein60 instances requires branching—using 82 branch-and-bound nodes.

The results in Table 4 show the capabilities of SCIP-Jack to solve instances in three dimensions. Specifically, all but one of the solids instances are solved to optimality. The unsolved instance, dodecahedron, is terminated after two hours with an optimality gap of 0.5 percent and 8435 branch-and-bound nodes. All other instances are solved in less than a second.

Finally, the cancer instances demonstrate the ability of SCIP-Jack to handle and solve RMST problems with up to eight dimensions. SCIP-Jack solves 13 of 14 instances to optimality at the root node. The remaining instance hits the memory limit after presolving—with SCIP-Jack computing a primal, but no dual bound. To the best of the authors' knowledge, SCIP-Jack is the first solver to solve any of the cancer instances to optimality. Remarkably, more than half of the instances can be solved during preprocessing, including the cancer13_8D instance with more than a million arcs (in its transformed shape). Furthermore, only two of the solved instances require more than four seconds to achieve optimality. As compared to the previous version of SCIP-Jack competing in the DIMACS Challenge, cf. [30], the run times have considerably improved, mainly due to the enhanced reduction techniques (most notably DA), but also due to the new heuristics (most notably ascend-and-prune). For example, the cancer4_6D instance was not solved within 12 hours with the previous version, while the new version of SCIP-Jack now proves optimality in less than 15 minutes.

3.3 The node-weighted Steiner tree problem

The *node-weighted Steiner tree problem* (*NWSTP*) is a generalization of the Steiner tree problem in graphs where the edges and, additionally, the vertices are assigned non-negative weights. The objective is to connect all terminals while minimizing the weight summed over both vertices and edges spanned by the corresponding tree.

The NWSTP is formally stated by: Given an undirected graph $G = (V, E)$, node costs $p : V \to \mathbb{Q}_{\geq 0}$, edge costs $c : E \to \mathbb{Q}_{\geq 0}$ and a set $T \subseteq V$ of terminals, the objective is to find a tree $S = (V_S, E_S)$ that spans T while minimizing

$$C(S) := \sum_{e \in E_S} c_e + \sum_{v \in V_S} p_v.$$

The NWSTP can be transformed to an SAP by substituting each edge by two anti-parallel arcs. Then, observing that in a tree there cannot be more than one arc going into the same vertex, the weight of each vertex is added to the weight of each of its incoming arcs.

Transformation 1 (NWSTP to SAP)
Given an NWSTP $P = (V, E, T, c, p)$ construct an SAP $P' = (V', A', T', c', r')$ as follows:

1. *Set $V' := V$, $T' := T$, $A' := \{(v, w) \in V' \times V' : \{v, w\} \in E\}$.*
2. *Define $c' : A' \to \mathbb{Q}_{\geq 0}$ by $c'_a = c_{\{v,w\}} + p_w$, for $a = (v, w) \in A'$.*
3. *Choose a root $r' \in T'$ arbitrarily.*

Lemma 1 (NWSTP to SAP) *Let $P = (V, E, T, c, p)$ be an NWSTP and $P' = (V', A', T', c')$ an SAP obtained by applying Transformation 1 on P. Denote by $\mathcal{S}$ and $\mathcal{S}'$ the set of solutions to P and P' respectively. Then $\mathcal{S}'$ can be bijectively mapped onto $\mathcal{S}$ by applying*

$$V_S := \{v \in V : \ v \in V'_{S'}\} \tag{9}$$

$$E_S := \{\{v, w\} \in E : \ (v, w) \in A'_{S'} \ \text{or} \ (w, v) \in A'_{S'}\} \tag{10}$$

for $S' = (V'_{S'}, A'_{S'}) \in \mathcal{S}'$ and it holds:

$$c'(A'_{S'}) + p_{r'} = c(E_S) + p(V_S). \tag{11}$$

The resulting SAP can be directly solved by SCIP-Jack. However, due to efficiency reasons only a subset of the heuristics and reduction techniques are employed, see Table 2.

Computational results Two NWSTP instances derived from a computational biology application are part of the DIMACS Challenge. The two instances differ drastically in their size. The first has more than 200 000 nodes—55 000 of them terminals—and almost 2.5 million edges, while the smaller instance comprises merely 386 nodes, 1477 edges, and 35 terminals.

The size of the first instance proves to be prohibitive for SCIP-Jack. The memory requirements of this instance quickly exceeds the limits of SCIP-Jack when applying the default settings on a modest machine. To evaluate the ability of SCIP-Jack to solve this particular instance, a runtime of 72 hours was used on a machine with 386 GB memory. To render the presolving more effective, modified versions of the STP tests NVO, SD and NTD$_3$ were employed to solve the NWSTP instance (in addition to the default preprocessing). After the application of the reduction techniques, the resulting graph contains 187 933 nodes and 986 703 edges. This equates to a 8.6 % and 60.4 % decrease in the number of nodes and edges respectively. SCIP-Jack fails to solve this instance to optimality, but it does achieve a nearly-optimal primal bound of 656 970.94 with an optimality gap of 0.0049%. The much smaller second instance is solved by SCIP-Jack at the root node within 0.1 seconds.

3.4 The prize-collecting Steiner tree problem

In contrast to the classical Steiner tree problem, the required tree for the *prize-collecting Steiner tree problem* (*PCSTP*) needs only to span a (possibly empty) subset of the terminals. However, a non-negative penalty is charged for each terminal not contained in the tree. Hence, the objective is to find a tree of minimum weight, given by both the sum of its edge costs and the penalties of all terminals not spanned by the tree. A profound discussion on the PCSTP is given in [4] that details real-world applications and introduces a sophisticated specialized solver.

A formal definition of the problem is stated as: Given an undirected graph $G = (V, E)$, edge-weights $c : E \to \mathbb{Q}_{\geq 0}$ and node-weights $p : V \to \mathbb{Q}_{\geq 0}$, a tree $S = (V_S, E_S)$ in G is required such that

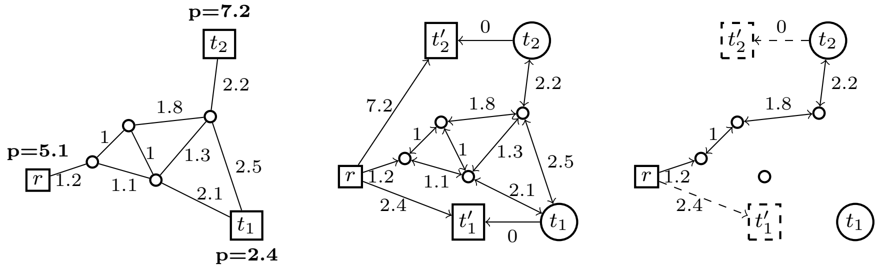
$$P(S) := \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \tag{12}$$

is minimized.

Prior to the discussion of the prize-collecting Steiner tree problem, a variation is introduced, the *rooted prize-collecting Steiner tree problem* (*RPCSTP*). The RPCSTP incorporates the additional condition that one distinguished node r, denoted the *root*, must be part of every feasible solution to the problem. It is assumed that $p_r = 0$. The RPCSTP can be transformed into an SAP as follows:

Transformation 2 (RPCSTP to SAP)
Given an RPCSTP $P = (V, E, p, r)$ construct an SAP $P' = (V', A', T', c', r')$ as follows:

1. *Set $V' := V$, $A' := \{(v, w) : \{v, w\} \in E\}$, $r' := r$ and $c' : A' \to \mathbb{Q}_{\geq 0}$ with $c'_a = c_{\{v,w\}}$ for $a = (v, w) \in A'$.*
2. *Denote the set of all $v \in V$ with $p_v > 0$ by $T = \{t_1, ..., t_s\}$. For each node $t_i \in T$, a new node t'_i and an arc $a = (t_i, t'_i)$ with $c'_a = 0$ is added to V' and E' respectively.*
3. *Add arcs (r', t'_i) for each $i \in \{1, ..., s\}$, setting their respective weight to p_{t_i}.*
4. *Define the set of terminals $T' := \{t'_1, ..., t'_s\}$.*



(a) RPCSTP instance (b) transformed instance (c) feasible solution

Fig. 1: Illustration of a rooted prize-collecting Steiner tree instance with root r (left), the equivalent SAP problem obtained by Transformation 2 (middle), and a solution to the SAP instance with value 8.6 (right).

After Transformation 2, for each terminal t_i' of the SAP P' there are exactly two incoming arcs $(t_i, t_i{}')$ and (r', t'). Thereupon, each solution $S' = (V_{S'}', A_{S'}') \in P'$ that contains t_i must also contain $(t_i, t_i{}')$, more succinctly:

$$\forall i \in \{1, ..., s\} : t_i \in V_{S'}' \implies (t_i, t_i{}') \in A_{S'}' \tag{13}$$

Condition (13) is satisfied by all optimal solutions to P' and each feasible solution can be easily modified to accomplish this, concomitantly improving its solution value. Transformation 2 is presented in [39], but without using condition (13). The latter gives rise to a one-to-one correspondence of the solution sets, stated in the following lemma.

Lemma 2 (RPCSTP to SAP) *Let $P' = (V', A', T', c')$ be an SAP obtained from an RPCSTP $P = (V, E, c, p)$ by applying Transformation 2. Denote by $\mathcal{S}$ and $\mathcal{S}'$ the set of solutions to P and P', satisfying condition* (13)*, respectively. P' can be mapped bijectively onto P by*

$$V_S := \{v \in V : \ v \in V_{S'}'\} \tag{14}$$

$$E_S := \{\{v, w\} \in E : \ (v, w) \in A_{S'}' \ \textit{or} \ (w, v) \in A_{S'}'\} \tag{15}$$

for $S' = (V_{S'}', A_{S'}') \in \mathcal{S}'$. The solution value is preserved.

Transformation 2 can be extended to cover the PCSTP by the inclusion of an artificial root node r' and arcs (r', t_i) of cost 0. However, only one of these arcs can be part of a feasible solution. This requirement is enforced by the following constraint:

$$\sum_{a \in \delta^+(r'), c_a' = 0} y_a = 1. \tag{16}$$

Furthermore, to allow a bijection between the original and the transformed problem, for all t_i included in a solution the arc (r', t_i) with the smallest index

i is required to be part of the solution. This condition can be expressed by using the following set of constraints:

$$\sum_{a \in \delta^-(t_j)} y_a + y_{(r',t_i)} \leq 1 \quad i = 1, ..., s;\ j = 1, ..., i-1. \tag{17}$$

An SAP that requires the conditions (13), (16) and (17) is referred to as *root constrained Steiner arborescence problem* (*rcSAP*). The constraints (16) and (17) can be incorporated into the cut-formulation (Formulation 1) without further alterations and each solution can be modified in order to meet condition (13). Although additional $\frac{s(s-1)}{2}$ constraints are introduced to fulfill (17), the solving time is considerably reduced by adding the constraints, as they exclude a plethora of symmetric solutions.

Transformation 3 (PCSTP to rcSAP)
Given an PCSTP $P = (V, E, c, p)$ construct an rcSAP $P' = (V', A', T', c', r')$ as follows:

1. *Add a vertex v_0 to V and set $r := v_0$.*
2. *Apply Transformation 2 to obtain $P' = (V', A', T', c', r')$.*
3. *Add arcs $a = (r', t_i)$ with $c'_a := 0$ for each $t_i \in T$.*
4. *Add constraints* (16) *and* (17)*.*

Lemma 3 (PCSTP to rcSAP) *Let $P = (V, E, c, p)$ be an PCSTP and $P' = (V', A', T', c', r')$ the corresponding rcSAP obtained by applying Transformation 3. Denote by $\mathcal{S}$ and $\mathcal{S}'$ the sets of solutions to P and P' respectively. Each solution $S' \in \mathcal{S}'$ can be bijectively mapped to a solution $S \in \mathcal{S}$ defined by:*

$$V_S := \{v \in V :\ v \in V'_{S'}\} \tag{18}$$

$$E_S := \{\{v, w\} \in E :\ (v, w) \in A'_{S'} \text{ or } (w, v) \in A'_{S'}\}. \tag{19}$$

The solution value is preserved.

For the PCSTP and RPCSTP a vast number of reduction techniques—described in [16]—are employed by SCIP-Jack, see Table 2. Furthermore, all heuristic used for the STP can be deployed, albeit with some alterations. For the RSPH in the case of a transformed PCSTP, i.e. an rcSAP, instead of commencing from different vertices, the starting point is always the (artificial) root. In each run all arcs between the root and non-terminals (denoted by (r', t) in Transformation 3) are temporarily removed, except for one. A tree is then computed on this new graph, by using the same process as the original constructive heuristic. Instead of starting from a new terminal as done by customary RSPH, a different arc (r', t) is chosen to remain in the graph.

Finally, the VQ heuristic requires an adaption for both the RPCSTP and the PCSTP: All terminals are temporarily removed from the (transformed) graph and VQ is executed with all t_i, as defined in Transformation 2 and Transformation 3, marked as key vertices.

Table 5: Computational results for PCSTP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
JMP	34	34	1.0	0.0	–	–
CRR	80	80	1.0	0.4	–	–
PUCNU	18	10	11.4	53.5	50.7	1.9

Computational results Table 5 shows aggregated results for three of the PCSTP test sets provided for the DIMACS Challenge. All but two JMP instances are solved during preprocessing in at most 0.1 seconds. The remaining problems require no more than 0.2 seconds. Similarly, reduction techniques alone can solve 72 of the 80 CRR instances. However, the hardest (and considerably larger) instances take comparably longer—up to 3.7 seconds—to be solved to optimality. The third test set, PUCNU, is derived from the PUC test set for the STP. SCIP-Jack is already unable to solve many of the original instances and the PCSTP versions also prove to be hard. However, ten of the instances are solved to optimality, with only four instances requiring branching. The remaining eight instances terminate with optimality gaps in the range 1.0 % to 2.8 %.

Comparing the above results with those obtained by the previous version of SCIP-Jack, a significant improvement is observed. Specifically, the JMP and CRR instances can be solved more than 10 times faster on average, with the longest single run time of the previous version being almost 1000 seconds. This is compared to 3.7 seconds for the current version of SCIP-Jack. Moreover, three additional PUCNU instances can be solved within the time limit. A notable result is that SCIP-Jack now exhibits a better performance for 10 of the 12 JMP, CRR and PUCNU instances that were part of the exact DIMACS competition than any other participating solver at the time of the competition.

The improved performance of SCIP-Jack can be traced to the vastly stronger new reduction techniques. However, it is also due to the new heuristics P and AP, and the general improvements of SCIP-Jack, such as the new propagator and the dual-ascent algorithm, which allows the start the branch-and-cut with a strong lower bound.

Table 6: Computational results for RPCSTP instances

			optimal	
test set	#	solved	∅ nodes	∅ time [s]
cologne1	14	14	1.0	0.2
cologne2	15	15	1.0	0.6

The average results for the RPCSTP instances are displayed in Table 6. Remarkably, the maximum run time observed is 0.7 seconds. While in the DIMACS Challenge SCIP-Jack already took first place in the exact solving category of the RPCTSP, the current version requires significantly less time to solve the Challenge instances—being on average more than a factor of 25 faster for the cologne1 test set and more than a factor of 75 for the cologne2 set. This improvement is the result of the vastly improved preprocessing techniques, which alone manage to solve all cologne1 and cologne2 instances to optimality.

3.5 The maximum-weight connected subgraph problem

At first glance, the *maximum-weight connected subgraph problem* (*MWCSP*) bears little resemblance to the Steiner problems introduced so far: Given an undirected graph (V, E) with (possibly negative) node weights p, the objective is to find a tree that maximizes the sum of its node weights. However, it is possible to transform this problem into a prize-collecting Steiner tree problem. One transformation is given in [5]. In this paper, an alternative transformation is presented that leads to a significant reduction in the number of terminals for the resulting PCSTP.

In the following it is assumed that at least one vertex is assigned a negative cost and at least one vertex is assigned a positive cost. Without this assumption the problem becomes trivial to solve.

Transformation 4 (MWCSP to rcSAP)
Let $P = (V, E, p)$ be an MWCSP, construct an rcSAP $P'' = (V'', A'', T'', c'', r'')$:

1. *Set $V' := V$, $A' := \{(v, w) : \{v, w\} \in E\}$.*
2. *$c' : A' \to \mathbb{Q}_{\geq 0}$ such that for $a = (v, w) \in A'$:*
   $c'_a = \begin{cases} -p_w, \text{ if } p_w < 0 \\ 0, \quad \text{otherwise} \end{cases}$
3. *$p' : V' \to \mathbb{Q}_{\geq 0}$ such that for $v \in V'$:*
   $p'(v) = \begin{cases} p_v, \text{ if } p_v > 0 \\ 0, \quad \text{otherwise} \end{cases}$
4. *Perform Transformation 3 to (V', A', c', p'), but in step 2 instead of constructing a new arc set, A' is being used. The resulting rcSAP gives us $P'' = (V'', A'', T'', c'', r'')$.*

Lemma 4 (MWCSP to rcSAP) *Let $P = (V, E, p)$ be an MWCSP and $P'' = (V'', A'', T'', c'', r'')$ an rcSAP obtained from P by Transformation 4. Then each solution S'' to P'' can be bijectively mapped to a solution S to P. The latter is obtained by:*

$$V_S := \{v \in V : \; v \in V''_{S''}\} \tag{20}$$

$$E_S := \{\{v, w\} \in E : \; (v, w) \in A''_{S''} \text{ or } (w, v) \in A''_{S''}\} \tag{21}$$

Furthermore, for the objective value $C(S)$ of S and the objective value $C''(S'')$ of S'' the following equality holds:

$$C(S) = \sum_{v \in V : p_v > 0} p_v - C''(S''). \tag{22}$$

Table 7: Number of terminals after transformation for test set ACTMOD

instance	Transformation 4	transformation from [5]
drosophila001	71	5226
drosophila005	194	5226
drosophila0075	250	5226
HCMV	55	3863
lymphoma	67	2034
metabol_expr_mice_1	150	3523
metabol_expr_mice_2	85	3514
metabol_expr_mice_3	114	2853

Since most of the vertex weights are non-positive for all real-world DIMACS instances, Transformation 4 results in problems with significantly less terminals compared to the transformation described in [5]. The differences in the number of terminals resulting from the two transformations are presented in Table 7. Even if the number of positive weight vertices is high in the original problem, after presolving it is typically much smaller, since adjacent non-negative vertices can be contracted [16].

For the MWCSP the computational settings of SCIP-Jack are similar to those of the (R)PCSTP. However, the VQ heuristic is not enabled since it cannot easily be adapted to handle anti-parallel arcs of different weight.

Computational results Computational experiments have been performed on the two MWCSP test sets that were part of the DIMACS Challenge. The first is the real-world derived ACTMOD test set (which contains eight instances), and second is the artificially created JMPALMK set (which contains 72 instances). The results, illustrated in Table 8, demonstrate the ability of SCIP-Jack to effectively handle real-world MWCSP instances of up to 93 000 edges in very short time: All eight instances can be solved within 1.4 seconds, in an average of less than half a second. The speedup of SCIP-Jack as compared to its DIMACS Challenge predecessor is impressive, ranging from a factor of ten to more than 4000. Furthermore, each instance is solved at least four times faster than by any solver during the DIMACS competition. A salient example is the drosophila001 instance which requires only 0.8 seconds with SCIP-Jack, but at least 21.6 seconds with any of the participating solvers at the time of the DIMACS competition. The drastically reduced run time of SCIP-Jack is mainly due to new reduction techniques, but also the dual-ascent algorithm is a notable factor.

The effectiveness of Transformation 4 is demonstrated by the performance of SCIP-Jack on the ACTMODPC test set, which consists of the ACTMOD problems transformed to PCSTP by the transformation described in [5]. Compared to the original ACTMOD test set the run time of SCIP-Jack increases for each instance of the ACTMODPC set; one cause of this increase is the DA method becoming less efficient due to a far higher amount of terminals in the transformed SAP. As such, Transformation 4 is a valuable addition to SCIP-Jack.

The results on the JMPALMK test set once again bespeak the strength of reduction techniques implemented in SCIP-Jack. All instances are solved during presolving, in an average of less than 0.1 seconds.

Table 8: Computational results for MWCSP instances

test set	#	solved	∅ nodes	∅ time [s]
ACTMOD	8	8	1.0	0.4
JMPALMK	72	72	1.0	0.0

3.6 The degree-constrained Steiner tree problem

The *degree-constrained Steiner tree problem* (*DCSTP*) is an STP with additional degree constraints on the vertices, described by a function $b : V \to \mathbb{N}$. The objective is to find a minimum cost Steiner tree $S = (V_S, E_S)$ such that $\delta_S(v) \leq b(v)$ is satisfied for all $v \in V_S$. A comprehensive discussion of the DCSTP, including its applications in biology, can be found in [6].

The implementation in SCIP-Jack to solve the DCSTP involves the extension of Formulation 1 by an additional (linear) degree constraint for each vertex. Since the degree restriction does not comply with any reduction techniques of SCIP-Jack, problem-specific preprocessing has not been performed on these instances. Only the constructive heuristic is used, albeit in a modified form. The implemented constructive heuristic performs the following two checks while choosing a new (shortest) path to be added to the current tree. First, whether attaching this path would violate any degree constraints. Second, whether after having added this path at least one additional edge could be added (or all terminals are spanned). If no such path can be found, a vertex of the tree is pseudo randomly chosen that allows to add at least one adjacent edge. Next, such an edge leading to a vertex of high degree and being of small cost is chosen.

Computational results Computational experiments are performed on the 20 instances in the TreeFam test set of the DIMACS Challenge with a time limit of two hours. All instances come with degree bounds of at most 3 and their underlying graphs are complete.

The results from computational experiments on these instances are illustrated in Table 9. SCIP-Jack finds the optimal solution to five instances and proves the infeasibility of another two. The remaining 13 instances cannot be solved by SCIP-Jack within the time limit. The gap is reduced to less than or equal to one percent for seven of these instances. However, the gaps of the remaining six instances range from 4.5 to as much as 55.0 percent.

As compared to what can be observed for other Steiner problem variants in this paper, the average number of branch-and-bound nodes is high, with 90.6 nodes for the solved and 758.6 for the unsolved instances.

Similar to the preceding variants, the current version of SCIP-Jack demonstrates improved performance over the previous version at the DIMACS competition. In particular, the insatisfiability of two instances can now be proven and both the run time for the solved instances and the gap for the remainder are significantly reduced: by a factor of more than 10 and by a factor of up to 100, respectively. This results in a reduction of the average gap (one of the criteria in the DIMACS Challenge) from 37.4 at the time of the competition to 9.3 with the latest SCIP-Jack version. Note that the winner of this category reached an average gap of 19.1 in the competition. The improved solving behavior of SCIP-Jack on the DCSTP can be attributed to general enhancements such as the new propagator, see Section 2.

Table 9: Computational results for DCSTP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
TreeFam	20	7	90.6	10.3	760.6	14.3

3.7 The group Steiner tree problem

The *group Steiner tree problem* (*GSTP*) is another generalization of the Steiner tree problem that originates from VLSI design [20]. For the GSTP the concept of terminals as a set of vertices to be interconnected is extended to a set of vertex groups: Given an undirected graph $G = (V, E)$, edge costs $c : E \to \mathbb{Q}_{\geq 0}$ and a series of vertex subsets $T_1, ..., T_s \subseteq V$, $s \in \mathbb{N}$, a minimum cost tree spanning at least one vertex of each subset is required. By interpreting each terminal t as a subset $\{t\}$, every STP can be considered as a GSTP, the latter likewise being $\mathcal{NP}$-hard. On the other hand, it is possible to transform each GSTP instance $(V, E, T_1, .., T_s, c)$ to an STP by using the following scheme:

Transformation 5 (GSTP to STP)
Given an GSTP $P = (V, E, T_1, ...T_s, c)$ construct an STP $P' = (V', E', T', c')$ as follows:

1. *Set $V' := V$, $E' := E$, $T' = \emptyset$, $c' := c$, $K := \sum_{e \in E} c_e + 1$.*
2. *For $i = 1, ..., s$ add a new node t'_i to V' and T' and for all $v_j \in T_i$ add an edge $e = \{t'_i, v_j\}$, with $c'_e := K$.*

Let $(V, E, T_1, ...T_s, c)$ be a GSTP and $P' = (V', A', T', c')$ an STP obtained by applying Transformation 5 on P. A solution S' to P' can then be reduced to a solution S to P by deleting all vertices and edges of S not in (V, E). The GSTP P can in this way be solved on the STP P' as shown in [20] and [40].

This approach has already been deployed by [41] to solve group Steiner tree problems and demonstrated to be competitive with specialized solvers at the time of publishing. In the case of SCIP-Jack, to solve a GSTP, Transformation 5 is applied and the resulting problem is treated as a customary STP that is solved without any alteration. An alternative approach would be to employ GSTP-specific heuristics or reduction techniques [42].

Computational results Computational experiments were performed on two test sets of unpublished group Steiner tree instances derived from a real-world wire routing problem. The results from these experiments are presented in Table 10. SCIP-Jack solves all but two of the first test set, with run times ranging from 3.3 to 563 seconds. Four of the instances solved to optimality only require a single node, with the remaining instances solved in 219 and 61 nodes, respectively. The two unsolved instances gstp34f2 and gstp39f2 exhibit optimality gaps of 2.7 % and 5.1 % respectively. The same performance is not observed for the second test set. None of the instances, are solved within the time limit and the optimality gaps range from 0.9 % to 7.8 %.

Table 10: Computational results for GSTP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
GSTP1	8	6	14.9	25.0	355.9	3.9
GSTP2	10	0	–	–	48.6	3.2

3.8 The hop-constrained directed Steiner tree problem

The *hop-constrained directed Steiner tree problem* (*HCDSTP*) searches for an SAP with the additional constraint that the number of selected arcs must not exceed a predetermined bound, called *hop limit*. The cut formulation (Formulation 1) used by SCIP-Jack is simply extended to cover this variation by adding one extra linear inequality bounding the sum of all binary arc variables. It should be noted that in the literature the term "hop-constrained Steiner tree" often refers to a problem for which the number of arcs in the path from the root to any terminal within a feasible solution is limited by a predefined bound [43], which differs from the definition used in this paper.

The hop limit brings significant ramifications for the preprocessing and heuristics approaches in its wake. Customarily, many presolving techniques for Steiner tree problems remove or include edges from the graph if a less costly path can be found, regardless whether this procedure leads to a solution with more edges. For the HCDSTP such techniques can therefore produce infeasibility. However, a number of HCDSTP-specific bound-based reduction techniques can be applied, as described in [16].

Similar to the presolving techniques, the heuristics implemented in SCIP-Jack for the other variants do not take into account the hop limit. As such, any identified solution may not be feasible. Therefore, a simple variation of the constructive heuristic is used for the HCDSTP: Each arc a, having original costs c_a, is assigned the new cost $c'_a := 1 + \lambda \frac{c_a}{c_{max}}$, with $\lambda \in \mathbb{Q}_+$ and $c_{max} := \max_{a \in A} c_a$. Initially λ is set to 3 but its value is decreased or increased after each iteration of the constructive heuristic, depending on whether the last computed solution exceeds or is below the hop limit, respectively. This modification to λ is performed relative to the deviation of the number of edges from the hop limit.

Computational results Three different test sets, consisting of the gr12, gr14 and gr16 instances, are used for the computational experiments. All three test sets were used in the evaluation of the DIMACS Challenge. SCIP-Jack is able to solve all gr12 instances at the root node in less than 100 seconds. The performance worsens for the gr14 test set, with 12 of 21 instances being solved to optimality within the time limit. The unsolved instances terminate with optimality gaps ranging from 2.4 % to 17.9 %, after 8.8 nodes on average. Similarly, the optimally solved instances require—with 433.9 seconds—much more time than the previous problems (in gr12).

Table 11: Computational results for HCDSTP instances

			optimal		timeout	
test set	#	solved	∅ nodes	∅ time [s]	∅ nodes	∅ gap [%]
gr12	19	19	1.0	4.0	–	–
gr14	21	12	6.2	396.6	8.8	10.2
gr16	20	0	–	–	1.1	81.3

Finally, more than half of the gr16 instances were terminated due to insufficient memory. Therefore, to solve these instances a different machine was used, consisting of Intel Xeon E5-2697 CPUs with 2.70 GHz and 128 GB RAM. Although this machine can boast more RAM than the machines of the cluster used for the other computational experiments reported in this paper (see Section 2.1), it is notably slower.

The results for the gr16 test set are significantly worse than for the other two sets. Specifically, all instances terminate within the time limit with an optimality gap of at least 27.4 %. For these larger instances, SCIP-Jack terminates within the cut loop at the root LP for all but one instance. Besides the size of the problems, a possible cause of this performance is the lack of stronger HCDSTP-specific reduction techniques and heuristics in SCIP-Jack.

3.9 Using CPLEX as underlying LP solver

As an extension of SCIP, SCIP-Jack provides a branch-and-cut search, but requires an external LP solver for solving the linear programming relaxations.

For all results previously presented the LP solver SoPlex—the default LP solver employed by SCIP—has been used for this purpose. However, SCIP provides interfaces to many different commercial and academic LP solvers. This section discusses the impact of exchanging the academic LP solver SoPlex for the commercial solver CPLEX 12.6.

Table 12: Results of using CPLEX as LP solver for SCIP-Jack.

		SCIP-Jack		SCIP-Jack/CPLEX		relative change [%]	
test set	type	solved	∅ time [s]	solved	∅ time [s]	solved	∅ time
vienna-i-simple	STP	68	298.6	75	218.9	+10.3	-26.7
estein60	RSMTP	6	6307.2	12	2672.5	+100	-57.6
PUCNU	PCSTP	10	127.0	10	56.3	–	-55.7
TreeFam	DCSTP	7	730.1	7	773.0	–	+5.9
GSTP2	GSTP	0	7200.1	6	2394.4	–	-66.7
gr14	HCDSTP	12	1134.9	14	523.7	+16.7	-53.9
all		103	866.5	124	501.0	+20.4	-42.2

The comparison between using the LP solvers of CPLEX and SoPlex in SCIP-Jack is performed by selecting one test set for each previously discussed Steiner tree problem variant. An exception is made for those problem variants that can be trivially solved after presolving—such as the SAP and MWCSP. This test set selection is made to provide instances for which the reduction techniques still leave large problems, to highlight the impact of the LP solver in the branch-and-cut algorithm.

Table 12 illustrates the comparative performance of SCIP-Jack/CPLEX. The test set and the problem variant are listed in columns one and two. Columns three and four show the number of solved instances and the shifted geometric mean of the running time on the test set for SCIP-Jack using SoPlex as LP solver. The next two columns show the corresponding information for SCIP-Jack/CPLEX. Finally, the last two columns provide the relative change in the number of solved instances and the average time. The last row of the table considers all instances of the six test sets jointly.

Table 12 reveals that the number of solved instance is significantly increased when CPLEX is used. This phenomenon becomes notably pronounced for the estein60 instances, for which more than 90 percent of time is spent in the LP solver. Specifically, the number of solved instances doubles. Even more salient is the behavior on the GSTP2 test set, with six instances solved to optimality by SCIP-Jack/CPLEX, but not a single one by SCIP-Jack/SoPlex.

The comparison between CPLEX and SoPlex shows that the solving time for most of the instances is significantly smaller when the former is used as LP solver of SCIP-Jack. The only exception to this pattern is the TreeFam class on which SCIP-Jack/SoPlex is around six percent faster than SCIP-Jack/CPLEX.

In summary, the results show a considerable potential to speed up SCIP-Jack by using a faster LP solver.

4 From single core to distributed parallel

SCIP has two parallel extensions, ParaSCIP [44] and FiberSCIP [45], which are built by using the *Ubiquity Generator Framework* (UG) [45]. In order to parallelize a problem-specific solver (such as SCIP-Jack), users of SCIP can simply modify their developed plugins by adding a small glue code and linking to one of the UG libraries for SCIP (UG can be used with different state-of-the-art MIP solvers). This glue code consists of an additional class with a function that issues calls to include all SCIP plugins required for the sequential version of the code. Importantly, no modification to the sequential version of the problem-specific solver is required.

In this way, users obtain their own problem-specific parallel optimization solver that can perform a parallel tree search on a distributed memory computing environment. The main features of UG are: several *ramp-up* mechanisms (the ramp-up is the process from the beginning of the computation until all available solvers become busy), a dynamic load balancing mechanism for parallel tree search and a check-pointing and restarting mechanism. More details about the parallelization provided by UG can be found in [44,45].

This section presents computational results for the PUC test set from SteinLib. However, it must be noted that the parallel version of SCIP-Jack can handle all of the variants presented throughout this paper. The main purpose of the parallel runs is to provide optimal solutions to as many instances as possible. As mentioned above, the parallelization of a problem-specific solver only requires a small glue code. As such, the parallel version of SCIP-Jack is identical to the sequential version. By pursuing this simple approach, large supercomputing resources can be employed to apply SCIP-Jack to solve computationally difficult Steiner tree problems. For the computations, various clusters and supercomputers were used as they were available. The largest computation performed for these experiments involved up to 864 cores, which was only required for eight instances (`bip52p, bip62u, bipa2p, bipa2u, cc11-2p, cc12-2p, cc3-12p, hc9p`). However, all other computations were conducted with 192 or less solvers. Since these experiments were performed with the goal to solve previously unsolved instances and cluster and supercomputer time was limited, CPLEX 12.6 was used as the underlying LP solver to reduce the expected run times, see Section 3.9. As a reference to the scalability of ParaSCIP, the largest computation previously performed was an 80 000 cores run on Titan at ORNL [46]. It is expected that SCIP-Jack can also run on such a large scale computing environment, although at this stage only relatively small scale computational experiments have been conducted.

Table 13 shows the results on the instances of the PUC test set as of 17th April 2015. This table lists the number of nodes, edges, and terminals, as well as the best primal bound known at the beginning of the DIMACS Challenge (August 2014), and the primal solution value obtained in experiments with the parallel version of SCIP-Jack. Prior to the experiments performed using SCIP-Jack, 32 instances of the PUC test set remained unsolved. Three of these instances have been solved by SCIP-Jack to proven optimality, which

Table 13: Primal bound improvements on the PUC instances

instance	$\|V\|$	$\|E\|$	$\|T\|$	best	SCIP-Jack	instance	$\|V\|$	$\|E\|$	$\|T\|$	best	SCIP-Jack
bip42p	1200	3982	200	24657	24657*	cc3-5u	125	750	13	36	36*
bip42u	1200	3982	200	236	236*	cc5-3p	243	1215	27	7299	7299*
bip52p	2200	7997	200	24535	**24526**	cc5-3u	243	1215	27	71	71*
bip52u	2200	7997	200	234	234	cc6-2p	64	192	12	3271	3271*
bip62p	1200	10002	200	22870	**22843**	cc6-2u	64	192	12	32	32*
bip62u	1200	10002	200	220	**219**	cc6-3p	729	4368	76	20456	**20270***
bipa2p	3300	18073	300	35379	**35326**	cc6-3u	729	4368	76	197	197*
bipa2u	3300	18073	300	341	**338**	cc7-3p	2187	15308	222	57088	57117
bipe2p	550	5013	50	5616	5616*	cc7-3u	2187	15308	222	552	552
bipe2u	550	5013	50	54	54*	cc9-2p	512	2304	64	17296	**17199**
cc10-2p	1024	5120	135	35379	**35227**	cc9-2u	512	2304	64	167	167*
cc10-2u	1024	5120	135	342	343	hc10p	1024	5120	512	60494	**59797**
cc11-2p	2048	11263	244	63826	**63636**	hc10u	1024	5120	512	581	**575**
cc11-2u	2048	11263	244	614	618	hc11p	2048	11264	1024	119779	**119689**
cc12-2p	4096	24574	473	121106	122099	hc11u	2048	11264	1024	1154	**1151**
cc12-2u	4096	24574	473	1179	1184	hc12p	4096	24576	2048	236949	**236080**
cc3-10p	1000	13500	50	12860	**12837**	hc12u	4096	24576	2048	2275	**2262**
cc3-10u	1000	13500	50	125	126	hc6p	64	192	32	4003	4003*
cc3-11p	1331	19965	61	15609	15648	hc6u	64	192	32	39	39*
cc3-11u	1331	19965	61	153	153	hc7p	128	448	64	7905	7905*
cc3-12p	1728	28512	74	18838	18997	hc7u	128	448	64	77	77*
cc3-12u	1728	28512	74	186	187	hc8p	256	1024	128	15322	15322*
cc3-4p	64	288	8	2338	2338*	hc8u	256	1024	128	148	148*
cc3-4u	64	288	8	23	23*	hc9p	512	2304	256	30258	**30242**
cc3-5p	125	750	13	3661	3661*	hc9u	512	2304	256	292	292

have been underlined and marked with an asterisk in Table 13. For a further 16 instances, SCIP-Jack improved the best known solution. All instances for which the best known primal bound has been improved are marked in bold. Finally, all previously solved instances of the PUC test set have also been solved by SCIP-Jack to proven optimality, which have been marked by an asterisk (without underline).

The instances presented in Table 13 differ widely in their solving behavior. Using the cc6-3u instance as an example, one obtains greater insight into the typical solving procedure when applying ParaSCIP. The cc6-3u instance was solved to optimality for the first time by SCIP-Jack and ParaSCIP. In order to solve this instance again in a single run without restarting, SCIP-Jack was run on the HLRN-III supercomputer consisting of a Cray XC30. This experiment was performed by using nodes equipped with two 12-core Intel Xeon Haswell CPUs sharing 64 GB of RAM and with a CPU clock of 2.5 GHz. By deploying 3072 MPI processes, the optimal solution with an objective value of 197 was proven in 961 seconds after having processed 123 210 branch-and-bound nodes. While the lower bound was already within a distance of one to the optimal objective value after 20 seconds, the primal bound dropped to 198 only after 100 seconds.

The above results demonstrate an overall strong performance of the parallel version of SCIP-Jack in solving computationally difficult STP instances.

5 Conclusions

This paper shows the multilayered impact of embedding a 15-year old Steiner tree branch-and-cut procedure into a state-of-the-art MIP framework and clustering new solving methods around it. First, the amount of problem-specific code is drastically reduced. At the same time the number of general solution methods available, e.g., cutting planes, has increased and will be kept up-to-date just by the continuous improvements in the framework. Furthermore, the opportunity to solve instances in a massively parallel distributed memory environment has been added at minimal cost. Attempts were made to solve open instances from the difficult PUC test set by using these massively parallel extensions. As a result, SCIP-Jack was not only able to solve three previously unsolved instances, but improve the best known solution for another 16.

The use of a general MIP solver allows a significant amount of flexibility in the model to be solved. SCIP-Jack is able to support solving ten variants of the Steiner tree problem with nearly the same code, and the support of further restrictions in the model is straightforward. On top of this versatility, the powerful solving framework for the underlying IP formulation combined with problem-specific methods such as reduction techniques allows SCIP-Jack to be highly competitive with problem-specific state-of-the-art solvers.

Yet, there certainly is potential for future work to improve the performance and scope of the solver. First, already implemented routines such as the branching rule could be improved. Second, additional reduction techniques and heuristics for specific Steiner tree problem variants could be implemented. Finally, SCIP-Jack could be extended to cover further Steiner problem variants described in the literature. By using the plugin structure of SCIP, the inclusion of some of these enhancements is expected in the future.

Ultimately, this paper has described the creation of a highly competitive exact solving framework of outstanding versatility that can veritably be designated as a Steiner *class* solver. Furthermore, to the best of our knowledge this is the first time that a powerful exact Steiner tree solver has been made available in source code to the scientific community. The SCIP Optimization Suite [9] already contains a previous version of our solver and the current version of SCIP-Jack is planned to be part of the next release of SCIP. We hope that the availability of such a device will foster the use of Steiner trees in modeling real-world phenomena.

6 Acknowledgements

The work for this article has been conducted within the *Research Campus Modal* funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM). It has been further supported by a Google Faculty Research Award.

The authors would like to thank the referees for constructive criticism and helpful comments on the paper. We are thankful to The Institute of Statistical Mathematics (ISM), Tokyo, and the HLRN-III facility at ZIB, Berlin, for providing supercomputing resources and their respective staff for their helpful support. Furthermore, we are thankful to Cees Duin for sending us a hard copy of his PhD thesis. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

References

1. Karp, R.: Reducibility among combinatorial problems. In Miller, R., Thatcher, J., eds.: Complexity of Computer Computations. Plenum Press (1972) 85–103
2. Leitner, M., Ljubić, I., Luipersbeck, M., Prossegger, M., Resch, M.: New real-world instances for the Steiner tree problem in graphs. Technical report, ISOR, Uni Wien (2014)
3. Chowdhury, S.A., Shackney, S., Heselmeyer-Haddad, K., Ried, T., Schffer, A.A., Schwartz, R.: Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. Bioinformatics **29**(13) (2013) 189–198
4. Ljubić, I.: Exact and Memetic Algorithms for Two Network Design Problems. PhD thesis, Technische Universität Wien (2004)
5. Dittrich, M.T., Klau, G.W., Rosenwald, A., Dandekar, T., Müller, T.: Identifying functional modules in protein-protein interaction networks: An integrated exact approach. Bioinformatics **24**(13) (2008) 223–231
6. Liers, F., Martin, A., Pape, S.: Steiner trees with degree constraints: Structural results and an exact solution approach. Technical report, Department Mathematik (2014)
7. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks **32** (1998) 207–232
8. Achterberg, T.: SCIP: Solving constraint integer programs. Mathematical Programming Computation **1**(1) (2009) 1–41
9. Gamrath, G., Fischer, T., Gally, T., Gleixner, A.M., Hendel, G., Koch, T., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Vigerske, S., Weninger, D., Winkler, M., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 3.2. Technical Report 15-60, ZIB, Takustr.7, 14195 Berlin (2016)
10. Borndörfer, R., Hoàng, N.D., Karbstein, M., Koch, T., Martin, A.: How many Steiner terminals can you connect in 20 years? In Jünger, M., Reinelt, G., eds.: Facets of Combinatorial Optimization. Springer (2013) 215–244
11. Koch, T., Martin, A., Pfetsch, M.E.: Progress in academic computational integer programming. In Jünger, M., Reinelt, G., eds.: Facets of Combinatorial Optimization. Springer (2013) 483–506
12. Rosseti, I., de Aragão, M., Ribeiro, C., Uchoa, E., Werneck, R.: New benchmark instances for the Steiner problem in graphs. In: Extended Abstracts of the 4th Metaheuristics International Conference (MIC'2001), Porto (2001) 557–561
13. Duin, C.: Steiner Problems in Graphs. PhD thesis, University of Amsterdam (1993)
14. Polzin, T.: Algorithms for the Steiner problem in networks. PhD thesis, Saarland University (2004)
15. Daneshmand, S.V.: Algorithmic Approaches to the Steiner Problem in Networks. PhD thesis, Universität Mannheim (2004)
16. Rehfeldt, D.: A generic approach to solving the Steiner tree problem and variants. Master's thesis, Technische Universität Berlin (2015)
17. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: On the solution of traveling salesman problems. Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung **Extra Volume ICM III** (1998) 645–656
18. Mittelmann, H.: Benchmarks for optimization software (last accessed Mar. 9, 2015) `http://plato.asu.edu/bench.html`.

19. Beasley, J.E.: An algorithm for the Steiner problem in graphs. Networks **14**(1) (1984) 147–159
20. Hwang, F., Richards, D., Winter, P.: The Steiner tree problem. Annals of Discrete Mathematics **53** (1992)
21. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)
22. de Aragao, M.P., Werneck, R.F.: On the implementation of MST-based heuristics for the Steiner problem in graphs. In: Proceedings of the 4th International Workshop on Algorithm Engineering and Experiments, Springer (2002) 1–15
23. Uchoa, E., Werneck, R.F.F.: Fast local search for Steiner trees in graphs. In Blelloch, G.E., Halperin, D., eds.: ALENEX, SIAM (2010) 1–10
24. Wong, R.: A dual ascent approach for Steiner tree problems on a directed graph. Mathematical Programming **28** (1984) 271–287
25. Pajor, T., Uchoa, E., Werneck, R.F.: A robust and scalable algorithm for the Steiner problem in graphs. Computing Research Repository (2014)
26. Takahashi, H., A., M.: An approximate solution for the Steiner problem in graphs. Math. Jap. **24** (1980) 573–577
27. Ribeiro, C.C., Uchoa, E., Werneck, R.F.: A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS Journal on Computing **14**(3) (2002) 228–246
28. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)
29. Beasley, J.: An SST-based algorithm for the Steiner problem in graphs. Networks **19** (1989) 1–16
30. Gamrath, G., Koch, T., Maher, S.J., Rehfeldt, D., Shinano, Y.: SCIP-Jack – A solver for STP and variants with parallelization extensions. 11th DIMACS Competition workshop paper (2014)
31. Polzin, T., Vahdati-Daneshmand, S.: The Steiner tree challenge: An updated study. Unpublished manuscript at http://dimacs11.zib.de/downloads.html (2014)
32. Johnston, J., Kelley, R., Crawford, T., Morton, D., Agarwala, R., Koch, T., Schäffer, A., Francomano, C., Biesecker, L.: A novel nemaline myopathy in the Amish caused by a mutation in troponin T1. American Journal of Human Genetics (October 2000) 814–821
33. Garey, M., Johnson, D.: The rectilinear Steiner tree problem is NP-complete. SIAM Journal of Applied Mathematics **32** (1977) 826–834
34. Warme, D., Winter, P., Zachariasen, M.: Exact algorithms for plane Steiner tree problems: A computational study. In Du, D.Z., Smith, J., Rubinstein, J., eds.: Advances in Steiner Trees. Kluwer (2000) 81–116
35. Zachariasen, M., Rohe, A.: Rectilinear group Steiner trees and applications in VLSI design. Technical Report 00906, Institute for Discrete Mathematics (2000)
36. Emanet, N.: The Rectilinear Steiner Tree Problem. Lambert Academic Publishing (2010)
37. Hanan, M.: On Steiner's problem with rectilinear distance. SIAM Journal on Applied Mathematics **14**(2) (1966) 255–265
38. Snyder, T.L.: On the exact location of Steiner points in general dimension. SIAM Journal on Computing **21**(1) (1992) 163–180
39. Rehfeldt, D., Koch, T.: Transformations for the Prize-Collecting Steiner Tree Problem and the Maximum-Weight Connected Subgraph Problem to SAP. Technical Report 16-36, ZIB, Takustr.7, 14195 Berlin (2016)
40. Vo, S.: A survey on some generalizations of Steiner's problem. 1st Balkan Conference on Operational Research Proceedings **1** (1988) 41–51
41. Duin, C.W., Volgenant, A., Vo, S.: Solving group Steiner problems as Steiner problems. European Journal of Operational Research **154**(1) (2004) 323–329
42. Ferreira, C.E., de Oliveira Filho, F.M.: New reduction techniques for the group Steiner tree problem. SIAM Journal on Optimization **17**(4) (December 2006) 1176–1188
43. Voß, S.: The Steiner tree problem with hop constraints. Annals of Operations Research **86**(0) (1999) 321–345
44. Shinano, Y., Achterberg, T., Berthold, T., Heinz, S., Koch, T.: ParaSCIP: a parallel extension of SCIP. In Bischof, C., Hegering, H.G., Nagel, W., Wittum, G., eds.: Competence in High Performance Computing 2010. (2012) 135–148

45. Shinano, Y., Heinz, S., Vigerske, S., Winkler, M.: FiberSCIP - a shared memory parallelization of SCIP. Technical Report 13–55, ZIB, Takustr.7, 14195 Berlin (2013)
46. Shinano, Y., Achterberg, T., Berthold, T., Heinz, S., Koch, T., Winkler, M.: Solving open MIP instances with ParaSCIP on supercomputers using up to 80,000 cores. In: 2016 IEEE International Parallel and Distributed Processing Symposium. (May 2016) 770–779

A Proofs

This section provides proofs to the lemmata stated in the course of this paper. For the respective transformation corresponding to each of these lemmata a one-to-one correspondence between the solution sets of the original and the transformed problem is proven as well as the linear relation between the respective solutions values. This implies that all these problems can be solved on their transformed solution spaces.

A.1 Proof of Lemma 1 (NWSTP to SAP)

Proof To see the one-to-one correspondence let $S = (V_S, E_S) \in \mathcal{S}$ and proceed as follows:
Surjective. Initially set $V'_{S'} := V_S$ and $A'_{S'} := \emptyset$. Traverse (V_S, E_S), e.g. using breadth-first search, starting from r' and add for each $w \in V_S$ visited from $v \in V_S$ the arc (v, w) to $A'_{S'}$. $S' := (V'_{S'}, A'_{S'})$ is a solution to P' and by applying (9) and (10), S is obtained.
Injective. S' is the only solution to P' that is mapped by (9) and (10) to S: Each $\tilde{S}' \in \mathcal{S}'$, $\tilde{S}' \neq S'$ contains at least one arc (v, w) such that $(v, w) \notin A'_{S'}$ and $(w, v) \notin A'_{S'}$, since only substituting arcs in $A'_{S'}$ by there anti-parallel counterparts would not allow directed paths from the root to all vertices. Therefore, $\tilde{S}'$ is not mapped onto S.

To acknowledge (11) one readily observes that for each node of S' except for the root there is exactly one incoming arc, so:

$$\sum_{(v,w)\in A'_{S'}} c'_{(v,w)} = \sum_{(v,w)\in A'_{S'}} \big(c_{\{v,w\}} + p_w\big) = \sum_{\{v,w\}\in E_S} c_{\{v,w\}} + \sum_{w\in V_S} p_w - p_{r'},$$

which implies (11).

A.2 Proof of Lemma 2 (RPCSTP to SAP)

Proof To acknowledge that (14) and (15) constitute a mapping $\mathcal{S}' \to \mathcal{S}$ it can be observed that first the root node is conserved and second the set of all arcs corresponding to edges in the original graph (V, E) forms a tree. To prove that a bijection is given, let $S = (V_S, E_S) \in \mathcal{S}$ and $T = \{t_1, ..., t_s\}$ as defined in Transformation 2.
Surjective. Initially, set $V'_{S'} := V_S$ and $A'_{S'} := \emptyset$. Analogously to the proof of Lemma 1, add for each edge in E_S an arc to $A'_{S'}$ in such a way that finally there is for each $v' \in V'_{S'}$ a directed path from r' to v'. Next, for each $i \in \{1, ...s\}$ set $a_i := (t_i, t'_i)$ if $t_i \in V_S$, otherwise $a_i := (r', t'_i)$ and add a_i to $A'_{S'}$. Thereupon, $S' := (V'_{S'}, A'_{S'})$ is a solution to P' and by applying (14) and (15), we obtain S.
Injective. Define the set of all arcs of P' corresponding to the edges of P as $A := \{(v, w) \in A' : \{v, w\} \in E\}$ and accordingly $A_{S'} := A'_{S'} \cap A$. Since (13) has been assumed, it holds that: $(t_i, t'_i) \in A'_{S'} \Leftrightarrow t_i \in V'_S$ and $(r', t'_i) \in A'_{S'} \Leftrightarrow t_i \notin V'_S$. This implies that $A'_{S'}$ is already determined by $A_{S'}$. Now let $\tilde{S}' = (\tilde{V}'_S, \tilde{A}'_S) \in \mathcal{S}'$, $\tilde{S}' \neq S'$. Consequently, there is at least one arc $(v, w) \in \tilde{A}'_S$ such that $(w, v) \notin A_{S'}$ and $(w, v) \notin A_{S'}$ and therefore is $\tilde{S}'$ not mapped

to S.
Finally, using the above notation one observes that:

$$\sum_{a \in A'_{S'}} c'_a \;=\; \sum_{a \in A_{S'}} c'_a + \sum_{a \in A'_{S'} \setminus A_{S'}} c'_a \;=\; \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v,$$

so the costs of S' and S are equal.

A.3 Proof of Lemma 3 (PCSTP to rcSAP)

Proof Likewise to the proof of Lemma 2 one observes that (18) and (19) constitute a mapping $\mathcal{S}' \to \mathcal{S}$. Let $S = (V_S, E_S) \in \mathcal{S}$ and $T = \{t_1, ..., t_s\}$ defined as in Transformation 3.
Surjective. Initially, define $V'_{S'} := V_S$, $A'_{S'} := \{(r, t_{i_0})\}$, with $i_0 := \min\{i \mid t_i \in V'_{S'}\}$. Then extend $A'_{S'}$ analogously to the proof of Lemma 2. The so constructed $S' := (V'_{S'}, A'_{S'})$ is a solution to P' and applying (18) and (19) S is obtained.
Injective. Parallelly to the proof of Lemma 2 it can be shown that for a solution $\tilde{S}' \neq S'$ to P' there must be at least one arc $(v, w) \in A_{\tilde{S}'}$ such that $(v, w) \notin A_{S'}$ and $(w, v) \notin A_{S'}$ with A defined as in the proof of Lemma 2. Therefore it follows that $\tilde{S}'$ is not mapped to S. The equality of the solution values of S and S' can be seen likewise.

A.4 Proof of Lemma 4 (MWCS to rcSAP)

Proof The one-to-one correspondence between the sets of solutions to P and P'' can be seen analogously to the proof of Lemma 3.
To prove (22) let $S = (V_S, E_S)$ be a solution to P and $S'' = (V''_{S''}, A''_{S''})$ the corresponding solution to P'', obtained by applying (20) and (21). Further, define $A := \{(v, w) \in A'' : \{v, w\} \in E\}$ and $A_{S''} = A \cap A''_{S''}$. First, one observes that for each $v \in S$ such that $p_v \leq 0$ there is exactly one incoming arc $a \in A_{S''}$, so:

$$\sum_{v \in V_S : p_v \leq 0} p_v = - \sum_{a \in A_{S''}} c''_a. \tag{23}$$

Second:

$$\sum_{v \in V_S : p_v > 0} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{v \in V \setminus V_S : p_v > 0} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{a \in A''_{S''} \setminus A_{S''}} c''_a. \tag{24}$$

Finally, by adding (23) and (24) the equation:

$$\sum_{v \in V_S} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{a \in A''_{S''}} c''_a \tag{25}$$

is obtained, which coincides with (22).

B Abbreviations of reduction methods

This section provides the names for all reduction methods used by SCIP-Jack, which are listed in Table 2 in abbreviated form. All methods are described in detail in [16]. Note that several methods can be used for several variants, see Table 2, but in this case almost always require (considerable) adaptations. These adaptations can likewise be found in [16].

Table 14 lists in the first column the respective abbreviation of each reduction method used by SCIP-Jack. The second column provides the full name of the method, while the third and last column states all Steiner problem variants that use this particular reduction method (in adapted form).

Table 14: Abbrevitations of reduction methods

Abbreviation	Name	Variant
BND	Bound	STP, SAP, PCSTP, RPCSTP, RSMTP, GSTP
BR	Basic Reduction	SAP
BT	Basic Test	MWCSP
CBND	Cost Bound	HCSTP
CNS	Connected Neighborhood Subset	MWCSP
CT	Close Terminals	SAP
DA	Dual-Ascent	SPG, SAP, NWSTP, PCSTP, RPCSTP, MWCSP, RSMTP, GSTP
DT	Degree Test	SPG, SAP, PCSTP, RPCSTP, RSMTP, GSTP
HBND	Hop Bound	HCSTP
NNP	Non-negative Path	MWCSP
NPV$_k$	Non-Positive Vertex of degree k	MWCSP
NTD$_k$	Non-Terminal of Degree k	SPG, PCSTP, RPCSTP, RSMTP, GSTP
NV	Nearest Vertex	SPG, PCSTP, RPCSTP, RSMTP, GSTP
PNT	Prohibitive Non-Terminal	SAP
RPT	Root Proximity Terminal	SAP
SD	Steiner bottleneck Distance	SPG, PCSTP, RPCSTP, RSMTP, GSTP
SDC	Steiner bottleneck Distance Circuit	SPG, PCSTP, RPCSTP, MWCSP, RSMTP, GSTP
SL	Short Links	SPG, PCSTP, RPCSTP, RSMTP, GSTP
UNT	Unreachable Non-Terminal	PCSTP, RPCSTP
UNPV	Unreachable Non-Positive Vertex	MWCSP

C Detailed computational results

This section presents detailed instance-wise results from the experiments performed in this paper for all test sets discussed in Sections 2 and 3. The tables list the original and the presolved problem size, i.e., number of nodes $|V|$, arcs $|A|$, and terminals $|T|$ as well as the preprocessing time (column t [s] in the Presolved columns). Moreover, the tables show the Dual and Primal bound upon termination and the corresponding Gap in percent. If an instance was solved to optimality, the optimal value is printed instead of primal and dual bound, and the gap is omitted. Similarly, "–" is printed if no primal bound was present at the time of termination. Additionally, the number of branch-and-bound nodes (N), and the total solving time in seconds (last column) is listed. The total solving time includes the preprocessing time. A timeout is marked by ">" before the termination time. In case of the DCSTP for which SCIP-Jack does not perform preprocessing, the statistics about the presolved model are omitted.

Table 15. Detailed computational results for the STP, test set X.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
berlin52	52	2652	16	0	0	0	0.0	**1044**	1	0.0
brasil58	58	3306	25	0	0	0	0.0	**13655**	1	0.0
world666	666	442890	174	0	0	0	1.1	**122467**	1	1.1

Table 16. Detailed computational results for the STP, test set E.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
e01	2500	6250	5	0	0	0	0.1	**111**	1	0.1
e02	2500	6250	10	0	0	0	0.1	**214**	1	0.1
e03	2500	6250	417	0	0	0	0.1	**4013**	1	0.1
e04	2500	6250	625	0	0	0	0.2	**5101**	1	0.2
e05	2500	6250	1250	0	0	0	0.2	**8128**	1	0.2
e06	2500	10000	5	0	0	0	0.2	**73**	1	0.2

cont. next page

Instance	Original			Presolved						
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
e07	2500	10000	10	0	0	0	0.4	**145**	1	0.4
e08	2500	10000	417	0	0	0	0.1	**2640**	1	0.1
e09	2500	10000	625	0	0	0	0.1	**3604**	1	0.1
e10	2500	10000	1250	0	0	0	0.1	**5600**	1	0.1
e11	2500	25000	5	0	0	0	0.5	**34**	1	0.5
e12	2500	25000	10	0	0	0	0.5	**67**	1	0.5
e13	2500	25000	417	439	1506	164	0.3	**1280**	1	0.8
e14	2500	25000	625	0	0	0	0.1	**1732**	1	0.1
e15	2500	25000	1250	0	0	0	0.4	**2784**	1	0.4
e16	2500	125000	5	0	0	0	0.6	**15**	1	0.6
e17	2500	125000	10	0	0	0	0.4	**25**	1	0.5
e18	2500	125000	417	2063	11702	245	0.8	**564**	13	104.5
e19	2500	125000	625	1203	5878	175	3.7	**758**	1	6.4
e20	2500	125000	1250	0	0	0	13.8	**1342**	1	13.8

Table 17. Detailed computational results for the STP, test set ALUE.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
alue2087	1244	3942	34	0	0	0	0.1	**1049**			1	0.1
alue2105	1220	3716	34	53	154	16	0.1	**1032**			1	0.1
alue3146	3626	11738	64	682	2364	58	0.7	**2240**			3	7.9
alue5067	3524	11120	68	605	1966	61	0.5	**2586**			1	4.6
alue5345	5179	16330	68	2728	9146	68	1.4	**3507**			5	946.5
alue5623	4472	13876	68	2037	6816	68	1.7	**3413**			5	729.5
alue5901	11543	36858	68	2749	9320	68	2.3	**3912**			1	665.0
alue6179	3372	10426	67	259	778	52	0.4	**2452**			1	0.9
alue6457	3932	12274	68	748	2388	62	0.8	**3057**			1	7.2
alue6735	4119	13392	68	799	2564	66	0.7	**2696**			1	5.5
alue6951	2818	8838	67	733	2362	67	0.7	**2386**			1	10.6
alue7065	34046	109682	544	28309	97592	512	6.4	23340.4247	23898	2.3	1	>7200.0
alue7066	6405	20908	16	3631	12480	11	10.0	2230.71143	2265	1.5	1	>7200.1
alue7080	34479	110988	2344	27631	95538	2002	14.6	61569.9419	62475	1.4	1	>7200.0
alue7229	940	2948	34	0	0	0	0.0	**824**			1	0.0

Table 18. Detailed computational results for the STP, test set I640.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
i640-001	640	1920	9	0	0	0	0.0	**4033**			1	0.0
i640-002	640	1920	9	0	0	0	0.1	**3588**			1	0.1
i640-003	640	1920	9	0	0	0	0.1	**3438**			1	0.1
i640-004	640	1920	9	0	0	0	0.0	**4000**			1	0.1
i640-005	640	1920	9	0	0	0	0.1	**4006**			1	0.1
i640-011	640	8270	9	0	0	0	0.1	**2392**			1	0.1
i640-012	640	8270	9	0	0	0	0.0	**2465**			1	0.1
i640-013	640	8270	9	0	0	0	0.1	**2399**			1	0.1
i640-014	640	8270	9	0	0	0	0.1	**2171**			1	0.1
i640-015	640	8270	9	0	0	0	0.1	**2347**			1	0.1
i640-021	640	408960	9	0	0	0	4.5	**1749**			1	4.5
i640-022	640	408960	9	0	0	0	4.4	**1756**			1	4.4
i640-023	640	408960	9	0	0	0	4.6	**1754**			1	4.7
i640-024	640	408960	9	0	0	0	4.5	**1751**			1	4.5
i640-025	640	408960	9	0	0	0	4.7	**1745**			1	4.7
i640-031	640	2560	9	0	0	0	0.1	**3278**			1	0.1
i640-032	640	2560	9	0	0	0	0.1	**3187**			1	0.1
i640-033	640	2560	9	0	0	0	0.0	**3260**			1	0.0
i640-034	640	2560	9	0	0	0	0.1	**2953**			1	0.1
i640-035	640	2560	9	0	0	0	0.1	**3292**			1	0.1
i640-041	640	81792	9	32	232	9	1.5	**1897**			1	1.5

cont. next page

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
i640-042	640	81792	9	44	338	9	1.4	**1934**			1	1.5
i640-043	640	81792	9	36	260	9	1.2	**1931**			3	1.4
i640-044	640	81792	9	21	102	9	1.2	**1938**			1	1.2
i640-045	640	81792	9	0	0	0	1.2	**1866**			1	1.2
i640-101	640	1920	25	62	194	22	0.0	**8764**			1	0.1
i640-102	640	1920	25	0	0	0	0.0	**9109**			1	0.0
i640-103	640	1920	25	0	0	0	0.0	**8819**			1	0.0
i640-104	640	1920	25	0	0	0	0.0	**9040**			1	0.0
i640-105	640	1920	25	173	658	25	0.1	**9623**			3	1.7
i640-111	640	8270	25	640	8270	25	0.2	**6167**			233	80.9
i640-112	640	8270	25	640	8270	25	0.1	**6304**			89	66.0
i640-113	640	8270	25	640	8270	25	0.1	**6249**			285	219.9
i640-114	640	8270	25	640	8270	25	0.1	**6308**			199	156.8
i640-115	640	8270	25	640	8270	25	0.1	**6217**			285	141.3
i640-121	640	408960	25	0	0	0	4.5	**4906**			1	4.5
i640-122	640	408960	25	0	0	0	4.6	**4911**			1	4.6
i640-123	640	408960	25	0	0	0	4.7	**4913**			1	4.7
i640-124	640	408960	25	0	0	0	4.6	**4906**			1	4.6
i640-125	640	408960	25	0	0	0	4.6	**4920**			1	4.6
i640-131	640	2560	25	95	322	25	0.1	**8097**			1	0.5
i640-132	640	2560	25	106	414	24	0.1	**8154**			3	0.8
i640-133	640	2560	25	100	384	23	0.1	**8021**			1	0.2
i640-134	640	2560	25	0	0	0	0.0	**7754**			1	0.0
i640-135	640	2560	25	0	0	0	0.1	**7696**			1	0.1
i640-141	640	81792	25	640	39442	25	2.5	**5199**			186	2886.1
i640-142	640	81792	25	636	40352	25	2.6	**5193**			77	1602.1
i640-143	640	81792	25	640	49152	25	2.5	**5194**			69	1226.3
i640-144	640	81792	25	396	9360	25	1.9	**5205**			105	342.2
i640-145	640	81792	25	640	80900	25	3.4	**5218**			225	3158.5
i640-201	640	1920	50	104	352	39	0.1	**16079**			1	0.2
i640-202	640	1920	50	0	0	0	0.1	**16324**			1	0.1
i640-203	640	1920	50	165	602	46	0.1	**16124**			1	1.5
i640-204	640	1920	50	0	0	0	0.0	**16239**			1	0.0
i640-205	640	1920	50	55	162	31	0.0	**16616**			1	0.1
i640-211	640	8270	50	640	8270	50	0.2	11848.9241	12034	1.5	2328	>7200.0
i640-212	640	8270	50	640	8270	50	0.2	**11795**			1332	1618.0
i640-213	640	8270	50	640	8268	50	0.3	**11879**			4481	5362.3
i640-214	640	8270	50	640	8270	50	0.3	11860.8781	11898	0.3	3316	>7200.0
i640-215	640	8270	50	640	8262	50	0.3	11962.5217	12102	1.2	3085	>7200.0
i640-221	640	408960	50	640	175732	50	15.6	**9821**			31	1035.8
i640-222	640	408960	50	568	64944	50	7.2	**9798**			21	237.9
i640-223	640	408960	50	320	26888	50	7.3	**9811**			17	59.9
i640-224	640	408960	50	109	5672	50	5.9	**9805**			7	10.5
i640-225	640	408960	50	272	21786	50	5.7	**9807**			13	71.1
i640-231	640	2560	50	448	2044	50	0.1	**15014**			55	27.9
i640-232	640	2560	50	483	2226	49	0.1	**14630**			11	7.4
i640-233	640	2560	50	489	2208	47	0.1	**14797**			9	20.5
i640-234	640	2560	50	196	808	47	0.2	**15203**			1	1.1
i640-235	640	2560	50	482	2238	50	0.1	**14803**			181	205.6
i640-241	640	81792	50	640	79734	50	4.5	10165.1069	10259	0.9	26	>7200.1
i640-242	640	81792	50	636	38180	50	4.1	**10195**			323	6254.2
i640-243	640	81792	50	640	81410	50	3.2	10174.2073	10245	0.7	19	>7200.0
i640-244	640	81792	50	640	81354	50	3.2	10171.5011	10256	0.8	28	>7200.1
i640-245	640	81792	50	640	81414	50	3.1	10168.0015	10235	0.7	24	>7200.1
i640-301	640	1920	160	318	1168	122	0.0	**45005**			1	0.9
i640-302	640	1920	160	291	1112	110	0.0	**45736**			1	3.5
i640-303	640	1920	160	270	956	116	0.1	**44922**			1	0.4
i640-304	640	1920	160	261	930	119	0.1	**46233**			1	0.9

cont. next page

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
i640-305	640	1920	160	301	1106	115	0.1	**45902**			1	2.6
i640-311	640	8270	160	640	8058	160	0.3	35356.8112	35889	1.5	1057	>7200.0
i640-312	640	8270	160	639	8034	160	0.3	35403.0294	35931	1.5	1067	>7200.0
i640-313	640	8270	160	640	8064	160	0.4	35226.7419	35592	1.0	1136	>7200.0
i640-314	640	8270	160	640	8054	160	0.5	35144.1966	35700	1.6	1314	>7200.0
i640-315	640	8270	160	640	8038	160	0.3	35299.5984	35972	1.9	1305	>7200.0
i640-321	640	408960	160	640	383854	160	22.9	31005.418	31101	0.3	1	>7200.6
i640-322	640	408960	160	640	383870	160	23.1	31001.5172	31077	0.2	1	>7213.3
i640-323	640	408960	160	640	383854	160	22.2	31007.5649	31096	0.3	1	>7200.6
i640-324	640	408960	160	640	383872	160	23.2	31015.009	31097	0.3	1	>7204.6
i640-325	640	408960	160	640	383870	160	24.1	31000.6727	31104	0.3	1	>7202.4
i640-331	640	2560	160	488	2204	145	0.1	**42796**			161	67.5
i640-332	640	2560	160	504	2250	152	0.2	**42548**			163	113.4
i640-333	640	2560	160	502	2230	147	0.1	**42345**			1023	522.0
i640-334	640	2560	160	512	2280	155	0.1	**42768**			2715	1135.1
i640-335	640	2560	160	516	2292	153	0.1	**43035**			826	411.1
i640-341	640	81792	160	640	77068	160	5.1	31865.446	32121	0.8	9	>7200.1
i640-342	640	81792	160	640	76886	160	4.9	31839.4952	32028	0.6	11	>7200.0
i640-343	640	81792	160	640	76970	160	4.8	31846.6237	32049	0.6	8	>7200.1
i640-344	640	81792	160	640	77206	160	4.8	31829.1739	32009	0.6	11	>7200.1
i640-345	640	81792	160	640	77076	160	5.1	31839.7831	32051	0.7	9	>7200.1

Table 19. Detailed computational results for the STP, test set PUC.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
bip42p	1200	7964	200	990	7236	200	0.1	24478.2215	24721	1.0	3361	>7200.1
bip42u	1200	7964	200	990	7220	200	1.2	233.081642	237	1.7	4076	>7200.0
bip52p	2200	15994	200	1819	14666	200	0.5	24235.7691	24880	2.6	1018	>7200.1
bip52u	2200	15994	200	1819	14652	200	1.4	229.754538	235	2.2	1435	>7200.0
bip62p	1200	20004	200	1199	20000	200	0.5	22505.0323	23002	2.2	493	>7200.0
bip62u	1200	20004	200	1199	20000	200	1.2	214.54281	221	2.9	545	>7200.0
bipa2p	3300	36146	300	3140	35594	300	1.0	34739.6741	35909	3.3	108	>7200.1
bipa2u	3300	36146	300	3140	35590	300	3.5	329.71166	343	3.9	144	>7200.0
bipe2p	550	10026	50	550	10026	50	0.2	**5616**			1577	520.1
bipe2u	550	10026	50	550	10026	50	0.5	**54**			81	111.8
cc10-2p	1024	10240	135	1024	10240	135	0.7	34472.1056	36569	5.7	1	>7206.5
cc10-2u	1024	10240	135	1024	10240	135	1.5	334.162023	346	3.4	1	>7200.0
cc11-2p	2048	22526	244	2048	22526	244	1.5	62105.0239	65297	4.9	1	>7200.0
cc11-2u	2048	22526	244	2048	22526	244	3.0	602.531609	623	3.3	1	>7201.0
cc12-2p	4096	49148	473	4096	49148	473	5.7	118446.22	123835	4.4	1	>7200.7
cc12-2u	4096	49148	473	4096	49148	473	6.2	1148.80514	1201	4.3	1	>7200.0
cc3-10p	1000	27000	50	1000	27000	50	1.2	12696.14	12860	1.3	193	>7200.0
cc3-10u	1000	27000	50	1000	27000	50	2.8	117.836203	128	7.9	1	>7200.0
cc3-11p	1331	39930	61	1331	39930	61	1.8	15448.5499	15676	1.5	47	>7200.0
cc3-11u	1331	39930	61	1331	39930	61	7.9	143.442501	155	7.5	1	>7200.0
cc3-12p	1728	57024	74	1728	57024	74	2.4	18635.6768	19422	4.0	1	>7200.1
cc3-12u	1728	57024	74	1728	57024	74	5.9	172.550821	190	9.2	1	>7200.0
cc3-4p	64	576	8	64	576	8	0.0	**2338**			6301	167.6
cc3-4u	64	576	8	64	576	8	0.1	**23**			123	19.7
cc3-5p	125	1500	13	125	1500	13	0.0	3568.76584	3661	2.5	18548	>7200.0
cc3-5u	125	1500	13	125	1500	13	0.3	34.6204106	36	3.8	18377	>7200.0
cc5-3p	243	2430	27	243	2430	27	0.0	7173.69143	7303	1.8	1772	>7200.0
cc5-3u	243	2430	27	243	2430	27	0.4	69.6769218	71	1.9	960	>7200.0
cc6-2p	64	384	12	64	384	12	0.0	**3271**			207	11.7
cc6-2u	64	384	12	64	384	12	0.0	**32**			33	4.0
cc6-3p	729	8736	76	729	8736	76	0.2	20133.1622	20428	1.4	117	>7200.0
cc6-3u	729	8736	76	729	8736	76	1.0	195.4	201	2.8	1	>7200.0
cc7-3p	2187	30616	222	2187	30616	222	2.0	55338.9773	58029	4.6	1	>7200.7
cc7-3u	2187	30616	222	2187	30616	222	2.9	535.826599	562	4.7	1	>7200.1

cont. next page

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
cc9-2p	512	4608	64	512	4608	64	0.2	16868.5668	17441	3.3	1	>7200.3
cc9-2u	512	4608	64	512	4608	64	1.1	163.524047	172	4.9	1	>7200.0
hc10p	1024	10240	512	1024	10240	512	0.6	59252.5345	61409	3.5	166	>7200.0
hc10u	1024	10240	512	1024	10240	512	3.6	567.777778	595	4.6	4	>7200.1
hc11p	2048	22528	1024	2048	22528	1024	1.2	117404.127	122258	4.0	20	>7200.0
hc11u	2048	22528	1024	2048	22528	1024	14.1	1124.45703	1175	4.3	1	>7200.0
hc12p	4096	49152	2048	4096	49152	2048	3.3	232883.175	244784	4.9	1	>7200.0
hc12u	4096	49152	2048	4096	49152	2048	702.8	2199.62873	2334	5.8	1	>7200.0
hc6p	64	384	32	64	384	32	0.0	**4003**			1629	20.0
hc6u	64	384	32	64	384	32	0.0	**39**			481	14.7
hc7p	128	896	64	128	896	64	0.0	7856.13043	7905	0.6	67035	>7200.0
hc7u	128	896	64	128	896	64	0.1	75.1343284	77	2.4	34718	>7200.0
hc8p	256	2048	128	256	2048	128	0.1	15182.5249	15327	0.9	8785	>7200.0
hc8u	256	2048	128	256	2048	128	0.3	145.252523	148	1.9	2788	>7200.0
hc9p	512	4608	256	512	4608	256	0.2	29925.8807	30281	1.2	942	>7200.0
hc9u	512	4608	256	512	4608	256	0.8	286.875	295	2.8	46	>7200.0

Table 20. Detailed computational results for the STP, test set vienna-i-simple.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
I001	30190	95496	1184	7539	23190	978	6.3	**253921201**			1	220.9
I002	49920	155742	1665	12947	39624	1324	9.9	**399809303**			3	2817.3
I003	44482	146838	3222	13687	41394	2346	18.3	788774400	788774494	0.0	1	>7200.0
I004	5556	17104	570	471	1300	198	0.6	**279512692**			1	1.5
I005	10284	31960	1017	1031	2950	378	1.3	**390876350**			1	4.1
I006	31754	105750	2202	11750	35452	1856	9.8	**504526035**			5	4903.3
I007	15122	48742	737	4310	13290	586	3.1	**177909660**			1	53.1
I008	15714	51134	871	4804	14630	713	3.3	**201788202**			3	224.0
I009	33188	104014	1262	10603	32962	1066	7.7	**275558727**			3	642.5
I010	29905	94914	943	7421	23194	821	5.7	**207889674**			1	155.2
I011	25195	82596	1428	7454	22826	1218	7.7	**317589880**			11	799.6
I012	12355	39924	503	2057	6400	383	1.7	**118893243**			1	8.8
I013	18242	57952	891	4818	14740	679	3.4	**193190339**			1	93.2
I014	12715	41264	475	1870	5934	336	1.7	**105173465**			1	7.2
I015	48833	159974	2493	16371	50858	2142	25.4	**592240832**			7	7167.2
I016	72038	230110	4391	23236	70546	3388	48.0	1110879760	1110921290	0.0	1	>7205.0
I017	15095	48182	478	3500	10972	391	2.8	**109739695**			1	22.6
I018	31121	102226	1898	10360	31530	1571	11.9	**463887832**			3	2289.2
I019	25946	83290	866	8703	28128	747	4.9	**217647693**			3	563.3
I020	21808	69842	594	4230	13532	513	2.9	**146515460**			1	83.8
I021	16013	50538	392	3097	10192	298	2.3	**106470644**			1	21.5
I022	16224	51382	437	3857	12068	355	2.7	**106799980**			1	22.7
I023	22805	70614	582	4315	13278	437	2.9	**131044872**			1	31.0
I024	68464	217464	3001	26094	81048	2566	38.0	758479100	758484240	0.0	1	>7202.3
I025	23412	75904	945	7573	23930	848	4.2	**232790758**			3	1001.4
I026	47429	158614	3334	14589	44204	2640	24.2	**928032223**			7	5191.4
I027	85085	277776	3954	33300	103702	3537	66.1	976783461	976821921	0.0	1	>7220.3
I028	72701	230860	1790	37098	116948	1674	34.4	384026141	384055351	0.0	1	>7200.1
I029	69988	223608	2162	29051	91656	2026	26.3	492190908	492197789	0.0	1	>7242.0
I030	33188	107360	1263	9217	29282	1077	6.7	**321646787**			3	563.8
I031	54351	176422	2182	16397	51858	1853	24.0	**578284709**			5	2486.9
I032	56023	182798	3017	16513	50380	2435	33.8	773096540	773096720	0.0	4	>7201.3
I033	18555	59460	636	4073	12460	559	3.0	**134461857**			1	91.3
I034	22311	71032	735	6006	19008	639	3.9	**165115148**			1	105.0
I035	30585	100908	1704	10392	31946	1420	8.3	**414440370**			27	625.7
I036	37208	120712	1411	13410	42622	1278	11.3	375260654	375261017	0.0	6	>7216.8
I037	13694	44252	427	4003	12906	390	2.4	**105720727**			1	73.5
I038	18747	61278	967	5883	18274	786	3.3	**255767543**			7	845.3

cont. next page

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
I039	8755	28898	347	2566	7958	314	1.8	**85566290**			1	28.1
I040	40389	131640	1762	14850	46752	1480	11.0	431490471	431504580	0.0	1	>7200.0
I041	47197	150614	1193	17972	57698	1047	10.6	**301914840**			5	1990.4
I042	51896	171100	2171	19763	62066	1972	21.8	532128593	532131496	0.0	1	>7229.5
I043	10398	33574	367	3200	10126	327	1.6	**95722094**			1	38.3
I044	68905	227778	3358	25796	80104	2999	47.9	804499605	804538034	0.0	1	>7215.8
I045	14685	46932	421	4965	15742	390	2.4	**105944062**			1	52.9
I046	70843	234418	3598	25770	80572	3152	46.4	925441426	925473901	0.0	1	>7200.0
I047	28524	92502	2354	8528	25526	1695	6.6	**695163406**			3	1886.4
I048	13189	42438	358	3418	10906	330	2.1	**91509264**			1	44.0
I049	30857	99182	990	11239	36292	834	7.2	**294811505**			1	1240.8
I050	43073	142552	2868	14736	44958	2226	18.0	792589745	792605078	0.0	1	>7203.3
I051	27028	90812	1524	9912	30402	1344	7.8	**357230839**			37	2872.0
I052	2363	7522	40	0	0	0	0.0	**13309487**			1	0.0
I053	3224	10570	126	433	1320	89	0.3	**30854904**			1	1.1
I054	3803	12426	38	204	632	29	0.1	**15841596**			1	0.1
I055	13332	43160	570	3225	10066	463	2.3	**144164924**			1	27.4
I056	1991	6352	51	0	0	0	0.0	**14171206**			1	0.0
I057	33231	110298	1569	11142	34578	1366	8.9	**412746415**			1	893.2
I058	23527	79256	1256	5915	18378	1008	4.8	**305024188**			1	133.8
I059	9287	29950	363	1576	4854	287	1.2	**107617854**			1	6.1
I060	42008	135144	1242	14838	47920	1199	10.8	**337290460**			1	2013.8
I061	39160	127318	1458	18198	57156	1329	10.5	363042657	363042722	0.0	9	>7201.1
I062	66048	220982	3343	17331	55012	2760	38.5	**792941137**			9	5654.5
I063	26840	87322	1645	7294	22254	1239	6.0	**459801704**			13	1035.7
I064	63158	214690	3458	27661	84210	3188	43.4	863037799	863120966	0.0	1	>7200.0
I065	3898	12712	144	812	2522	117	0.7	**32965718**			1	2.2
I066	15038	49192	551	2732	8708	425	2.3	**174219813**			1	16.3
I067	20547	66460	627	7945	25222	569	4.3	**175540750**			1	380.9
I068	33118	110254	1553	9019	27914	1281	8.1	**420730046**			3	744.2
I069	9574	32416	543	2731	8312	455	1.4	**135161583**			1	44.7
I070	15079	49216	550	4956	15844	510	2.0	**136700139**			1	243.0
I071	33203	108854	1494	9631	29822	1291	11.0	**382539099**			1	348.0
I072	26948	88388	993	7870	25248	848	6.1	**289019226**			3	246.1
I073	21653	70342	1847	6106	18106	1274	4.5	**663004987**			1	354.5
I074	13316	44066	653	3118	9648	519	1.6	**165573383**			1	17.0
I075	57551	190762	2973	17983	55874	2487	27.4	**815404026**			11	4786.3
I076	14023	45790	598	3472	10934	489	1.9	**166249692**			1	26.8
I077	20856	68474	1787	7821	23236	1467	5.4	**472503150**			1	1644.0
I078	13294	43896	835	4874	14766	696	2.4	**185525490**			7	209.7
I079	19867	62542	565	5853	18694	520	3.5	**150506933**			1	818.5
I080	18695	59416	548	5530	17452	515	4.3	**164299652**			1	273.0
I081	25081	81478	888	8199	25950	746	4.3	**247527679**			1	276.8
I082	15592	49576	515	3951	12476	437	2.3	**147407632**			1	20.4
I083	89596	297166	4991	26313	80678	4045	89.7	1405586980	1405595600	0.0	1	>7200.0
I084	44934	147454	2319	12753	39698	1883	16.6	**627187559**			5	4346.2
I085	9113	28982	301	1727	5422	242	1.3	**80628079**			1	8.3

Table 21. Detailed computational results for the SAP, test set gene.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
gene41x	335	910	43	27	68	9	0.1	**126**	1	0.1
gene42	335	912	43	31	80	11	0.1	**126**	1	0.1
gene61a	395	1024	82	28	78	11	0.1	**205**	1	0.1
gene61b	570	1616	82	31	86	10	0.1	**199**	1	0.1
gene61c	549	1580	82	102	304	39	0.2	**196**	1	0.2
gene61f	412	1104	82	37	110	17	0.2	**198**	1	0.2
gene425	425	1108	86	29	82	12	0.2	**214**	1	0.2
gene442	442	1188	86	38	114	18	0.2	**207**	1	0.2
gene575	575	1648	86	22	58	8	0.2	**207**	1	0.2
gene602	602	1716	86	41	116	15	0.2	**209**	1	0.2

cont. next page

	Original			Presolved						
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	t [s]	Optimum	N	t [s]

Table 22. Detailed computational results for the SAP, test set gene2002.

	Original			Presolved						
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	t [s]	Optimum	N	t [s]
microtri1	347	952	47	28	76	9	0.0	**128**	1	0.1
microtri3	400	1112	47	30	74	9	0.0	**146**	1	0.0
microtri5	416	1124	47	44	126	17	0.1	**150**	1	0.1
microtri6	419	1164	47	30	74	9	0.0	**146**	1	0.1
microtri7	437	1172	47	24	62	8	0.1	**159**	1	0.1
microtri8	484	1412	47	80	218	24	0.1	**151**	1	0.1
microtri9	297	792	47	30	78	10	0.0	**131**	1	0.0
microtri10	319	836	47	35	98	15	0.0	**136**	1	0.0
microtri11	382	1024	47	25	70	10	0.1	**152**	1	0.1

Table 23. Detailed computational results for the RSMTP, test set estein40.

	Original			Presolved								
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	t [s]	Dual	Primal	Gap %	N	t [s]
estein40-0	1600	6240	40	1362	5462	40	1.2	**4.484154**			1	80.0
estein40-10	1600	6240	40	1369	5452	40	1.6	**4.673421**			1	422.2
estein40-11	1600	6240	40	867	3424	40	1.2	**4.384339**			1	5.5
estein40-12	1600	6240	40	1371	5334	40	1.3	**5.188453**			1	227.2
estein40-13	1600	6240	40	1576	6188	40	0.9	**4.916698**			1	191.7
estein40-14	1600	6240	40	1512	6002	40	1.1	**5.082803**			1	419.7
estein40-1	1600	6240	40	1291	5174	40	2.1	**4.681131**			1	261.0
estein40-2	1600	6240	40	1466	5834	40	1.4	**4.997415**			1	689.5
estein40-3	1600	6240	40	1337	5330	40	1.7	**4.528989**			1	317.4
estein40-4	1600	6240	40	1543	6096	40	1.0	5.18228937	5.194038	0.2	144	>7200.1
estein40-5	1600	6240	40	1403	5574	40	1.3	**4.97534**			1	296.0
estein40-6	1600	6240	40	1394	5538	40	1.4	**4.563901**			1	169.5
estein40-7	1600	6240	40	1397	5548	40	0.7	**4.874601**			1	314.8
estein40-8	1600	6240	40	1548	6108	40	1.2	**5.176179**			1	1923.4
estein40-9	1600	6240	40	1547	6116	40	0.9	**5.713686**			1	743.9

Table 24. Detailed computational results for the RSMTP, test set estein50.

	Original			Presolved								
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	t [s]	Dual	Primal	Gap %	N	t [s]
estein50-0	2500	9800	50	2475	9744	50	1.1	**5.494867**			1	1570.1
estein50-10	2500	9800	50	2471	9718	50	1.4	**5.253293**			1	4541.1
estein50-11	2500	9800	50	2397	9460	50	1.1	5.32773868	5.34093	0.2	149	>7200.0
estein50-12	2500	9800	50	2401	9464	50	1.4	**5.389099**			1	2006.4
estein50-13	2500	9800	50	2436	9612	50	2.2	**5.355143**			1	3005.2
estein50-14	2500	9800	50	2427	9586	50	1.4	**5.218085**			1	1190.2
estein50-1	2500	9800	50	2442	9672	50	1.3	**5.548422**			1	3080.2
estein50-2	2500	9800	50	2313	9218	50	2.1	**5.469105**			1	2605.8
estein50-3	2500	9800	50	2222	8850	50	1.6	**5.153576**			1	445.2
estein50-4	2500	9800	50	2129	8458	50	2.2	**5.518601**			1	631.7
estein50-5	2500	9800	50	2426	9598	50	1.4	**5.58043**			1	3470.8
estein50-6	2500	9800	50	2443	9650	50	2.4	4.96438487	5.000242	0.7	1	>7200.1
estein50-7	2500	9800	50	2325	9194	50	2.0	**5.375465**			1	699.2
estein50-8	2500	9800	50	2441	9670	50	1.4	**5.345677**			7	4239.8
estein50-9	2500	9800	50	2472	9738	50	1.2	**5.403795**			1	2830.3

Table 25. Detailed computational results for the RSMTP, test set estein60.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
estein60-0	3600	14160	60	3438	13596	60	2.0	**5.376143**			1	4188.3
estein60-10	3600	14160	60	3566	14082	60	1.7	**5.614167**			1	6480.2
estein60-11	3600	14160	60	3573	14100	60	2.5	5.88235049	5.979133	1.6	1	>7200.0
estein60-12	3600	14160	60	3573	14102	60	1.8	6.03294593	6.121356	1.4	1	>7200.3
estein60-13	3600	14160	60	3575	14110	60	2.0	**5.603556**			1	6003.0
estein60-14	3600	14160	60	3559	14052	60	1.6	**5.662257**			1	5588.5
estein60-1	3600	14160	60	3508	13906	60	3.2	5.5143905	5.536782	0.4	1	>7201.6
estein60-2	3600	14160	60	3534	13964	60	2.0	5.64039151	5.656678	0.3	1	>7201.4
estein60-3	3600	14160	60	3573	14102	60	3.1	5.48713677	5.542169	1.0	1	>7200.1
estein60-4	3600	14160	60	3539	13996	60	1.4	5.462872	5.470499	0.1	82	>7200.0
estein60-5	3600	14160	60	3573	14092	60	2.1	6.02892818	6.042196	0.2	1	>7200.1
estein60-6	3600	14160	60	3555	14058	60	1.9	5.83360294	5.897848	1.1	1	>7200.2
estein60-7	3600	14160	60	3565	14094	60	1.8	**5.813816**			1	6749.6
estein60-8	3600	14160	60	3568	14096	60	2.0	5.57171307	5.587713	0.3	1	>7201.9
estein60-9	3600	14160	60	3570	14104	60	1.5	**5.762446**			1	4019.2

Table 26. Detailed computational results for the RSMTP, test set solids.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
cube	8	24	8	0	0	0	0.0	**7**			1	0.0
dodecahedron	343	1764	20	317	1642	20	0.2	7.65777665	7.69398	0.5	8435	>7200.0
icosahedron	125	600	12	90	436	12	0.0	**20.944264**			27	1.0
octahedron	27	108	6	0	0	0	0.0	**6**			1	0.0
tetrahedron	18	66	4	0	0	0	0.0	**2.682521**			1	0.0

Table 27. Detailed computational results for the RSMTP, test set cancer.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
cancer10_6D	10000	94000	82	0	0	0	0.5	**92**			1	0.5
cancer11_8D	4762800	64777860	75	1389461	16249270	51	642.6	–	185	100	1	>12349.8
cancer12_8D	918750	12031250	58	2282	13394	33	67.8	**113**			1	82.9
cancer13_8D	86400	1039680	70	0	0	0	3.8	**88**			1	3.8
cancer14_8D	27648	308736	54	0	0	0	1.3	**63**			1	1.3
cancer1_4D	600	3820	20	0	0	0	0.0	**28**			1	0.0
cancer2_4D	256	1536	20	0	0	0	0.0	**21**			1	0.0
cancer3_6D	20580	197078	110	331	1568	33	1.0	**146**			1	1.4
cancer4_6D	34560	340416	93	6585	47340	26	2.1	**136**			1	786.1
cancer5_6D	8000	74400	48	312	1614	16	1.5	**69**			1	2.7
cancer6_6D	5120	46592	50	0	0	0	0.2	**55**			1	0.2
cancer7_6D	21000	203300	109	522	2788	29	1.0	**140**			1	2.1
cancer8_6D	8640	80064	77	0	0	0	0.3	**89**			1	0.3
cancer9_6D	6000	54800	46	0	0	0	0.3	**59**			1	0.3

Final/

Table 28. Detailed computational results for the PCSTP, test set JMP.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
K100-10	115	722	15	3	6	2	0.0	**133567**	1	0.0
K100-1	112	762	12	3	6	2	0.0	**124108**	1	0.0
K100-2	114	756	14	3	6	2	0.0	**200262**	1	0.0
K100-3	111	874	11	3	6	2	0.0	**115953**	1	0.0
K100-4	111	788	11	3	6	2	0.0	**87498**	1	0.0
K100-5	117	812	17	3	6	2	0.0	**119078**	1	0.0
K100-6	112	680	12	3	6	2	0.0	**132886**	1	0.0
K100-7	114	708	14	3	6	2	0.0	**172457**	1	0.0
K100-8	116	776	16	3	6	2	0.0	**210869**	1	0.0
K100-9	112	732	12	3	6	2	0.0	**122917**	1	0.0

cont. next page

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
K100-0	115	786	15	3	6	2	0.0	**135511**	1	0.0
K200-0	234	1580	34	3	6	2	0.0	**329211**	1	0.0
K400-10	450	3308	50	3	6	2	0.0	**394191**	1	0.0
K400-1	465	3324	65	3	6	2	0.0	**490771**	1	0.0
K400-2	462	3420	62	17	68	8	0.0	**477073**	1	0.1
K400-3	456	3314	56	3	6	2	0.0	**415328**	1	0.0
K400-4	456	3182	56	3	6	2	0.0	**389451**	1	0.0
K400-5	477	3368	77	3	6	2	0.0	**519526**	1	0.0
K400-6	456	3482	56	3	6	2	0.0	**374849**	1	0.0
K400-7	468	3286	68	3	6	2	0.0	**474466**	1	0.0
K400-8	461	3392	61	3	6	2	0.0	**418614**	1	0.0
K400-9	454	3318	54	3	6	2	0.0	**383105**	1	0.0
K400-0	463	3402	63	3	6	2	0.0	**350093**	1	0.0
P100-1	133	760	33	3	6	2	0.0	**926238**	1	0.0
P100-2	127	750	27	3	6	2	0.0	**401641**	1	0.0
P100-3	125	776	25	3	6	2	0.0	**659644**	1	0.0
P100-4	133	760	33	3	6	2	0.0	**827419**	1	0.0
P100-0	134	832	34	3	6	2	0.0	**803300**	1	0.0
P200-0	249	1462	49	3	6	2	0.0	**1317874**	1	0.0
P400-1	521	3144	121	3	6	2	0.1	**2808440**	1	0.1
P400-2	508	3034	108	3	6	2	0.1	**2518577**	1	0.1
P400-3	514	3028	114	52	212	21	0.1	**2951725**	1	0.2
P400-4	495	2852	95	3	6	2	0.1	**2852956**	1	0.1
P400-0	495	2964	95	3	6	2	0.1	**2459904**	1	0.1

Table 29. Detailed computational results for the PCSTP, test set CRR.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
C01-A	506	1280	6	3	6	2	0.0	**18**	1	0.0
C01-B	506	1280	6	3	6	2	0.0	**85**	1	0.0
C02-A	511	1310	11	3	6	2	0.0	**50**	1	0.0
C02-B	511	1310	11	3	6	2	0.0	**141**	1	0.0
C03-A	584	1748	84	3	6	2	0.0	**414**	1	0.0
C03-B	584	1748	84	3	6	2	0.0	**737**	1	0.0
C04-A	626	2000	126	3	6	2	0.0	**618**	1	0.0
C04-B	626	2000	126	3	6	2	0.0	**1063**	1	0.0
C05-A	751	2750	251	3	6	2	0.0	**1080**	1	0.0
C05-B	751	2750	251	3	6	2	0.0	**1528**	1	0.0
C06-A	506	2030	6	3	6	2	0.1	**18**	1	0.1
C06-B	506	2030	6	3	6	2	0.1	**55**	1	0.1
C07-A	511	2060	11	3	6	2	0.1	**50**	1	0.1
C07-B	511	2060	11	3	6	2	0.1	**102**	1	0.1
C08-A	584	2498	84	3	6	2	0.1	**361**	1	0.1
C08-B	584	2498	84	3	6	2	0.1	**500**	1	0.1
C09-A	626	2750	126	3	6	2	0.1	**533**	1	0.1
C09-B	626	2750	126	3	6	2	0.2	**694**	1	0.2
C10-A	751	3500	251	3	6	2	0.1	**859**	1	0.1
C10-B	751	3500	251	3	6	2	0.1	**1069**	1	0.1
C11-A	506	5030	6	3	6	2	0.1	**18**	1	0.1
C11-B	506	5030	6	3	6	2	0.1	**32**	1	0.1
C12-A	511	5060	11	3	6	2	0.1	**38**	1	0.1
C12-B	511	5060	11	3	6	2	0.1	**46**	1	0.1
C13-A	584	5498	84	3	6	2	0.2	**236**	1	0.2
C13-B	584	5498	84	3	6	2	0.2	**258**	1	0.2
C14-A	626	5750	126	3	6	2	0.2	**293**	1	0.2
C14-B	626	5750	126	3	6	2	0.2	**318**	1	0.2
C15-A	751	6500	251	3	6	2	0.1	**501**	1	0.1
C15-B	751	6500	251	3	6	2	0.1	**551**	1	0.1
C16-A	506	25030	6	3	6	2	0.7	**11**	1	0.7

cont. next page

Instance	Original			Presolved						
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
C16-B	506	25030	6	3	6	2	0.7	**11**	1	0.7
C17-A	511	25060	11	3	6	2	0.7	**18**	1	0.7
C17-B	511	25060	11	3	6	2	0.6	**18**	1	0.6
C18-A	584	25498	84	269	1418	46	1.0	**111**	1	1.1
C18-B	584	25498	84	378	2246	48	1.0	**113**	1	1.2
C19-A	626	25750	126	3	6	2	0.7	**146**	1	0.7
C19-B	626	25750	126	3	6	2	0.6	**146**	1	0.6
C20-A	751	26500	251	3	6	2	0.5	**266**	1	0.5
C20-B	751	26500	251	3	6	2	0.5	**267**	1	0.5
D01-A	1006	2530	6	3	6	2	0.1	**18**	1	0.1
D01-B	1006	2530	6	3	6	2	0.0	**106**	1	0.0
D02-A	1011	2560	11	3	6	2	0.1	**50**	1	0.1
D02-B	1011	2560	11	3	6	2	0.1	**218**	1	0.1
D03-A	1168	3502	168	3	6	2	0.0	**807**	1	0.0
D03-B	1168	3502	168	3	6	2	0.1	**1509**	1	0.1
D04-A	1251	4000	251	3	6	2	0.0	**1203**	1	0.0
D04-B	1251	4000	251	3	6	2	0.1	**1881**	1	0.1
D05-A	1501	5500	501	3	6	2	0.1	**2157**	1	0.1
D05-B	1501	5500	501	3	6	2	0.2	**3135**	1	0.2
D06-A	1006	4030	6	3	6	2	0.1	**18**	1	0.1
D06-B	1006	4030	6	3	6	2	0.1	**67**	1	0.1
D07-A	1011	4060	11	3	6	2	0.1	**50**	1	0.1
D07-B	1011	4060	11	3	6	2	0.1	**103**	1	0.1
D08-A	1168	5002	168	3	6	2	0.2	**755**	1	0.2
D08-B	1168	5002	168	3	6	2	0.3	**1036**	1	0.3
D09-A	1251	5500	251	3	6	2	0.2	**1070**	1	0.2
D09-B	1251	5500	251	3	6	2	0.3	**1420**	1	0.3
D10-A	1501	7000	501	3	6	2	0.2	**1671**	1	0.2
D10-B	1501	7000	501	3	6	2	0.3	**2079**	1	0.3
D11-A	1006	10030	6	3	6	2	0.3	**18**	1	0.3
D11-B	1006	10030	6	3	6	2	0.3	**29**	1	0.3
D12-A	1011	10060	11	3	6	2	0.3	**42**	1	0.3
D12-B	1011	10060	11	3	6	2	0.3	**42**	1	0.3
D13-A	1168	11002	168	84	352	31	0.8	**445**	1	0.8
D13-B	1168	11002	168	3	6	2	0.4	**486**	1	0.4
D14-A	1251	11500	251	3	6	2	0.5	**602**	1	0.5
D14-B	1251	11500	251	3	6	2	0.4	**665**	1	0.4
D15-A	1501	13000	501	3	6	2	0.4	**1042**	1	0.4
D15-B	1501	13000	501	3	6	2	1.1	**1108**	1	1.1
D16-A	1006	50030	6	3	6	2	1.1	**13**	1	1.1
D16-B	1006	50030	6	3	6	2	1.1	**13**	1	1.1
D17-A	1011	50060	11	3	6	2	1.2	**23**	1	1.2
D17-B	1011	50060	11	3	6	2	1.1	**23**	1	1.1
D18-A	1168	51002	168	547	3026	92	2.0	**218**	1	2.4
D18-B	1168	51002	168	837	5550	95	2.3	**223**	1	3.7
D19-A	1251	51500	251	502	2582	95	2.0	**306**	1	2.5
D19-B	1251	51500	251	765	4494	98	2.4	**310**	1	3.0
D20-A	1501	53000	501	3	6	2	1.6	**536**	1	1.6
D20-B	1501	53000	501	3	6	2	1.9	**537**	1	1.9

Table 30. Detailed computational results for the PCSTP, test set PUCNU.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
bip42nu	1401	9164	201	1191	8420	201	1.7	223.804925	226	1.0	4063	>7200.1
bip52nu	2401	17194	201	2020	15852	201	2.0	219.793866	223	1.4	2262	>7200.0
bip62nu	1401	21204	201	1400	21200	201	2.0	210.047825	216	2.8	29	>7200.0
bipa2nu	3601	37946	301	3441	37390	301	6.7	320.113009	329	2.7	1	>7200.0
bipe2nu	601	10326	51	601	10326	51	0.5	**53**			9	50.0
cc10-2nu	1160	11050	136	1066	9872	89	0.3	165.703651	169	2.0	25	>7200.4
cc11-2nu	2293	23990	245	2123	21678	160	1.2	300.604788	307	2.1	1	>7206.9
cc12-2nu	4570	51986	474	4220	46846	299	2.4	558.870723	568	1.6	1	>7200.9
cc3-10nu	1051	27300	51	1051	16500	51	0.7	**61**			110	754.9

cont. next page

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
cc3-11nu	1393	40296	62	1393	23826	62	0.8	**79**			57	2815.9
cc3-12nu	1803	57468	75	1803	33048	75	1.1	**95**			39	4226.7
cc3-4nu	73	624	9	3	6	2	0.0	**10**			1	0.0
cc3-5nu	139	1578	14	138	996	14	0.0	**17**			1	1.2
cc5-3nu	271	2592	28	267	2282	26	0.1	**36**			1	14.1
cc6-2nu	77	456	13	3	6	2	0.0	**15**			1	0.0
cc6-3nu	806	9192	77	736	7268	42	0.3	**95**			1	34.5
cc7-3nu	2410	31948	223	2250	26534	143	1.1	268.372944	273	1.7	1	>7200.4
cc9-2nu	577	4992	65	567	4874	60	0.2	**83**			7	426.4

Table 31. Detailed computational results for the RPCSTP, test set cologne1.

Instance	Original			Presolved						
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
i101M1	758	12704	11	0	0	0	0.1	**109271.503**	1	0.1
i101M2	758	12704	11	0	0	0	0.2	**315925.31**	1	0.2
i101M3	758	12704	11	0	0	0	0.2	**355625.409**	1	0.2
i102M1	760	12730	12	0	0	0	0.1	**104065.801**	1	0.1
i102M2	760	12730	12	0	0	0	0.2	**352538.819**	1	0.2
i102M3	760	12730	12	0	0	0	0.2	**454365.927**	1	0.2
i103M1	764	12738	14	0	0	0	0.1	**139749.407**	1	0.1
i103M2	764	12738	14	0	0	0	0.2	**407834.228**	1	0.2
i103M3	764	12738	14	0	0	0	0.2	**456125.488**	1	0.2
i104M2	744	12598	4	0	0	0	0.1	**89920.8353**	1	0.1
i104M3	744	12598	4	0	0	0	0.2	**97148.789**	1	0.2
i105M1	744	12604	4	0	0	0	0.1	**26717.2025**	1	0.1
i105M2	744	12604	4	0	0	0	0.1	**100269.619**	1	0.1
i105M3	744	12604	4	0	0	0	0.2	**110351.163**	1	0.2

Table 32. Detailed computational results for the RPCSTP, test set cologne2.

Instance	Original			Presolved						
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
i201M2	1812	33522	10	0	0	0	0.5	**355467.684**	1	0.5
i201M3	1812	33522	10	0	0	0	0.6	**628833.614**	1	0.6
i201M4	1812	33522	10	0	0	0	0.7	**773398.303**	1	0.7
i202M2	1814	33520	11	0	0	0	0.5	**288946.832**	1	0.5
i202M3	1814	33520	11	0	0	0	0.6	**419184.159**	1	0.6
i202M4	1814	33520	11	0	0	0	0.6	**430034.264**	1	0.6
i203M2	1824	33584	16	0	0	0	0.5	**459894.776**	1	0.5
i203M3	1824	33584	16	0	0	0	0.6	**643062.02**	1	0.6
i203M4	1824	33584	16	0	0	0	0.6	**677733.067**	1	0.6
i204M2	1805	33454	5	0	0	0	0.5	**161700.545**	1	0.5
i204M3	1805	33454	5	0	0	0	0.6	**245287.203**	1	0.6
i204M4	1805	33454	5	0	0	0	0.6	**245287.203**	1	0.6
i205M2	1823	33640	14	0	0	0	0.5	**571031.415**	1	0.5
i205M3	1823	33640	14	0	0	0	0.6	**672403.143**	1	0.6
i205M4	1823	33640	14	0	0	0	0.6	**713973.623**	1	0.6

Table 33. Detailed computational results for the MWCSP, test set ACTMOD.

Instance	Original			Presolved					
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	t [s]	Optimum	N	t [s]
drosophila001	5298	187214	72	3	6	0.8	**24.3855064**	1	0.8
drosophila005	5421	187952	195	37	134	1.4	**178.663952**	1	1.4
drosophila0075	5477	188288	251	3	6	1.1	**260.523557**	1	1.1
HCMV	3919	58916	56	3	6	0.2	**7.55431486**	1	0.2
lymphoma	2102	15914	68	3	6	0.1	**70.1663087**	1	0.1
metabol_expr_mice_1	3674	9590	151	3	6	0.1	**544.94837**	1	0.1

cont. next page

	Original			Presolved					
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	t [s]	Optimum	N	t [s]
metabol_expr_mice_2	3600	9174	86	3	6	0.0	**241.077524**	1	0.0
metabol_expr_mice_3	2968	7354	115	3	6	0.1	**508.260877**	1	0.1

Table 34. Detailed computational results for the MWCSP, test set JMPALMK.

	Original			Presolved					
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	t [s]	Optimum	N	t [s]
10-0-a-0-6-d-0-25-e-0-25	1193	11024	193	3	6	0.1	**931.538552**	1	0.1
10-0-a-0-6-d-0-25-e-0-5	1388	12194	388	3	6	0.1	**1872.2754**	1	0.1
10-0-a-0-6-d-0-25-e-0-75	1564	13250	564	3	6	0.5	**2789.57911**	1	0.5
10-0-a-0-6-d-0-5-e-0-25	1114	10550	114	3	6	0.1	**522.525615**	1	0.1
10-0-a-0-6-d-0-5-e-0-5	1250	11366	250	3	6	0.0	**1197.85102**	1	0.0
10-0-a-0-6-d-0-5-e-0-75	1374	12110	374	3	6	0.0	**1762.70747**	1	0.0
10-0-a-0-6-d-0-75-e-0-25	1062	10238	62	3	6	0.1	**332.791924**	1	0.1
10-0-a-0-6-d-0-75-e-0-5	1141	10712	141	3	6	0.1	**754.300601**	1	0.1
10-0-a-0-6-d-0-75-e-0-75	1196	11042	196	3	6	0.1	**998.215414**	1	0.1
10-0-a-1-d-0-25-e-0-25	1193	27710	193	3	6	0.0	**939.39337**	1	0.0
10-0-a-1-d-0-25-e-0-5	1388	28880	388	3	6	0.0	**1883.21361**	1	0.0
10-0-a-1-d-0-25-e-0-75	1564	29936	564	3	6	0.1	**2789.57911**	1	0.1
10-0-a-1-d-0-5-e-0-25	1114	27236	114	3	6	0.0	**533.4294**	1	0.0
10-0-a-1-d-0-5-e-0-5	1250	28052	250	3	6	0.0	**1205.42131**	1	0.0
10-0-a-1-d-0-5-e-0-75	1374	28796	374	3	6	0.1	**1770.27776**	1	0.1
10-0-a-1-d-0-75-e-0-25	1062	26924	62	3	6	0.0	**336.829944**	1	0.0
10-0-a-1-d-0-75-e-0-5	1141	27398	141	3	6	0.0	**760.284581**	1	0.0
10-0-a-1-d-0-75-e-0-75	1196	27728	196	3	6	0.0	**1004.19939**	1	0.0
150–a-0-6-d-0-25-e-0-25	1785	17028	285	3	6	0.1	**1333.47643**	1	0.1
150–a-0-6-d-0-25-e-0-5	2078	18786	578	3	6	0.1	**2799.67722**	1	0.1
150–a-0-6-d-0-25-e-0-75	2353	20436	853	3	6	0.2	**4230.25112**	1	0.2
150–a-0-6-d-0-5-e-0-25	1680	16398	180	3	6	0.1	**847.452011**	1	0.1
150–a-0-6-d-0-5-e-0-5	1881	17604	381	3	6	0.1	**1858.0926**	1	0.1
150–a-0-6-d-0-5-e-0-75	2060	18678	560	3	6	0.1	**2697.45876**	1	0.1
150–a-0-6-d-0-75-e-0-25	1594	15882	94	3	6	0.1	**502.17599**	1	0.1
150–a-0-6-d-0-75-e-0-5	1705	16548	205	3	6	0.1	**1089.77117**	1	0.1
150–a-0-6-d-0-75-e-0-75	1779	16992	279	3	6	0.1	**1423.61063**	1	0.1
150–a-1-d-0-25-e-0-25	1785	42758	285	3	6	0.1	**1377.0144**	1	0.1
150–a-1-d-0-25-e-0-5	2078	44516	578	3	6	0.1	**2820.05174**	1	0.1
150–a-1-d-0-25-e-0-75	2353	46166	853	3	6	0.2	**4230.25112**	1	0.2
150–a-1-d-0-5-e-0-25	1680	42128	180	3	6	0.1	**860.618961**	1	0.1
150–a-1-d-0-5-e-0-5	1881	43334	381	3	6	0.2	**1865.66289**	1	0.2
150–a-1-d-0-5-e-0-75	2060	44408	560	3	6	0.1	**2707.70001**	1	0.1
150–a-1-d-0-75-e-0-25	1594	41612	94	3	6	0.1	**502.17599**	1	0.1
150–a-1-d-0-75-e-0-5	1705	42278	205	3	6	0.1	**1089.77117**	1	0.1
150–a-1-d-0-75-e-0-75	1779	42722	279	3	6	0.1	**1423.61063**	1	0.1
50–a-0-62-d-0-25-e-0-25	590	5728	90	3	6	0.0	**460.577357**	1	0.0
50–a-0-62-d-0-25-e-0-5	696	6364	196	3	6	0.0	**992.967111**	1	0.0
50–a-0-62-d-0-25-e-0-75	788	6916	288	3	6	0.0	**1447.54452**	1	0.0
50–a-0-62-d-0-5-e-0-25	556	5524	56	3	6	0.0	**280.832378**	1	0.0
50–a-0-62-d-0-5-e-0-5	629	5962	129	3	6	0.0	**655.623217**	1	0.0
50–a-0-62-d-0-5-e-0-75	696	6364	196	3	6	0.0	**965.554694**	1	0.0
50–a-0-62-d-0-75-e-0-25	531	5374	31	3	6	0.0	**171.628785**	1	0.0
50–a-0-62-d-0-75-e-0-5	566	5584	66	3	6	0.0	**362.188212**	1	0.0
50–a-0-62-d-0-75-e-0-75	593	5746	93	3	6	0.0	**490.623986**	1	0.0
50–a-1-d-0-25-e-0-25	590	13572	90	3	6	0.0	**471.393285**	1	0.0
50–a-1-d-0-25-e-0-5	696	14208	196	3	6	0.0	**995.313181**	1	0.0
50–a-1-d-0-25-e-0-75	788	14760	288	3	6	0.0	**1447.54452**	1	0.0
50–a-1-d-0-5-e-0-25	556	13368	56	3	6	0.0	**286.920868**	1	0.0
50–a-1-d-0-5-e-0-5	629	13806	129	3	6	0.0	**661.711707**	1	0.0
50–a-1-d-0-5-e-0-75	696	14208	196	3	6	0.0	**965.554694**	1	0.0

cont. next page

	Original			Presolved					
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	t [s]	Optimum	N	t [s]
50–a-1-d-0-75-e-0-25	531	13218	31	3	6	0.0	**171.628785**	1	0.0
50–a-1-d-0-75-e-0-5	566	13428	66	3	6	0.0	**362.188212**	1	0.0
50–a-1-d-0-75-e-0-75	593	13590	93	3	6	0.0	**490.623986**	1	0.0
750-a-0-647-d-0-25-e-0-25	891	9278	141	3	6	0.0	**702.644057**	1	0.0
750-a-0-647-d-0-25-e-0-5	1041	10178	291	3	6	0.0	**1419.77986**	1	0.0
750-a-0-647-d-0-25-e-0-75	1176	10988	426	3	6	0.0	**2116.58233**	1	0.0
750-a-0-647-d-0-5-e-0-25	830	8912	80	3	6	0.0	**403.177763**	1	0.0
750-a-0-647-d-0-5-e-0-5	939	9566	189	3	6	0.0	**946.129495**	1	0.0
750-a-0-647-d-0-5-e-0-75	1036	10148	286	3	6	0.0	**1382.77203**	1	0.0
750-a-0-647-d-0-75-e-0-25	799	8726	49	3	6	0.0	**266.983922**	1	0.0
750-a-0-647-d-0-75-e-0-5	856	9068	106	3	6	0.0	**580.407832**	1	0.0
750-a-0-647-d-0-75-e-0-75	895	9302	145	3	6	0.0	**764.156726**	1	0.0
750-a-1-d-0-25-e-0-25	891	20484	141	3	6	0.0	**708.143835**	1	0.0
750-a-1-d-0-25-e-0-5	1041	21384	291	3	6	0.0	**1426.44904**	1	0.0
750-a-1-d-0-25-e-0-75	1176	22194	426	3	6	0.0	**2116.58233**	1	0.0
750-a-1-d-0-5-e-0-25	830	20118	80	3	6	0.0	**403.177763**	1	0.0
750-a-1-d-0-5-e-0-5	939	20772	189	3	6	0.0	**946.129495**	1	0.0
750-a-1-d-0-5-e-0-75	1036	21354	286	3	6	0.0	**1382.77203**	1	0.0
750-a-1-d-0-75-e-0-25	799	19932	49	3	6	0.0	**266.983922**	1	0.0
750-a-1-d-0-75-e-0-5	856	20274	106	3	6	0.0	**580.407832**	1	0.0
750-a-1-d-0-75-e-0-75	895	20508	145	3	6	0.0	**764.156726**	1	0.0

Table 35. Detailed computational results for the STP, test set TreeFam.

Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	Dual	Primal	Gap %	N	t [s]
TF101057-t1	52	2652	35	**infeasible**			1	0.1
TF101057-t3	52	2652	35	**2756**			1027	21.7
TF101125-t1	304	92112	155	**infeasible**			1	5.6
TF101125-t3	304	92112	155	55083.0778	55338	0.5	585	>7200.4
TF101202-t1	188	35156	72	79661.8973	80037	0.5	3680	>7200.3
TF101202-t3	188	35156	72	77859.6474	78102	0.3	13595	>7200.0
TF102003-t1	832	691392	407	194783.06	396842	50.9	2	>7220.7
TF102003-t3	832	691392	407	181270.819	190431	4.8	3	>7258.2
TF105035-t1	237	55932	104	34857.087	47145	26.1	1210	>7200.8
TF105035-t3	237	55932	104	32858.3179	32967	0.3	6335	>7200.4
TF105272-t1	476	226100	223	135288.813	300315	55.0	59	>7202.3
TF105272-t3	476	226100	223	126694.275	132597	4.5	19	>7204.3
TF105419-t1	55	2970	24	**18668**			8720	110.2
TF105419-t3	55	2970	24	**18223**			6	0.9
TF105897-t1	314	98282	133	106872.613	179907	40.6	244	>7202.1
TF105897-t3	314	98282	133	97485.0445	98452	1.0	339	>7200.0
TF106403-t1	119	14042	46	**54124**			532	196.5
TF106403-t3	119	14042	46	**53760**			1	2.5
TF106478-t1	130	16770	54	54979.9159	55413	0.8	59877	>7200.1
TF106478-t3	130	16770	54	54765.125	54849	0.2	88211	>7200.1

Table 36. Detailed computational results for the GSTP, test set GSTP1.

	Original			Presolved								
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert T\rvert$	t [s]	Dual	Primal	Gap %	N	t [s]
gstp30f2	474	1828	30	192	726	24	0.8	**569**			1	1.1
gstp31f2	349	1284	31	312	1170	30	0.5	**635**			1	3.8
gstp33f2	452	1746	33	0	0	0	0.5	**513**			1	0.5
gstp34f2	1253	5000	34	1234	4952	34	1.3	628.611868	646	2.7	67	>7200.0
gstp36f2	442	1672	36	410	1552	36	1.0	**610**			1	6.5
gstp37f2	1054	4216	37	1044	4190	37	1.1	**485**			219	4520.8
gstp38f2	618	2504	38	590	2416	38	1.3	**656**			61	601.6
gstp39f2	707	3310	39	700	3294	39	1.1	429.130633	452	5.1	1729	>7200.0

Table 37. Detailed computational results for the GSTP, test set GSTP2.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
gstp50f2	1142	4622	50	1120	4572	50	1.8	654.179976	674	2.9	244	>7200.0
gstp55f2	1751	6804	55	1691	6680	55	2.0	849.40167	892	4.8	45	>7200.0
gstp60f2	838	3528	60	835	3522	60	1.7	1153.20028	1164	0.9	602	>7200.0
gstp64f2	1860	7380	64	1790	7218	60	1.7	903.513293	932	3.1	18	>7200.0
gstp66f2	2623	10100	66	2483	9812	62	2.3	893.440207	920	2.9	8	>7200.2
gstp73f2	1911	7308	73	1797	7044	65	2.2	1195.80281	1209	1.1	25	>7200.0
gstp76f2	1818	6990	76	1686	6696	68	1.5	1016.0365	1026	1.0	14	>7200.0
gstp78f2	2355	9384	78	2275	9204	74	2.6	1053.98605	1095	3.7	59	>7200.0
gstp83f2	3177	12530	83	3052	12272	80	2.3	835.124263	906	7.8	18	>7200.1
gstp84f2	2358	9134	84	2184	8754	74	1.7	1055.89293	1095	3.6	58	>7200.1

Table 38. Detailed computational results for the STP, test set gr12.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Optimum	N	t [s]
wo11-cr100-se10	809	7432	10	335	3390	10	0.1	**136516**	1	0.9
wo11-cr100-se11	809	7430	10	502	5168	10	0.1	**145251**	1	1.4
wo11-cr100-se1	809	7444	10	616	6622	10	0.0	**182082**	1	2.5
wo11-cr100-se2	809	7394	10	480	5070	10	0.1	**163872**	1	0.4
wo11-cr200-se10	809	15262	10	471	9550	10	0.2	**59523**	1	2.5
wo11-cr200-se11	809	15260	10	661	14210	10	0.1	**66786**	1	4.9
wo11-cr200-se1	809	15274	10	663	14582	10	0.1	**76353**	1	8.8
wo11-cr200-se2	809	15224	10	645	13726	10	0.1	**75434**	1	1.9
wo12-cr100-se10	809	9360	10	510	6256	10	0.0	**167223**	1	2.2
wo12-cr100-se11	809	9852	10	569	7056	10	0.1	**199679**	1	1.4
wo12-cr100-se1	809	9446	10	554	6812	10	0.0	**164198**	1	2.0
wo12-cr100-se7	809	9702	10	301	3702	10	0.1	**136232**	1	0.6
wo12-cr200-se9	809	28346	10	293	8588	10	0.1	**46408**	1	1.1
wo10-cr100-se0	809	14396	10	809	14396	10	0.1	**171486**	1	74.1
wo10-cr100-se10	809	14428	10	603	10548	10	0.1	**117081**	1	4.7
wo10-cr100-se11	809	14386	10	643	10942	10	0.1	**125785**	1	7.7
wo10-cr200-se7	809	44696	10	454	19898	10	0.1	**46306**	1	3.3
wo10-cr200-se8	809	44654	10	805	44392	10	0.3	**61177**	1	34.5
wo10-cr200-se9	809	44670	10	723	37912	10	0.4	**51737**	1	28.4

Table 39. Detailed computational results for the STP, test set gr14.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
wo10-cr100-se0	3209	215940	10	3209	215940	10	2.6	160723.537	178284	9.8	1	>7200.2
wo10-cr100-se11	3209	215932	10	2831	187356	10	3.0	**120466**			1	2300.5
wo10-cr200-se3	3209	643552	10	3187	635486	10	21.9	51380.983	61148	16.0	1	>7203.6
wo10-cr200-se4	3209	643414	10	3167	628838	10	16.6	50725.3688	57593	11.9	1	>7202.7
wo11-cr100-se6	3209	115502	10	2773	115502	10	1.5	206671.764	218292	5.3	7	>7200.1
wo11-cr200-se2	3209	232858	10	2684	220074	10	2.8	**71134**			1	526.0
wo11-cr200-se3	3209	233104	10	2041	158058	10	2.3	**57930**			1	189.7
wo11-cr200-se4	3209	233038	10	2763	231592	10	3.1	**63313**			1	384.6
wo12-cr100-se0	3209	153366	10	904	38542	10	0.8	**116288**			1	22.0
wo12-cr100-se5	3209	156578	10	1374	63594	10	1.2	**131631**			1	293.4
wo12-cr100-se6	3209	157214	10	2030	99256	10	1.4	**146049**			234	6697.8
wo12-cr100-se7	3209	158984	10	1349	63790	10	0.8	**122306**			1	267.2
wo12-cr100-se8	3209	157912	10	1423	68700	10	0.8	**116077**			1	627.6
wo12-cr100-se9	3209	156658	10	633	23838	10	0.7	**99170**			1	6.3
wo12-cr200-se0	3209	445774	10	1515	181670	10	2.8	**53883**			1	442.4
wo12-cr200-se10	3209	446040	10	2317	311642	10	7.1	62137.1904	72475	14.3	1	>7201.2
wo12-cr200-se11	3209	457496	10	2383	330774	10	4.8	66663.9	81213	17.9	9	>7200.5
wo12-cr200-se4	3209	460250	10	2395	329386	10	6.7	71599.6445	78710	9.0	1	>7200.7
wo12-cr200-se5	3209	456998	10	2172	280816	10	6.8	**57694**			40	7062.8
wo12-cr200-se6	3209	460500	10	2377	327552	10	9.4	60423.1974	63892	5.4	7	>7201.1

cont. next page

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
wo12-cr200-se7	3209	464090	10	1823	224594	10	4.6	60445.221	61938	2.4	317	>7200.4

Table 40. Detailed computational results for the HCDSTP, test set gr16. All instances have 10 terminals (before and after preprocessing).

	Original		Presolved							
Instance	$\|V\|$	$\|A\|$	$\|V\|$	$\|A\|$	t [s]	Dual	Primal	Gap %	N	t [s]
wo10-cr100-se0	12509	2843882	11604	2843678	6.8	67934.3155	178781	163.2	1	>7208.4
wo10-cr100-se10	12509	2844058	11319	2772610	129.4	68639.4849	122284	78.2	1	>7331.9
wo10-cr100-se6	12509	2843894	11604	2843690	6.9	69686.5234	199237	185.9	3	>7207.2
wo10-cr200-se0	12509	8741560	11604	8738884	29.7	36160	68834	90.4	1	>7231.1
wo10-cr200-se3	12509	8741850	11604	8739162	29.8	32976	59383	80.1	1	>7235.9
wo10-cr200-se4	12509	8741234	11604	8738558	29.7	34218.3333	66166	93.4	1	>7240.3
wo10-cr200-se5	12509	8740874	11604	8738198	29.7	35158	68277	94.2	1	>7244.3
wo10-cr200-se7	12509	8741906	9692	7159770	2939.8	32432.3125	46438	43.2	1	>10143.0
wo11-cr100-se0	12509	1634066	10654	1634018	3.9	92733.2864	204001	120.0	1	>7204.4
wo11-cr100-se10	12509	1633968	8811	1319422	383.4	85769.1902	124389	45.0	1	>7583.5
wo11-cr200-se2	12509	3416158	10654	3415928	8.9	45476.5249	76168	67.5	1	>7211.3
wo11-cr200-se3	12509	3416916	10449	3341260	117.3	45394.1664	57820	27.4	1	>7319.8
wo12-cr100-se2	12509	2172502	10486	2145056	5.3	96880.9782	194788	101.1	1	>7207.4
wo12-cr100-se3	12509	2173508	10426	2122636	7.9	90073.8988	151797	68.5	1	>7209.1
wo12-cr200-se2	12509	6560440	10543	6530350	22.8	43813.1724	81064	85.0	1	>7230.0
wo12-cr200-se3	12509	6557828	10494	6465210	22.8	40141.5455	62201	55.0	1	>7224.9
wo12-cr200-se4	12509	6420904	10422	6281784	19.9	43269.8722	83053	91.9	1	>7224.6
wo12-cr200-se7	12509	6766046	9903	6190724	1016.1	41470.7083	64796	56.2	1	>8231.3
wo12-cr200-se8	12509	6207724	10434	6178476	111.6	38677.7129	54757	41.6	1	>7313.8
wo12-cr200-se9	12509	6571406	9928	6168132	924.0	36254.0664	50364	38.9	1	>8124.9