Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

GERALD GAMRATH, THORSTEN KOCH, DANIEL REHFELDT,
YUJI SHINANO

SCIP-Jack – A massively parallel STP solver

This article was submitted to the 11th DIMACS Implementation Challenge on Steiner Tree Problems

ZIB Report 14-35 (September 2014)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

SCIP-Jack – A massively parallel STP solver*

Gerald Gamrath† · Thorsten Koch · Daniel Rehfeldt · Yuji Shinano

Abstract

In this article we describe the impact from embedding a 15 year old model for solving the Steiner tree problem in graphs in a state-of-the-art MIP-Framework, making the result run in a massively parallel environment and extending the model to solve as many variants as possible. We end up with a high-perfomance solver that is capable of solving previously unsolved instances and, in contrast to its predecessor, is freely available for academic research.

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 $c : E \to \mathbb{Q}^+$ and a set $T \subset V$ of *terminals*, the problem is to find a minimum weight tree $S \subseteq G$ which spans T.

While the STP is said to have numerous applications, it is hard to find a pure one in the wild. The authors have collected more than a thousand instances in the SteinLib [2], but extremly few of them originate from real-world applications. In this the STP resembles the *Traveling Salesmen Problem* for which Vasel Chvatal said in an Interview: *The traveling salesman problem is to mathematical programming what chess is to artificial intelligence: thoroughly useless and fiercely competitive sport that serves as a testing ground of your techniques.*[3] We would claim the same to be true for the STP. Nevertheless, there are many applications which include STPs as a subproblem or in some variant.

When the 11th DIMACS Challenge, dedicated to the study of Steiner tree problems, was announced, the starting point of our investigation was the model and code described in [4]. Taking this and the developments of the last 15 years, we will show:

- in Section 2 the impact of the transition from a simple self-made branch-and-cut code to the use of a full fledged, state-of-the-art MIP-Framework.
- in Section 3 how to use the versatility of MIP models to solve a class of related problem variations.
- in Section 4 the possible gains from using hundredes of CPU cores to solve a single problem.

As we will see in the end, it is worthwhile to revisit topics after some time. Further details on this can be found in [5, 6].

*This article was submitted to the 11th DIMACS Implementation Challenge on Steiner Tree Problems

†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, {gamrath, koch, rehfeldt, shinano}@zib.de

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

In this section, we present the solver SCIP-Jack. To this end, we first describe the model that we use and the simple self-made branch-and-cut code Jack-III. Then, we will show how the MIP solving framework SCIP can be extended by STP specific plugins to a full-scale branch-and-cut based STP solver which profits from the state-of-the-art MIP solving algorithms implemented within SCIP. Their impact is then analyzed and computational results are discussed.

The model we will use in the following is the *directed cut formulation*: We replace each edge $\{u, v\} \in E$ by two anti-parallel arcs (u, v) and (v, u). Let A denote this set of arcs and $D = (V, A)$ the resulting digraph. We chose some terminal $r \in T$, which we call root. A *Steiner arborescence rooted at* r is a subgraph $S \subseteq D$ such that (V_S, A_S) contains exactly one directed path from r to t for all $t \in T \setminus \{r\}$. Obviously, there is a one-to-one correspondence between (undirected) Steiner trees in G and Steiner arborescences in D which contain at most one of two anti-parallel arcs. Thus, if we choose arc weights $\vec{c}_{(u,v)} := \vec{c}_{(v,u)} := c_{\{u,v\}}$, for $\{u, v\} \in E$, the Steiner tree problem can be solved by finding the minimal Steiner Arborescence with respect to $\vec{c}$. Introducing variables y_a for $a \in A$ with the interpretation $y_a := 1$, if a is in the Steiner arborecence, and $y_a := 0$ otherwise, we obtain the integer program:

Formulation 1. *Cut Formulation*

$$\min \vec{c}^T y \tag{1}$$

$$y(\delta^+(W)) \geq 1, \forall W \subset V, r \in W, (V \setminus W) \cap T \neq \emptyset \tag{2}$$

$$y(\delta^-(v)) \begin{cases} = & 0, \textit{if } v = r; \\ = & 1, \textit{if } v \in T \setminus r; \\ \leq & 1, \textit{if } v \in N; \end{cases} \tag{3}$$

$$y(\delta^-(v)) \leq y(\delta^+(v)), \forall v \in N; \tag{4}$$

$$y(\delta^-(v)) \geq y_a, \forall a \in \delta^+(v), v \in N; \tag{5}$$

$$0 \leq y_a \leq 1, \forall a \in A; \tag{6}$$

$$y_a \in \{0, 1\}, \forall a \in A, \tag{7}$$

where $N = V \setminus T$, $\delta^+(X) := \{(u, v) \in A | u \in X, v \in V \setminus X\}$ for $X \subset V$, i.e. the set of arcs with tail in X and head in its complement. The details are given in [4]. It is shown in [7] that the flow-cuts described are indeed facets.

Since the model contains a potentially exponential number of constraints, we use a branch-and-cut approach and separate constraints during the execution of the algorithm, if they are violated. Jack-III uses this model and implements its own branch-and-bound search. It uses strong branching, a depth-first search node selection, and tries to generate feasible solutions with a primal heuristic.

Our reimplementation is based on the academic MIP solver SCIP [8]. Besides being one of the fastest non-commercial MIP solvers [9], SCIP is a framework for several branch-and-bound based algorithm types including branch-and-cut. Thereby, its plugin-based design allows to easily extend SCIP to handle specific problem classes more effectively.

In the case of SCIP-Jack, the first thing to be implemented was a reader to read in an instance and a problem data to store the graph and build the model within SCIP. Thereby, we re-used the reading methods, data structures, and preprocessing algorithms of Jack-III. The heart of the new implementation is a constraint handler which checks solutions for feasibility and separates the model constraints, if needed. Again, we re-use the separation methods of the

15-year old code, while SCIP provides a separation storage which filters the cuts and dynamic aging of the generated cuts. Additionally, we also use the general-purpose separation methods that come with SCIP, which create, e.g., Gomory or mixed-integer rounding cuts.

The preprocessing of SCIP-Jack mainly relies on STP-specific reduction techniques [4]. Those techniques strive to reduce the size of an original problem instance in terms of both vertices and edges, without changing the optimal solution value. They exert a dramatic effect on the solving time for many instances, see [10, 7]. There have been only slight changes concerning the reduction techniques implemented in SCIP-Jack compared to those of the 15 years old solver Jack-III. For more detail on the applied reduction techniques see [4]. Much remains to be done here, taking into account for example the much stronger reduction techniques deployed by [7].

The branch-and-bound search is organized by SCIP: we use the default hybrid branching rule [11] which combines strong branching and pseudo costs with history information like conflict and inference scores. Node selection is done with respect to a best estimate criterion with interleaved best bound and depth-first search phases [12].

Additionally, two STP-specific primal heuristics have been implemented, a constructive heuristic based on [13] and an improvement heuristic [14]. The constructive heuristic is both coherent and empirically successful: Starting with one vertex, in each step the current subtree is connected to a nearest terminal by a shortest path. This procedure is reiterated until all terminals are spanned. Finally, in a pruning step a minimum spanning tree is constructed on the vertices of the computed tree and nonterminals of degree one are removed. The heuristic has already been implemented in Jack-III, where it is used not only as an initial heuristic but also, with altered edge weights, during the branch-and-cut: Having solved an LP with $x \in \mathbb{Q}^{\mathbb{E}}$ being the optimal solution, the heuristic is called with the edge weights $(1 - x_e) \cdot c(e)$ for all $e \in E$. Thus, a stimulus for the heuristic to choose edges contained in the LP solution is provided. Moreover, the heuristic is started from several distinct vertices, making it empirically much more potent (100 start vertices for the intial call, 10 after each LP). For the SCIP-based solver this heuristic is used in the same way, with the exception that terminals are prefered as starting points, which has turned out to bring a slight empirical advantage. However, for problems for which at least ten percent of the vertices are terminals (after presolving) a variation based on the concept of Voronoi regions (see [7]) is used, following the same scheme but often providing a computational advantage.

The improvement heuristic is called whenever a new incumbent solution has been found. It is in fact a combination of the three local search heuristics: vertex insertion, key-path exchange and key-vertex elimination, as described in [14], with the most impact by far being achieved by the latter two. Although some alterations had to be made in order to adapt the algorithms originally designed for undirected problems to our model (Formulation 1). The basic idea of vertex insertion is to connect further vertices to an existing Steiner tree in such a way that expensive edges can be removed from it. 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, connecting two key-vertices and containing none else. In key-path exchange attempts are made to replace existing key-paths by less costly ones. 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 so disconnected subtrees at lower costs. The combination of the constructive and the local heuristic considerably helps generate good primal solutions quickly and even finds optimal solutions to many problems.

Further primal heuristics for SCIP-Jack are currently being implemented. Most promising among them is a recombination approach, where several good solutions are merged, the STP on the corresponding graph is solved and finally the improvement heuristic is applied on top. This

Table 1: Computational results for STP instances

test set	size	solved	nodes	time
SP	8	6	2.8	4.7
I640	100	65	9.4	62.8
PUC	50	8	1708.5	330.1
vienna-i-simple	85	58	1.8	2673.0
vienna-i-advanced	85	61	1.8	1727.5

approach can be implemented using SCIP without much difficulty as the latter not only stores a number of best found solutions, but also allows to create and solve a subproblem during the superordinate solving process.

2.0.1 Computational Results

We present computational experiments showing the solving capabilities of SCIP-Jack for STP instances included in the DIMACS Challenge. All computational experiments described in this and the next section were performed on a cluster of Intel Xeon E5-2670 CPUs with 2.50 GHz and 32 GB RAM, running Kubuntu 14.4. We used a development version of SCIP 3.1 (git hash `4eca267`) with SoPlex [15] version 2.0.1 (git hash `708d128`) as underlying LP solver and a time limit of 12 hours. We average over results by taking the shifted geometric mean [12] with a shift of 1.0.

Table 1 summarizes the solving capabilities of SCIP-Jack for some of the DIMACS Challenge test sets. Detailed results are presented in the appendix. The test sets SP, I640, and PUC are all relatively hard test sets from SteinLib and contain still unsolved instances. SCIP-Jack solves most of the instances of the SP test set and about two thirds of the I640 set. On average, this needs just a couple of nodes and a few minutes, although there are also instances which need a significant amount of branching or several hours to solve. The PUC test set is even harder and more than half of the instances in the set are still unsolved. SCIP-Jack can only solve eight of 50 instances, none of them at the root node. The vienna test sets [16] were newly submitted to the DIMACS Challenge. We can solve about two thirds of the instances, most of them at the root node or after a couple of branchings, but still a lot of time is spent in the separation process. Nevertheless, SCIP-Jack seems to be competitive to the results presented in [16] at least for test set vienna-i-simple.

3 From single problem to class solver

In this section several variants of the Steiner problem, as described in the previous section, are introduced and transformations to render them tractable for our solver are presented. For most of the discussed variants we show computational results in order to evaluate the effectiveness of our generic STP solver for the respective variant. Throughout this section the weights of an edge e and an arc a are denoted by $c(e)$ and $c(a)$ respectively and the weight of a vertex v by $p(v)$.

The Steiner Arborescence Problem

Since the Steiner tree solver transforms each Steiner tree problem to a Steiner arborescence problem (SAP) in the first place, as described in section 2, the branch-and-cut substruction and the constructive heuristic can be used without much furter ado. However, the same does not hold for the presolving and the local heuristics. There have (yet) been no Steiner arborescence problems published in the DIMACS Challenge, so this section goes without computational evaluations.

Nevertheless, the Steiner arborescence problem is of importance in the context of the DIMACS Challenge as several of the following Steiner variants are reduced to Steiner arborescence problems.

The Rectilinear Steiner Minimum Tree Problem

Perhaps the most famous of all Steiner problem variants, the *rectilinear Steiner minimum tree problem* (*RSMTP*) can be described as follows: given a number of $n \in N$ points in the plane, the task is to find a shortest tree consisting only of vertical and horizontal line segments and containing all n points. The *RSMTP* is $\mathcal{NP}$-hard, as proved in [17], and has been the subject of various publications, [18, 19, 20]. In addition to this two-dimensional variant, a generalisation of the problem to the d-dimensional case with $d \geq 2$ will be considered, having real-world applications in up to eight dimensions, e.g. in cancer research, see [21].

Hanan [22] proved that the RSMTP can be restricted to the grid obtained by constructing vertical and horizontal lines through each given point of the RSMTP. Subsequently, this observation and its multi-dimensional generalisation [23] is exploited in order to adapt the RSMTP to our solver: Given a $d-$dimensional, $d \in \mathbb{N} \setminus \{1\}$, RSMTP represented by a set of $n \in N$ points in $\mathbb{Q}^d$, first a $d-$dimensional Hanan grid is built. Thereafter, a STP $P = (V, E, T, c)$ is constructed in a straightforward way: The vertex set V is constructed by assigning each intersection point of the grid a vertex. Analogously, E is obtained by interconnecting the vertices V according to the grid structure, i.e. in such a way that two vertices in V are adjacent if and only if their corresponding intersection points are adjacent in the grid. The set of terminals T consists of all vertices in V corresponding to one of the original n RSMTP points. Finally, the cost of an edge $\{v, w\} \in E$ is defined by the (euclidean) distance of the two d-dimensional points in the grid corresponding to v and w respectively.

Claim 1. *Each solution to the STP obtained from a RSMTP as delineated above, can be transformed to the original RSMTP, by applying the described bijective mapping. Optimality is preserved.*

A few remarks concerning the implementation: presolving is by default not used for our solver in the case of RSMTP problems, as it performs exceptionally poorly for this structure. Also, we do not expect this simple approach to be competitive with highly specialized solvers, such as GeoSteiner [18] in the cases $d = 2$ and $d = 3$. However, in dimensions $d \geq 4$ there seems to be a lack of specialized solvers, so this approach might be of help here. 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 number of dimensions.

Finally, it shall be noted that the related *Euclidean Steiner tree problem*, which differs from the RSMTP insofar as the rectilinear distance is replaced with the Euclidean distance, cannot be handled by our solver. It seems difficult to render this problem solvable for SCIP-Jack, as a clear structure, such as the Hanan grid for the RSMTP, is ostensibly missing here.

3.0.2 Computational Results

For the DIMACS Challenge, multiple test sets of RSMTP instances were posted, which range from very small instances in two dimensions with only two or three terminals to eight-dimensional instances with 70 or more terminals.

In our experiments, we focused on 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 cancer instances with up to eight dimensions. Computational results

Table 2: Computational results for RSMTP instances

test set	size	solved	nodes	time
estein1	46	46	1.0	0.3
estein10	15	15	1.0	0.2
estein20	15	15	1.4	8.6
estein30	15	15	1.2	172.4
estein40	15	14	1.0	1216.5
estein50	15	13	1.5	5881.0
estein60	15	10	1.0	11602.2
solids	5	5	14.4	7.2
cancer	14	11	1.0	132.7

are summarized in Table 2, detailed results are listed in the appendix. We see that the estein instances of these small sizes need just a few nodes to solve, but solving times increase significantly with increasing number of nodes. The solids instances can all be solved to optimality, but the dodecahedron instance needs a significant amount of branching. Finally, the cancer instances show that SCIP-Jack can handle and solve instances of dimension up to eight. However, we also see that the grid can lead to a huge number of nodes and edges, in particular in high dimension, so that some the transformed instances could not be handled with the available memory.

The Obstacle-Avoiding Rectilinear Steiner Minimum Tree Problem

The slightly awkwardly named *obstacle-avoiding rectilinear Steiner minimum tree problem* (*OARSMTP*) is a variation of the RSMTP in such a way that the required minimum-length rectilinear tree is not allowed to go through the interior of any of the given axis-aligned rectangles, denoted *obstacles*. The Hanan grid approach applied for the RSMTP can be naturally altered to fit this new problem by just removing all vertices located in the interior of an obstacle together with their incident edges.

The Node-Weighted Steiner Tree Problem

The *node-weighted Steiner tree problem* (*NWSTP*) is a generalization of the Steiner tree problem in graphs insofar as not only the edges but also the vertices are assigned nonnegative weights. The objective is to interconnect all terminals while minimizing the weight summed over both vertices and edges spanned by the corresponding tree.
Formally stated, 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 $T \subset V$ of terminals, the objective is to find a tree $S = (V_S, E_S)$ that spans T while minimizing

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

Transforming a node-weighted Steiner tree problem in such a way that it can be solved by our solver is straightforward: First, each edge is substituted by two anti-parallel arcs, then, observing that in a tree there cannot be more than one arc going into the same vertex, we add the weight of each vertex to the weight of each of its ingoing arcs.

Transformation 1 (NWSTP to SAP)**.**
Let $P = (V, E, T, c, p)$ be a NWSTP, construct a transformed Steiner arborescence problem $P' = (V', A', T', c', r')$ 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.*

Claim 2. *Let* S' *be a solution to the SAP* $P' = (V', A', T', c')$ *obtained from a NWSTP* $P = (V, E, T, c, p)$ *by applying Transformation 1.* S' *can be reduced to a solution* S *to* P *defined by*

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

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

Optimality is preserved by the reduction.

The resulting problem is a SAP, so for SCIP-Jack the configurations described in the Steiner arborescence section are used.

3.0.3 Computational Results

Two NWSTP instances derived from a computational biology application are part of the DIMACS Challenge, which differ drastically in their size: the first instance has more than 200,000 nodes—55,000 of them terminals—and 2.4 million edges, while the smaller instance comprises 386 nodes, 1477 edges, and 35 terminals.

Solving the first instance with our generic solver is beyond hope since SCIP-Jack currently cannot apply its preprocessing routines to the transformed problem and we soon run out of memory for the complete unpreprocessed problem.

The second instance, however, proves that smaller instances can easily be solved: our constructive heuristic finds the optimal solution right away and optimality is proven after 4 rounds of separation at the root node. Altogether this needs less then 0.1 seconds.

The Prize-Collecting Steiner Tree Problem

As opposed to the classical Steiner tree problem, the required tree for the the *prize-collecting Steiner tree problem* (*PCSTP*) needs only to span a (possibly empty) subset of the terminals; however, for each terminal not contained in the tree a nonnegative penalty is charged. So the objective is to find a tree of minimum weight, given by both the sum of its edge costs and the prizes of all terminals not spanned by the tree. For a profound discussion, real-world applications and a sophisticated specialized solver, see [24].

A formal definition of the problem might be stated as follows: 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:

$$P(S) := \sum_{e \in E_S} c(e) + \sum_{v \in V \setminus V_S} p(v) \tag{8}$$

is minimized.

Before discussing the prize-collecting Steiner tree problem, we introduce a variation, the *rooted prize-collecting Steiner tree problem* (*RPCSTP*), which incorporates the additional condition that one distinguished node, denoted *root*, must be part of every feasible solution to the problem. In order to adapt this problem to our solver, the following transformation is applied:

Transformation 2 (RPCSTP to SAP)**.**
For a RPCSTP $P = (V, E, p, r)$*, construct a Steiner arborescence problem* $P' = (V', A', T', c', r')$ *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* $T' := \{t'_1, ..., t'_s\}$.

Claim 3. *Let* $P' = (V', A', T', c')$ *be a SAP obtained from a RPCSTP* $P = (V, E, c, p)$ *by applying Transformation 2. Each solution* S' *to* P' *can be reduced to a solution* S *to* P *by setting* $V_S := \{v \in V : \ v \in V'_{S'}\}$ *and* $E_S := \{\{v,w\} \in E : \ (v,w) \in A'_{S'} \text{ or } (w,v) \in A'_{S'}\}$. *If* S' *is an optimal solution to* P', S *is optimal with respect to* P.

The resulting problem is a SAP, so for SCIP-Jack the configurations described in the Steiner arborescence section are used.

The transformation used for the (unrooted) PCSTP is only a slight variation of the above.

Transformation 3 (PCSTP to SAP)**.**
Let $P = (V, E, p)$ *be a PCSTP, construct a SAP* $P' = (V', A', T', c', r')$ *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* $t_i \in T$.

Each SAP $P' = (V', A', T', c', r')$, obtained from a PCSTP $P = (V, E, c, p)$ by applying Transformation 3, that incorporates the following constraint into the cut-formulation (Formulation 1):

$$\sum_{a \in \delta^+(r'), c'(a)=0} y_a \leq 1 \tag{9}$$

is henceforth referred to as *root constrained Steiner arborescence problem.*

Claim 4. *Let* S' *be a solution to the root constrained SAP* $P' = (V', A', T', c', r')$, *obtained from a PCSTP* $P = (V, E, c, p)$ *by applying Transformation 3.* S' *can be reduced to a solution* S *to* P *defined by*

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

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

If S' *is an optimal solution to* P', *so is* S *to* P.

To enforce the root condition (9) for a root constrained SAP obtained from a PCSTP as described above, a new constraint is added to the SCIP-Jack model. Further, neither reduction techniques are currently applied by SCIP-Jack nor any of the heuristics presented so far. However, a variation of the constructive heuristic is deployed: Instead of starting from different vertices, the starting point is always the (artficial) root but 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 as before on the new graph; in each run the choice of the one edge not removed is varied. It shall be remarked, that some diligence is required to ensure that the shortest path computations are always up to date before starting a new run.

Table 3: Computational results for PCSTP instances

test set	size	solved	nodes	time
JMP	34	34	1.0	2.1
CRR	80	72	1.7	20.9
PUCNU	18	6	4.5	30.5

3.0.4 Computational Results

The currently available collection of instances for the DIMACS Challenge contains several test sets of PCSTP instances, but no RPCSTP instances.

Table 3 shows aggregated results for some of the PCSTP test sets of the DIMACS Challenge. For each test set, we list the number of instances and the number of instances solved to optimality as well as the shifted geometric mean of branch-and-bound nodes and solving time for the subset of instances solved to optimality. Test set JMP seems to be rather easy: all instances are solved without any branching in just a couple of seconds. Test set CRR is more challenging: still, most instances are solved at the root node, but some instances need a significant amount of branching—most of those cannot be solved within the time limit. The third test set we regard is the PUCNU test set. We cannot solve many of the instances of the PUC test set from which these instances originated, and this behaviour recurs for the PCSTP versions of the instances. Some of the instances are solved, mostly with only few branchings, but many cannot be solved within the time limit.

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 endowed with (possibly negative) node weights, the objective is to find a tree that maximizes the sum of its node weights. However, a transformation as described by [25] allows to transform this problem to a prize-collecting Steiner tree problem. Yet, having performed this tranformation, a great number of vertices are usually assigned strictly positive weights, which can set the stage for a strenuous PCST computation. In order to avoid this issue a different transformation is introduced here (note that at least one edge may be assumed to be assigned negative cost, otherwise the problem is reduced to finding a minimum spanning tree):

Transformation 4 (MWCSP to SAP)**.**
Let $P = (V, E, p)$ be a MWCSP, construct a SAP $P'' = (V'', A'', T'', c'', r'')$:

1. *Set $V' := V$, $A' := \{(v, w) : \{v, w\} \in E\}$,*
2. *$c' : A' \to \mathbb{Q}_{\geq 0}$ such that for $a = (v, w) \in A'$:*
   $$c'(a) = \begin{cases} -p(w), & \text{if } p(w) < 0 \\ 0, & \text{otherwise} \end{cases}$$
3. *$p' : V' \to \mathbb{Q}_{\geq 0}$ such that for $v \in V'$:*
   $$p'(v) = \begin{cases} p(v), & \text{if } p(v) > 0 \\ 0, & \text{otherwise} \end{cases}$$
4. *Perform Transformation 3 to (V', A', c', p'), slightly changed in such a way, that in step 2 instead of constructing a new arc set, A' is being used.*

Table 4: Number of terminals after transforming

instance	Transformation 4	transformation from [25]
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

Claim 5. *Let $P = (V, E, p)$ be a MWCSP instance and P'' its transformation obtained by Transformation 4 (which is a SAP incorporating the constraint (9)), then each solution S'' to P'' corresponds to a solution S to P. The latter is obtained by the set of vertices $V_S = \{v \in V : v \in V''_{S''}\}$ and the set of edges $E_S = \{\{v, w\} \in E : (v, w) \in A''_{S''} \text{ or } (w, v) \in A''_{S''}\}$. If S'' is optimal with respect to P'', so is S to P. Furthermore the objective value $C(S)$ of S can be computed out of the objective value $C''(S'')$ of S'':*

$$C(S) = -C''(S'') + \sum_{v \in V : p(v) > 0} p(v)$$

For all (real-world) DIMACS instances, Transformation 4 results in problems with fewer terminals by far compared to the tranformation described in [25], as most of the vertex weights are nonpositive, see Table 4.

3.0.5 Computational Results

We performed computational experiments on the ACTMOD test set of the DIMACS Challenge. Since the test set consists of only eight instances, we present detailed computational results in Table 5. We see that for all but the three largest instances, SCIP-Jack was able to solve the problem to optimality, the five smaller instances are solved without branching in less then 200 seconds. It should be noted that the performance of SCIP-Jack on the ACTMODPC test set which contains the same problems, but already transformed to PCSTP by the transformation described in [25], is significantly worse. Thus, at least for SCIP-Jack, our transformation is clearly superior.

Table 5. Detailed computational results for test set ACTMOD.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
drosophila001	5298	187214	72	24.8802052	23.6640647	5.1	8350	timeout
drosophila005	5421	187952	195	179.291307	113.379745	58.1	65	timeout
drosophila0075	5477	188288	251	261.034691	260.523557	0.2	93	timeout
HCMV	3919	58916	56	7.55431486	7.55431486	0.0	1	186.9
lymphoma	2102	15914	68	70.1663087	70.1663087	0.0	1	21.1
metabol_expr_mice_1	3674	9590	151	544.94837	544.94837	0.0	1	89.1
metabol_expr_mice_2	3600	9174	86	241.077524	241.077524	0.0	1	9.0
metabol_expr_mice_3	2968	7354	115	508.260877	508.260877	0.0	1	20.8

The Degree-Constrained Steiner Tree Problem

The *degree-constrained Steiner tree problem* (*DCSTP*), is a STP with an additional degree constraint for each node. The objective is to find a minimum solution to the STP such that the

degree of each node in the Steiner tree is not larger than the given limit. The DCSTP is implemented by just adding the additional degree constraints for each node as linear constraints to the directed-cut-formulation (Formulation 1).

3.0.6 Computational Results

We did computational experiments for the 20 instances in the TreeFam test set of the DIMACS Challenge. Our current constructive heuristic does not take the degree constraints into account and therefore does not find any feasible solutions. We are currently in the process of implementing a DCSTP specific primal heuristic in order to change this situation. With the current version, a primal bound is found for only 8 of the 20 instances in the test set, and only 5 instances can be solved to optimality. This shows how important good primal heuristics are in the context of STP.

Table 6. Detailed computational results for test set TreeFam.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
TF101057-t1	52	2652	35	3477.12274	–	–	519787	timeout
TF101057-t3	52	2652	35	2756	2756	0.0	1861	71.1
TF101125-t1	304	92112	155	66252.2458	–	–	1176	timeout
TF101125-t3	304	92112	155	53680.8338	–	–	3232	timeout
TF101202-t1	188	35156	72	79447.8948	–	–	10178	timeout
TF101202-t3	188	35156	72	77800.9395	78043	0.3	22467	timeout
TF102003-t1	832	691392	407	190335.559	–	–	29	timeout
TF102003-t3	832	691392	407	176175.97	–	–	100	timeout
TF105035-t1	237	55932	104	34479.267	–	–	6141	timeout
TF105035-t3	237	55932	104	32460.0283	–	–	4232	timeout
TF105272-t1	476	226100	223	131751.126	–	–	231	timeout
TF105272-t3	476	226100	223	122822.143	–	–	305	timeout
TF105419-t1	55	2970	24	18668	18668	0.0	17901	806.6
TF105419-t3	55	2970	24	18223	18223	0.0	53	12.3
TF105897-t1	314	98282	133	105499.898	–	–	1082	timeout
TF105897-t3	314	98282	133	96168.5484	–	–	2637	timeout
TF106403-t1	119	14042	46	54124	54124	0.0	853	738.9
TF106403-t3	119	14042	46	53760	53760	0.0	94	181.2
TF106478-t1	130	16770	54	55041.963	55192	0.3	88895	timeout
TF106478-t3	130	16770	54	54757.8148	54849	0.2	101916	memout

The Group Steiner Tree Problem

The *group Steiner tree problem* (*GSTP*) is another generalization of the Steiner tree problem, where 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 $T_1, ...T_s \subset V$, $s \in \mathbb{N}$, a minimum cost tree spanning at least one vertex of each subset is required. Since by substituting each terminal t by $\{t\}$ every STP can be considered as a GSTP, the latter is $\mathcal{NP}$-hard as well. Altough a generalization, each GSP instance $(V, E, T_1, .., T_S, c)$ can be transformed to a STP following the subsequent scheme: Enlarge the graph by adding for each group T_i a new vertex t_i and edges $\{t_i, v\}$ of high cost to each $v \in V$. Regarding the set $\{t_1, ..., t_s\}$ as terminals, each solution to the STP on the enlarged graph corresponds to a solution to the GSP, obtained by removing all artificial edges $\{t_i, v\}$, $i = 1, ..., s$, $v \in V$. This approach has already been deployed by [26] to solve group Steiner tree problems and proved to be competitive with specialized solvers at the time of publishing. Since no *GSTP* instances have been published in the course of the DIMACS Challenge yet and the raison d'être of the *GSTP* as a part of this paper might therefore be questioned anyway, we refrain from presenting a more formal delineation of the transformation and refer to [26] instead.

The Length-Constrained Steiner Tree Problem

The *length-constrained Steiner tree problem* (*LCSTP*) as compared to a STP incorporates the additional condition, that for a Steiner tree to be feasible the number of its edges may not exceed a predetermined bound. The cut formulation (Formulation 1) used by SCIP-Jack can be easily extended to cover this variation by adding one extra inequality bounding the sum of all (binary) arc variables (denoted by y_a). No LCSTP instances have yet been published in the DIMACS Challenge, but shall be in the near future.

4 From single core to distributed parallel

SCIP has two parallel extensions ParaSCIP [27] and FiberSCIP [28], which are built by using the *Ubiquity Generator Framework (*UG*)* [28]. We have developed libraries of them to preserve the extendability of SCIP also for ParaSCIP and FiberSCIP. In order to make a solver for a specific problem, users of SCIP need to write their own user defined plugin code. When using the libraries, this problem specific code can be used as it is for a parallelization by adding a small glue code and linking to one of the libraries.

In this way, users obtain their own problem specific parallel optimization solver, which can do parallel tree search basically on a distributed memory computing environment. The user gets the three main features of UG: several *ramp-up* mechanisms (the ramp-up is the process from the begining 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. For more details about the parallelization, see [27, 28].

This has been developed and tested for several problem specific parallel optimization solvers based on SCIP and with both libraries. The parallel version of SCIP-Jack is one of them. Therefore, we did not need to describe anything about parallelization in the previous sections and realized it by just using the libraries. However, there is one major difference in the parallelization of SCIP-Jack compared to ParaSCIP for general MIP: When a branch-and-bound node is transferred from one solver to the other via the work load coordinating process, local cuts generated within the first solver are transferred to the receiving solver. For general MIP, we only transfer bound changes of variables in the default setting.

We present computational results for the open instances of the PUC test set from SteinLib. The main purpose of the parallel runs is to provide the optimal solutions to as many instances as possible. The development of the sequential and the parallel version of SCIP-Jack had been continuing concurrently, so our results for the open instances were obtained by using the intermediate development code. When the development code made a significant improvement in the sequential version, we just restarted the computation by giving the best incumbent solution found so far and when the code change was not significant, we continued the computation from a check point of the previous run. For the computations, we used PC clusters and supercomputers as they were available. We conducted a few big jobs which used over 4,000 solvers, but still, most of the computations were conducted with less than 192 solvers. In contrast to the previous experiments, we used CPLEX 12.6 as the underlying LP solver. The biggest scale computation we did with ParaSCIP before was a 35,200 cores run on Titan which is located at Oak Ridge National Laboratory. We expect SCIP-Jack to also run on such a large scale computing environment, though we have conducted relatively small scale computational experiments so far.

Table 7 shows the improvements on the open instances of the PUC test set as of 11th September. We list the number of nodes, edges, and terminals, as well as the best known primal bound and the primal solution value obtained by our experiments with the parallel SCIP-Jack version. For 13 of the 32 unsolved instances, we were able to to improve the best known solution. In this

Table 7: Primal bound improvements for unsolved PUC instances

instance	$\|V\|$	$\|E\|$	$\|T\|$	best	SCIP-Jack	instance	$\|V\|$	$\|E\|$	$\|T\|$	best	SCIP-Jack
bip52p	2200	7997	200	24535	**24526**	cc3-12p	1728	28512	74	18838	19144
bip52u	2200	7997	200	234	234	cc3-12u	1728	28512	74	186	188
bip62p	1200	10002	200	22870	**22843**	cc6-3p	729	4368	76	20456	**20270***
bip62u	1200	10002	200	220	**219**	cc6-3u	729	4368	76	197	197*
bipa2p	3300	18073	300	35379	**35337**	cc7-3p	2187	15308	222	57088	59210
bipa2u	3300	18073	300	341	**339**	cc7-3u	2187	15308	222	552	562
cc10-2p	1024	5120	135	35379	35531	cc9-2p	512	2304	64	17296	**17221**
cc10-2u	1024	5120	135	342	345	cc9-2u	512	2304	64	167	167*
cc11-2p	2048	11263	244	63826	63968	hc10p	1024	5120	512	60494	**59839**
cc11-2u	2048	11263	244	614	623	hc10u	1024	5120	512	581	**575**
cc12-2p	4096	24574	473	121106	122910	hc11p	2048	11264	1024	119779	**119689**
cc12-2u	4096	24574	473	1179	1199	hc11u	2048	11264	1024	1154	**1153**
cc3-10p	1000	13500	50	12860	12915	hc12p	4096	24576	2048	236949	237000
cc3-10u	1000	13500	50	125	126	hc12u	4096	24576	2048	2275	**2274**
cc3-11p	1331	19965	61	15609	15729	hc9p	512	2304	256	30258	**30242**
cc3-11u	1331	19965	61	153	153	hc9u	512	2304	256	292	292

case, our result is printed bold, the three instances for which we also proved optimality of the solution are marked by an asterisk.

5 Conclusions

We have shown that incorporating the mathematical progress of the last 15 years into a well-known model for Steiner trees can have a significant impact in several dimensions. First, the amount of problem specific code is notably reduced while 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 continous improvements in the framework. Furthermore, the opportunity to solve instances in a massively parallel distributed memory enviroment has been added at minimal cost.

The use of a general MIP solver allows us to be extremly flexible with the model to be solved. We were able to support solving seven variants of the Steiner Tree problem with the same code, and the support of further restrictions in the model is straightforward. Using all this prebuild-technology we were able to solve three previously unsolved instances and improve the best known solution for another 11 instances. And finally, to our best knowledge this is the first time that a powerful exact Steiner tree solver code is available in source code to the scientific community. We hope that this will foster the use of Steiner trees in modelling real-world phenomena as it has already been the case in genetics.

References

[1] Karp, R.: Reducibility among combinatorial problems. In Miller, R., Thatcher, J., eds.: Complexity of Computer Computations. Plenum Press (1972) 85–103

[2] Koch, T., Martin, A., Voß, S.: SteinLib: An updated library on Steiner tree problems in graphs. In Du, D.Z., Cheng, X., eds.: Steiner Trees in Industries. Kluwer (2001) 285–325 ZIB-Report 00-37.

[3] Chvatal, V.: A farmer's daughter in our midst: An interview with vasek chvatal (1996) `http://www.cs.rutgers.edu/~mcgrew/Explorer/1.2/`.

[4] Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks **32** (1998) 207–232 ZIB-Report SC 96-42.

[5] 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 ZIB-Report 13-57.

[6] 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

[7] Polzin, T.: Algorithms for the Steiner problem in networks. PhD thesis, Saarland University (2004)

[8] Achterberg, T.: SCIP: Solving constraint integer programs. Mathematical Programming Computation **1**(1) (2009) 1–41

[9] Mittelmann, H.: Benchmarks for optimization software (last accessed Sep. 8, 2014) `http://plato.asu.edu/bench.html`.

[10] Duin, C.: Steiner Problems in Graphs. PhD thesis, University of Amsterdam (1993)

[11] Achterberg, T., Berthold, T.: Hybrid branching. In van Hoeve, W.J., Hooker, J.N., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009. Volume 5547 of Lecture Notes in Computer Science., Springer (May 2009) 309–311

[12] Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)

[13] Takahashi, H., A., M.: An approximate solution for the steiner problem in graphs. Math. Jap. **24** (1980) 573 – 577

[14] 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

[15] Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)

[16] Leitner, M., Ljubic, I., Luipersbeck, M., Prossegger, M., Resch, M.: New real-world instances for the steiner tree problem in graphs. Technical report, ISOR, Uni Wien (2014)

[17] Garey, M., Johnson, D.: The rectilinear Steiner tree problem is np-complete. SIAM Journal of Applied Mathematics **32** (1977) 826–834

[18] 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

[19] Zachariasen, M., Rohe, A.: Rectilinear group steiner trees and applications in VLSI design. Technical Report 00906, Institute for Discrete Mathematics (2000)

[20] Emanet, N.: The rectilinear steiner tree problem (2010)

[21] 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

[22] Hanan, M.: On Steiner's problem with rectilinear distance. SIAM Journal of Applied Mathematics **14**(2) (1966) 255–265

[23] Snyder, T.L.: On the exact location of steiner points in general dimension. SIAM J. Comput. **21**(1) (1992) 163–180

[24] Ljubi, I.: Exact and memetic algorithms for two network design problems (2004)

[25] Dittrich, M.T., Klau, G.W., Rosenwald, A., Dandekar, T., Mller, T.: Identifying functional modules in protein-protein interaction networks: an integrated exact approach. In: ISMB. (2008) 223–231

[26] Duin, C.W., Volgenant, A., Vo, S.: Solving group steiner problems as steiner problems. European Journal of Operational Research **154**(1) (2004) 323–329

[27] 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

[28] 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)

A Detailed Computational Results

Here, we present instance-wise results for the STP (Tables 8–12), RSMTP (Tables 13–21), and PCSPG (Tables 22–24) test sets discussed in Sections 2 and 3. For all instances, we list the number of nodes, arcs, and terminals in the presolved graph, together with the dual bound, primal bound, and optimality gap in percent at the end of the solving process, i.e., after the instance was solved or the time or memory limit was reached. Additionally, we show the number of nodes and the solution time. We print "timeout" and "memout" if the time or memory limit was reached, respectively.

For the STP instances (Tables 8–12), we additionally depict the presolving results. This includes the number of nodes, arcs, and terminals in the presolved problem together with the presolving time.

Table 8. Detailed computational results for test set SP.

	Original			Presolved								
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time	Dual Bound	Primal Bound	Gap	Nodes	Time
antiwheel5	10	30	5	10	30	5	0.0	7	7	0.0	1	0.0
design432	8	40	4	8	38	4	0.0	9	9	0.0	1	0.0
oddcycle3	6	18	3	8	0	0	0.0	4	4	0.0	0	0.0
oddwheel3	7	18	4	7	18	4	0.0	5	5	0.0	1	0.0
se03	13	42	4	13	42	4	0.0	12	12	0.0	1	0.0
w13c29	783	4524	406	783	4524	406	0.2	507	507	0.0	185	33826.8
w23c23	1081	6348	552	1081	6348	552	0.4	689	697	1.2	152	timeout
w3c571	3997	20556	2284	3997	20556	2284	1.9	2853	2856	0.1	1	timeout

Table 9. Detailed computational results for test set I640.

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
i640-001	640	1920	9	314	1260	9	0.1	4033	4033	0.0	1	0.5
i640-002	640	1920	9	301	1224	9	0.1	3588	3588	0.0	1	0.4
i640-003	640	1920	9	301	1224	9	0.2	3438	3438	0.0	1	0.3
i640-004	640	1920	9	303	1238	9	0.1	4000	4000	0.0	1	0.8
i640-005	640	1920	9	318	1272	9	0.1	4006	4006	0.0	1	0.6
i640-011	640	8270	9	640	8270	9	0.4	2392	2392	0.0	1	6.3
i640-012	640	8270	9	640	8270	9	0.3	2465	2465	0.0	1	13.0
i640-013	640	8270	9	640	8270	9	0.4	2399	2399	0.0	1	8.8
i640-014	640	8270	9	640	8270	9	0.3	2171	2171	0.0	1	3.1
i640-015	640	8270	9	640	8270	9	0.4	2347	2347	0.0	5	23.1
i640-021	640	408960	9	640	408904	9	71.4	1749	1749	0.0	1	17056.3
i640-022	640	408960	9	640	408904	9	73.7	1756	1756	0.0	1	30626.4
i640-023	640	408960	9	640	408904	9	75.5	1754	1754	0.0	1	26586.6
i640-024	640	408960	9	640	408904	9	71.4	1751	1751	0.0	1	23828.4
i640-025	640	408960	9	640	408904	9	70.8	1745	1745	0.0	1	21453.1
i640-031	640	2560	9	483	2234	9	0.7	3278	3278	0.0	1	1.8
i640-032	640	2560	9	475	2226	9	0.7	3187	3187	0.0	1	1.5
i640-033	640	2560	9	484	2244	9	0.7	3260	3260	0.0	1	2.4
i640-034	640	2560	9	479	2232	9	0.7	2953	2953	0.0	1	1.7
i640-035	640	2560	9	478	2232	9	0.3	3292	3292	0.0	1	1.8
i640-041	640	81792	9	640	81788	9	12.9	1897	1897	0.0	1	369.4
i640-042	640	81792	9	640	81792	9	6.7	1934	1934	0.0	151	3784.2
i640-043	640	81792	9	640	81792	9	6.9	1931	1931	0.0	167	4551.1
i640-044	640	81792	9	640	81790	9	12.7	1938	1938	0.0	259	6021.5
i640-045	640	81792	9	640	81792	9	6.4	1866	1866	0.0	1	457.3
i640-101	640	1920	25	320	1262	25	0.2	8764	8764	0.0	1	2.9
i640-102	640	1920	25	312	1240	25	0.1	9109	9109	0.0	1	0.6
i640-103	640	1920	25	305	1232	24	0.1	8819	8819	0.0	1	1.3
i640-104	640	1920	25	304	1234	24	0.1	9040	9040	0.0	1	1.4
i640-105	640	1920	25	324	1270	25	0.1	9623	9623	0.0	5	22.4
i640-111	640	8270	25	640	8270	25	0.4	6167	6167	0.0	375	563.9
i640-112	640	8270	25	640	8270	25	0.4	6304	6304	0.0	127	441.5
i640-113	640	8270	25	640	8270	25	0.4	6249	6249	0.0	941	1865.1
i640-114	640	8270	25	640	8270	25	0.4	6308	6308	0.0	267	633.3
i640-115	640	8270	25	640	8270	25	0.4	6217	6217	0.0	1373	2242.4
i640-121	640	408960	25	640	408408	25	49.0	4906	4906	0.0	3	40975.2
i640-122	640	408960	25	640	408408	25	64.9	4899	4911	0.2	5	timeout
i640-123	640	408960	25	640	408408	25	80.3	4904.92308	4913	0.2	4	timeout
i640-124	640	408960	25	640	408408	25	72.5	4897.30405	4908	0.2	3	timeout
i640-125	640	408960	25	640	408408	25	71.5	4908.6	4920	0.2	6	timeout
i640-131	640	2560	25	481	2234	25	0.3	8097	8097	0.0	1	3.5
i640-132	640	2560	25	480	2228	24	0.3	8154	8154	0.0	1	19.6
i640-133	640	2560	25	482	2236	25	0.3	8021	8021	0.0	1	2.4
i640-134	640	2560	25	485	2244	25	0.3	7754	7754	0.0	1	4.7
i640-135	640	2560	25	479	2226	25	0.3	7696	7696	0.0	1	5.5
i640-141	640	81792	25	640	81714	25	13.9	5177.37742	5200	0.4	1636	timeout
i640-142	640	81792	25	640	81720	25	13.7	5178.35294	5193	0.3	1067	timeout
i640-143	640	81792	25	640	81726	25	13.8	5194	5194	0.0	693	25448.7
i640-144	640	81792	25	640	81716	25	13.7	5205	5205	0.0	1091	37225.1
i640-145	640	81792	25	640	81722	25	12.8	5194.0884	5218	0.5	1437	timeout
i640-201	640	1920	50	311	1236	47	0.3	16079	16079	0.0	1	1.2
i640-202	640	1920	50	320	1250	48	0.3	16324	16324	0.0	1	1.7
i640-203	640	1920	50	326	1274	48	0.2	16124	16124	0.0	1	3.8
i640-204	640	1920	50	324	1270	49	0.1	16239	16239	0.0	1	1.7
i640-205	640	1920	50	327	1272	48	0.2	16616	16616	0.0	2	4.8
i640-211	640	8270	50	640	8270	50	0.5	11874.1184	11991	1.0	11030	timeout
i640-212	640	8270	50	640	8270	50	0.4	11795	11795	0.0	5621	11844.3
i640-213	640	8270	50	640	8268	50	1.2	11879	11879	0.0	18049	31721.9
i640-214	640	8270	50	640	8270	50	0.8	11808.77	11898	0.8	15240	timeout
i640-215	640	8270	50	640	8262	50	1.4	11989.5913	12081	0.8	16974	timeout
i640-221	640	408960	50	640	406608	50	55.1	9783.10008	9826	0.4	7	timeout
i640-222	640	408960	50	640	406608	50	74.2	9768.21097	9798	0.3	6	timeout

cont. next page

Instance	Original			Presolved			Time	Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$						
i640-223	640	408960	50	640	406608	50	73.6	9777.07161	9815	0.4	7	timeout
i640-224	640	408960	50	640	406608	50	77.1	9774.44793	9805	0.3	7	timeout
i640-225	640	408960	50	640	406608	50	70.1	9774.57873	9810	0.4	5	timeout
i640-231	640	2560	50	492	2260	50	0.3	15014	15014	0.0	53	108.9
i640-232	640	2560	50	493	2260	49	0.8	14630	14630	0.0	3	25.3
i640-233	640	2560	50	507	2282	48	1.0	14797	14797	0.0	5	74.7
i640-234	640	2560	50	488	2236	49	0.7	15203	15203	0.0	1	4.6
i640-235	640	2560	50	484	2244	50	0.7	14803	14803	0.0	81	200.1
i640-241	640	81792	50	640	81392	50	13.3	10150.0275	10250	1.0	221	timeout
i640-242	640	81792	50	640	81404	50	12.7	10126.7859	10195	0.7	436	timeout
i640-243	640	81792	50	640	81408	50	13.2	10154.9661	10215	0.6	226	timeout
i640-244	640	81792	50	640	81356	50	13.0	10155.3296	10263	1.1	201	timeout
i640-245	640	81792	50	640	81414	50	13.2	10154.9237	10223	0.7	377	timeout
i640-301	640	1920	160	352	1294	132	0.3	45005	45005	0.0	1	4.6
i640-302	640	1920	160	302	1158	113	0.3	45736	45736	0.0	1	6.0
i640-303	640	1920	160	342	1260	127	0.3	44922	44922	0.0	1	1.1
i640-304	640	1920	160	345	1248	136	0.3	46233	46233	0.0	1	4.3
i640-305	640	1920	160	304	1120	120	0.4	45902	45902	0.0	1	4.4
i640-311	640	8270	160	640	8058	160	1.1	35343.4495	35958	1.7	4303	timeout
i640-312	640	8270	160	639	8034	160	1.6	35377.61	35973	1.7	4058	timeout
i640-313	640	8270	160	640	8064	160	1.6	35240.2339	35543	0.9	7643	timeout
i640-314	640	8270	160	640	8054	160	1.6	35167.5667	35596	1.2	10793	timeout
i640-315	640	8270	160	640	8038	160	1.6	35337.8654	35784	1.3	8566	timeout
i640-321	640	408960	160	640	383838	160	71.0	30995.5736	31218	0.7	10	timeout
i640-322	640	408960	160	640	383838	160	74.1	30989.6609	31140	0.5	9	timeout
i640-323	640	408960	160	640	383838	160	62.0	30999.9283	31172	0.6	7	timeout
i640-324	640	408960	160	640	383838	160	60.6	30999.3726	31112	0.4	9	timeout
i640-325	640	408960	160	640	383838	160	54.2	30987.5937	31300	1.0	8	timeout
i640-331	640	2560	160	495	2222	151	0.9	42796	42796	0.0	480	349.8
i640-332	640	2560	160	506	2260	154	0.3	42548	42548	0.0	69	140.3
i640-333	640	2560	160	503	2232	148	1.0	42345	42345	0.0	217	265.1
i640-334	640	2560	160	511	2276	155	0.8	42768	42768	0.0	719	811.1
i640-335	640	2560	160	516	2294	153	0.8	43035	43035	0.0	433	448.5
i640-341	640	81792	160	640	77066	160	12.4	31864.8529	32108	0.8	75	timeout
i640-342	640	81792	160	640	76886	160	11.4	31822.5062	32001	0.6	83	timeout
i640-343	640	81792	160	640	76968	160	11.8	31841.3245	32044	0.6	83	timeout
i640-344	640	81792	160	640	77204	160	12.7	31829.3452	32014	0.6	75	timeout
i640-345	640	81792	160	640	77070	160	13.0	31816.5146	32020	0.6	108	timeout

Table 10. Detailed computational results for test set PUC.

Instance	Original			Presolved			Time	Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$						
bip42p	1200	7964	200	990	7234	200	1.2	24493.4799	24670	0.7	43156	timeout
bip42u	1200	7964	200	990	7220	200	0.9	233.432193	236	1.1	47264	timeout
bip52p	2200	15994	200	1819	14666	200	7.0	24247.4802	24693	1.8	13882	timeout
bip52u	2200	15994	200	1819	14652	200	3.4	229.77802	234	1.8	11288	timeout
bip62p	1200	20004	200	1199	20000	200	3.1	22493.7726	22927	1.9	3280	timeout
bip62u	1200	20004	200	1199	20000	200	2.7	214.079797	221	3.2	2878	timeout
bipa2p	3300	36146	300	3140	35592	300	21.6	34707.1104	36093	4.0	249	timeout
bipa2u	3300	36146	300	3140	35590	300	13.0	329.708754	345	4.6	222	timeout
bipe2p	550	10026	50	550	10026	50	0.7	5616	5616	0.0	26323	14915.6
bipe2u	550	10026	50	550	10026	50	0.6	54	54	0.0	106	9407.0
cc10-2p	1024	10240	135	1024	10240	135	0.8	34510.6406	36320	5.2	1	timeout
cc10-2u	1024	10240	135	1024	10240	135	0.5	334.569953	349	4.3	1	timeout
cc11-2p	2048	22526	244	2048	22526	244	3.7	62152.4689	64870	4.4	1	timeout
cc11-2u	2048	22526	244	2048	22526	244	2.2	602.858649	636	5.5	1	timeout
cc12-2p	4096	49148	473	4096	49148	473	16.3	118552.69	125228	5.6	1	timeout
cc12-2u	4096	49148	473	4096	49148	473	9.4	1149.69926	1218	5.9	1	timeout
cc3-10p	1000	27000	50	1000	27000	50	1.4	12227.5067	13285	8.6	1	timeout
cc3-10u	1000	27000	50	1000	27000	50	1.7	117.783609	130	10.4	1	timeout
cc3-11p	1331	39930	61	1331	39930	61	3.3	14790.4955	16195	9.5	1	timeout
cc3-11u	1331	39930	61	1331	39930	61	2.4	143.348092	159	10.9	1	timeout
cc3-12p	1728	57024	74	1728	57024	74	5.5	17799.313	19483	9.5	1	timeout
cc3-12u	1728	57024	74	1728	57024	74	4.5	171.826667	191	11.2	1	timeout

cont. next page

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
cc3-4p	64	576	8	64	576	8	0.0	2338	2338	0.0	20448	745.2
cc3-4u	64	576	8	64	576	8	0.0	23	23	0.0	935	118.0
cc3-5p	125	1500	13	125	1500	13	0.0	3468.35525	3661	5.6	102403	timeout
cc3-5u	125	1500	13	125	1500	13	0.0	33.6241486	36	7.1	102042	timeout
cc5-3p	243	2430	27	243	2430	27	0.1	–	–	–	0	memout
cc5-3u	243	2430	27	243	2430	27	0.0	69.6282699	71	2.0	13356	timeout
cc6-2p	64	384	12	64	384	12	0.0	3271	3271	0.0	593	40.0
cc6-2u	64	384	12	64	384	12	0.0	32	32	0.0	19	17.4
cc6-3p	729	8736	76	729	8736	76	0.3	20163.9566	20551	1.9	801	timeout
cc6-3u	729	8736	76	729	8736	76	0.3	195.58884	203	3.8	1	timeout
cc7-3p	2187	30616	222	2187	30616	222	4.5	55305.5296	58817	6.3	1	timeout
cc7-3u	2187	30616	222	2187	30616	222	2.9	536.098276	572	6.7	1	timeout
cc9-2p	512	4608	64	512	4608	64	0.2	16879.0806	17559	4.0	266	timeout
cc9-2u	512	4608	64	512	4608	64	0.1	163.647679	171	4.5	426	timeout
hc10p	1024	10240	512	1024	10240	512	1.4	59230.2518	61201	3.3	304	timeout
hc10u	1024	10240	512	1024	10240	512	0.9	567.777778	596	5.0	35	timeout
hc11p	2048	22528	1024	2048	22528	1024	4.5	117382.864	123373	5.1	20	timeout
hc11u	2048	22528	1024	2048	22528	1024	2.7	1125.22675	1198	6.5	1	timeout
hc12p	4096	49152	2048	4096	49152	2048	19.0	232821.911	244673	5.1	1	timeout
hc12u	4096	49152	2048	4096	49152	2048	10.8	2221.43833	2324	4.6	1	timeout
hc6p	64	384	32	64	384	32	0.0	4003	4003	0.0	16459	171.7
hc6u	64	384	32	64	384	32	0.0	39	39	0.0	6919	88.0
hc7p	128	896	64	128	896	64	0.0	7809.64565	7913	1.3	946134	timeout
hc7u	128	896	64	128	896	64	0.0	74.3284618	77	3.6	432639	timeout
hc8p	256	2048	128	256	2048	128	0.1	15169.7011	15328	1.0	134132	timeout
hc8u	256	2048	128	256	2048	128	0.0	145.186012	148	1.9	52209	timeout
hc9p	512	4608	256	512	4608	256	0.2	29919.6174	30293	1.2	8279	timeout
hc9u	512	4608	256	512	4608	256	0.1	286.875	293	2.1	1281	timeout

Table 11. Detailed computational results for test set vienna-i-simple.

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
I001	30190	95496	1184	14202	44196	980	6439.2	253762410	253947459	0.1	1	timeout
I002	49920	155742	1665	23433	72608	1334	21524.2	398742243	399832427	0.3	1	timeout
I003	44482	146838	3222	14756	44758	2438	21519.0	788688122	788806491	0.0	1	timeout
I004	5556	17104	570	820	2296	297	122.8	279512692	279512692	0.0	1	137.4
I005	10284	31960	1017	1577	4596	530	636.2	390876350	390876350	0.0	1	696.4
I006	31754	105750	2202	12541	37898	1904	6346.4	504526027	504526124	0.0	3	timeout
I007	15122	48742	737	6728	20848	616	969.8	177909660	177909660	0.0	1	6376.1
I008	15714	51134	871	6022	18440	742	1526.8	201788202	201788202	0.0	1	8168.4
I009	33188	104014	1262	14909	46400	1090	9097.2	275510588	275583163	0.0	1	timeout
I010	29905	94914	943	13043	41200	833	4585.5	207889674	207889674	0.0	1	21467.5
I011	25195	82596	1428	8843	27312	1257	2135.1	317589880	317589880	0.0	15	6335.9
I012	12355	39924	503	3191	10182	402	425.4	118893243	118893243	0.0	1	971.6
I013	18242	57952	891	6916	21284	712	1347.1	193190339	193190339	0.0	1	6032.7
I014	12715	41264	475	3611	11608	397	310.5	105173465	105173465	0.0	1	428.0
I015	48833	159974	2493	19326	60326	2208	14043.4	592129138	592346523	0.0	1	timeout
I016	72038	230110	4391	26428	80322	3604	44170.9	–	–	–	0	timeout
I017	15095	48182	478	7399	23470	414	1376.3	109739695	109739695	0.0	1	3087.2
I018	31121	102226	1898	11931	36540	1605	6399.5	463887832	463887832	0.0	3	29082.9
I019	25946	83290	866	10714	34534	767	2695.5	217647693	217647693	0.0	3	39371.1
I020	21808	69842	594	6483	20884	516	1285.6	146515460	146515460	0.0	1	3212.3
I021	16013	50538	392	5216	17150	314	564.9	106470644	106470644	0.0	1	2364.9
I022	16224	51382	437	8582	27268	378	1287.6	106799980	106799980	0.0	1	14003.7
I023	22805	70614	582	13470	41972	430	4419.1	131032600	131045335	0.0	1	timeout
I024	68464	217464	3001	30317	94236	2592	43367.2	–	–	–	0	timeout
I025	23412	75904	945	8924	28278	850	2086.7	232790758	232790758	0.0	3	27044.1
I026	47429	158614	3334	16221	49496	2734	27277.1	927917236	928108049	0.0	1	timeout
I027	85085	277776	3954	39033	121424	3665	45607.4	–	–	–	0	timeout
I028	72701	230860	1790	42555	133906	1639	43220.3	–	–	–	0	timeout
I029	69988	223608	2162	31948	100798	1977	43512.7	–	–	–	0	timeout
I030	33188	107360	1263	11653	36996	1139	3430.2	321646787	321646787	0.0	2	23821.5
I031	54351	176422	2182	19827	62754	1909	16850.0	577615634	578557879	0.2	1	timeout

cont. next page

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
I032	56023	182798	3017	19885	61072	2494	20257.6	773096651	773096651	0.0	7	40655.9
I033	18555	59460	636	8396	26136	564	2081.8	134461857	134461857	0.0	1	4728.8
I034	22311	71032	735	8466	26930	639	1640.7	165115148	165115148	0.0	1	5985.7
I035	30585	100908	1704	11793	36370	1465	6098.7	414440370	414440370	0.0	8	9306.9
I036	37208	120712	1411	15196	48196	1286	5704.3	375231130	375270860	0.0	1	timeout
I037	13694	44252	427	5696	18462	400	573.8	105720727	105720727	0.0	1	3500.7
I038	18747	61278	967	6945	21676	819	1099.3	255767543	255767543	0.0	11	5323.1
I039	8755	28898	347	3573	11254	316	176.6	85566290	85566290	0.0	2	1038.5
I040	40389	131640	1762	16942	53390	1530	9075.4	431192266	431580458	0.1	1	timeout
I041	47197	150614	1193	20988	66954	1052	11913.3	301541112	302010236	0.2	1	timeout
I042	51896	171100	2171	22146	69624	1969	9976.5	531659490	532211823	0.1	1	timeout
I043	10398	33574	367	4050	12806	337	260.6	95722094	95722094	0.0	1	1166.2
I044	68905	227778	3358	29204	91078	3035	43414.1	–	–	–	0	timeout
I045	14685	46932	421	6387	20334	390	532.9	105944062	105944062	0.0	1	1542.9
I046	70843	234418	3598	28621	89474	3204	27276.6	922196299	925766976	0.4	1	timeout
I047	28524	92502	2354	9548	28596	1843	7629.2	695163406	695163406	0.0	3	28867.5
I048	13189	42438	358	4950	15908	322	342.3	91509264	91509264	0.0	1	1132.8
I049	30857	99182	990	13446	43362	843	3733.7	–	–	–	0	memout
I050	43073	142552	2868	16281	49762	2324	18588.4	792423060	792672885	0.0	1	timeout
I051	27028	90812	1524	10913	33580	1348	3758.8	357230839	357230839	0.0	21	21282.0
I052	2363	7522	40	193	632	23	1.9	13309487	13309487	0.0	1	1.9
I053	3224	10570	126	634	1988	105	8.0	30854904	30854904	0.0	1	10.7
I054	3803	12426	38	639	2102	28	16.7	15841596	15841596	0.0	1	23.1
I055	13332	43160	570	4511	14176	482	496.4	144164924	144164924	0.0	1	850.0
I056	1991	6352	51	288	966	33	2.1	14171206	14171206	0.0	1	2.5
I057	33231	110298	1569	12501	38948	1374	5101.4	412746415	412746415	0.0	3	12126.5
I058	23527	79256	1256	6878	21426	1036	1772.5	305024188	305024188	0.0	1	2882.3
I059	9287	29950	363	2787	8840	288	204.5	107617854	107617854	0.0	1	449.9
I060	42008	135144	1242	17201	55516	1170	5778.6	337290460	337290460	0.0	1	32276.3
I061	39160	127318	1458	19607	61462	1350	14805.5	362806517	363072768	0.1	1	timeout
I062	66048	220982	3343	20790	66376	2914	28286.0	792505427	793194884	0.1	1	timeout
I063	26840	87322	1645	8589	26352	1306	4348.6	459801704	459801704	0.0	5	13411.7
I064	63158	214690	3458	29063	88664	3227	26830.8	861816269	863190298	0.2	1	timeout
I065	3898	12712	144	1085	3412	128	18.0	32965718	32965718	0.0	1	66.1
I066	15038	49192	551	4050	13048	456	282.1	174219813	174219813	0.0	1	552.6
I067	20547	66460	627	9701	30734	578	1045.5	175540750	175540750	0.0	1	9628.1
I068	33118	110254	1553	10833	33748	1340	2846.5	420730046	420730046	0.0	6	7215.9
I069	9574	32416	543	3121	9600	466	148.8	135161583	135161583	0.0	1	762.5
I070	15079	49216	550	5867	18786	509	468.3	136700139	136700139	0.0	1	3345.2
I071	33203	108854	1494	11720	36726	1300	3667.3	382539099	382539099	0.0	1	6109.1
I072	26948	88388	993	10154	32760	871	1653.8	289019226	289019226	0.0	1	11373.8
I073	21653	70342	1847	6820	20324	1367	2734.0	663004987	663004987	0.0	1	6073.6
I074	13316	44066	653	3872	12098	558	299.7	165573383	165573383	0.0	2	518.5
I075	57551	190762	2973	20636	64314	2529	17838.9	815292228	815502785	0.0	1	timeout
I076	14023	45790	598	4434	14022	508	372.4	166249692	166249692	0.0	1	2091.9
I077	20856	68474	1787	8572	25834	1504	3110.6	472503150	472503150	0.0	3	9812.7
I078	13294	43896	835	5442	16630	720	599.0	185525490	185525490	0.0	11	2770.8
I079	19867	62542	565	6936	22058	504	1116.0	150506933	150506933	0.0	1	9160.2
I080	18695	59416	548	6642	20928	516	819.6	164299652	164299652	0.0	1	5697.9
I081	25081	81478	888	9563	30208	770	2417.7	247527679	247527679	0.0	1	14752.1
I082	15592	49576	515	5155	16406	448	640.4	147407632	147407632	0.0	1	3504.1
I083	89596	297166	4991	35552	108774	4600	45781.9	–	–	–	0	timeout
I084	44934	147454	2319	14998	46704	1958	10527.6	627187559	627187559	0.0	1	36099.5
I085	9113	28982	301	2762	8748	256	100.4	80628079	80628079	0.0	1	409.2

Table 12. Detailed computational results for test set vienna-i-advanced.

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
I001a	14675	44110	941	12565	38920	881	2121.6	253836156	253949289	0.0	1	timeout
I002a	23800	71516	1282	20753	64086	1226	10811.0	399121823	399831580	0.2	1	timeout
I003a	16270	47838	2336	13295	40280	2225	3251.3	788769523	788777471	0.0	1	timeout
I004a	867	2476	263	551	1588	203	3.3	279512692	279512692	0.0	1	7.9
I005a	1677	4860	491	1041	3054	357	14.4	390876350	390876350	0.0	1	26.4
I006a	13339	39064	1842	11431	34412	1784	1803.3	504526035	504526035	0.0	5	16163.1
I007a	6873	20598	599	5869	18118	576	511.0	177909660	177909660	0.0	1	3765.8
I008a	6522	19258	708	5424	16536	695	410.2	201788202	201788202	0.0	1	3155.5

cont. next page

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time	Dual Bound	Primal Bound	Gap	Nodes	Time
I009a	14977	44870	1053	12851	39724	1015	3029.2	275488552	275583456	0.0	1	timeout
I010a	13041	39090	782	10545	32890	738	3235.1	207889674	207889674	0.0	1	10891.0
I011a	9298	27370	1202	7382	22558	1150	581.9	317589880	317589880	0.0	21	4737.1
I012a	3500	10428	387	2314	7300	346	60.7	118893243	118893243	0.0	1	220.2
I013a	7147	21216	670	5733	17582	630	480.5	193190339	193190339	0.0	1	3898.2
I014a	3577	10622	364	2478	7806	329	54.3	105173465	105173465	0.0	1	104.0
I015a	20573	61082	2119	16470	50918	2044	4698.1	592240832	592240832	0.0	3	21463.2
I016a	27214	79648	3434	21954	66412	3262	11972.9	1110848130	1110940660	0.0	1	timeout
I017a	7571	23142	386	6618	20846	379	399.8	109739695	109739695	0.0	1	2062.2
I018a	12258	36028	1549	9965	30276	1495	1761.4	463887832	463887832	0.0	1	16113.2
I019a	11693	35248	732	8962	28532	717	898.2	217647693	217647693	0.0	3	13647.0
I020a	6405	19128	508	4672	14802	477	233.3	146515460	146515460	0.0	1	1191.3
I021a	5195	15722	295	3639	11808	277	126.3	106470644	106470644	0.0	1	1282.3
I022a	8869	27102	356	7453	23574	345	932.4	106799980	106799980	0.0	1	13596.7
I023a	13724	41726	403	12167	37854	360	2412.2	131044872	131044872	0.0	1	41245.6
I024a	32357	96500	2511	27179	84068	2435	9598.9	756672940	758557437	0.2	1	timeout
I025a	10055	29922	833	7563	23750	807	734.2	232790758	232790758	0.0	3	13328.0
I026a	18155	53136	2661	14694	44534	2566	3132.3	928032223	928032223	0.0	7	37424.3
I027a	40772	121110	3490	32761	101492	3391	17889.6	975965014	976964371	0.1	1	timeout
I028a	43690	132922	1597	38201	119772	1562	43256.4	–	–	–	0	timeout
I029a	32979	99254	1946	26985	84432	1885	13118.9	489241617	492266866	0.6	1	timeout
I030a	12941	38558	1093	9662	30324	1062	1625.1	321646787	321646787	0.0	3	5277.8
I031a	21054	62820	1832	16103	50500	1715	4044.2	–	–	–	0	memout
I032a	21345	62706	2454	16977	51802	2302	4909.8	773096651	773096651	0.0	7	42145.1
I033a	8500	25400	548	6963	21572	528	783.9	134461857	134461857	0.0	1	3185.9
I034a	9128	27336	606	6875	21732	577	425.8	165115148	165115148	0.0	1	7877.9
I035a	13129	38840	1428	10481	32168	1375	2636.7	–	–	–	0	memout
I036a	17036	50964	1258	13484	42368	1213	2401.6	375260864	375260864	0.0	5	39870.4
I037a	5886	17738	392	4557	14556	389	181.8	105720727	105720727	0.0	1	2107.1
I038a	7733	22956	798	6098	18920	767	321.4	255767543	255767543	0.0	3	3374.4
I039a	3719	11066	306	2953	9182	294	70.4	85566290	85566290	0.0	2	667.3
I040a	18837	56312	1501	14946	46730	1445	3520.8	431461983	431566911	0.0	1	timeout
I041a	22466	67736	1014	17905	56692	973	4601.1	301713564	301947909	0.1	1	timeout
I042a	23925	71612	1923	19446	60668	1875	4914.2	531908798	532213772	0.1	1	timeout
I043a	4511	13480	335	3531	11078	324	97.5	95722094	95722094	0.0	1	881.5
I044a	31500	93514	2954	25471	78746	2866	8693.4	804530061	804534188	0.0	1	timeout
I045a	6775	20454	378	5476	17306	367	316.1	105944062	105944062	0.0	1	1078.0
I046a	32376	96108	3154	25699	79822	3033	8655.0	925331983	925505793	0.0	1	timeout
I047a	10622	30880	1791	8673	26012	1683	1162.9	695163406	695163406	0.0	3	42910.9
I048a	4920	14712	320	3665	11660	298	122.6	91509264	91509264	0.0	1	546.9
I049a	15045	45426	821	11740	37542	785	1246.5	294811505	294811505	0.0	1	17433.1
I050a	17787	52352	2232	14461	44028	2146	4753.3	792563895	792610551	0.0	1	timeout
I051a	12130	35784	1337	9924	30348	1287	1220.1	357230839	357230839	0.0	20	12131.3
I052a	160	474	23	91	278	16	0.0	13309487	13309487	0.0	1	0.1
I053a	693	2046	102	514	1594	93	0.9	30854904	30854904	0.0	1	1.8
I054a	540	1634	25	375	1214	21	0.8	15841596	15841596	0.0	1	3.1
I055a	4701	13958	483	3470	10806	451	123.1	144164924	144164924	0.0	1	442.8
I056a	290	878	34	156	502	26	0.1	14171206	14171206	0.0	1	0.2
I057a	13078	38736	1346	10377	32042	1282	1724.6	412746415	412746415	0.0	5	6537.3
I058a	7877	23314	997	5798	17974	922	401.0	305024188	305024188	0.0	1	1216.7
I059a	2800	8314	286	1685	5294	247	34.0	107617854	107617854	0.0	6	59.4
I060a	18991	57072	1158	14554	46326	1127	2424.9	337290460	337290460	0.0	1	23550.3
I061a	20958	62930	1337	17563	54742	1317	7525.1	362908665	363077130	0.0	1	timeout
I062a	23714	70610	2812	17492	54870	2660	6552.0	–	–	–	0	memout
I063a	9600	28084	1291	7337	22308	1209	754.6	459801704	459801704	0.0	1	5681.3
I064a	31712	93422	3182	27236	82676	3137	15243.8	862202260	863191346	0.1	1	timeout
I065a	1185	3512	119	898	2786	116	4.6	32965718	32965718	0.0	1	61.3
I066a	4551	13642	417	3260	10414	393	80.2	174219813	174219813	0.0	1	291.3
I067a	10318	31176	579	8496	26706	557	506.7	175540750	175540750	0.0	1	5293.6
I068a	12191	36046	1302	9183	28354	1219	1203.5	–	–	–	0	memout
I069a	3508	10312	452	2784	8512	431	51.4	135161583	135161583	0.0	1	717.6
I070a	6739	20128	511	5172	16386	497	210.5	136700139	136700139	0.0	1	3049.8

cont. next page

Instance	Original			Presolved				Dual Bound	Primal Bound	Gap	Nodes	Time
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	Time					
l071a	12772	37772	1281	10026	31000	1236	1019.6	382539099	382539099	0.0	1	2976.8
l072a	11628	34822	851	8650	27612	819	719.0	289019226	289019226	0.0	1	7775.9
l073a	7510	21746	1337	6011	17888	1223	478.0	663004987	663004987	0.0	1	3475.3
l074a	4441	13124	548	3169	9788	504	84.7	165573383	165573383	0.0	1	359.7
l075a	23195	68724	2498	18229	56492	2399	3590.6	815024079	815505116	0.1	1	timeout
l076a	4909	14536	498	3504	10968	459	131.4	–	–	–	0	memout
l077a	9153	26726	1490	7886	23544	1434	690.7	472503150	472503150	0.0	3	8114.8
l078a	5864	17324	692	4945	14992	672	227.8	185525490	185525490	0.0	15	1347.7
l079a	7933	23614	497	5831	18392	481	354.8	150506933	150506933	0.0	1	12664.5
l080a	7589	22512	499	5650	17666	483	223.3	164299652	164299652	0.0	1	2756.9
l081a	10747	32058	751	8282	26008	715	616.5	247527679	247527679	0.0	1	12797.4
l082a	5850	17386	435	4161	13132	410	214.3	147407632	147407632	0.0	1	2292.5
l083a	34221	100602	4138	26297	80404	3863	13940.6	1405059510	1405705190	0.0	1	timeout
l084a	17050	50402	1918	12995	40152	1823	2101.5	627187559	627187559	0.0	3	19063.4
l085a	2780	8246	243	1973	6202	219	26.4	80628079	80628079	0.0	1	119.9

Table 13. Detailed computational results for test set estein1.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein1-00	15	44	5	1.87	1.87	0.0	1	0.0
estein1-01	12	34	6	1.64	1.64	0.0	1	0.0
estein1-02	28	90	7	2.36	2.36	0.0	1	0.0
estein1-03	64	224	8	2.54	2.54	0.0	1	0.3
estein1-04	12	34	6	2.26	2.26	0.0	1	0.0
estein1-05	24	76	12	2.42	2.42	0.0	1	0.0
estein1-06	30	98	12	2.48	2.48	0.0	1	0.0
estein1-07	24	74	12	2.36	2.36	0.0	1	0.0
estein1-08	15	44	7	1.64	1.64	0.0	1	0.0
estein1-09	36	120	6	1.77	1.77	0.0	1	0.0
estein1-10	30	98	6	1.44	1.44	0.0	1	0.0
estein1-11	27	84	9	1.8	1.8	0.0	1	0.0
estein1-12	42	142	9	1.5	1.5	0.0	1	0.1
estein1-13	36	120	12	2.6	2.6	0.0	1	0.0
estein1-14	100	360	14	1.48	1.48	0.0	1	0.3
estein1-15	9	24	3	1.6	1.6	0.0	1	0.0
estein1-16	48	164	10	2	2	0.0	1	0.0
estein1-17	182	674	62	4.04	4.04	0.0	1	0.3
estein1-18	168	620	14	1.88	1.88	0.0	1	0.7
estein1-19	6	14	3	1.12	1.12	0.0	1	0.0
estein1-20	15	44	5	1.92	1.92	0.0	1	0.0
estein1-21	16	48	4	0.63	0.63	0.0	1	0.0
estein1-22	16	48	4	0.65	0.65	0.0	1	0.0
estein1-23	16	48	4	0.3	0.3	0.0	1	0.0
estein1-24	9	24	3	0.23	0.23	0.0	1	0.0
estein1-25	9	24	3	0.15	0.15	0.0	1	0.0
estein1-26	16	48	4	1.33	1.33	0.0	1	0.0
estein1-27	12	34	4	0.24	0.24	0.0	1	0.0
estein1-28	9	24	3	2	2	0.0	1	0.0
estein1-29	28	90	12	1.1	1.1	0.0	1	0.0
estein1-30	130	474	14	2.59	2.59	0.0	1	0.7
estein1-31	195	724	19	3.12	3.12	0.0	1	1.4
estein1-32	132	482	18	2.68	2.68	0.0	1	0.6
estein1-33	272	1022	19	2.41	2.41	0.0	1	7.2
estein1-34	240	898	18	1.51	1.51	0.0	1	2.3
estein1-35	6	14	4	0.9	0.9	0.0	1	0.0
estein1-36	49	168	8	0.9	0.9	0.0	1	0.0
estein1-37	100	360	14	1.66	1.66	0.0	1	0.6
estein1-38	100	360	14	1.66	1.66	0.0	1	0.4
estein1-39	64	224	10	1.55	1.55	0.0	1	0.2
estein1-40	144	526	20	2.24	2.24	0.0	1	0.2
estein1-41	81	288	15	1.53	1.53	0.0	1	0.2

cont. next page

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein1-42	195	724	16	2.55	2.55	0.0	1	1.2
estein1-43	196	728	17	2.52	2.52	0.0	1	2.0
estein1-44	270	1014	19	2.2	2.2	0.0	1	4.6
estein1-45	16	48	16	1.5	1.5	0.0	1	0.0

Table 14. Detailed computational results for test set estein10.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein10-0	100	360	10	2.292075	2.292075	0.0	1	0.3
estein10-10	100	360	10	2.223952	2.223952	0.0	1	0.5
estein10-11	100	360	10	1.962632	1.962632	0.0	1	0.2
estein10-12	100	360	10	1.948392	1.948392	0.0	1	0.1
estein10-13	100	360	10	2.185612	2.185612	0.0	1	0.3
estein10-14	100	360	10	1.864192	1.864192	0.0	1	0.3
estein10-1	100	360	10	1.913409	1.913409	0.0	1	0.3
estein10-2	100	360	10	2.600368	2.600368	0.0	1	0.2
estein10-3	100	360	10	2.046109	2.046109	0.0	1	0.3
estein10-4	100	360	10	1.881893	1.881893	0.0	1	0.1
estein10-5	100	360	10	2.654077	2.654077	0.0	1	0.3
estein10-6	100	360	10	2.602508	2.602508	0.0	1	0.2
estein10-7	100	360	10	2.50562	2.50562	0.0	1	0.2
estein10-8	100	360	10	2.206235	2.206235	0.0	1	0.3
estein10-9	100	360	10	2.39361	2.39361	0.0	1	0.1

Table 15. Detailed computational results for test set estein20.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein20-0	400	1520	20	3.370387	3.370387	0.0	1	6.0
estein20-10	400	1520	20	2.712391	2.712391	0.0	1	6.3
estein20-11	400	1520	20	3.04514	3.04514	0.0	1	10.1
estein20-12	400	1520	20	3.443865	3.443865	0.0	1	4.1
estein20-13	400	1520	20	3.406237	3.406237	0.0	1	12.1
estein20-14	400	1520	20	3.230378	3.230378	0.0	1	14.4
estein20-1	400	1520	20	3.263948	3.263948	0.0	1	6.9
estein20-2	400	1520	20	2.784744	2.784744	0.0	1	3.3
estein20-3	400	1520	20	2.762439	2.762439	0.0	7	35.7
estein20-4	400	1520	20	3.403317	3.403317	0.0	3	18.6
estein20-5	400	1520	20	3.601423	3.601423	0.0	1	3.9
estein20-6	400	1520	20	3.493487	3.493487	0.0	1	9.7
estein20-7	400	1520	20	3.801638	3.801638	0.0	3	13.4
estein20-8	400	1520	20	3.673995	3.673995	0.0	1	5.7
estein20-9	400	1520	20	3.402477	3.402477	0.0	1	6.7

Table 16. Detailed computational results for test set estein30.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein30-0	900	3480	30	4.069296	4.069296	0.0	1	234.6
estein30-10	900	3480	30	4.164799	4.164799	0.0	1	251.7
estein30-11	900	3480	30	3.841669	3.841669	0.0	1	38.3
estein30-12	900	3480	30	3.740663	3.740663	0.0	1	62.4
estein30-13	900	3480	30	4.2897	4.2897	0.0	1	96.0
estein30-14	900	3480	30	4.303555	4.303555	0.0	1	282.3
estein30-1	900	3480	30	4.090005	4.090005	0.0	7	135.2
estein30-2	900	3480	30	4.312045	4.312045	0.0	1	432.1
estein30-3	900	3480	30	4.215096	4.215096	0.0	1	205.1
estein30-4	900	3480	30	4.173974	4.173974	0.0	1	209.9
estein30-5	900	3480	30	3.995514	3.995514	0.0	1	383.6
estein30-6	900	3480	30	4.376138	4.376138	0.0	1	176.9
estein30-7	900	3480	30	4.169121	4.169121	0.0	1	292.2
estein30-8	900	3480	30	3.713363	3.713363	0.0	1	141.5
estein30-9	900	3480	30	4.268661	4.268661	0.0	1	128.1

Table 17. Detailed computational results for test set estein40.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein40-0	1600	6240	40	4.484154	4.484154	0.0	1	630.6
estein40-10	1600	6240	40	4.673421	4.673421	0.0	1	1101.3
estein40-11	1600	6240	40	4.384339	4.384339	0.0	1	714.7
estein40-12	1600	6240	40	5.188453	5.188453	0.0	1	2123.0
estein40-13	1600	6240	40	4.916698	4.916698	0.0	1	743.9
estein40-14	1600	6240	40	5.082803	5.082803	0.0	1	1606.4
estein40-1	1600	6240	40	4.681131	4.681131	0.0	1	1483.6
estein40-2	1600	6240	40	4.997415	4.997415	0.0	1	2042.8
estein40-3	1600	6240	40	4.528989	4.528989	0.0	1	1263.5
estein40-4	1600	6240	40	5.18999254	5.194038	0.1	2030	timeout
estein40-5	1600	6240	40	4.97534	4.97534	0.0	1	1481.2
estein40-6	1600	6240	40	4.563901	4.563901	0.0	1	458.6
estein40-7	1600	6240	40	4.874601	4.874601	0.0	1	1049.7
estein40-8	1600	6240	40	5.176179	5.176179	0.0	1	2337.5
estein40-9	1600	6240	40	5.713686	5.713686	0.0	1	1931.7

Table 18. Detailed computational results for test set estein50.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein50-0	2500	9800	50	5.494867	5.494867	0.0	1	5828.3
estein50-10	2500	9800	50	5.253293	5.253293	0.0	1	20820.2
estein50-11	2500	9800	50	5.32673654	5.34093	0.3	448	timeout
estein50-12	2500	9800	50	5.389099	5.389099	0.0	1	5841.1
estein50-13	2500	9800	50	5.355143	5.355143	0.0	1	16948.9
estein50-14	2500	9800	50	5.218085	5.218085	0.0	1	5981.4
estein50-1	2500	9800	50	5.548422	5.548422	0.0	1	3684.2
estein50-2	2500	9800	50	5.469105	5.469105	0.0	1	11805.2
estein50-3	2500	9800	50	5.153576	5.153576	0.0	1	1447.5
estein50-4	2500	9800	50	5.518601	5.518601	0.0	1	2284.0
estein50-5	2500	9800	50	5.58043	5.58043	0.0	1	5858.3
estein50-6	2500	9800	50	4.996117	4.996117	0.0	29	23352.9
estein50-7	2500	9800	50	5.375465	5.375465	0.0	1	1375.5
estein50-8	2500	9800	50	5.3431982	5.345677	0.0	380	timeout
estein50-9	2500	9800	50	5.403795	5.403795	0.0	1	5171.6

Table 19. Detailed computational results for test set estein60.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
estein60-0	3600	14160	60	5.376143	5.376143	0.0	1	5737.2
estein60-10	3600	14160	60	5.614167	5.614167	0.0	1	14913.2
estein60-11	3600	14160	60	5.979133	5.979133	0.0	1	20954.6
estein60-12	3600	14160	60	6.07503679	6.138029	1.0	1	timeout
estein60-13	3600	14160	60	5.603556	5.603556	0.0	1	5819.2
estein60-14	3600	14160	60	5.662257	5.662257	0.0	1	9987.0
estein60-1	3600	14160	60	5.53025109	5.536782	0.1	121	timeout
estein60-2	3600	14160	60	5.656678	5.656678	0.0	1	8370.9
estein60-3	3600	14160	60	5.537102	5.537102	0.0	1	34545.6
estein60-4	3600	14160	60	5.46369806	5.470499	0.1	786	timeout
estein60-5	3600	14160	60	6.03886585	6.042196	0.1	1	timeout
estein60-6	3600	14160	60	5.89326687	5.897803	0.1	1	timeout
estein60-7	3600	14160	60	5.813816	5.813816	0.0	1	16811.2
estein60-8	3600	14160	60	5.587713	5.587713	0.0	1	8110.2
estein60-9	3600	14160	60	5.762446	5.762446	0.0	1	10757.2

Table 20. Detailed computational results for test set solids.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
cube	8	24	8	7	7	0.0	1	0.0
dodecahedron	343	1764	20	7.69398	7.69398	0.0	10709	5671.1
icosahedron	125	600	12	20.944264	20.944264	0.0	9	5.5
octahedron	27	108	6	6	6	0.0	1	0.0
tetrahedron	18	66	4	2.682521	2.682521	0.0	1	0.0

Table 21. Detailed computational results for test set cancer.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
cancer1_4D	600	3820	20	28	28	0.0	1	1.3
cancer2_4D	256	1536	20	21	21	0.0	1	0.1
cancer3_6D	20580	197078	110	146	146	0.0	1	380.9
cancer4_6D	34560	340416	93	134.159732	137	2.1	1	timeout
cancer5_6D	8000	74400	48	69	69	0.0	1	9901.0
cancer6_6D	5120	46592	50	55	55	0.0	1	73.2
cancer7_6D	21000	203300	109	140	140	0.0	1	2358.3
cancer8_6D	8640	80064	77	89	89	0.0	1	103.1
cancer9_6D	6000	54800	46	59	59	0.0	1	34.1
cancer10_6D	10000	94000	82	92	92	0.0	1	45.8
cancer11_8D	4762800	64777860	75	–	–	–	0	memout
cancer12_8D	918750	12031250	58	–	–	–	0	memout
cancer13_8D	86400	1039680	70	88	88	0.0	1	5240.8
cancer14_8D	27648	308736	54	63	63	0.0	1	161.9

Table 22. Detailed computational results for test set JMP.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
K100.10	115	722	15	133567	133567	0.0	1	0.0
K100.1	112	762	12	124108	124108	0.0	1	0.0
K100.2	114	756	14	200262	200262	0.0	1	0.3
K100.3	111	874	11	115953	115953	0.0	1	0.1
K100.4	111	788	11	87498	87498	0.0	1	0.1
K100.5	117	812	17	119078	119078	0.0	1	0.1
K100.6	112	680	12	132886	132886	0.0	1	0.0
K100.7	114	708	14	172457	172457	0.0	1	0.1
K100.8	116	776	16	210869	210869	0.0	1	0.2
K100.9	112	732	12	122917	122917	0.0	1	0.2
K100	115	786	15	135511	135511	0.0	1	0.2
K200	234	1580	34	329211	329211	0.0	1	1.2
K400.10	450	3308	50	394191	394191	0.0	1	11.7
K400.1	465	3324	65	490771	490771	0.0	1	11.2
K400.2	462	3420	62	477073	477073	0.0	1	12.2
K400.3	456	3314	56	415328	415328	0.0	1	4.7
K400.4	456	3182	56	389451	389451	0.0	1	7.2
K400.5	477	3368	77	519526	519526	0.0	1	10.7
K400.6	456	3482	56	374849	374849	0.0	1	3.5
K400.7	468	3286	68	474466	474466	0.0	1	8.3
K400.8	461	3392	61	418614	418614	0.0	1	4.2
K400.9	454	3318	54	383105	383105	0.0	1	6.0
K400	463	3402	63	350093	350093	0.0	1	4.2
P100.1	133	760	33	926238	926238	0.0	1	0.9
P100.2	127	750	27	401641	401641	0.0	1	0.8
P100.3	125	776	25	659644	659644	0.0	1	0.5
P100.4	133	760	33	827419	827419	0.0	1	0.7
P100	134	832	34	803300	803300	0.0	1	0.3
P200	249	1462	49	1317874	1317874	0.0	1	1.1
P400.1	521	3144	121	2808440	2808440	0.0	1	7.7
P400.2	508	3034	108	2518577	2518577	0.0	1	7.2
P400.3	514	3028	114	2951725	2951725	0.0	1	8.4
P400.4	495	2852	95	2852956	2852956	0.0	1	8.0
P400	495	2964	95	2459904	2459904	0.0	1	5.3

Table 23. Detailed computational results for test set CRR.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
C01-A	506	1280	6	18	18	0.0	1	0.1
C01-B	506	1280	6	85	85	0.0	1	0.3
C02-A	511	1310	11	50	50	0.0	1	0.0
C02-B	511	1310	11	141	141	0.0	1	0.5
C03-A	584	1748	84	414	414	0.0	1	0.8
C03-B	584	1748	84	737	737	0.0	1	3.8
C04-A	626	2000	126	618	618	0.0	1	2.2
C04-B	626	2000	126	1063	1063	0.0	1	14.1

cont. next page

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
C05-A	751	2750	251	1080	1080	0.0	1	19.4
C05-B	751	2750	251	1528	1528	0.0	3	154.5
C06-A	506	2030	6	18	18	0.0	1	0.1
C06-B	506	2030	6	55	55	0.0	1	1.1
C07-A	511	2060	11	50	50	0.0	1	0.2
C07-B	511	2060	11	102	102	0.0	1	0.6
C08-A	584	2498	84	361	361	0.0	1	3.1
C08-B	584	2498	84	500	500	0.0	1	7.3
C09-A	626	2750	126	533	533	0.0	1	6.0
C09-B	626	2750	126	694	694	0.0	1	18.3
C10-A	751	3500	251	859	859	0.0	1	68.8
C10-B	751	3500	251	1069	1069	0.0	1	66.3
C11-A	506	5030	6	18	18	0.0	1	0.6
C11-B	506	5030	6	32	32	0.0	1	1.8
C12-A	511	5060	11	38	38	0.0	1	1.1
C12-B	511	5060	11	46	46	0.0	1	2.3
C13-A	584	5498	84	236	236	0.0	1	12.3
C13-B	584	5498	84	258	258	0.0	1	11.9
C14-A	626	5750	126	293	293	0.0	2	77.4
C14-B	626	5750	126	318	318	0.0	1	10.1
C15-A	751	6500	251	501	501	0.0	2	192.8
C15-B	751	6500	251	551	551	0.0	1	56.0
C16-A	506	25030	6	11	11	0.0	1	6.5
C16-B	506	25030	6	11	11	0.0	1	7.0
C17-A	511	25060	11	18	18	0.0	1	10.9
C17-B	511	25060	11	18	18	0.0	1	10.3
C18-A	584	25498	84	111	111	0.0	1	143.5
C18-B	584	25498	84	112	113	0.9	101750	timeout
C19-A	626	25750	126	146	146	0.0	4767	12026.2
C19-B	626	25750	126	146	146	0.0	1	43.8
C20-A	751	26500	251	266	266	0.0	5	245.5
C20-B	751	26500	251	267	267	0.0	1	42.1
D01-A	1006	2530	6	18	18	0.0	1	0.0
D01-B	1006	2530	6	106	106	0.0	1	0.6
D02-A	1011	2560	11	50	50	0.0	1	0.0
D02-B	1011	2560	11	218	218	0.0	1	0.7
D03-A	1168	3502	168	807	807	0.0	7	12.3
D03-B	1168	3502	168	1509	1509	0.0	1	81.1
D04-A	1251	4000	251	1203	1203	0.0	1	37.2
D04-B	1251	4000	251	1881	1881	0.0	1	336.0
D05-A	1501	5500	501	2157	2157	0.0	1	523.6
D05-B	1501	5500	501	3135	3135	0.0	1	15117.9
D06-A	1006	4030	6	18	18	0.0	1	0.1
D06-B	1006	4030	6	67	67	0.0	1	4.5
D07-A	1011	4060	11	50	50	0.0	1	0.2
D07-B	1011	4060	11	103	103	0.0	1	1.2
D08-A	1168	5002	168	755	755	0.0	1	34.3
D08-B	1168	5002	168	1036	1036	0.0	1	104.9
D09-A	1251	5500	251	1070	1070	0.0	4	194.5
D09-B	1251	5500	251	1420	1420	0.0	4	342.1
D10-A	1501	7000	501	1671	1671	0.0	3	6448.8
D10-B	1501	7000	501	2079	2079	0.0	6	9741.0
D11-A	1006	10030	6	18	18	0.0	1	1.1
D11-B	1006	10030	6	29	29	0.0	1	7.0
D12-A	1011	10060	11	42	42	0.0	1	6.1
D12-B	1011	10060	11	42	42	0.0	1	5.3
D13-A	1168	11002	168	445	445	0.0	7	345.4
D13-B	1168	11002	168	486	486	0.0	4	318.1
D14-A	1251	11500	251	602	602	0.0	5	1872.6
D14-B	1251	11500	251	665	665	0.0	1	258.7
D15-A	1501	13000	501	1042	1042	0.0	10	4285.1
D15-B	1501	13000	501	1107	1108	0.1	4214	timeout

cont. next page

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
D16-A	1006	50030	6	13	13	0.0	1	32.4
D16-B	1006	50030	6	13	13	0.0	1	26.7
D17-A	1011	50060	11	23	23	0.0	1	52.3
D17-B	1011	50060	11	23	23	0.0	1	51.6
D18-A	1168	51002	168	217	218	0.5	7013	timeout
D18-B	1168	51002	168	222	223	0.5	18908	timeout
D19-A	1251	51500	251	305	306	0.3	9541	timeout
D19-B	1251	51500	251	309	310	0.3	15951	timeout
D20-A	1501	53000	501	535	540	0.9	2005	timeout
D20-B	1501	53000	501	536	537	0.2	2455	timeout

Table 24. Detailed computational results for test set PUCNU.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Dual Bound	Primal Bound	Gap%	Nodes	Time
bip42nu	1401	9164	201	224.170552	228	1.7	48242	timeout
bip52nu	2401	17194	201	219.987945	224	1.8	49721	timeout
bip62nu	1401	21204	201	210.393707	214	1.7	552	timeout
bipa2nu	3601	37946	301	320.276245	330	3.0	40	timeout
bipe2nu	601	10326	51	53	53	0.0	61	1668.3
cc10-2nu	1160	11050	136	165.263052	172	4.1	461	timeout
cc11-2nu	2293	23990	245	300.067047	321	7.0	36	timeout
cc12-2nu	4570	51986	474	558.518991	595	6.5	27	timeout
cc3-10nu	1051	27300	51	58.3827899	61	4.5	1218	timeout
cc3-11nu	1393	40296	62	75.0409485	81	7.9	133	timeout
cc3-12nu	1803	57468	75	90.2119729	98	8.6	20	timeout
cc3-4nu	73	624	9	10	10	0.0	1	0.2
cc3-5nu	139	1578	14	17	17	0.0	1	1.5
cc5-3nu	271	2592	28	36	36	0.0	1	115.3
cc6-2nu	77	456	13	15	15	0.0	1	0.2
cc6-3nu	806	9192	77	94	99	5.3	1	timeout
cc7-3nu	2410	31948	223	267.573727	287	7.3	1	timeout
cc9-2nu	577	4992	65	83	83	0.0	28	1399.6