TECHNISCHE UNIVERSITÄT WIEN
Institut für Computergraphik und Algorithmen

Solving the Minimum Label Spanning Tree Problem by Mathematical Programming Techniques

Andreas M. Chwatal, Günther R. Raidl

Forschungsbericht / Technical Report

TR–186–1–10–03

June, 2010

Favoritenstraße 9-11 / E186, A-1040 Wien, Austria
Tel. +43 (1) 58801-18601, Fax +43 (1) 58801-18699
www.cg.tuwien.ac.at

Solving the Minimum Label Spanning Tree Problem by Mathematical Programming Techniques

Andreas M. Chwatal, Günther R. Raidl[a]

[a] *Vienna University of Technology, Favoritenstraße 9-11, 1040 Vienna, Austria*

Abstract

In this work we present exact mixed integer programming approaches including branch-and-cut and branch-and-cut-and-price for the minimum label spanning tree problem as well as a variant of it having multiple labels assigned to each edge. We compare formulations based on network flows and directed connectivity cuts. Further we show how to use odd hole inequalities and additional inequalities to strengthen the formulation. Label variables can be added dynamically to the model in the pricing step. Primal heuristics are incorporated into the framework to speed up the overall solution process. After a polyhedral comparison of the involved formulations, comprehensive computational experiments are presented in order to compare and evaluate the underlying formulations and the particular algorithmic building blocks of the overall branch-and-cut(-and-price) framework.

1. Introduction

The minimum label spanning tree (MLST) problem was first introduced in [4] and has for instance applications in telecommunication network design and data compression [8]. For the MLST problem we are given an undirected graph $G = (V, E, l)$ with nodes (or vertices) $v \in V$ and edges $e \in E$ connecting pairs of nodes. In addition a labelling function $l : E \to L$ is given, assigning to each edge an element, called "label", from a finite set L. The objective is to find a minimum cardinality label subset $L' \subseteq L$ inducing a spanning tree in the sense that for each edge in the spanning tree its corresponding label is selected. We also consider the situation of $l : E \to 2^L$ where more than one label can be assigned to an edge.

2. Related Work

The minimum label spanning tree (MLST) problem has been introduced by Chang and Leu [4] for the first time. In this work the authors showed the MLST problem to be $\mathcal{NP}$-complete, and proposed an exact and an approximative algorithm called *Maximum Vertex Covering Algorithm* (MVCA). Krumke and Wirth [18] proposed a modified construction algorithm and derived a performance guarantee for it. Moreover it has been shown that the problem cannot be approximated with a

constant factor. An improved performance bound has been obtained by Wan, Chen and Xu [25], and a tight bound has then been found by Xiong, Golden and Wasil [26]. An experimental comparison of further MVCA variations is presented in [13].

Besides approximative methods many metaheuristic algorithms have been proposed and studied in the literature during the last decade. Various genetic algorithms have been developed in [27] and [23]. Methods based on local search have been treated from a theoretical point of view in [1], and from a more practical one in [3, 14, 11, 10], and [12]. In particular, the latter publications also cover metaheuristics like greedy randomized search procedures, local search, variable neighborhood search and the pilot method.

Less work does exist regarding exact algorithms. An exact algorithm based on A^*-search has been proposed in [4], a similar approach, however, not using the guidance function of the A^*-algorithm, has been proposed in [11]. So far, only two mathematical-programming approaches have been considered in the literature. The first MIP formulation proposed by Chen et al. [5] is based on Miller-Tucker-Zemlin inequalities (cf. Section 3.1) which ensure that the decision variables for the edges induce a connected subgraph covering all nodes of the initial graph. In a recent work of Captivo et al. [2], the authors propose a MIP formulation based on single commodity flows, a frequently used modelling technique for spanning trees. A branch-and-cut algorithm based on directed connection cuts and cycle elimination cuts for an extension of the MLST problem has been described in [9]. For a general introduction to integer linear programming (ILP) based algorithms like branch-and-cut and branch-and-price we refer the reader to [22].

In this work we propose a branch-and-cut(-and-price) (BCP) framework for the solution of moderately sized problem instances. We present a polyhedral and computational comparison of an underlying flow-formulation to a formulation based on directed connection cuts. For the latter we show how the cut-separation can be performed more efficiently than for many other spanning tree problems. New inequalities are introduced to strengthen the formulations. Optionally also cycle-elimination cuts are separated. Furthermore we show how to use odd hole inequalities to strengthen the formulation by cutting off fractional values of the label variables. We further consider branch-and-cut-and-price, where instead of starting the algorithm with a full model, we start with a restricted set of labels and include further (label) variables only on demand. In order to obtain valid integral solutions in each node of the B&B tree fast, we apply primal heuristics based on the well known MVCA-heuristic [18]. A detailed description of the formulations and algorithmic building blocks is given in Section 3. In Section 4 we finally present a comparison of the described formulations and algorithmic components based on computational experiments.

3. Mixed Integer Programming Framework

In this section we first give a rather abstract formulation of the MLST as mixed integer program (MIP). For the spanning-tree property we present two concrete instantiations: 1) based on a flow-formulation and 2) a formulation based on directed connectivity cuts, respectively. Both formulations as well as additional inequalities to strengthen the formulations and methods for cutting-plane separation and dynamic variable generation are described within one generic framework, as they can be used in different combinations.

We use the following variables: variables $z_l \in \{0,1\}, \text{forall } l \in L$ indicate if label l is part of the solution; edge variables $x_e, \text{ forall } e \in E$, denote if edge e is used in the final spanning tree; variables $y_{i,j}, \text{ forall } i,j \in V$, denote directed arc variables used for the cut-based formulation, where we introduce for each edge $e = \{i,j\} \in E$ two arcs (i,j) and $(j,i) \in A$. For the flow formulation we analogously introduce two directed flow variables $f_{ij}, f_{ji} \in [0, n-1]$. Let further $L(e)$ denote the set of labels associated to edge e.

3.1. Mixed integer formulation

The basic formulation is given by the following abstract integer linear program:

$$\text{min.} \quad \sum_{l \in L} z_l \tag{1a}$$

$$\text{s.t.} \quad \sum_{l \in L(e)} z_l \geq x_e \qquad \text{for all } e \in E \tag{1b}$$

$$x \equiv \text{“spanning tree”} \tag{1c}$$

$$z_l \in \{0,1\} \qquad \text{for all } l \in L \tag{1d}$$

The objective function (1a) minimizes the number of required labels, inequalities (1b) ensure that for each selected edge (at least) one label is selected. For the abstract condition (1c) we will subsequently introduce alternative formulations.

Fixing the number of selected edges according to a valid spanning tree The number of selected edges may be fixed according to a valid spanning tree:

$$\sum_{e \in E} x_e = |V| - 1. \tag{2}$$

Note, that Equation 2 is however not required for a valid description of the MLST problem.

Single-Commodity Flow Formulation. A single-commodity flow formulation, also considered in [2], is given as follows:

$$\sum_{(0,i) \in A} f_{0i} = |V| - 1 \tag{3a}$$

$$\sum_{(i,t)\in A} f_{it} - \sum_{(t,j)\in A} f_{tj} = 1 \qquad \text{for all } t \in V\backslash\{0\} \tag{3b}$$

$$f_{ij} \leq (|V|-1)\cdot x_e \qquad \text{for all } \{i,j\} \in E \text{ and } e = \{i,j\} \tag{3c}$$

Equation (3a) ensures the correct quantity of flow leaving the (arbitrary) root node with index 0. For all other nodes flow consumption (3b) must hold, i.e. one unit of flow is consumed at each node. Inequalities (3c) finally ensure that only edges with a sufficient amount of flow may be selected. Flow formulations have the big advantage that they permit to formulate a spanning tree by a polynomial number of variables and therefore provide a relatively compact model.

Multi-Commodity Flow Formulation. The single-commodity flow formulation's major shortcoming is, however, that it provides a relatively poor LP-relaxation [20]. This is particularly due to the weak coupling of f to x-variables in Inequalities (3c), the linking constraints. This drawback can be circumvented by the introduction of multiple commodities k for each node $v \in V$. Again, all flows of commodity k originate from node 0 and must be delivered to node k. The formulation is given by the following equalities:

$$\sum_{(i,t)\in A} f^k_{it} - \sum_{(t,j)\in A} f^k_{tj} = \begin{cases} -1 & t = 0, \\ 0 & t \neq 0 \wedge t \neq k, \\ 1 & t = k, \end{cases} \qquad \text{for all } k \in V\backslash\{0\}. \tag{4}$$

Linkage of flow to edge variables is then given by

$$x_e \leq f^k_{ij} \qquad \text{for all commodities } k, \text{ for all } e = \{i,j\} \in E \tag{5}$$

This formulation, however, has the drawback of having more variables than the single commodity flow formulation, i.e. $O(|V|\cdot|E|)$ flow variables in contrast to only $O(|E|)$.

Directed Cut Formulation. An alternative formulation is given by directed connection inequalities, stating that to each node a valid (directed) path must exist. In contrast to the flow model this formulation consists of an exponential number of inequalities and therefore cannot be directly passed to an ILP-solver for larger instances. However, this formulation provides a better LP-relaxation to many spanning tree problems, as it exactly describes the convex hull of the minimum spanning tree polytope. The corresponding inequalities are given by (6a), linkage to the edge variables is given by (6b).

$$\sum_{(i,j)\in\delta^-(S)} y_{ij} \geq 1 \qquad \text{for all } S \subseteq V,\ 0 \notin S \tag{6a}$$

$$x_e \geq y_{ij} \qquad \text{for all } \{i,j\} \in E \text{ and } e = \{i,j\} \tag{6b}$$

Here $\delta^-(S)$ denotes the set of ingoing arcs to some node set $S \subset V$. Instead of Inequalities (6b) we could also directly link the labels to the directed arcs. However, we proceed with Inequalities (6b) for sake of a unified notation. The separation of these directed-connection inequalities is discussed in Section 3.2.

It is well known to be practically advantageous to initially add the inequalities

$$y_{ij} + y_{ji} \leq 1, \qquad \text{for all } \{i,j\} \in E \tag{7}$$

and

$$\sum_{(i,j)\in\delta^-(j)} y_{ij} \geq 1, \qquad \text{for all } j \in V\backslash\{0\} \tag{8}$$

to directed (cut-based) formulations, see [7, 19]. Inequalities (7) avoid short cycles corresponding to a single edge, inequalities (8) assure that each node has one incoming arc. By $\delta^-(i)$ we denote the set of incoming arcs to node i.

Cycle-Elimination Formulation. We can also ensure feasibility for integer solutions by *cycle-elimination inequalities*. These inequalities enforce the resulting graph not to contain any cycles, which is together with the enforced number of arcs also a sufficient condition for spanning trees, and are given by the following Inequalities (9):

$$\sum_{e\in C} x_e \leq |C| - 1, \qquad \text{for all cycles } C \in G, |C| > 2. \tag{9}$$

Miller–Tucker-Zemlin Formulation. A further way for prohibiting cycles are models based on the well known *Miller-Tucker-Zemlin* inequalities [21]. Such a model for the MLST problem has been proposed in [5], however with some differences. Let $u_i \in \mathbb{R}$ for all $i \in V$ denote variables assigning numeric values to teach node. By inequalities

$$u_i - u_j + |V| \cdot y_{ij} \leq |V| - 1 \qquad \text{for all } (i,j) \in A \tag{10a}$$

$$u_i \leq |V| \qquad \text{for all } i \in V \tag{10b}$$

cycles can be inhibited by just using a polynomial number of variables, however with the drawback, that a large multiplicative factor appears, usually leading to bad LP-relaxations. Main difference to the formulation proposed in [5] is the meaning of the variables. Whereas we use distinct variables for labels and edges ($O(|E| + |L|)$ variables), and link them by Inequalities (3c) which are in total $O(|E|)$ constraints, they introduce $O(|E| \cdot |L|)$ variables x_{ijk} with i, j corresponding to edges $\{i,j\}$ and index k corresponding to labels.

In [2] the authors pointed out an important property of the flow formulation: They showed that the edge variables are not required to be integer in order to obtain the correct (optimal) objective function value. Furthermore it is easy to derive a valid MLST solution based on the set of labels provided by the MIP solution.

Based on this reasoning, we can establish the following theorem, which extends this result to further MLST formulations, and also immediately provides an improved cut formulation with a fast separation method.

Epsilon-Connectivity Formulation.

Theorem 3.1. *For any MIP formulation given by equations 1a, 1b and 2, $z_l \in \{0,1\}$, for all $l \in L$ any set of labels corresponding to an optimal solution to this formulation, and additionally meeting the following inequalities ("epsilon-connectivity")*

$$\sum_{e \in \delta(S)} x_e \geq \epsilon \qquad \text{for all } S \subset V, S \neq \emptyset \tag{11}$$

implies a valid MLST. Here, $\epsilon > 0$ denotes some arbitrary small real number.

Proof The number of edges is fixed by (2), but a solution may still contain fractional edges. However, as the label variables z are integer and required to be greater than the value of the corresponding edge variables by inequalities (1b), they are always one if the corresponding edge variable has a value greater than ϵ. Consequently, fractional edge variables will only appear in the final solution if they do not raise the objective function value (by requiring additional labels). Due to Inequalities (11) the labels obtained from the MIP solution facilitate paths between all pairs of nodes. □

Given a label set of an optimal MIP solution, a feasible spanning tree can easily be derived in polynomial time, by determining an arbitrary spanning tree on the edges induced by the label set, as described in [2]. As a direct consequence of Theorem 3.1 the domain of the variables x and y need not be restricted to Boolean values, restricting them to non-negative values by inequalities

$$x_e \geq 0, \quad \text{for all } e \in E, \tag{12}$$

and

$$y_{i,j} \geq 0, \quad \text{for all } \{i,j\} \in E, \tag{13a}$$
$$y_{j,i} \geq 0, \quad \text{for all } \{i,j\} \in E, \tag{13b}$$

is already sufficient.

Theorem 3.1 also suggests a further formulation for the MLST problem. Although not explicitly containing any constraints describing a valid spanning tree, equations (1a), (1b), (2) and (11) already provide a complete description to the MLST problem, and could be further strengthened by (16) and

$$\sum_{e \in \delta(i)} x_e \geq 1, \qquad \text{for all } i \in V. \tag{14}$$

Inequalities (11) will again be separated on demand as cutting planes, which can, however, be performed more efficiently than the separation for the directed connection cuts, which will be discussed in detail in Section 3.2.

Note that epsilon-connectivity as defined by Theorem 3.1 is not guaranteed if cycle-elimination inequalities (9) are used exclusively to describe a valid spanning tree. A fractional LP-solution not containing a cycle may still contain a *subtour*, i.e. a subgraph where the sum over corresponding edges is larger than the size of its nodes minus one. Such a situation is depicted in Figure 1. As a consequence, the domain of the x-variables must be restricted to Boolean values if only cycle-elimination inequalities are used to describe a valid spanning tree. The same is true for the Miller-Tucker-Zemlin formulation given by Inequalities (10a).

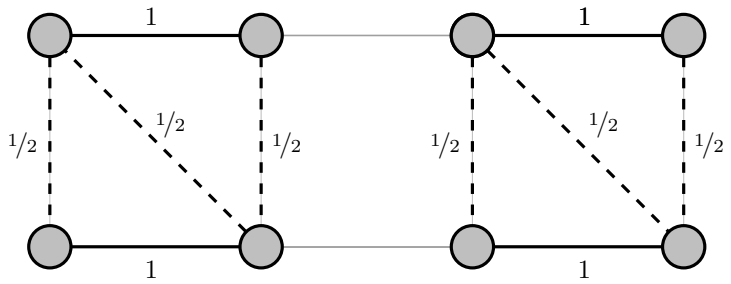

Figure 1: LP-solution that does not contain a cycle w.r.t. Inequalities (9), but still violates subtour elimination constraints. Corresponding (integer) label solutions are not necessarily feasible.

We now draw our attention to the special case of having only one single label assigned to each edge. If we have not fixed the number of edges we can impose further equalities

$$\sum_{l \in L(e)} z_l = x_e \quad \text{for all } e \in E, \tag{15}$$

instead of Inequalities (1b), which provide a more direct link between labels and their corresponding edges. This approach emphasizes the search for a feasible label set of minimal cardinality rather then the search for a feasible spanning tree.

3.2. Cutting-Plane Separation

The directed connection Inequalities (6a) can be separated by computing the maximum flow from the root node r to each nodes i as target node. This provides a minimum (r, i)-cut. We have found a violated inequality if the value of the corresponding arcs according to the sum of the LP-values is less than 1. Our separation procedure utilizes Cherkassky and Goldberg's implementation of the push-relabel method for the maximum flow problem [6] to perform the required minimum cut computations.

The cycle-elimination cuts (9) can be easily separated by shortest path computations with Dijkstra's algorithm. Hereby we use $1 - y_{ij}^{\text{LP}}$ as the arc weights with

y_{ij}^{LP} denoting the current value of the LP-relaxation for arc (i, j) in the current node of the B&B-tree. We obtain cycles by iteratively considering each arc $(i, j) \in A$ and searching for the shortest path from j to i. If the value of a shortest path plus y_{ij}^{LP} is less than 1, we have found a cycle violating Inequalities (9). We add this inequality to the LP and resolve it. In each node of the B&B-tree we perform these cutting plane separations until no further cuts can be found.

Theorem 3.1 suggested a formulation not requiring any auxiliary variables (like flow or arc variables), where validity of the labels is obtained by Inequalities (11) exclusively. Instead of using the minimum cut based separation routine (which would also be valid), we can perform a faster separation by a simple depth first search (DFS). Given an LP-solution, we first select an arbitrary start node for which we call the DFS procedure. Within this procedure we only consider edges e with $x_e \geq \epsilon$. Within the DFS we keep track of all visited nodes, if there are unvisited nodes at the end of the DFS, we have found a valid cut. The DFS can be carried out in $O(|V| + |E|)$ time, which is clearly superior to the time of the maximum flow algorithm running in $O(|V| \cdot |E| + |V|^{2+\varepsilon})$.

3.3. Strengthening the Formulations

As each node must be connected to the spanning tree by one of its incident edges, we can further impose additional inequalities to strengthen the formulation w.r.t. the label variables:

$$\sum_{l \in L(v)} z_l \geq 1, \qquad \text{for all } v \in V. \tag{16}$$

Here, $L(v)$, $v \in V$ denotes the set of labels being associated to the edges incident to node v. We will subsequently refer to this set of $|V|$ inequalities as *node-label-inequalities*. Figure 2 gives a simple example of an LP solution where the node is sufficiently connected according to the sum of the LP-values of the ingoing arcs and therefore its incident edges, but the corresponding sum over the labels associated to these edges is clearly infeasible w.r.t. Inequalities (16). Therefore Inequalities (16) strengthen the presented formulations w.r.t. their LP-relaxation. In Section 3.6 we formally prove this property with respect to the particular proposed MIP-formulations for the MLST. Note, that we will use MIP variables and their corresponding graph-entities equivalently in the the context of subsequent figures and proofs for simplicity, e.g. we will simply designate a label by $a, b, \ldots$ (or $l_a, l_b, \ldots$) instead of explicitly refering to the MIP variables $z_a, z_b, \ldots$.

This basic idea used in Inequalities (16) can be pursued by considering sets of two nodes, say v_1 and v_2. Let e_{12} denote the edge joining v_1 and v_2. Let further $L(e_{12})$ denote the set of labels associated to this edge. For set $L(v_1) \cup L(v_2)$ we can observe, that at least two labels are required to feasibly connect the nodes v_1 and v_2, if $L(v_1) \cap L(v_2) = \emptyset$. However, if $L(v_1) \cap L(v_2) = L(e_{12})$ we still require two

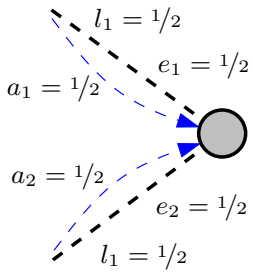

Figure 2: Example of node that is feasible connected w.r.t. its incoming arcs, but not w.r.t. inequalities (16). Edges $e_1 = e_2 = 1/2$ in the current LP-solutions, but as both edges have assigned the same label $l_1 = 1/2$ the sum over the set of all labels assigned to incident edges of the considered node is also $1/2$. Such situations are forbidden by Inequalities (16).

labels from $L(v_1) \cup L(v_2)$. We therefore obtain the following valid inequalities,

$$\sum_{l \in L(v_1) \cup L(v_2)} z_l \geq 2, \qquad \text{for all } v_1, v_2 \in V \text{ with } L(v_1) \cap L(v_2) = L(e_{12}), \tag{17}$$

which are not directly implied by Inequalities (16). Figure 3 shows an example where Inequalities (17) dominate Inequalities (16).

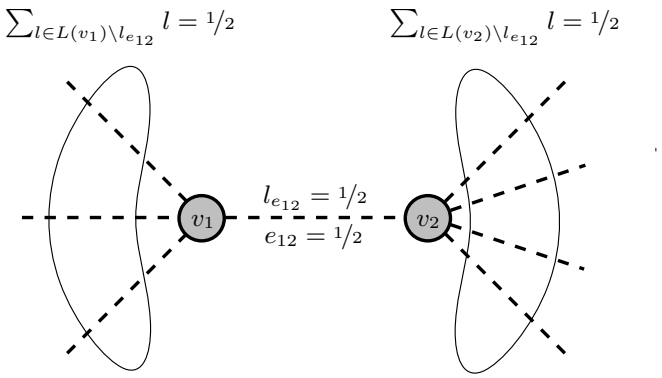

Figure 3: Example of node-label-constraints for sets of two nodes (17) dominating Inequalities (16), i.e. the node-label constraints for single nodes. For both nodes v_i, $i = 1, 2$ it holds that $\sum_{l \in L(v_i)} l \geq 1$. Corresponding Inequality (17) is however violated, as $\sum_{l \in L(v_1) \cup L(v_2)} l = 3/2$.

As we can expect a lot of branching on the label variables, in particular for GMLST instances, further cutting-planes cutting of fractional label solutions may be helpful. In order to identify such valid inequalities, we consider situations where fractional label variables lower the objective value of LP solutions. Such a situation is depicted in Figure 4. If labels $a = b = c = 1/2$ in the LP solutions the corresponding arcs can be set to $1/2$ as well without violating any directed connectivity inequality. However, w.r.t. these arc set, at least two labels must be selected in an integer

solution. Consequently adding the inequality $a+b+c \geq 2$ will cut-off this fractional solution, but is only valid if no additional arcs/edges are incident to these nodes.

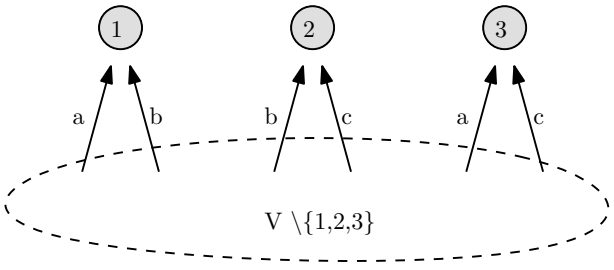

Figure 4: Example of fractional label solution

In the following we show how to apply *odd hole inequalities* to cut-off such and more general situations. These inequalities are well known from studies of the set-covering polytope, their application becomes evident by the observation that the MLST problem can be seen as a set covering problem where each *node* v needs to be covered by a label from the set $L(v)$ and the corresponding edges fulfilling further constraints (i.e. forming a valid spanning tree). In particular we use a MIP based heuristic to separate valid inequalities for the set-covering problem with coefficients $\{0, 1\}$, which have been proposed in [15].

Let $\boldsymbol{\Lambda}$ be a $|V| \times |L|$ matrix with $\lambda_{ij} = 1$ if node i is labeled with j, $\lambda_{ij} = 0$ otherwise. A $|V'| \times |L'|$ submatrix $\boldsymbol{\Lambda'}$ of $\boldsymbol{\Lambda}$ of odd order is called an *odd hole* if it contains exactly two ones per row and column. For the subproblem $\boldsymbol{\Lambda' z'} \geq 1$ the inequality

$$\sum_{l \in L'} z_l \geq \frac{|L'|+1}{2} \tag{18}$$

is valid. In [15] the authors showed that this inequality even remains valid if $\boldsymbol{H} \leq \boldsymbol{\Lambda'} \leq \boldsymbol{H}^*$, where $\boldsymbol{H}$ is an odd hole, and $\boldsymbol{H}^*$ being a special matrix closely related to $\boldsymbol{H}$. Finding an odd hole $\boldsymbol{H}$ to a given matrix $\boldsymbol{\Lambda'}$ is $\mathcal{NP}$-complete, but if we have found such an odd hole, it is possible to decide in polynomial time whether $\boldsymbol{H} \leq \boldsymbol{\Lambda'} \leq \boldsymbol{H}^*$ and therefore (18) is valid [15].

3.3.1. Separation-Heuristic for the Odd Hole Inequalities

In order to cut-off fractional label solutions we consider the subset of nodes $V'' \subseteq V$ whose labels are either fractional or zero in the current LP solution. Let $\tilde{\boldsymbol{\Lambda}}^{V''}$ denote the matrix where each entry λ_{ij} represents the current LP value of label j associated to node i, or -1 if the label j is not associated to node i. Let further $\boldsymbol{\Lambda}^{V''}$ denote the corresponding matrix representing which labels are assigned to particular nodes, i.e. its elements $\lambda_{ij}^{V''}$ are one if label $j \in L(\delta(i))$, and zero otherwise. Our goal is to heuristically search for odd holes in $\boldsymbol{\Lambda}^{V''}$, based on the information provided

by matrix $\tilde{\boldsymbol{\Lambda}}^{V''}$, and then transform the related inequality to a valid inequality for the initial problem by the according lifting steps. We are hence searching for an odd hole $\boldsymbol{H}$ with $\boldsymbol{H} \leq \boldsymbol{\Lambda}^{V',L'}$ with $V' \subseteq V'', L' \subseteq L''$ and $|V'| = |L'|$ being odd. By the procedure of [15] we can now decide if

$$\sum_{l \in L'' \setminus L'} \gamma_l \cdot z_l + \sum_{l \in L'} z_l \geq \frac{|L'| + 1}{2} \tag{19}$$

is valid for $\boldsymbol{\Lambda}^{V',L'}$. The term $\sum_{l \in L'' \setminus L'} \gamma_l \cdot z_l$ results from lifting all labels which are associated to a node $v \in V'$ but are not part of the odd hole induced by V' and L'. The lifting-coefficient is denoted by γ_l, the calculation of its value will be discussed later on. By the following MIP (20) we aim to find subsets V' and L' forming an odd hole and for which inequality (19) is violated according to the current LP solution. For this purpose we define a bipartite directed graph $\tilde{G} = (\tilde{V} = \tilde{V}_1 \cup \tilde{V}_2, \tilde{A}), \tilde{V}_1 = V'', \tilde{V}_2 = L'', \tilde{A} = \{(i,j) \mid i \in V'' \wedge j \in L''(V'')\}$. Each cycle with length $4 \cdot k + 2$ corresponds to an odd cycle w.r.t. the number labels, and is therefore a potential odd hole. Variables $x_{ij} \in \{0,1\}$ represent the arcs from node $i \in V''$ to label $j \in L''(V'')$ and are intended to finally describe a valid odd hole. Variables $a_{ij} \in [0,1]$ denote other arcs which connect nodes $i \in V''$ being part of the odd hole (described by the x variables) and other labels not being part of the odd hole. For each arc $a = (i,j)$ the coefficient c_a is the LP value of label j if $j \in L'$ and zero otherwise.

$$\max \quad k + 1 - \sum_{i \in \tilde{A}} x_i \cdot c_i - \sum_{i \in \tilde{A}} a_i \cdot c_i \tag{20a}$$

$$\text{s.t.} \quad k + 1 - \sum_{i \in \tilde{A}} x_i \cdot c_i - \sum_{i \in \tilde{A}} a_i \cdot c_i \geq 0 \tag{20b}$$

$$\sum_{i \in \tilde{A}} x_i = 4 \cdot k + 2 \tag{20c}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} \leq 1 \qquad \text{for all } j \in L'' \tag{20d}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} \leq 1 \qquad \text{for all } i \in V'' \tag{20e}$$

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,k) \in A} x_{jk} = 0 \qquad \text{for all } i \in \tilde{V} \tag{20f}$$

$$y_i - y_j + 1 + |\tilde{V}| \cdot x_{ij} - |\tilde{V}| \cdot z_i \leq |\tilde{V}| \tag{20g}$$

$$\sum_{i \in V} z_i \leq 1 \tag{20h}$$

$$\sum_{(k,i) \in \delta^-(i)} x_{ki} - \sum_{(j,l) \in \delta^-(j)} x_{jl} \leq a_{ij} \qquad \text{for all } (i,j) \in \tilde{A} \tag{20i}$$

$$y_i \leq |\tilde{V}| \qquad \text{for all } i \in \tilde{V} \qquad (20j)$$
$$z_i \in \{0,1\} \qquad \text{for all } i \in \tilde{V} \qquad (20k)$$
$$x_i \in \{0,1\} \qquad \text{for all } i \in \tilde{A} \qquad (20l)$$
$$0 \leq a_i \leq 1 \qquad \text{for all } i \in \tilde{A} \qquad (20m)$$

From (20c) we can see that $\frac{|L'|+1}{2} = k+1$. As we prefer solutions where (19) is considerably violated we maximize the difference between $\frac{|L'|+1}{2}$ and $\sum_{i\in\tilde{A}} x_i \cdot c_i$. The term $\sum_{i\in\tilde{A}} a_i \cdot c_i$ gives a lower bound for the sum over all labels we need to lift w.r.t. some particular x. The correct coefficient which is to be discussed later on, cannot be formulated by a linear expression. By Equation (20b) this particular expression is enforced to be larger than zero, as the resulting inequality to be added to the MLST-MIP would not be violated otherwise. As a consequence all feasible solutions to MIP (20) fulfill this property which is desirable for the heuristic separation procedure discussed subsequently. For each node on the cycle the numbers of ingoing and outgoing arcs are limited to one by equations (20d) and (20e) and flow-conservation is imposed for each node (20f). The integer variables y_i assign numeric values to the nodes $i \in V'' \cup L''$ and prevent multiple cycles in the solution by Miller-Tucker-Zemlin-inequalities (20g), i.e. by enforcing for each arc on the cycle (except the one going out from the node i with $z_i = 1$ (20h)) to have an at least by one smaller source than target node. By Inequalities (20i) all arcs connecting nodes $i \in V'$ which *are* part of the odd cycle to be determined (by x-variables) to nodes $j \in L''(i)$ *not* being part of this cycle. Finally, y_i, for all $i \in \tilde{V}$ are enforced to be smaller than $|\tilde{V}|$ (20j), and the node selection and arc variables are required to be Boolean (20k, 20l). The a-variables only need to be restricted to $0 \leq a_i \leq 1$, for all $i \in \tilde{A}$, as they are implicitly integer by Inequalities (20i). Figure 5 shows an example for a solution to the MIP. The arcs selected by x-variables are depicted in red color, the dashed ones do not contribute to the objective function. The blue arcs correspond to the "lifting-arcs", selected by a-variables.

Given a solution to the MIP (20), we still need to check, if (19) is valid for this particular solution. The z-variables are derived by taking all labels $j \in L'$ selected by x_{ij} in (20). For this purpose we use the criterions described in [15] – here we only provide a rough explanation. An arc connecting two nodes on the odd cycle determined by (20) which is not part of the cycle itself is called a *chord*. In order to fulfill (18), and therefore (19) after the lifting, all chords of the odd cycle must be *compatible*. The chord set is called compatible, if 1) no chord induces even cycles (w.r.t. nodes $i \in V'$ on the cycle), and 2) every pair of crossing chords is compatible. Compatibility for crossing chords is defined on the basis of the mutual distances of their adjacent nodes on the cycle. Let $a_{ij} = (v_i, l_j),\ v_i \in V',\ l_j \in L'$ and $a_{hk} = (v_h, l_k),\ v_h \in V',\ l_k \in L'$ be two crossing chords. We now remove l_j and its two incident arcs from the odd hole. The chords are compatible, if the unique path from v_i to v_h has an even distance w.r.t. nodes in V' in this graph.

It remains to determine the lifting-coefficients γ_l. If a lifting-label only covers

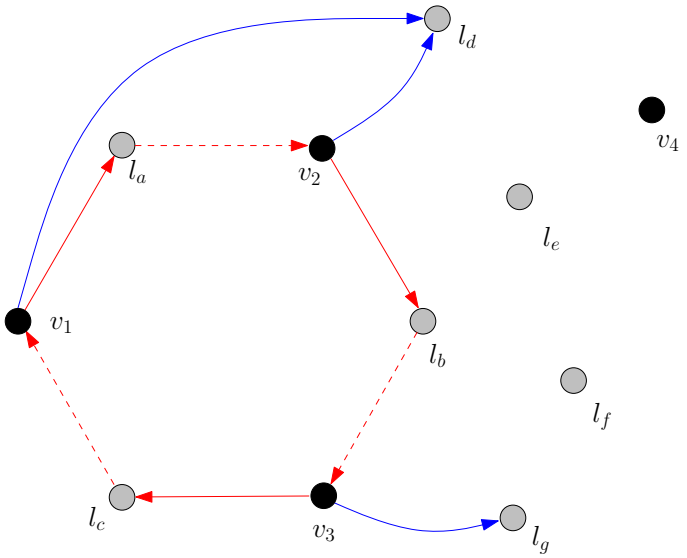

Figure 5: Example for a solution to (20). The octagon-shaped cycle constitutes the odd hole. The dashed arcs do not contribute to the objective function, whereas the solid arcs (which connect nodes to labels) contribute with the LP-value of the target-label as coefficient. The further arcs provide a lower bound for the contribution of all labels that need to be lifted in order to obtain a valid inequality for the initial problem.

one node of the odd hole, the sum over all labels necessary to feasibly cover all nodes from the odd hole does not change. The label can, however, be used alternatively for one of the odd hole labels and therefore gets coefficient one. Otherwise, if one lifting-label covers all odd hole nodes, the coefficient must equal the right hand side of (19), i.e. $\gamma_l = \frac{|L'|+1}{2}$ in this case. Suppose some lifting-label l covers ν_l odd hole nodes, then the size of the remaining odd hole nodes is $\frac{|L'|+1}{2} = \left\lceil \frac{|L'|}{2} \right\rceil$. These remaining nodes are still adjacent to two labels in the odd hole, pairwise having one label in common. We can therefore derive the following value for the lifting coefficient

$$\gamma_l = \left\lceil \frac{|L'|}{2} \right\rceil - \left\lceil \frac{|L'| - \nu_l}{2} \right\rceil = \frac{|L'|+1}{2} - \left(\frac{|L'|+1}{2} - \left\lceil \frac{\nu_l}{2} \right\rceil \right) = \left\lceil \frac{\nu_l}{2} \right\rceil . \tag{21}$$

During the branch-and-bound MLST solution process the MIP (20) is solved with very tight runtime-limits. As soon as an incumbent integer solution has been found, this solution is checked for validity by the mentioned criterions. Obtained valid MLST-inequalities are added immediately. Then the incumbent integer solution is rejected to the MIP solver by which we enforce to search for further solutions. This process continues until the time limit is reached.

3.4. Heuristics

In order to improve the overall performance – in particular the ability to generate feasible integer solutions fast – we embed a primal heuristic into the framework. For this purpose we adopt the well known MVCA heuristic [4, 18, 13]. This heuristic can create feasible solutions itself, but also complete partial solutions $\tilde{L} \subset L$. Creating complete solutions is important for the acquisition of strong upper bounds to efficiently cut-off unprofitable branches of the B&B-tree from the beginning on, but also to obtain an initial solution for BCP (Section 3.5). On the other hand the MVCA heuristic can be used to obtain feasible integer solutions and therefore upper bounds for each B&B-node based upon some variables already fixed to integer values. Many further fast metaheuristic techniques do exist for this problem, which could also easily be integrated into this framework. This is however beyond the scope of this work, as we primarily focus on mathematical programming methods for the MLST.

3.5. Pricing Problem

Problem formulations with a large (usually exponential) number of variables are frequently solved by column generation or branch-and-price algorithms. Such algorithms start with a restricted set of variables and add potentially improving variables during the solution process on demand. If these algorithms also include cutting-plane generation we call them branch-and-cut-and-price (BCP). Although the presented MLST formulation only has a polynomial number of label variables, these particular variables typically lead to extensive branching on them, requiring a special treatment. Hence we based a solution approach on BCP, operating on just a subset of variables. Such approaches follow the same idea as sparse graph techniques as proposed in [16].

We obtain the *restricted master problem* by replacing the complete set of labels L by a subset $L' \subseteq L$ in (1a). The set L' is required to imply a feasible solution and is obtained by the MVCA heuristic. Then, new variables and therefore columns potentially improving the current objective function value in the simplex tableau are created during the B&B process. These new variables are obtained from the solution of the *pricing problem* which is based upon the dual variables. Let π_i denote the dual variables corresponding to constraints (1b), and μ_i the ones corresponding to (16). They reflect a measure for the costs of some particular edge e w.r.t. the currently selected labels (π_e), and the costs of connecting some node v w.r.t. the currently selected labels (μ_v). The pricing problem is to find a variable with negative reduced costs

$$\bar{c}_l = 1 - \sum_{(i,j)\in A(l)} \pi_{ij} - \sum_{i \in V(l)} \mu_i, \tag{22}$$

within the set of all labels L. Here $A(l)$ denotes all arcs having label l, $V(l)$ denotes the set of nodes incident to arcs with label l. Finding such a variable or even the one with maximal reduced costs can be done by enumeration. Although only a

polynomial number of labels is involved, we may benefit from the pricing scheme as we only need to solve smaller LPs within the B&B procedure.

3.6. Polyhedral Comparison

In this section we compare various formulations resulting from combining the equations and inequalities from Section 3 as listed in Table 1. The only formulation just requiring a polynomial number of constraints is the flow-formulation with roughly $O(|L| + 3 \cdot |E|)$ variables and $O(|L| + |V| + |E|)$ constraints. The directed cut-formulation requires $O(|L| + 3 \cdot |E|)$ variables and an exponential number of constraints. Also the modified "epsilon" cut-formulation requires exponentially many constraints, but only has $O(|L| + |E|)$ variables.

Table 1: MLST formulations resulting from combining the equations and inequalities from Section 3. Further variants are given by the use of the components listed in the second part of the table, to be used as index for the formulation to be used with.

abbrevation	**involved equations and inequalities**
SCF	(1a), (1b), (3a) - (3c)
MCF	(1a), (1b), (2), (4), (3c)
DCut	(1a), (1b), (2), (6a), (6b), (7), (8)
EC	(1a), (1b), (2), (14), (11)
MTZ	(1a), (1b), (10a)
CEF	(1a), (1b), (9)
n	node-label-constraints (16)
$\tilde{n}$	extended node-label-constraints (17)
t	tree search, i.e. fixed number of edges (2)
s	strong linkage (15)
c	cycle elimination inequalities (9)
o	odd-hole inequalities
p	variable pricing

In the following we use the graph depicted in Figure 6 to show the properties of the polyhedra defined by the formulations listed in Table 1.

Proposition 3.2.

$$P^{\mathrm{SCF}_{tno}} \subsetneq P^{\mathrm{SCF}_{tn}} \subsetneq P^{\mathrm{SCF}_{t}} \subsetneq P^{\mathrm{SCF}} \tag{23}$$

Proof As $P^{\mathrm{SCF}_{tn}}$ contains the same equations and inequalities as $P^{\mathrm{SCF}_{\mathrm{t}}}$, but additionally Inequalities (16); thus we have $P^{\mathrm{SCF}_{\mathrm{tn}}} \subseteq P^{\mathrm{SCF}_{\mathrm{t}}}$. Figure 7 shows an LP solution of $P^{\mathrm{SCF}_{\mathrm{t}}}$ that is not contained in $P^{\mathrm{SCF}_{\mathrm{tn}}}$, which implies $P^{\mathrm{SCF}_{\mathrm{tn}}} \subsetneq P^{\mathrm{SCF}_{\mathrm{t}}}$. Such an LP solution may still contain fractional labels due to odd holes, as shown in Figure 5, by which we obtain $P^{\mathrm{SCF}_{\mathrm{tno}}} \subsetneq P^{\mathrm{SCF}_{\mathrm{tn}}}$.

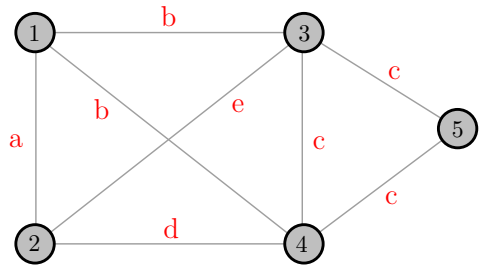

Figure 6: Example graph used in the following to show the properties of the formulations listed in Table 1. The set of labels is given by $L = \{a, b, c, d, e\}$, the optimal solution value is $f = 3$.

If the values of the edge and label variables in Figure 7 are decreased as much as possible for SCF, we obtain $l_a = 1/4$, $l_b = 3/8$ and $l_c = 1/8$ implying $f^{lp} = 3/4$. As SCF_t contains the additional Inequality (2), we can conclude that $P^{\text{SCF}_\text{t}} \subsetneq P^{\text{SCF}}$. □

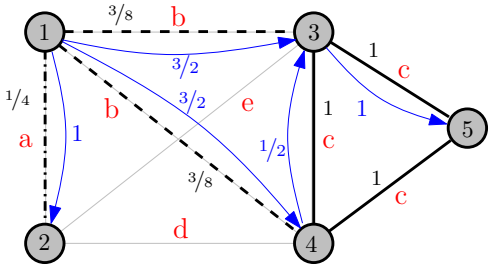

Figure 7: LP solution of SCF_t with objective value $f^{lp} = 1 + 5/8$ ($l_a = 1/4, l_b = 3/8, l_c = 1$). The blue arcs depict the flow variables with their according LP-values. This solution is not valid for SCF_tn, as the sum over the set of labels adjacent to node v_2 is smaller than one.

Proposition 3.3.

$$P^{\text{DCut}_\text{tno}} \subsetneq P^{\text{DCut}_\text{tn}} \subsetneq P^{\text{DCut}_\text{t}} \subsetneq P^{\text{DCut}} \tag{24}$$

Proof The proof of $P^{\text{DCut}_\text{tno}} \subsetneq P^{\text{DCut}_\text{tn}} \subsetneq P^{\text{DCut}_\text{t}}$ follows by the same reasoning as for the proof of theorem 3.2. Figure 8 shows that $P^{\text{DCut}_\text{tn}} \subsetneq P^{\text{DCut}_\text{t}}$. However, the requirement that each directed cut must have a value greater than one already implies that $\sum_{e \in \delta(v)} x_e \geq 1$, for all $v \in V$. This implies $\sum_{e \in E} x_e \geq |V| - 1$. An LP-solution to DCut may contain more edges than an LP-solution to DCut_t, which does, however, due to the minimality not affect the objective value of the LP-relaxation, i.e. $P_z^{\text{DCut}_\text{t}} = P_z^{\text{DCut}}$. □

Let P_S denote the projection of some polyhedron P to a subspace S.

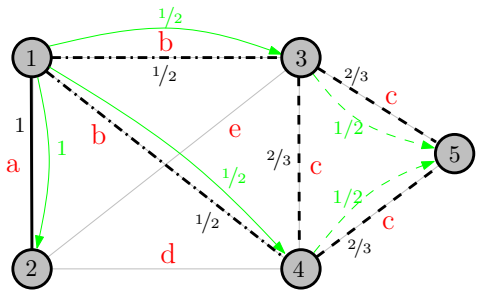

Figure 8: LP solution of $\mathrm{DCut_t}$ with objective value $f^{lp} = 2 + 1/6$ ($l_b = 1/2,\ l_c = 2/3,\ a + d + e \geq 1,$ w.l.o.g. $l_a = 1$). The green arcs depict the arc variables with their according LP values. The solution is not valid for $\mathrm{DCut_{tn}}$, as the sum over the set of labels adjacent to node v_5 is smaller than one.

Proposition 3.4.

$$P_x^{\mathrm{ECtno}} \subsetneq P_x^{\mathrm{ECtn}} \subsetneq P_x^{\mathrm{ECt}} \subsetneq P_x^{\mathrm{EC}} \tag{25}$$

Proof By applying the same reasoning as for the proofs of the last two theorems, we can prove Proposition 3.4. Figure 9 gives an example for $P^{\mathrm{ECtn}} \subsetneq P^{\mathrm{ECt}}$. □

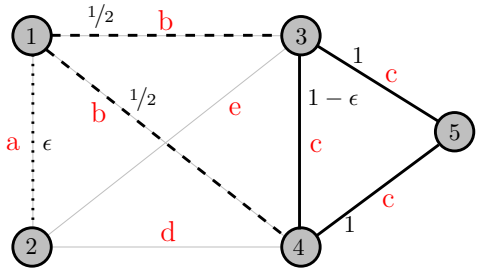

Figure 9: LP solution of $\mathrm{EC_t}$ with objective value $f^{lp} = 3/2 + \epsilon$ ($l_a = \epsilon, l_b = 1/2, l_c = 1$). The solution is not valid for $\mathrm{EC_{tn}}$, as the sum over the set of labels adjacent to nodes v_1 and v_2 are smaller than one.

In the following we will show the relations between the formulations $\mathrm{SCF_t}$, $\mathrm{DCut_t}$ and $\mathrm{EC_t}$.

Theorem 3.5.

$$P_x^{\mathrm{DCut_t}} \subsetneq P_x^{\mathrm{SCF_t}} \subsetneq P_x^{\mathrm{EC_t}} \tag{26}$$

Proof Figures 8, 7 and 9 already showed that the polyhedrons are not equal. To prove that $P_x^{\mathrm{DCut_t}} \subsetneq P_x^{\mathrm{SCF_t}}$ we show a procedure to transform all x-variables of any valid LP-solution of $\mathrm{DCut_t}$ to a valid x-solution in $\mathrm{SCF_t}$. For all $i, j \in V$ there exists at least one path from i to j with all edges (k, l) having LP-values x_{kl}^{lp} greater than zero. If we consider a network with source i and target j, only containing

edges e being part of one of these paths and having capacities x_e^{lp} there exists a flow of at least one unit from s to t. We now arbitrarily select a root node r (w.l.o.g. $r = 0$) and show how to construct a valid flow permitting the same x-configuration for SCF_t as in DCut_t. For an edge e to have LP value x_e^{lp} a corresponding flow variable must be larger than $x_e^{\text{lp}}/(n-1)$. We start by setting all flow variables to zero. Then for each node $t_i, i = 1, \ldots, n-1$ we construct all paths from r to t, considering all edges with $x_e^{\text{lp}} > 0$. Summing up $x_e^{\text{lp}} > 0$ for all edges e on these paths may not exceed $n-1$, as the number of edges is fixed by (2) when $i = 2$. However, this sum may usually be smaller than $n-1$, say λ_l, but integer. Now we backtrack all these paths and set their flow values to minimal values according to flow conservation (3b) and LP-values for the edges. Note that $\sum_{i \in \delta(r)} f_{ri} = \lambda_1$ after this first step. We then continue this procedure for all further $t_i, i = 2, \ldots, n-1$. According to (2) in step t_k at most $(n-1) - \sum_{l<k} \lambda_k$ not yet considered edges need to be added, possibly increasing $\sum_{i \in \delta(r)} f_{ri}$ by exactly this amount. We finally end up with all nodes being feasibly connected and $\sum_{i \in \delta(r)} f_{ri} = (n-1)$ fulfilling (3a) and flow conservation (3b) being fulfilled at each node.

It is trivial to see that the x-variables of a valid LP-solution of SCF_t is also valid for EC_t. □

Theorem 3.6.

$$P_x^{\text{DCut}_\text{tn}} \subsetneq P_x^{\text{SCF}_\text{tn}} \subsetneq P_x^{\text{EC}_\text{tn}} \tag{27}$$

Proof In the proof of Theorem 3.5 we already showed how each projection of a solution of DCut_t to the subspace defined by the x-variables can be transformed into a solution of SCF_t, and likewise SCF_t to EC_t. The only difference of the polyhedrons considered in Theorem 3.6 are the constraints (16), which clearly do not affect this transformation. It needs to be shown, that the polyhedrons are not equal, which is done by the example in Figure 10. The depicted EC_tn solution is not valid for SCF_tn or DCut_n respectively, although the node-label constraints (16) are fulfilled. However, the value of edge $\{3, 4\}$ can be increased to $1/5$ (implying the need to decrease the values of edges $\{1, 4\}$ and $\{3, 6\}$ accordingly), which makes the solution feasible to SCF_tn. Nevertheless, this solution remains infeasible to DCut_tn, by which we have shown the theorem. □

4. Results

In this section we present a comprehensive computational comparison of the presented formulations and separation strategies, and compare our methods to other work. Three different data sets are used for our computational tests. We start by a description of the test instances used for our experiments and tests.

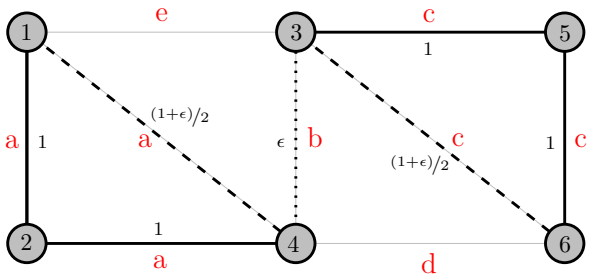

Figure 10: Valid LP-solution of EC_{tn} with $f^{lp} = 2 + \epsilon$ ($l_a = 1, l_b = \epsilon, l_c = 1$) that is not valid for SCF_{tn}. It can however be transformed to such, by increasing $x_{3,4}$ to $1/5$, yielding $f^{lp} = 2 + 1/5$. It is easy to see, that this solution is still not valid for DCut_{tn}.

4.1. Test Instances

The first set is the publicly available benchmark set used in [13, 11, 10, 3]. We refer to this data set as **Set-I**. It consists of graphs with 100 to 500 nodes and various densities $d \in \{0.2, 0.5, 0.8\}$, defined by $|E| = d \cdot \frac{|V| \cdot (|V|-1)}{2}$, and different numbers of labels $|L| = l/4, l \in \{1, 2, 4, 5\}$. The instances are organized in groups of ten for each configuration of d and $|L|$ for each $|V|$. So far, primarily metaheuristics have been applied to this instance set, but also an exact algorithm based on A^*-search, as reported in [11].

The second test set **Set-II** is created following the specification of the instances used in [2], in order to obtain comparable results to the MIPs presented therein. This set is organized in four groups. In contrast to Set-I, the instances of the first two groups just contain very few labels, i.e. $|L| \in \{5, 10, 20\}$. The number of nodes ranges from 20 to 1000, network densities are set to $|E| = 4 \cdot |V|$ Moreover, this set contains various *grid-graphs* (group 3) of sizes 2×10, 4×5, 2×18, 3×12, and 6×6. The fourth group contains instances with $|V| \in \{20, 50\}$ and $|L| = |V|$ and various network densities $d \in \{0.2, 0.5, 0.8\}$.

In addition to Set-I and Set-II we created a further test set **Set-III** containing also instances with multiple labels assigned to the edges. The construction is performed by first creating a spanning tree and assigning labels from set $L^* \subseteq L$ to its edges. Usually $L^* = L$ if not stated otherwise, but $|L^*| \ll |L|$ is used to study the effect of having optimal solutions with significantly less labels than for completely random label assignment for the particular graph properties. Next further edges are added until a specified density $d \cdot \frac{n \cdot (n-1)}{2}, 0 < d \leq 1$ or specified number of edges $m := |E|$ is reached. Then we randomly assign all labels not used yet. In the final step we iterate over all edges and assign further labels by uniform random decision. Parameter a specifies how many labels can be assigned to each edge, if not stated otherwise $a = 1$. Instead of directly using $|L|$ as a parameter, we may also specify the size of the label set by parameter $r = \frac{|L|}{|E|}, 0 < r \leq 1$. In contrast to the other instances, the instances of Set-III have relatively high values of r, i.e. $r = 1/4$ and $r = 3/4$. Although such instances are less likely to occur within practical

applications regarding telecommunication network design, they may be relevant for other scenarios, as for instance the compression model based on the MLST problem presented in [9]

4.2. Test environment

The generic framework presented in Section 3 has been implemented in C++ (`gpp-4.3`) within the SCIP framework [24]. The standard-plugins have been used for all computational tests unless explicitly stated otherwise. In addition some branch-and-cut algorithms (not involving any pricing procedures) have been implemented within the ILOG CONCERT framework [17] for comparison purposes. As LP solver ILOG CPLEX (in version 12.0) [17] has been used for both frameworks.

All computational tests have been performed on an Intel Xeon E5540 processor operating at 2.53 GHz and having 24 GB for totally 8 cores. The operation system is Ubuntu 9.10 with Linux-kernel 2.6.31. All runs have been performed in single-threaded mode, CPU times have been limited to 7200 seconds, unless stated otherwise.

4.3. Comparison of Described Methods

In this section we present a comparison of the described formulations based on computational tests. Furthermore we analyze the impact of particular "components" to each of the formulations. These components consist of the node-label-inequalities (16), the extended node-label-inequalities (17), the strong linkage of the edges to the edges (15), which can however only be used if only one label is assigned to the edges and the number of edges is not fixed by Equation (2). Table 1 provides an overview of these components and corresponding notation. After the comprehensive analysis and comparison of the particular methods in this section, we compare the results of the newly proposed methods to previous work in Section 4.4.

4.3.1. MIP formulations

In this section we primarily focus on the comparison of formulations EC, DCut and SCF. However, particularities like node-label-constraints (16), or fixed number of edges (2), or the direct linkage of labels to edges (15), may significantly change the picture regarding the superiority of one method over another one. For this reason we present the results not only for three formulations, but rather four to five variants of each formulation. Recall, that directly linking the labels to edges by Equations (15) is only possible for instances with one label assigned to each edge (15), i.e. $a = 1$ and is generally not possible for flow-formulations. In order not to be biased towards some particular class of instances we report these results for each of the three instance sets.

Tables 2 and 3 show the results for instances of SET-I with $|V| = 100$ and $|V| = 200$. These instances include graphs with various densities $d \in \{0.2, 0.5, 0.8\}$, where $|E| = d \cdot \frac{|V| \cdot (|V|-1)}{2}$, and different numbers of labels, i.e. $|L| = 1/2 \cdot |V|$, $|L| = |V|$, and $|L| = 5/4 \cdot |V|$. In these tables, as well as in the following ones, we report the

Table 2: Comparison of selected variants of formulations EC, DCut and SCF on the instances from SET-I with $|V| = 100$.

		$\|L\| = 50$						$\|L\| = 100$						$\|L\| = 125$					
d	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
0.2	EC	10	10	6.7	7	234	65	10	10	9.7	62	3876	1072	9	9	11.2	123	9365	2843
	EC_t	10	10	6.7	8	347	196	10	10	9.7	25	2482	414	9	9	11.2	34	3769	871
	EC_{tn}	10	10	6.7	4	100	173	10	10	9.7	11	857	339	9	9	11.2	16	1401	402
	EC_n	**10**	**10**	**6.7**	**0**	**24**	**3**	10	10	9.7	9	1298	179	9	9	11.2	33	2648	594
	EC_{sn}	**10**	**10**	**6.7**	**0**	**31**	**3**	**10**	**10**	**9.7**	**6**	**1127**	**34**	**9**	**9**	**11.2**	**12**	**2570**	**89**
	DCut	10	10	6.7	80	1167	691	10	10	9.7	1013	14378	7798	9	8	11.2	1470	19646	11403
	$DCut_t$	10	10	6.7	125	1458	954	10	10	9.7	478	6676	3919	9	9	11.2	740	10273	6450
	$DCut_{tn}$	10	10	6.7	21	127	116	10	10	9.7	86	1021	729	9	9	11.2	115	1426	1017
	$DCut_n$	10	10	6.7	11	90	96	10	10	9.7	57	1157	599	9	9	11.2	176	3186	1868
	$DCut_{sn}$	10	10	6.7	12	94	89	10	10	9.7	51	1190	516	9	9	11.2	129	3086	1536
	SCF	10	2	6.8	7028	136930	-1	10	0	10.3	7200	19854	-1	9	0	12.4	7200	17296	-1
	SCF_t	10	1	6.7	7133	143226	-1	10	0	10.3	7200	101319	-1	9	0	11.7	7200	119683	-1
	SCF_{tn}	10	10	6.7	15	83	-1	10	10	9.7	67	673	-1	9	9	11.2	85	1033	-1
	SCF_n	10	10	6.7	15	83	-1	10	10	9.7	63	673	-1	9	9	11.2	84	1033	-1
0.5	EC	10	10	3.0	17	46	5	10	10	4.7	374	5584	282	10	10	5.2	446	5307	630
	EC_t	10	10	3.0	14	69	148	10	10	4.7	175	5039	330	10	10	5.2	129	3249	382
	EC_{tn}	10	10	3.0	9	7	136	10	10	4.7	34	524	165	10	10	5.2	39	452	277
	EC_n	**10**	**10**	**3.0**	**0**	**2**	**0**	**10**	**10**	**4.7**	**8**	**221**	**4**	**10**	**10**	**5.2**	**9**	**127**	**4**
	EC_{sn}	**10**	**10**	**3.0**	**0**	**2**	**0**	10	10	4.7	9	173	4	10	10	5.2	11	370	12
	DCut	10	10	3.0	217	488	272	10	8	4.8	3858	15810	7214	10	10	5.2	3150	9316	4187
	$DCut_t$	10	10	3.0	190	465	256	10	9	4.7	3471	11770	6246	10	10	5.2	1794	5366	3175
	$DCut_{tn}$	10	10	3.0	56	13	28	10	10	4.7	401	1133	450	10	10	5.2	305	633	342
	$DCut_n$	10	10	3.0	27	12	28	10	10	4.7	261	1597	518	10	10	5.2	250	1179	401
	$DCut_{sn}$	10	10	3.0	27	23	55	10	10	4.7	216	1539	390	10	10	5.2	225	1234	362
	SCF	10	10	3.0	850	1475	-1	10	0	5.0	7200	11453	-1	10	0	5.8	7200	7586	-1
	SCF_t	10	10	3.0	722	1597	-1	10	6	4.7	5319	23618	-1	10	0	5.5	7200	18169	-1
	SCF_{tn}	10	10	3.0	22	1	-1	10	10	4.7	211	617	-1	10	10	5.2	171	298	-1
	SCF_n	10	10	3.0	20	1	-1	10	10	4.7	203	617	-1	10	10	5.2	176	298	-1
0.8	EC	10	10	2.0	12	2	12	10	10	3.0	161	848	98	10	10	4.0	999	4310	20
	EC_t	10	10	2.0	11	7	196	10	10	3.0	36	142	179	10	10	4.0	135	2344	130
	EC_{tn}	10	10	2.0	14	4	102	10	10	3.0	24	11	187	10	10	4.0	44	394	83
	EC_n	**10**	**10**	**2.0**	**0**	**1**	**0**	**10**	**10**	**3.0**	**1**	**3**	**2**	**10**	**10**	**4.0**	**17**	**102**	**0**
	EC_{sn}	**10**	**10**	**2.0**	**0**	**1**	**0**	10	10	3.0	2	4	1	**10**	**10**	**4.0**	**17**	**51**	**0**
	DCut	10	10	2.0	255	97	65	10	9	3.1	2367	2440	1286	10	0	4.0	7200	6958	3525
	$DCut_t$	10	10	2.0	198	127	131	10	9	3.1	1997	2702	1635	10	9	4.0	5083	6363	3410
	$DCut_{tn}$	10	10	2.0	87	7	19	10	10	3.0	314	370	216	10	10	4.0	923	1193	408
	$DCut_n$	10	10	2.0	40	5	22	10	10	3.0	418	932	457	10	10	4.0	780	1853	546
	$DCut_{sn}$	10	10	2.0	33	3	29	10	10	3.0	128	161	93	10	10	4.0	740	2154	668
	SCF	10	10	2.0	274	103	-1	10	0	3.6	7200	2826	-1	10	0	4.0	7200	2752	-1
	SCF_t	10	10	2.0	33	5	-1	10	0	4.0	7200	3035	-1	10	0	4.0	7200	2849	-1
	SCF_{tn}	10	10	2.0	28	1	-1	10	10	3.0	29	2	-1	10	10	4.0	243	349	-1
	SCF_n	10	10	2.0	28	1	-1	10	10	3.0	29	2	-1	10	10	4.0	230	349	-1

following entities for each method and group of instances: Columns "cnt" contain the number of instances within each group, which is 10 in most of the cases. The reason for less than ten instances reported is not being able to finish some instances

Table 3: Comparison of selected variants of formulations EC, DCut and SCF on the instances from SET-I with $|V| = 200$.

		$\|L\| = 100$						$\|L\| = 200$						$\|L\| = 250$					
d	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
0.2	EC	10	8	8.0	3770	52682	16621	10	0	12.9	7200	42289	22303	10	0	15.4	7200	32009	18669
	EC_t	10	7	8.0	4541	43756	7665	10	0	13.6	7200	43298	8007	10	0	16.0	7200	43792	7293
	EC_{tn}	10	10	7.9	1296	14254	539	10	2	13.3	6878	60643	1436	10	1	14.9	6761	46296	1658
	EC_n	10	10	7.9	201	8379	5	10	5	12.2	5146	189033	973	10	3	13.8	6387	144597	5207
	EC_{sn}	**10**	**10**	**7.9**	**191**	**8322**	**4**	**10**	**7**	**12.1**	**4331**	**182237**	**81**	**10**	**3**	**13.9**	**6153**	**250129**	**158**
	DCut	10	0	8.8	7200	10929	8971	10	0	14.5	7200	3850	3632	10	0	18.9	7200	3681	3548
	$DCut_t$	10	0	9.0	7200	4392	4036	10	0	14.5	7200	2896	2866	10	0	16.1	7200	2653	2622
	$DCut_{tn}$	10	7	8.1	5487	13056	8036	10	0	13.0	7200	5588	4763	10	0	15.7	7200	3982	3552
	$DCut_n$	10	9	8.0	3398	14653	8637	10	0	12.9	7200	15877	12200	10	0	14.9	7200	11207	9494
	$DCut_{sn}$	10	9	8.0	2996	14937	7658	10	0	12.7	7200	19280	13569	10	0	14.7	7200	13712	10837
	SCF	10	0	9.3	7200	1017	-1	10	0	14.5	7200	634	-1	10	0	16.8	7200	559	-1
	SCF_t	10	0	8.9	7200	3204	-1	10	0	13.5	7200	4733	-1	10	0	15.8	7200	5326	-1
	SCF_{tn}	10	8	8.0	3353	9362	-1	10	0	12.4	7200	6854	-1	10	0	14.1	7200	4343	-1
	SCF_n	10	8	8.0	3498	8308	-1	10	0	12.4	7200	6468	-1	10	0	14.1	7200	4099	-1
0.5	EC	10	10	3.4	769	2082	69	10	0	5.8	7200	8603	539	10	0	6.5	7200	5380	456
	EC_t	10	10	3.4	1452	2744	558	10	0	5.8	7200	10307	647	10	0	6.4	7200	9084	1024
	EC_{tn}	10	10	3.4	570	469	678	10	7	5.5	4249	6550	908	10	0	6.5	7200	15870	861
	EC_n	10	10	3.4	25	126	1	10	9	5.4	1291	14284	12	10	8	6.4	4323	57715	31
	EC_{sn}	**10**	**10**	**3.4**	**19**	**92**	**1**	**10**	**9**	**5.4**	**1176**	**14653**	**6**	**10**	**9**	**6.4**	**4049**	**62371**	**18**
	DCut	9	0	4.1	7200	1086	728	10	0	7.2	7200	323	265	10	0	7.9	7200	272	215
	$DCut_t$	9	0	4.3	7200	613	469	10	0	7.9	7200	298	290	10	0	8.2	7200	335	307
	$DCut_{tn}$	10	8	3.5	5135	954	481	10	0	6.6	7200	557	349	10	0	7.4	7200	420	282
	$DCut_n$	10	8	3.5	3132	1079	507	10	0	6.2	7200	1795	929	10	0	7.1	7200	1097	564
	$DCut_{sn}$	10	9	3.4	2054	979	412	10	0	6.0	7200	2072	912	10	0	6.7	7200	1432	671
	SCF	10	0	4.3	7200	124	-1	10	0	7.0	7200	69	-1	10	0	7.8	7200	54	-1
	SCF_t	10	0	3.9	7200	207	-1	10	0	6.5	7200	166	-1	10	0	7.3	7200	150	-1
	SCF_{tn}	10	10	3.4	1102	270	-1	10	0	5.7	7200	828	-1	10	0	6.5	7200	839	-1
	SCF_n	10	10	3.4	1204	270	-1	10	0	5.7	7200	749	-1	10	0	6.4	7200	728	-1
0.8	EC	10	10	2.6	2803	2968	16	10	0	4.0	7200	1132	16	10	0	5.0	7200	1640	48
	EC_t	10	10	2.6	3040	3650	505	10	0	4.0	7200	2146	656	10	2	4.4	7064	6763	757
	EC_{tn}	10	9	2.7	2739	103	613	10	10	4.0	5038	6819	634	10	9	4.1	2902	1331	776
	EC_n	10	10	2.6	76	2	73	10	10	4.0	1122	5975	1	10	10	4.0	609	3324	3
	EC_{sn}	**10**	**10**	**2.6**	**28**	**1**	**1**	**10**	**10**	**4.0**	**911**	**4845**	**1**	**10**	**10**	**4.0**	**301**	**777**	**1**
	DCut	4	0	3.0	7200	152	134	10	0	8.2	7201	105	110	10	0	7.1	7200	69	70
	$DCut_t$	10	0	3.9	7200	151	171	10	0	5.2	7202	110	112	10	0	6.8	7200	111	112
	$DCut_{tn}$	10	3	3.3	6344	142	67	10	0	4.6	7200	177	102	10	0	5.5	7200	120	98
	$DCut_n$	10	3	2.7	6528	1103	444	10	0	4.2	7200	329	170	10	0	4.9	7200	158	126
	$DCut_{sn}$	10	6	2.6	5135	799	292	10	0	4.0	7200	556	227	10	0	5.0	7200	226	140
	SCF	10	0	2.9	7200	69	-1	10	0	4.9	7200	35	-1	10	0	5.5	7200	33	-1
	SCF_t	10	2	2.8	5923	54	-1	10	0	4.3	7200	58	-1	10	0	5.1	7201	35	-1
	SCF_{tn}	10	9	2.6	4177	74	-1	10	1	4.0	6929	391	-1	10	3	4.7	6712	264	-1
	SCF_n	10	9	2.6	4183	79	-1	10	1	4.0	6973	375	-1	10	2	4.7	6871	229	-1

with particular formulations due to high memory requirements. Columns "opt" report the number of instances that have been solved and proved to be optimal within the timelimit. In columns "obj" the average objective value for all instances in the group is reported. If not all instances have been solved to optimality, this value corresponds to the average value of feasible solutions that have been found within the timelimit. Average running times in seconds are then reported in columns

"t". The average number of branch-and-bound nodes is listed in columns "bbn", the average number of generated cuts in column "cuts". Results of the fastest method(s) for each group are emphasized with bold letters.

From Tables 2 and 3 we can already observe that the difficulty of solving these instances is strongly correlated to the objective function values of the instances. Higher values, in particular those larger than ten, require significantly more B&B-nodes, and the separation of more cuts. This also implies longer average running times. This property holds for all of the considered formulations. The results in Tables 2 and 3 show that formulation $\mathrm{EC_{sn}}$ consistently gives the best results for these instances. The single commodity flow formulations show a slightly better performance than the directed cut formulations for most of the instances.

The strength of the node-label-inequalities (16) is also demonstrated by the results in Tables 2 and 3. Their addition to the plain formulations does not only yield a significant speedup, but also enables to solve more instances regarding the set with $|V| = 200$. The difference between Inequalities (16) and their extended form, given by Inequalities (17) is examined in Section 4.3.2. Regarding Equation (2) no clear conclusion can be drawn from these instances. If, however, combinations of these components are considered, the variants only using the node-label-constraints are superior in most of the cases. For formulations EC and DCut it is also possible to directly link the edges to the labels by Equations (15). In most of the cases, this yields the best results, when combined with the node-label-inequalities for both formulations, and in particular in combination with EC the overall best results.

Table 4 reports the results for the same formulations for the instances of Set-II. These instances have the major difference to contain only graphs of extremely low density d and just very few labels. Again, we can observe a clear superiority of formulations $\mathrm{EC_{sn}}$ and $\mathrm{EC_{n}}$, which are able to solve all these instances with average running times of less than a half second.

In Tables 5, 6, and 7 results for the instances from Set-III are reported. Table 5 shows the results for instances with $|V| = 100$ and $a = 1$, i.e. one single label assigned to the edges. As already mentioned in Section 4.1, this instances differ from the previous ones in the way that they contain a higher number of labels, i.e. $r = 1/4$ and $r = 3/4$ with $r = \frac{|L|}{|E|}$. It can be observed that it is beneficial to limit the number of edges to $|V| - 1$ by Equation (2) in this case. Thus, the stronger LP-relaxation implied by this restriction is beneficial in the case of higher values of r. For instances with $r = 1/4$ formulation EC still shows the best performance, but DCut provides better results in the case of $r = 3/4$. Hence, the strong LP-relaxation becomes even more important if $|L|$ is in the same order of magnitude as $|E|$.

With a single exception the same effect can be observed for the instances with $a \in \{2, 5\}$ reported in Table 6. The effect of more than one label being assigned to the edges seems to make the problem easier to solve, but the effect is relatively small. It is important to note, that directly linking the labels to the edges, which was beneficial for the instances with $a = 1$, cannot be applied to instances with

Table 4: Comparison of selected variants of formulations EC, DCut and SCF on the instances from SET-II.

		$\|L\| = 5$						$\|L\| = 10$						$\|L\| = 20$					
$\|V\|$, $\|E\|$	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
200, 800	EC	10	10	3.0	0	1	0	10	10	5.0	0	3	0	10	10	7.9	0	13	2
	EC_t	10	10	3.0	0	2	236	10	10	5.0	0	8	315	10	10	7.9	2	47	582
	EC_{tn}	10	10	3.0	0	2	157	10	10	5.0	0	6	249	10	10	7.9	1	20	515
	EC_n	10	10	3.0	0	1	0	10	10	5.0	0	1	0	10	10	7.9	0	6	2
	EC_{sn}	10	10	3.0	0	1	0	10	10	5.0	0	1	0	10	10	7.9	0	6	1
	DCut	10	10	3.0	1	6	38	10	10	5.0	4	14	63	10	10	7.9	14	127	171
	$DCut_t$	10	10	3.0	3	6	26	10	10	5.0	7	18	37	10	10	7.9	18	161	173
	$DCut_{tn}$	10	10	3.0	0	4	29	10	10	5.0	2	7	36	10	10	7.9	8	21	63
	$DCut_n$	10	10	3.0	0	2	24	10	10	5.0	0	6	49	10	10	5.0	0	6	49
	$DCut_{sn}$	10	10	3.0	0	1	0	10	10	5.0	0	1	0	10	10	7.9	0	6	1
	SCF	10	10	3.0	2	2	-1	10	10	5.0	9	32	-1	10	10	7.9	69	800	-1
	SCF_t	10	10	3.0	3	2	-1	10	10	5.0	9	32	-1	10	10	7.9	71	800	-1
	SCF_{tn}	10	10	3.0	0	1	-1	10	10	5.0	1	3	-1	10	10	7.9	6	16	-1
	SCF_n	10	10	3.0	0	1	-1	10	10	5.0	1	3	-1	10	10	7.9	6	16	-1
500, 2000	EC	10	10	3.5	0	1	0	10	10	5.9	0	2	0	10	10	9.9	0	12	0
	EC_t	10	10	3.5	3	2	621	10	10	5.9	6	7	833	10	10	9.9	16	44	1129
	EC_{tn}	10	10	3.5	2	1	541	10	10	5.9	5	6	763	10	10	9.9	16	25	1385
	EC_n	10	10	3.5	0	1	0	10	10	5.9	0	1	0	10	10	9.9	0	8	0
	EC_{sn}	10	10	3.5	0	1	0	10	10	3.5	0	1	0	10	10	9.9	0	7	0
	DCut	10	10	3.5	5	5	34	10	10	5.9	14	14	76	10	10	9.9	48	152	184
	$DCut_t$	10	10	3.5	13	7	25	10	10	5.9	28	15	34	10	10	9.9	68	181	144
	$DCut_{tn}$	10	10	3.5	2	3	16	10	10	5.9	9	8	33	10	10	5.9	9	8	33
	$DCut_n$	10	10	3.5	1	2	46	10	10	5.9	3	6	67	10	10	9.9	20	20	139
	$DCut_{sn}$	10	10	3.5	1	2	82	10	10	5.9	3	6	60	10	10	9.9	20	19	129
	SCF	10	10	3.5	10	3	-1	10	10	5.9	28	18	-1	10	10	9.9	372	661	-1
	SCF_t	10	10	3.5	11	3	-1	10	10	5.9	29	18	-1	10	10	9.9	384	661	-1
	SCF_{tn}	10	10	3.5	0	1	-1	10	10	5.9	4	3	-1	10	10	9.9	20	20	-1
	SCF_n	10	10	3.5	0	1	-1	10	10	5.9	3	3	-1	10	10	9.9	18	20	-1
1000, 4000	EC	10	10	4.1	0	1	0	10	10	6.6	0	1	0	10	10	11.3	0	13	0
	EC_t	10	10	4.1	20	1	1182	10	10	6.6	46	6	1762	10	10	11.3	121	54	3514
	EC_{tn}	10	10	4.1	300	1	3823	10	10	6.6	234	6	2660	10	10	11.3	108	26	2909
	EC_n	10	10	4.1	0	1	0	10	10	6.6	0	1	0	10	10	11.3	0	7	0
	EC_{sn}	10	10	4.1	0	1	0	10	10	6.6	0	1	0	10	10	11.3	0	6	0
	DCut	10	10	4.1	16	5	40	10	10	6.6	47	13	259	10	10	11.3	144	191	275
	$DCut_t$	10	10	4.1	54	6	22	10	10	6.6	90	20	36	10	10	11.3	240	189	150
	$DCut_{tn}$	10	10	4.1	7	3	23	10	10	6.6	26	7	26	10	10	11.3	103	36	47
	$DCut_n$	10	10	4.1	12	1	184	10	10	6.6	13	5	195	10	10	11.3	64	26	355
	$DCut_{sn}$	10	10	4.1	11	1	178	10	10	6.6	47	6	495	10	10	11.3	57	24	253
	SCF	10	10	4.1	30	2	-1	10	10	6.6	99	14	-1	10	10	11.3	1243	416	-1
	SCF_t	10	10	4.1	31	2	-1	10	10	6.6	96	14	-1	10	10	11.3	1303	416	-1
	SCF_{tn}	10	10	4.1	1	1	-1	10	10	6.6	12	3	-1	10	10	11.3	52	31	-1
	SCF_n	10	10	4.1	1	1	-1	10	10	6.6	12	3	-1	10	10	11.3	48	31	-1

larger a.

Table 7 shows the result for grid-graphs with 100 and 400 nodes and $|L| \in \{30, 50, 80\}$. The average optimal objective value on these graphs is relatively high,

Table 5: Comparison of selected variants of formulations EC, DCut and SCF on the instances from SET-III with $|V| = 100,\ a = 1$.

		$\|L\| = 1/4 \cdot \|E\|$						$\|L\| = 3/4 \cdot \|E\|$					
d	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
0.05	EC	10	10	19.6	2	1892	1722	10	0	61.6	7200	108393	124719
	EC_t	10	10	19.6	23	6887	6026	10	1	51.3	6480	178187	195559
	EC_{tn}	10	10	19.6	14	4944	4486	10	2	51.2	5760	163875	182801
	EC_n	10	10	19.6	2	1491	1246	10	0	62.5	7200	103413	119245
	EC_{sn}	**10**	**10**	**19.6**	**1**	**1313**	**966**	10	0	55.5	7200	133833	151889
	DCut	10	10	19.6	35	5214	3026	10	0	50.7	7200	367146	337417
	$DCut_t$	10	10	19.6	34	4174	2487	10	4	49.8	4381	76629	55867
	$DCut_{tn}$	10	10	19.6	12	1148	759	10	6	49.8	3195	87152	60318
	$DCut_n$	10	10	19.6	13	1437	941	10	0	50.4	7200	337882	315950
	$DCut_{sn}$	10	10	19.6	1	1313	966	10	0	50.4	7200	360801	343056
	SCF	10	3	19.6	5576	987720	-1	10	0	51.5	7200	1815207	-1
	SCF_t	10	9	19.6	1354	513872	-1	10	5	50.0	3823	3081605	-1
	SCF_{tn}	10	10	19.6	31	5160	-1	**10**	**8**	**49.9**	**1508**	**800971**	**-1**
	SCF_n	10	10	19.6	49	9415	-1	10	0	50.6	7200	894051	-1
0.2	EC	10	1	15.1	7099	208082	117362	10	0	45.1	7200	62300	63102
	EC_t	10	10	14.8	675	54742	23790	10	4	36.5	6326	138547	137549
	EC_{tn}	**10**	**10**	**14.8**	**344**	**36386**	**16745**	10	2	37.1	6450	120465	121687
	EC_n	10	2	15.3	6369	136657	94927	10	0	46.0	7200	40163	41557
	EC_{sn}	10	4	14.8	4894	231148	95062	10	0	39.2	7200	65167	63256
	DCut	10	0	16.3	7200	48073	35316	10	0	38.9	7200	119789	87861
	$DCut_t$	10	6	14.8	3196	55169	37144	10	6	35.8	3706	61762	47626
	$DCut_{tn}$	10	10	14.8	835	13677	8698	**10**	**7**	**35.8**	**2432**	**36099**	**27852**
	$DCut_n$	10	0	15.7	7200	39132	29700	10	0	38.5	7200	48576	40239
	$DCut_{sn}$	10	1	15.6	7134	78339	57546	10	0	38.0	7200	88645	73235
	SCF	10	0	17.0	7200	14435	-1	10	0	40.5	7200	39472	-1
	SCF_t	10	0	15.5	7200	173479	-1	10	0	37.8	7200	480980	-1
	SCF_{tn}	10	9	14.8	2401	31073	-1	10	1	36.1	6495	152260	-1
	SCF_n	10	0	15.2	7200	63078	-1	10	0	38.4	7200	29479	-1
0.5	EC	10	0	15.8	7200	69554	36260	10	0	38.8	7200	54153	55751
	EC_t	10	8	13.3	2570	62487	30425	10	5	30.7	4064	51442	50552
	EC_{tn}	**9**	**8**	**13.2**	**1400**	**34106**	**18769**	10	4	31.0	5038	59605	57375
	EC_n	10	0	16.2	7200	35944	18323	10	0	38.9	7200	28984	31127
	EC_{sn}	10	0	13.7	7200	296782	88962	10	0	33.8	7200	63430	64801
	DCut	10	0	15.4	7200	7645	4291	10	0	36.0	7200	31008	19324
	$DCut_t$	10	6	13.5	5152	9463	9036	10	6	30.3	4331	16003	13285
	$DCut_{tn}$	10	5	13.5	4557	9195	8543	**10**	**7**	**30.2**	**3552**	**12337**	**9780**
	$DCut_n$	10	0	15.4	7200	5807	3785	10	0	36.6	7200	6935	5131
	$DCut_{sn}$	10	0	14.4	7200	12834	8154	10	0	32.4	7200	14108	9913
	SCF	10	0	16.4	7200	1345	-1	10	0	34.5	7200	3160	-1
	SCF_t	10	0	14.6	7200	19313	-1	10	0	32.3	7200	35869	-1
	SCF_{tn}	10	5	13.5	4756	9842	-1	10	2	30.7	5842	16960	-1
	SCF_n	10	0	14.5	7200	6765	-1	10	0	33.1	7200	2327	-1

which makes them difficult to solve. However, all instances with $|L| \in \{30, 50\}$ could be solved to optimality by formulation EC_{sn}, which showed the overall best performance on this class of instances.

Table 6: Comparison of selected variants of formulations EC, DCut and SCF on the instances from SET-III with $|V| = 100,\ a \in \{2, 5\}$.

		$\lvert L\rvert = 1/4 \cdot \lvert E\rvert,\ a = 2$						$\lvert L\rvert = 3/4 \cdot \lvert E\rvert,\ a = 2$						$\lvert L\rvert = 1/4 \cdot \lvert E\rvert,\ a = 5$						$\lvert L\rvert = 3/4 \cdot \lvert E\rvert,\ a = 5$					
d	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
0.05	EC	10	10	16.6	7	3222	3277	10	0	42.8	7200	112158	123363	10	10	10.5	0	152	452	10	0	21.4	7200	111275	106501
	EC_t	10	10	16.6	63	11657	9901	10	7	35.0	2558	113102	106934	10	10	10.5	0	362	400	10	4	20.8	4763	178547	160611
	EC_{tn}	10	10	16.6	12	4707	4080	10	7	35.0	2375	102179	93532	10	10	10.5	0	306	359	10	6	20.6	3467	143690	125295
	EC_n	**10**	**10**	**16.6**	**2**	**1883**	**1956**	10	0	42.4	7200	116472	124838	10	10	10.5	0	158	426	10	0	21.5	7200	107718	103093
	DCut	10	10	16.6	31	4173	2409	10	0	35.1	7200	374197	286064	10	10	10.5	7	247	238	10	0	20.5	7200	375009	236293
	$DCut_t$	10	10	16.6	36	3523	2114	10	10	34.7	112	6725	4842	10	10	10.5	10	350	301	10	9	20.5	1066	75587	46230
	$DCut_{tn}$	10	10	16.6	16	1162	787	**10**	**10**	**34.7**	**61**	**4661**	**3696**	10	10	10.5	5	115	134	**10**	**10**	**20.5**	**698**	**45870**	**25073**
	$DCut_n$	10	10	16.6	11	1026	694	10	0	35.0	7200	376389	268567	10	10	10.5	3	95	115	10	7	20.5	4126	215290	129837
	SCF	10	1	16.7	7153	896512	-1	10	0	37.1	7200	701281	-1	10	9	10.5	2318	507224	-1	10	0	21.8	7200	276624	-1
	SCF_t	10	7	16.6	4038	1339695	-1	10	1	35.0	7046	3581766	-1	10	10	10.5	1580	440888	-1	10	0	21.0	7200	1707957	-1
	SCF_{tn}	10	10	16.6	29	3894	-1	10	10	34.7	609	239014	-1	10	10	10.5	4	284	-1	10	9	20.5	2218	462195	-1
	SCF_n	10	10	16.6	67	11755	-1	10	0	35.8	7200	666248	-1	10	10	10.5	5	329	-1	10	0	20.9	7200	720521	-1
0.2	EC	10	3	12.3	6521	268240	117079	10	0	31.8	7200	72720	69345	10	10	7.8	3364	236585	34987	10	0	19.5	7200	159231	107082
	EC_t	10	8	12.1	1840	81598	40109	10	3	26.2	5845	141568	127229	10	10	7.8	265	39875	773	10	3	15.1	5184	171490	89976
	EC_{tn}	**10**	**10**	**11.9**	**629**	**42052**	**20637**	10	3	26.3	5242	119550	107358	**10**	**10**	**7.8**	**128**	**12441**	**651**	10	3	15.1	5140	139983	90176
	EC_n	10	7	12.1	4285	166054	79457	10	0	34.0	7200	49859	50940	10	10	7.8	1041	56147	8166	10	0	19.0	7200	115188	89434
	DCut	10	0	13.4	7200	43412	32497	10	0	27.3	7200	72779	48772	10	1	7.9	6892	70690	36507	10	0	16.6	7200	33350	23851
	$DCut_t$	10	10	11.9	1477	17841	13183	10	7	25.6	2957	55433	38121	10	5	7.8	5653	67754	42560	10	4	14.9	5423	55800	43216
	$DCut_{tn}$	10	10	11.9	681	5906	5180	**10**	**9**	**25.6**	**1489**	**24931**	**17663**	10	10	7.8	1628	15222	9552	**10**	**6**	**14.8**	**4406**	**44628**	**32312**
	$DCut_n$	10	0	13.2	7200	39599	30225	10	0	27.4	7200	31085	23420	10	7	7.8	4679	65415	25812	10	0	15.8	7200	15761	11375
	SCF	10	0	14.3	7200	13282	-1	10	0	30.0	7200	19389	-1	10	0	8.8	7200	22933	-1	10	0	18.1	7200	12399	-1
	SCF_t	10	0	12.9	7200	105555	-1	10	0	27.0	7200	320786	-1	10	0	8.1	7200	107916	-1	10	0	15.8	7200	109149	-1
	SCF_{tn}	10	8	12.1	1909	18528	-1	10	4	25.9	5338	87496	-1	10	10	7.8	1669	11756	-1	10	2	15.1	6327	44397	-1
	SCF_n	10	0	12.8	7200	47542	-1	10	0	27.6	7200	15439	-1	10	9	7.8	4151	59801	-1	10	0	16.0	7200	10494	-1
0.5	EC	10	0	11.5	7200	98257	38000	10	0	28.4	7200	93989	77494	10	1	6.9	6785	101172	7242	10	0	14.5	7200	50518	29040
	EC_t	10	10	10.9	632	42951	5268	10	1	23.4	6487	85485	71003	10	10	6.9	612	27936	785	9	5	13.1	3804	100666	38284
	EC_{tn}	**10**	**10**	**10.9**	**506**	**22467**	**6432**	10	3	26.3	5242	119550	107358	**10**	**10**	**6.9**	**255**	**6604**	**532**	**9**	**6**	**13.0**	**3472**	**72582**	**38165**
	EC_n	10	0	11.3	7200	37858	18365	10	0	29.0	7200	44406	39926	10	0	7.0	7200	29141	2143	10	0	16.2	7200	11751	6165
	DCut	10	0	11.8	7200	9769	5503	10	0	30.4	7200	15852	9516	10	0	7.1	7200	11487	5995	10	0	15.2	7200	5742	3186
	$DCut_t$	10	0	11.3	7200	13089	11560	**10**	**8**	**22.5**	**4745**	**9585**	**8851**	10	0	7.4	7200	7977	6213	10	0	13.8	7200	7827	7958
	$DCut_{tn}$	10	5	11.2	5664	11813	9786	10	5	22.8	4259	8850	8481	10	6	7.1	5089	5318	3529	10	4	13.1	5743	7934	7389
	$DCut_n$	10	0	11.9	7200	7129	4426	10	0	27.1	7200	4738	2996	10	1	7.1	6717	7337	3172	10	0	15.2	7200	2682	1652
	SCF	10	0	13.2	7200	1477	-1	10	0	26.8	7200	1628	-1	10	0	8.0	7200	2606	-1	10	0	16.2	7200	923	-1
	SCF_t	10	0	12.1	7200	13970	-1	10	0	24.3	7200	26439	-1	10	0	7.6	7200	6757	-1	10	0	14.3	7200	10634	-1
	SCF_{tn}	10	6	10.9	4841	6975	-1	10	0	23.4	7200	12529	-1	10	7	7.0	4238	3003	-1	10	2	13.3	5923	6203	-1
	SCF_n	10	0	11.8	7200	9232	-1	10	0	25.0	7200	3000	-1	10	0	7.0	7200	21470	-1	10	0	14.4	7200	4668	-1

Table 7: Comparison of selected variants of formulations EC, DCut and SCF on the grid graph instances from SET-III with $|E| \approx 4 \cdot |V|$.

		$\|V\| = 10 \times 10$						$\|V\| = 20 \times 20$					
$\|L\|$	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
30	EC	10	10	9.2	4	2041	1540	10	10	11.5	61	7496	400
	EC_t	10	10	9.2	6	2641	1843	10	10	11.5	183	7547	5819
	EC_{tn}	10	10	9.2	4	1639	1544	10	10	11.5	91	3757	4231
	EC_n	**10**	**10**	**9.2**	**1**	**1369**	**656**	**10**	**10**	**11.5**	**17**	**3602**	**200**
	EC_{sn}	**10**	**10**	**9.2**	**1**	**1281**	**601**	**10**	**10**	**11.5**	**17**	**3558**	**175**
	DCut	10	10	9.2	61	4948	3240	2	2	11.0	5224	6233	37505
	$DCut_t$	10	10	9.2	93	6196	4135	10	10	11.5	3029	22956	19238
	$DCut_{tn}$	10	10	9.2	34	1892	1710	10	10	11.5	627	4058	10961
	$DCut_n$	10	10	9.2	18	1512	1364	1	0	12.0	7200	39	24766
	$DCut_{sn}$	10	10	9.2	19	1356	2314	1	0	13.0	7200	26	23113
	SCF	10	10	9.2	815	118270	-1	10	0	11.8	7200	22192	-1
	SCF_t	10	10	9.2	641	90448	-1	10	0	11.7	7200	22091	-1
	SCF_{tn}	10	10	9.2	17	1280	-1	10	10	11.5	251	3127	-1
	SCF_n	10	10	9.2	12	1008	-1	10	10	11.5	218	3076	-1
50	EC	9	8	13.0	1018	54653	41036	10	9	17.1	4341	464602	31132
	EC_t	9	7	13.1	1959	99397	73687	10	0	17.0	7200	301858	167539
	EC_{tn}	9	8	13.0	1578	83996	67834	10	10	17.0	3834	212265	107170
	EC_n	9	9	12.9	129	25299	16378	10	10	17.0	1347	208221	11217
	EC_{sn}	**9**	**9**	**12.9**	**66**	**22339**	**13131**	**10**	**10**	**17.0**	**1087**	**193213**	**8470**
	DCut	9	9	12.9	867	67360	37873	7	0	17.3	7200	29390	47832
	$DCut_t$	9	9	12.9	1032	74718	42088	10	0	17.2	7200	32248	33668
	$DCut_{tn}$	9	9	12.9	302	20596	12907	10	0	17.0	7200	86172	64236
	$DCut_n$	9	9	12.9	336	28974	17613	7	0	16.3	7200	3345	36785
	$DCut_{sn}$	9	9	12.9	207	20951	13323	7	0	15.7	7200	648	23682
	SCF	9	0	13.3	7200	368322	-1	10	0	18.5	7200	13103	-1
	SCF_t	9	0	13.2	7200	741954	-1	10	0	18.5	7200	19592	-1
	SCF_{tn}	9	9	12.9	215	21632	-1	10	0	17.1	7200	131296	-1
	SCF_n	9	9	12.9	299	34004	-1	10	8	17.1	5532	169723	-1
80	EC	10	0	19.5	7200	134608	118377	10	0	25.0	7200	199143	139246
	EC_t	10	0	19.9	7200	231789	208661	10	0	24.9	7200	143214	104486
	EC_{tn}	10	0	19.6	7200	229619	202787	10	0	24.8	7200	146616	112030
	EC_n	10	0	19.5	7200	162192	138904	10	0	24.6	7200	305748	138953
	EC_{sn}	10	0	18.9	7200	228167	176741	10	0	24.6	7200	299188	134344
	DCut	10	0	18.8	7200	252826	225273	10	0	25.5	7200	31756	32402
	$DCut_t$	10	0	18.9	7200	248123	216513	9	0	19.8	7200	24121	25391
	$DCut_{tn}$	9	0	18.7	7200	283625	220940	10	0	25.2	7200	43604	44755
	$DCut_n$	10	0	18.8	7200	239489	197073	8	0	24.6	7200	54226	62124
	$DCut_{sn}$	10	0	18.9	7200	248966	213157	7	0	25.1	7200	25262	46754
	SCF	10	0	19.7	7200	285092	-1	10	0	27.0	7200	9767	-1
	SCF_t	10	0	19.2	7200	948572	-1	10	0	28.0	7200	22180	-1
	SCF_{tn}	10	0	19.0	7200	689593	-1	10	0	25.3	7200	68435	-1
	SCF_n	10	0	18.8	7200	863039	-1	10	0	25.2	7200	99643	-1

Having now analyzed the main variations of the discussed formulations we draw

our attention to further approaches and enhancements that have been proposed in Section 3.

4.3.2. Further Methods

In Section 4.3.1 the node-label-inequalities (16) have been shown to be of utter importance for a strong formulation. In Section 3.3 we have also presented an extension of this idea, where two nodes are considered instead of just one. This led to the class of Inequalities (17). Table 8 shows a comparison of formulations $\mathrm{EC_t}$ and $\mathrm{DCut_t}$ with on the one hand the node-label-inequalities (16) and on the other hand additional Inequalities (17). In particular for formulation $\mathrm{EC_t}$ these further inequalities turn out to be useful in many cases. They do not only speedup the solution process, but moreover frequently enable to solve more instances to provable optimality. However, also the opposite is often the case. It is therefore not possible to decide which approach is superior over the other based on the available data. On grid-graphs Inequalities (17) have not been beneficial at all.

Further formulations, considered in Section 3, are based on the property, that a tree must not contain a cycle by definition. Formulation $\mathrm{MTZ_{tn}}$ requires just a polynomial number of variables, but contains constraints with infamous "Big-M" constants, as the SCF formulation does. On the contrary CEF contains an exponential number of Inequalities (9), which need to be separated as cutting-planes as for the DCut or EC formulation. Due to their fast separation by a simple shortest-path computation, also other formulations may benefit from additionally using cycle-elimination cuts. Corresponding results are reported in Table 9, column "cec" lists the average number of separated cycle-elimination cuts. Whereas $\mathrm{MTZ_{tn}}$ and $\mathrm{CEF_{tn}}$ show a relatively weak performance on the instances with $r = 1/4$, they provide good results in the case of $r = 3/4$. In particular for the low density graphs $\mathrm{CEF_{tn}}$ could solve all instances to optimality, which no other method was able to. For the dense graphs best results are obtained by $\mathrm{DCut_{tnc}}$ and $\mathrm{EC_{tnc}}$.

Table 10 shows the results that have been obtained by including primal heuristics into the branch-and-bound algorithm. Formulations $\mathrm{EC_{tn}}$, $\mathrm{EC_{sn}}$, $\mathrm{DCut_{tn}}$, and $\mathrm{DCut_{sn}}$ are considered for this purpose. As indicated by preliminary experiments it turned out to be advantageous only to use the primal heuristics in the root node, as they were generally not able to find improved solutions based on the information provided by the LP-solution in other B&B-nodes. Embedding MVCA in B&B has a positive effect w.r.t. the variants "tn" of formulations EC and DCut, but a negative impact concerning variants "ts".

4.3.3. Odd-Hole Inequalities

We now draw our attention to the odd-hole inequalities. Within preliminary tests we determined a tight timelimit of 10^{-3} seconds for solving the MIP (20) to show a generally good performance. Two algorithmic variants are considered for the results reported in Table 11. The first version (denoted with index o) simply adds the found valid cutting-planes to the MIP. Alternatively, the set of labels corresponding to the

obtained odd-hole can also be used to deduce a branching rule. This was motivated by the observation that many lifted odd-hole cutting planes, found by MIP (20), were not strong enough to define facets w.r.t. the involved label variables. As a consequence, these variables remained fractional after the cutting-plane was added to the MIP. However, odd-holes provide an important information and references to situations where special configurations of label-variables artificially reduce the LP-relaxation. Hence it is likely that immediately branching over these variables may be beneficial. This is done by inserting all labels of the odd-hole into a global queue, and always branch over such a variable unless the queue is empty. Index *ob* denotes this approach in Table 11. Odd-hole cuts are separated with lowest priority amongst the user-defined cutting-planes, and are only separated in levels of the B&B-tree which are multiples of ten.

The results in Table 11 show that the odd-hole inequalities are beneficial in many cases, in particular when used to deduce branching-rules from the corresponding label-variables. For instances from SET-I and SET-II almost no odd-holes have been found with the described parameter settings. For dense graphs it is less likely to find odd-holes that are violated by the current LP-solution, as each node is incident to many edges. Hence $|L(v)|$ is in the same order of magnitude as $|V|$ in the expected case. This implies many non-zero lifting coefficients in Inequalities (19), reducing the chance of finding a valid inequality that is actually violated by the current LP-solution. Hence, the separation of odd-hole inequalities is most beneficial for sparse graphs. Also the number of labels compared to the number of edges has an impact on the efficiency of the odd-hole separation. If the number of labels is relatively low, the expected label frequency ν_l will be high. This implies high values for the lifting coefficients γ_l, which in turn reduces the chance of finding violated odd-hole inequalities. If, on the other hand, the number of edges is too high, odd-holes are generally less likely to occur, as the sets $L(v) \cap L(u)$, for all $v, u \in V$ can be expected to be very small or even empty.

4.3.4. Branch-and-Cut-and-Price

Additionally using the column generation approach within the B&C framework, i.e. branch-and-cut-and-price (BCP) is only beneficial for a very special class of instances. For most of the instances almost all variables are priced in during the solution process. The computational overhead for solving the pricing problem and resolving the MIP implies significantly higher running times in this case. However, if the instances consist of a high number of labels, and have an optimal solution that is significantly lower than the average optimal solution value when assigning the labels to the edges randomly in the instance construction process, BCP shows a superior performance. To study this effect, special instances have been created containing single optima having a relatively low number of labels. The computational results for these instances are reported in Table 12. In particular for the larger instances a clear superiority of the BCP approach w.r.t. the corresponding B&C algorithm can be observed. For this special class of instances, the percentage of created label

variables is always less than 30% of the total number of labels (reported in column "priced"). Although the importance of such instances may be quite limited for many purposes, the instances used for the data compression approach presented in [9] exhibit comparable properties. For the data-compression application presented therein, the BCP approach is thus a valuable and important mean for exactly solving large instances.

4.3.5. Summary

In Table 13 we finally report the best method for each group of instances from the three instance sets. For this purpose, also variations including primal heuristic and using cycle-elimination cut separation are considered. In the case a variant including a primal heuristic yields the best performance, we additionally report the best method not using primal heuristics. Formulations $\mathrm{EC}_{\mathrm{sn}}$ and $\mathrm{EC}_{\mathrm{snh}}$ are the best formulations for almost all instances of SET-I, with the primal heuristic often yielding small improvements. The same is true for the instances of SET-II, where almost all variations of formulation EC are able to solve the considered instances in less than a second. For SET-III formulation DCut is superior for many instances with $|L| = 3/4 \cdot |E|$, whereas EC is better for instances with $|L| = 1/4 \cdot |E|$. In contrast to SET-I it is beneficial to restrict the number of edges to $|V| - 1$ as indicated with index "t". Additionally separating cycle-elimination cuts frequently yields the overall best method, in particular for instances with $|L| = 3/4 \cdot |E|$. Furthermore it can be observed that variants using separation of odd-hole inequalities are frequently the overall best methods for this group.

4.4. Comparison to Other Work

In this section we present direct comparisons to existing work, in particular [2]. Table 14 shows the results presented in [2], running times have been rounded to integers. Formulation "MLSTb" corresponds to formulation SCF of this work. Formulation "MLSTc" only uses a weaker coupling of labels to edges, given by the following inequalities

$$\sum_{(i,j)\in A} x_{ij} \leq \min\{|V| - 1, A(l)\} z_l, \quad \text{for all } l \in L. \tag{28}$$

Table 14 furthermore reports results for the implementation of the exact backtracking method from [4], labelled with "MLST-CL". Table 15 shows the running times of selected MIP variants in comparison to our reimplementation of the flow formulation "MLSTb" from [2] (SCF). Formulation $\mathrm{EC}_{\mathrm{tn}}$ is clearly superior to the others, all instances have been solved in less than one second. Higher running times of SCF as opposed to "MLSTb" can be explained due to the fact that the SCIP framework [24] has been used for the implementation of SCF, whereas "MLSTb" has been implemented with the ILOG CONCERT framework [17].

Table 16 shows the results of selected MIP variants in comparison to the exact A^* backtracking-search procedure used in [11]. The A^*-algorithm is very effective

for instances with small optimal objective value, but instances with larger objective values or large sets of labels cannot be solved. The time limit imposed by the authors of [11] was three hours. It is important to note that the running times listed in Table 16 are not directly comparable, as the authors of [11] list the computation time at which the best solution was obtained, and also different hardware has been used. For some groups, where A^* could not solve all instance (indicated by "NF"), the MIP method was able to do so. Furthermore, it is reported if the MIP method could solve some but not all instances within some group. In any case the average objective value for the ten instances of each group is reported in column "avg($|L_T|$)", also considering the best feasible solutions that have been found within the time limit of two hours. If not all instances have been solved to optimality, this is indicated with "(*)" in this particular columns.

In general it can be observed that relatively small instances could be solved efficiently by the MIP approach, but for larger instances with $|V| = 400$ and $|V| = 500$ it generally fails to produce provable optimal solutions within the allowed time limit.

4.5. Summary

For all formulations the node-label-constraints (16) significantly improved running-times and reduced the number of branch-and-bound nodes. Despite its relatively poor LP-relaxation, formulation $\mathrm{EC}_{\mathrm{tn}}$ turned out to be superior to the other ones for a broad class of test instances, which is mainly to the fast cut separation and the low number of involved variables. Amongst the other considered formulations $\mathrm{DCut}_{\mathrm{tn}}$ is superior over $\mathrm{EC}_{\mathrm{tn}}$ for dense graphs with a huge number of labels.

The odd-hole cuts (19) significantly improved running-times and number of branch-and-bound nodes for some classes of instances, in particular when branching-rules are deduced from the label sets corresponding to the found odd-holes. Using BCP for dynamically adding new labels during the solution process turned out to be only beneficial in the case where the input instances significantly deviate from random label assignments, i.e. where the optimal solution is much lower than the expectation value of randomly assigned labels. However, such solutions may likely easily be found also by heuristic methods. Nevertheless, this could remain the only way to prove optimality for "easy" large-scale instances.

5. Conclusions

In this work we presented a branch-and-cut(-and-price) framework for solving MLST instances exactly. We gave a comparison of an underlying flow-formulation in comparison to the (better) directed cut-based formulations. Furthermore, a new connectivity formulation permitting a fast cutting-plane separation has been presented. We further introduced the application of odd-hole inequalities to this problem. To separate cutting-planes based on these odd-hole inequalities, a separation

heuristic based on a mixed integer program using Miller-Tucker-Zemlin inequalities has been proposed.

Moreover, a detailed comparison of the contribution of the presented algorithmic building blocks has been presented. Our results show that the presented framework is able to solve small to medium sized instances to optimality within a relatively short amount of time. Existing benchmark instances could be solved within a significantly shorter computation time than before and new (larger) instances could be solved to proven optimality for the first time.

References

[1] T. Brüggemann, J. Monnot, and G. J. Woeginger. Local search for the minimum label spanning tree problem with bounded color classes. *Oper. Res. Lett.*, 31(3):195–201, 2003.

[2] M. Captivo, J. C. Clímaco, and M. M. Pascoal. A mixed integer linear formulation for the minimum label spanning tree problem. *Computers & Operations Research*, 36(11):3082 – 3085, 2009.

[3] R. Cerulli, A. Fink, M. Gentili, and S. Voß. Metaheuristics comparison for the minimum labelling spanning tree problem. In *Operations Research/Computer Science Interfaces Series*, volume 29, pages 93–106. Springer US, 2005.

[4] R.-S. Chang and S.-J. Leu. The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277–282, 1997.

[5] Y. Chen, N. Cornick, A. O. Hall, R. Shajpal, J. Silberholz, I. Yahav, and B. L. Golden. *Operations Research/Computer Science Interfaces*, chapter Comparison of Heuristics for Solving the Gmlst Problem, pages 191–217. Springer, 2008.

[6] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[7] M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Obtaining optimal k-cardinality trees fast. *J. Exp. Algorithmics*, 14:2.5–2.23, 2009.

[8] A. M. Chwatal, G. R. Raidl, and O. Dietzel. Compressing fingerprint templates by solving an extended minimum label spanning tree problem. In *Proceedings of the Seventh Metaheuristics International Conference (MIC)*, Montreal, Canada, 2007.

[9] A. M. Chwatal, G. R. Raidl, and K. Oberlechner. Solving a k-node minimum label spanning arborescence problem to compress fingerprint templates. *Journal of Mathematical Modelling and Algorithms*, 8:293–334, 2009.

[10] S. Consoli, K. Darby-Dowman, N. Mladenovic, and J. Moreno-Pérez. Solving the minimum labelling spanning tree problem using hybrid local search. Technical report, Brunel University, 2007.

[11] S. Consoli, K. Darby-Dowman, N. Mladenovic, and J. Moreno-Pérez. Heuristics based on greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem. *European Journal of Operational Research*, 196(2):440–449, 2009.

[12] S. Consoli, N. Darby-Dowman, K.and Mladenovic, and J. Moreno-Pèrez. Variable neighbourhood search for the minimum labelling steiner tree problem. *Annals of Operations Research*, 174(1):71–96, 2009.

[13] S. Consoli, J. A. Moreno, N. Mladenovic, and K. Darby-Dowman. Constructive heuristics for the minimum labelling spanning tree problem: a preliminary comparison. Technical report, DEIOC Technical Report, 2006.

[14] S. Consoli, J. A. Moreno, N. Mladenovic, and K. Darby-Dowman. Mejora de la exploración y la explotación de las heurísticas constructivas para el mlstp. In *Spanish Meeting on Metaheuristics 2007*, 2007.

[15] G. Cornuéjols and A. Sassano. On the 0, 1 facets of the set covering polytope. *Math. Program.*, 43(1):45–55, 1989.

[16] M. Grötschel and O. Holland. Solving matching problems with linear programming. *Mathematical Programming*, 1985.

[17] ILOG Concert Technology, CPLEX. ILOG. http://www.ilog.com. Version 12.0.

[18] S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85, 1998.

[19] I. Ljubić. *Exact and Memetic Algorithms for Two Network Design Problems*. PhD thesis, Technische Universität Wien, 2004.

[20] T. Magnanti and L. Wolsey. Optimal trees. *Handbook in Operations Research and Management Science*, Network Models:503–615, 1995.

[21] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960.

[22] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, November 1999.

[23] J. Nummela and B. A. Julstrom. An effective genetic algorithm for the minimum-label spanning tree problem. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 553–558, New York, NY, USA, 2006. ACM.

[24] SCIP – Solving Constraint Integer Programs. Konrad-Zuse-Zentrum für Informationstechnik Berlin. http://scip.zib.de/. Version 1.2.

[25] Y. Wan, G. Chert, and Y. Xu. A note on the minimum label spanning tree. *Information Processing Letters*, 84(2):99–101, 2002.

[26] Y. Xiong, B. Golden, and E. Wasil. A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 9(1):55–60, 2 2005.

[27] Y. Xiong, B. Golden, and E. Wasil. Improved heuristics for the minimum label spanning tree problem. In *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, volume 10, December 2006.

Table 8: Comparison of formulations $\mathrm{EC_t}$ and $\mathrm{DCut}_{\boldsymbol{t}}$ with Inequalities (16), indicated with index n, and with additional Inequalities (17), indicated with index $\tilde{n}$.

	$\mathrm{EC_{tn}}/\mathrm{DCut_{tn}}$						$\mathrm{EC_{t\tilde{n}}}/\mathrm{DCut_{t\tilde{n}}}$					
$\lvert V\rvert, \lvert E\rvert, a, \lvert L\rvert$	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
$100, 247, 1, 61$	10	10	19.6	14	4944	4486	**10**	**10**	**19.6**	**0**	**499**	**492**
	10	**10**	**19.6**	**12**	**1148**	**759**	10	10	19.6	19	1577	1034
$100, 247, 1, 185$	10	2	51.2	5760	163875	182801	**10**	**7**	**50.0**	**2174**	**65781**	**68723**
	10	**6**	**49.8**	**3195**	**87152**	**60318**	10	5	49.8	4284	86769	62183
$100, 900, 1, 247$	10	10	14.8	344	36386	16745	**10**	**10**	**14.8**	**177**	**18352**	**6741**
	10	**10**	**14.8**	**835**	**13677**	**8698**	10	10	14.8	1949	26973	16494
$100, 900, 1, 742$	10	2	37.1	6450	120465	121687	**10**	**7**	**35.8**	**2403**	**48038**	**45398**
	10	7	35.8	2432	36099	27852	**10**	**8**	**35.7**	**1822**	**20522**	**15153**
$100, 2475, 1, 618$	**10**	**8**	**13.2**	**1400**	**34106**	**18769**	9	8	13.2	1801	46469	22363
	10	**5**	**13.5**	**4557**	**9195**	**8543**	10	4	13.6	5417	10773	9933
$100, 2475, 1, 1856$	10	4	31.0	5038	59605	57375	**10**	**7**	**30.3**	**2506**	**27241**	**24957**
	10	7	30.2	3552	12337	9780	**10**	**8**	**30.2**	**2907**	**3715**	**3689**
$100, 247, 2, 61$	10	10	16.6	12	4707	4080	**10**	**10**	**16.6**	**1**	**601**	**498**
	10	**10**	**16.6**	**16**	**1162**	**787**	10	10	16.6	21	1348	907
$100, 247, 2, 185$	10	7	35.0	2375	102179	93532	**10**	**10**	**34.7**	**9**	**3269**	**3274**
	10	10	34.7	61	4661	3696	**10**	**10**	**34.7**	**19**	**1425**	**1226**
$100, 900, 2, 247$	**10**	**10**	**11.9**	**629**	**42052**	**20637**	10	10	11.9	912	55106	20864
	10	**10**	**11.9**	**681**	**5906**	**5180**	10	10	11.9	1523	16435	12011
$100, 900, 2, 742$	10	3	26.3	5242	119550	107358	**10**	**7**	**25.8**	**3265**	**69685**	**53785**
	10	**9**	**25.6**	**1489**	**24931**	**17663**	10	9	25.6	1583	17076	12240
$100, 2475, 2, 618$	**10**	**10**	**10.9**	**506**	**22467**	**6432**	10	10	10.9	558	37153	3921
	10	**5**	**11.2**	**5664**	**11813**	**9786**	10	1	11.7	7050	12503	11444
$100, 2475, 2, 1856$	**10**	**4**	**23.2**	**5213**	**61908**	**53294**	10	3	23.2	5757	73145	57657
	10	5	22.8	4259	8850	8481	**10**	**8**	**22.5**	**3649**	**4185**	**4254**
$100, 247, 5, 61$	**10**	**10**	**10.5**	**0**	**306**	**359**	**10**	**10**	**10.5**	**0**	**248**	**306**
	10	**10**	**10.5**	**5**	**115**	**134**	10	10	10.5	11	316	321
$100, 247, 5, 185$	10	6	20.6	3467	143690	125295	**10**	**9**	**20.5**	**1202**	**60571**	**49983**
	10	10	20.5	698	45870	25073	**10**	**10**	**20.5**	**498**	**36774**	**20977**
$100, 900, 5, 247$	**10**	**10**	**7.8**	**128**	**12441**	**651**	10	10	7.8	288	39324	825
	10	**10**	**7.8**	**1628**	**15222**	**9552**	10	5	7.8	5675	66499	42125
$100, 900, 5, 742$	10	3	15.1	5140	139983	90176	**9**	**4**	**15.0**	**4344**	**155589**	**73198**
	10	**6**	**14.8**	**4406**	**44628**	**32312**	10	4	14.9	5171	50434	38692
$100, 2475, 5, 618$	**10**	**10**	**6.9**	**255**	**6604**	**532**	10	10	6.9	624	27936	785
	10	**6**	**7.1**	**5089**	**5318**	**3529**	10	0	7.4	7200	8303	6431
$100, 2475, 5, 1856$	**10**	**6**	**13.0**	**3472**	**72582**	**38165**	10	6	13.0	3848	103635	41651
	10	**4**	**13.1**	**5743**	**7934**	**7389**	10	1	13.6	7191	8419	8617
$10 \times 10, 360, 1, 30$	**10**	**10**	**9.2**	**4**	**1639**	**1544**	10	10	9.2	6	2641	1843
	10	**10**	**9.2**	**34**	**1892**	**1710**	10	10	9.2	90	6196	4135
$10 \times 10, 360, 1, 50$	**9**	**8**	**13.0**	**1578**	**83996**	**67834**	9	8	13.0	1877	102421	75692
	9	**9**	**12.9**	**302**	**20596**	**12907**	9	9	12.9	1034	74718	42088
$10 \times 10, 360, 1, 80$	**10**	**0**	**19.6**	**7200**	**229619**	**202787**	10	0	19.9	7200	251957	226941
	10	**0**	**18.7**	**7200**	**283625**	**220940**	9	0	18.8	7200	283288	241399
$20 \times 20, 1520, 1, 30$	**10**	**10**	**11.5**	**91**	**3757**	**4231**	10	10	11.5	176	7547	5819
	10	**10**	**11.5**	**627**	**4058**	**10961**	10	10	11.5	2866	22956	19238
$20 \times 20, 1520, 1, 50$	**10**	**10**	**17.0**	**3834**	**212265**	**107170**	10	1	17.0	7194	326051	178058
	10	**0**	**17.0**	**7200**	**86172**	**64236**	10	0	17.2	7200	34742	35779
$20 \times 20, 1520, 1, 80$	**10**	**0**	**24.8**	**7200**	**146616**	**112030**	10	0	24.9	7200	150529	110120
	10	0	25.2	7200	43604	44755	**10**	**0**	**20.6**	**7200**	**25718**	**26997**

Table 9: Comparison of various formulations based on cycle elimination, i.e. the Miller-Tucker-Zemlin formulation MTZ and the CEF on the instances from SET-III with $|V| = 100$, $a = 1$. Furthermore results for connectivity-based formulations (EC and DCut), enhanced by cycle elimination inequalities are reported.

		$\|L\| = 1/4 \cdot \|E\|$							$\|L\| = 3/4 \cdot \|E\|$						
d	alg	cnt	opt	obj	t	bbn	cuts	cec	cnt	opt	obj	t	bbn	cuts	cec
	MTZ_{tn}	10	5	19.7	3931	502063	-1	-1	10	7	49.9	3112	721279	-1	-1
0.05	CEF_{tn}	10	6	19.6	3000	135407	-1	16638	10	10	49.8	901	94205	-1	6389
	$CEF_{t\tilde{n}}$	10	5	19.6	3998	155632	-1	17642	10	7	49.9	2208	208113	-1	14321
	EC_{nc}	10	10	19.6	14	1353	121	59	10	0	50.3	7200	532836	170798	1783
	EC_{tnc}	10	10	19.6	12	915	153	96	10	7	49.8	2566	62656	44607	5502
	$DCut_{nc}$	10	10	19.6	13	1433	931	55	10	0	50.4	7200	376649	351288	1630
	$DCut_{tnc}$	10	10	19.6	13	1029	700	94	10	7	49.8	2291	36745	27748	2859
	MTZ_{tn}	10	7	15.0	4276	45276	-1	-1	10	5	35.8	4272	87003	-1	-1
0.2	CEF_{tn}	10	7	14.9	3217	36913	-1	3298	10	7	35.7	2313	91215	-1	5422
	$CEF_{t\tilde{n}}$	10	3	15.2	5307	61399	-1	5690	10	7	35.7	2668	40566	-1	2478
	EC_{nc}	10	0	15.6	7200	31835	143	118	10	0	37.8	7200	59337	1533	30
	EC_{tnc}	10	10	14.8	701	10687	196	670	10	8	35.7	1871	51079	15544	3426
	$DCut_{nc}$	10	0	15.9	7200	39225	29755	171	10	0	39.4	7200	47294	39206	3
	$DCut_{tnc}$	10	10	14.8	737	11214	7212	581	10	8	35.7	1537	21721	13795	1400
	MTZ_{tn}	10	5	13.5	5555	7818	-1	-1	10	3	30.9	5658	13851	-1	-1
0.5	CEF_{tn}	10	5	13.6	5038	8686	-1	763	10	7	30.1	4444	19156	-1	1653
	$CEF_{t\tilde{n}}$	10	3	13.6	5791	8399	-1	1063	10	5	30.5	5570	6646	-1	711
	EC_{nc}	10	0	14.1	7200	3463	26	35	10	0	32.7	7200	6497	118	25
	EC_{tnc}	10	7	13.5	3865	8772	116	913	10	9	30.1	2112	9120	1344	665
	$DCut_{nc}$	10	0	15.6	7200	5353	3395	24	10	0	38.2	7200	6964	5357	6
	$DCut_{tnc}$	10	8	13.5	3394	7452	6433	675	10	9	30.0	2427	7475	5918	576

Table 10: Comparison of best formulations used without and with primal heuristics, i.e. MVCA and ACO.

		$\lvert L\rvert = 1/4 \cdot \lvert E\rvert$						$\lvert L\rvert = 3/4 \cdot \lvert E\rvert$					
d	alg	cnt	opt	obj	t	bbn	cuts	cnt	opt	obj	t	bbn	cuts
	EC_{tn}	10	10	19.6	14	4944	4486	10	2	51.2	5760	163875	182801
0.05	EC_{sn}	10	10	19.6	1	1313	966	10	0	55.5	7200	133833	151889
	$DCut_{tn}$	10	10	19.6	12	1148	759	10	6	49.8	3195	87152	60318
	$DCut_{sn}$	10	10	19.6	10	1336	895	10	6	49.8	3195	87152	60318
	EC_{tn} + MVCA	10	10	19.6	12	5474	4866	10	2	51.3	5771	145352	166307
	EC_{sn} + MVCA	10	10	19.6	2	1635	1255	10	0	52.3	7200	137776	154933
	$DCut_{tn}$ + MVCA	10	10	19.6	11	1056	714	10	6	49.8	2902	57282	41009
	$DCut_{sn}$ + MVCA	10	10	19.6	12	1530	993	10	0	50.4	7200	371927	343330
	EC_{tn} + ACO	10	10	19.6	12	4425	3916	8	4	49.6	3600	107155	110775
	EC_{sn} + ACO	10	10	19.6	2	1339	978	10	0	49.9	7200	134930	151941
	$DCut_{tn}$ + ACO	10	10	19.6	10	937	627	9	4	49.8	4019	74506	52846
	$DCut_{sn}$ + ACO	10	10	19.6	11	1303	828	10	0	50.4	7200	371927	343330
	EC_{tn}	10	10	14.8	344	36386	16745	10	2	37.1	6450	120465	121687
0.2	EC_{sn}	10	4	14.8	4894	231148	95062	10	0	39.2	7200	65167	63256
	$DCut_{tn}$	10	10	14.8	835	13677	8698	10	7	35.8	2432	36099	27852
	$DCut_{sn}$	10	10	14.8	835	13677	8698	10	0	38.0	7200	88645	73235
	EC_{tn} + MVCA	10	10	14.8	767	52496	28531	10	2	36.9	6004	120042	120862
	EC_{sn} + MVCA	10	4	14.8	4904	212222	89873	10	0	38.7	7200	63308	60554
	$DCut_{tn}$ + MVCA	10	10	14.8	799	12503	8284	10	8	35.7	2169	34219	25835
	$DCut_{sn}$ + MVCA	10	1	15.5	7067	72658	54894	10	0	38.1	7200	82575	67712
	EC_{tn} + ACO	10	10	14.8	345	31157	14313	9	5	36.0	4135	67362	66292
	EC_{sn} + ACO	10	3	14.9	5323	228273	94410	10	0	36.1	7200	64340	60405
	$DCut_{tn}$ + ACO	10	10	14.8	640	11034	7149	9	6	35.8	2524	35560	26632
	$DCut_{sn}$ + ACO	10	2	15.0	6977	80725	54925	10	0	36.2	7200	83553	69016
	EC_{tn}	9	8	13.2	1400	34106	18769	10	4	31.0	5038	59605	57375
0.5	EC_{sn}	10	0	13.7	7200	296782	88962	10	0	33.8	7200	63430	64801
	$DCut_{tn}$	10	5	13.5	4557	9195	8543	10	7	30.2	3552	12337	9780
	$DCut_{sn}$	10	0	14.4	7200	12834	8154	10	7	30.2	3552	12337	9780
	EC_{tn} + MVCA	10	8	13.3	1991	40440	23995	10	5	30.5	4274	53202	50948
	EC_{sn} + MVCA	10	0	13.7	7200	279895	79633	10	0	32.4	7200	67026	66888
	$DCut_{tn}$ + MVCA	10	7	13.3	3321	6462	5852	10	9	30.0	2728	8479	7236
	$DCut_{sn}$ + MVCA	10	0	14.3	7200	11154	7224	10	0	32.2	7200	14878	10082
	EC_{tn} + ACO	9	7	13.3	2261	44962	25718	6	2	31.3	5419	61481	59297
	EC_{sn} + ACO	10	0	13.7	7200	345358	80054	10	0	31.1	7200	64469	63099
	$DCut_{tn}$ + ACO	10	7	13.4	3534	7659	6933	8	5	30.5	3374	10577	8304
	$DCut_{sn}$ + ACO	10	0	13.8	7200	12035	7319	10	0	31.2	7200	13919	9175

Table 11: Comparison of formulations EC_t and $DCut_{\boldsymbol{t}}$ with and without using odd-hole inqualities. Index o denotes if odd-hole inequalities are separated, index b indicates that odd-hole inequalities have been used to induce branching over related label variables.

	$EC_{tn}/DCut_{tn}$						$EC_{tno}/DCut_{tno}$						$EC_{tnob}/DCut_{tnob}$					
$\|V\|, \|E\|, a, \|L\|$	cnt	opt	obj	t	bbn	cuts	opt	obj	t	bbn	cuts	oh	opt	obj	t	bbn	cuts	oh
$100, 247, 1, 61$	10	**10**	**19.6**	**14**	**4944**	**4486**	10	19.6	25	5692	5141	23	10	19.6	17	5226	4701	19
	10	**10**	**19.6**	**12**	**1148**	**759**	10	19.6	13	1163	772	1	10	19.6	13	1121	733	1
$100, 247, 1, 185$	10	**2**	**51.2**	**5760**	**163875**	**182801**	2	51.2	5760	144151	159964	194	2	51.1	5760	143562	159321	173
	10	**6**	**49.8**	**3195**	**87152**	**60318**	6	49.8	3214	83347	58322	11	6	49.8	3197	85997	60585	13
$100, 900, 1, 247$	10	10	14.8	344	36386	16745	10	14.8	417	37993	18284	134	**10**	**14.8**	**343**	**34639**	**15817**	**120**
	10	**10**	**14.8**	**835**	**13677**	**8698**	10	14.8	892	14659	9364	1	10	14.8	987	15816	10131	2
$100, 900, 1, 742$	10	2	37.1	6450	120465	121687	**3**	**36.8**	**6377**	**126819**	**128924**	**1251**	2	36.7	6007	119218	119667	1308
	10	7	35.8	2432	36099	27852	**7**	**35.8**	**2369**	**34717**	**26348**	**77**	7	35.8	2371	33757	26083	61
$100, 2475, 1, 618$	10	**8**	**13.2**	**1400**	**34106**	**18769**	8	13.3	2499	48599	28883	174	8	13.3	1820	35479	21775	138
	10	5	13.5	4557	9195	8543	**7**	**13.5**	**4261**	**8305**	**7615**	**9**	7	13.3	4318	8322	7769	8
$100, 2475, 1, 1856$	10	4	31.0	5038	59605	57375	**5**	**30.8**	**4133**	**53079**	**52191**	**709**	4	31.0	4499	54945	54254	614
	10	7	30.2	3552	12337	9780	9	30.0	2827	8780	6809	51	**9**	**30.0**	**2778**	**8061**	**6898**	**39**
$100, 247, 2, 61$	10	**10**	**16.6**	**12**	**4707**	**4080**	10	16.6	14	5148	4496	23	10	16.6	13	4987	4363	20
	10	**10**	**16.6**	**16**	**1162**	**787**	10	16.6	17	1161	788	0	10	16.6	16	1146	775	1
$100, 247, 2, 185$	10	**7**	**35.0**	**2375**	**102179**	**93532**	6	35.1	3485	137028	125999	272	7	35.0	2431	102595	90520	178
	10	10	34.7	61	4661	3696	10	34.7	69	5073	4081	8	**10**	**34.7**	**22**	**1758**	**1630**	**4**
$100, 900, 2, 247$	10	**10**	**11.9**	**629**	**42052**	**20637**	9	12.0	997	48954	22866	137	10	11.9	857	41933	21903	129
	10	**10**	**11.9**	**681**	**5906**	**5180**	10	11.9	891	8242	6868	6	10	11.9	1000	8936	7734	11
$100, 900, 2, 742$	10	3	26.3	5242	119550	107358	3	26.2	5395	118483	104341	1312	**4**	**26.3**	**4841**	**107072**	**94593**	**1220**
	10	9	25.6	1489	24931	17663	9	25.6	1603	24112	18031	82	**9**	**25.6**	**1443**	**23837**	**16801**	**67**
$100, 2475, 2, 618$	10	10	10.9	506	22467	6432	10	10.9	748	28917	9107	44	**10**	**10.9**	**388**	**17770**	**3684**	**14**
	10	5	11.2	5664	11813	9786	5	11.2	5736	10958	9467	7	**6**	**11.1**	**5727**	**11155**	**9316**	**8**
$100, 2475, 2, 1856$	10	**4**	**23.2**	**5213**	**61908**	**53294**	1	23.8	6490	74525	64822	822	2	23.5	6298	72256	62059	754
	10	5	22.8	4259	8850	8481	7	22.6	3642	7480	7073	22	**8**	**22.5**	**3479**	**7831**	**7173**	**20**
$100, 247, 5, 61$	10	**10**	**10.5**	**0**	**306**	**359**	10	10.5	0	306	359	1	10	10.5	1	267	338	1
	10	**10**	**10.5**	**5**	**115**	**134**	10	10.5	5	115	134	0	10	10.5	5	115	134	0
$100, 247, 5, 185$	10	6	20.6	3467	143690	125295	4	20.7	4461	157779	137051	937	**6**	**20.6**	**3047**	**116829**	**103068**	**673**
	10	**10**	**20.5**	**698**	**45870**	**25073**	9	20.5	787	43883	24921	12	9	20.5	783	45144	25575	10
$100, 900, 5, 247$	10	**10**	**7.8**	**128**	**12441**	**651**	10	7.8	134	12521	631	2	10	7.8	157	13710	862	3
	10	10	7.8	1628	15222	9552	**10**	**7.8**	**1513**	**15071**	**9222**	**1**	10	7.8	1540	15153	9365	0
$100, 900, 5, 742$	10	3	15.1	5140	139983	90176	4	15.0	4838	132756	83094	702	**4**	**15.0**	**4458**	**125665**	**78327**	**651**
	10	**6**	**14.8**	**4406**	**44628**	**32312**	5	14.9	4520	45818	33172	22	5	14.9	4392	44435	32267	18
$100, 2475, 5, 618$	10	**10**	**6.9**	**255**	**6604**	**532**	10	6.9	253	6581	519	0	10	6.9	262	6701	577	0
	10	**6**	**7.1**	**5089**	**5318**	**3529**	6	7.0	5092	5503	3685	0	6	7.0	5093	5447	3608	0
$100, 2475, 5, 1856$	10	6	13.0	3472	72582	38165	7	12.9	3132	62473	33275	255	**7**	**12.9**	**2343**	**50106**	**24087**	**158**
	10	**4**	**13.1**	**5743**	**7934**	**7389**	3	13.3	6505	8896	8420	9	3	13.1	6213	8259	7770	8
$10 \times 10, 360, 1, 30$	10	**10**	**9.2**	**4**	**1639**	**1544**	10	9.2	5	1427	1350	10	10	9.2	5	1427	1384	11
	10	10	9.2	34	1892	1710	**10**	**9.2**	**32**	**1685**	**1510**	5	10	9.2	34	1711	1576	6
$10 \times 10, 360, 1, 50$	9	8	13.0	1578	83996	67834	8	13.0	1550	81112	66089	310	**9**	**12.9**	**619**	**54900**	**42488**	**145**
	9	9	12.9	302	20596	12907	9	12.9	387	25142	16180	23	**9**	**12.9**	**298**	**19938**	**12261**	**8**
$10 \times 10, 360, 1, 80$	10	0	19.6	7200	229619	202787	0	19.8	7200	211903	186885	1086	**0**	**19.5**	**7200**	**211561**	**188812**	**1102**
	10	**0**	**18.7**	**7200**	**283625**	**220940**	0	18.8	7200	232626	190432	403	0	19.0	7200	223970	188832	381
$20 \times 20, 1520, 1, 30$	10	**10**	**11.5**	**91**	**3757**	**4231**	10	11.5	107	3749	4310	0	10	11.5	106	3799	4249	0
	10	**10**	**11.5**	**627**	**4058**	**10961**	10	11.5	753	4028	11567	0	10	11.5	705	4112	11261	0
$20 \times 20, 1520, 1, 50$	10	**10**	**17.0**	**3834**	**212265**	**107170**	9	17.0	4232	209923	105755	32	9	17.0	4324	210003	107385	39
	10	**0**	**17.0**	**7200**	**86172**	**64236**	0	17.0	7200	77221	60603	6	0	17.0	7200	80453	59244	4
$20 \times 20, 1520, 1, 80$	10	**0**	**24.8**	**7200**	**146616**	**112030**	0	24.8	7200	133250	102086	626	0	24.9	7200	131641	101479	577
	10	**0**	**25.2**	**7200**	**43604**	**44755**	0	25.2	7200	39262	41704	21	0	25.2	7200	39451	42952	19

Table 12: Branch-and-cut-and-price results for a special class of instances containing many labels and isolated optima with a relatively low number of labels.

$\|V\|, \|E\|, a, \|L\|$	method	cnt	opt	obj	t	bbn	cuts	priced
$100, 247, 2, 61$	EC_{tn}	10	10	5.0	0	1	32	-1
	DCut_{tn}	10	10	5.0	0	1	7	-1
	EC_{tnp}	10	10	5.0	0	1	64	14
	DCut_{tnp}	10	10	5.0	0	1	13	17
$100, 247, 2, 185$	EC_{tn}	10	10	10.0	0	1	1	-1
	DCut_{tn}	10	10	10.0	0	1	2	-1
	EC_{tnp}	10	10	10.0	0	1	2	11
	DCut_{tnp}	10	10	10.0	0	1	3	7
$100, 900, 2, 247$	EC_{tn}	10	10	5.0	0	1	30	-1
	DCut_{tn}	10	10	5.0	1	1	15	-1
	EC_{tnp}	10	10	5.0	0	1	72	29
	DCut_{tnp}	10	10	5.0	0	2	19	28
$100, 900, 2, 742$	EC_{tn}	10	10	10.0	0	14	42	-1
	DCut_{tn}	10	10	10.0	8	13	25	-1
	EC_{tnp}	10	10	10.0	2	497	328	30
	DCut_{tnp}	10	10	10.0	12	32	41	25
$100, 2475, 2, 618$	EC_{tn}	10	10	5.0	1	2	46	-1
	DCut_{tn}	10	10	5.0	19	4	15	-1
	EC_{tnp}	10	10	5.0	1	6	51	27
	DCut_{tnp}	10	10	5.0	11	4	19	26
$100, 2475, 2, 1856$	EC_{tn}	10	10	10.0	2	15	48	-1
	DCut_{tn}	10	10	10.0	40	11	23	-1
	EC_{tnp}	10	10	10.0	10	237	174	24
	DCut_{tnp}	10	10	10.0	36	23	26	16
$300, 22425, 2, 1856$	EC_{tn}	10	10	10.0	228	1	273	-1
	DCut_{tn}	10	10	10.0	617	1	6	-1
	EC_{tnp}	10	10	10.0	105	1	257	2
	DCut_{tnp}	10	10	10.0	459	1	13	2
$300, 35880, 2, 8970$	EC_{tn}	9	6	6.7	3846	1	600	-1
	DCut_{tn}	9	8	8.9	4113	1	17	-1
	EC_{tnp}	9	9	10.0	880	1	674	14
	DCut_{tnp}	9	9	10.0	1131	1	20	12
$300, 35880, 2, 26910$	EC_{tn}	10	10	10.0	627	1	254	-1
	DCut_{tn}	10	10	10.0	2735	1	10	-1
	EC_{tnp}	10	10	10.0	259	1	262	2
	DCut_{tnp}	10	10	10.0	1212	1	18	3

Table 13: Overview of all test instances from Set-I, Set-II and Set-III and corresponding best formulations.

Set	$\|V\|$	$d/\|E\|$	$\|L\|$	a	Best Formulation
Set-I	100	0.2	50	1	EC_{sn}
			100		EC_{sn}, EC_{snh}
			125		EC_{sn}
		0.5	50		EC_{sn}, EC_{n}, EC_{snh}
			100		EC_{n}, EC_{snh}
			125		EC_{n}, EC_{snh}
		0.8	50		EC_{sn}, EC_{n}
			100		EC_{n}, EC_{snh}
			125		EC_{sn}, EC_{n}, EC_{snh}
	200	0.2	100		EC_{sn}, EC_{snh}
			200		EC_{sn}
			250		EC_{sn}, EC_{snh}
		0.5	100		EC_{sn}
			200		EC_{sn}
			250		EC_{sn}, EC_{snh}
		0.8	100		EC_{sn}
			200		EC_{sn}, EC_{snh}
			250		EC_{sn}, EC_{snh}
Set-II	1000	4000	5		EC_{*} (several variants having same performance)
			10		EC_{*} (several variants having same performance)
			20		EC_{*} (several variants having same performance)
Set-III	100	0.05	$1/4 \cdot \|E\|$		several methods having same performance
			$3/4 \cdot \|E\|$		CEF_{tn}
		0.2	$1/4 \cdot \|E\|$		$EC_{t\tilde{n}}$
			$3/4 \cdot \|E\|$		$DCut_{tnc}$
		0.5	$1/4 \cdot \|E\|$		$EC_{t\tilde{n}o}$
			$3/4 \cdot \|E\|$		EC_{tnc}
		0.05	$1/4 \cdot \|E\|$	2	EC_{*} (several variants having same performance)
			$3/4 \cdot \|E\|$		$EC_{t\tilde{n}ob}$, EC_{tnc}
		0.2	$1/4 \cdot \|E\|$		$EC_{t\tilde{n}ob}$, EC_{tnh}
			$3/4 \cdot \|E\|$		$DCut_{t\tilde{n}obc}$
		0.5	$1/4 \cdot \|E\|$		EC_{tnob}
			$3/4 \cdot \|E\|$		EC_{tnoc}
		0.05	$1/4 \cdot \|E\|$	5	several methods having $t \leq 0$
			$3/4 \cdot \|E\|$		$DCut_{t\tilde{n}c}$
		0.2	$1/4 \cdot \|E\|$		EC_{tn}
			$3/4 \cdot \|E\|$		$DCut_{tnco}$
		0.5	$1/4 \cdot \|E\|$		EC_{tn}, EC_{tnh}
			$3/4 \cdot \|E\|$		EC_{tnob}, EC_{tnh}
	10×10		30	1	EC_{*} (several variants having same performance)
			50		EC_{snob}
			80		$DCut_{s\tilde{n}ob}$ (best relaxation)
	20×20		30		EC_{sn}, EC_{n}
			50		EC_{sn}
			80		$DCut_{t\tilde{n}}$ (best relaxation)

Table 14: Running times in seconds reported in [2], rounded to integers.

l	5	10	20	5	10	20	5	10	20
n	20			50			100		
MLSTb	0	0	0	0	0	1	0	1	3
MLSTc	0	0	0	0	0	1	0	1	7
MLST-CL	0	0	0	0	0	0	0	0	1
l	5	10	20	5	10	20	5	10	20
n	200			500			1000		
MLSTb	0	3	15	1	9	136	2	43	621
MLSTc	1	6	34	4	38	371	5	132	1994
MLST-CL	0	0	6	0	0	71	0	0	360
l	5	10	20	5	10	20	5	10	20
n	20			50			-	-	-
MLSTb	0	0	0	10	9	8	-	-	-
MLSTc	0	0	0	6	9	4	-	-	-
MLST-CL	0	0	0	45	0	0	-	-	-

Table 15: Running times for instances that have been created according to specification from [2]. The first column lists the method for the corresponding row. In parenthesis the corresponding method from [2] is reported.

l	5	10	20	5	10	20	5	10	20
n	20			50			100		
avg($\|L_T\|$)	2.0	2.5	3.8	2.4	3.3	5.0	3.0	4.1	6.6
SCF (MLSTb)	0	0	0	0	0	0	0	0	19
SCF_{tn}	0	0	0	0	0	0	0	0	1
$DCut_{tn}$	0	0	0	0	0	0	0	0	1
EC_{tn}	0	0	0	0	0	0	0	0	0
EC_{sn}	0	0	0	0	0	0	0	0	0
A* (MLST-CL)	0	0	0	0	0	0	0	0	1
l	5	10	20	5	10	20	5	10	20
n	200			500			1000		
avg($\|L_T\|$)	3.0	5.0	7.9	3.5	5.9	9.9	4.1	6.6	11.3
SCF (MLSTb)	3	3	9	71	29	384	31	96	1303
SCF_{tn}	0	1	6	0	4	19	1	13	51
$DCut_{tn}$	0	0	4	1	3	21	12	13	67
EC_{tn}	0	0	0	0	0	0	0	0	0
EC_{sn}	0	0	0	0	0	0	0	0	0
A* (MLST-CL)	0	0	13	0	0	159	0	0	609
d	0.2	0.5	0.8	0.2	0.5	0.8	-	-	-
n	20			50			-	-	-
avg($\|L_T\|$)	7.1	3.5	2.2	3.0	3.9	7.6	-	-	-
SCF (MLSTb)	0	0	0	23	40	25	-	-	-
SCF_{tn}	0	0	0	3	0	1	-	-	-
$DCut_{tn}$	0	0	0	5	2	2	-	-	-
EC_{n}	0	0	0	0	0	0	-	-	-
EC_{sn}	0	0	0	0	0	0	-	-	-
A* (MLST-CL)	0	0	0	67	0	0	-	-	-

Table 16: Comparison to results reported in [11] for the A^*-algorithm. Columns $\text{MLST}^{\text{EC}_n}$ list the average total running times for each group of this particular MIP in seconds, columns A^* list the running times in seconds (rounded to integers) reported in [11], at which the best solution was found.

$\|V\|$	$\|L\|$	d	avg($\|L_T\|$)	A^*	$\text{MLST}^{\text{EC}_{\text{tn}}}$	opt	$\|V\|$	$\|L\|$	d	avg($\|L_T\|$)	A^*	$\text{MLST}^{\text{EC}_{\text{tn}}}$	opt
100	25	0.8	1.8	0	0	10	400	100	0.8	2.0	n/a	60	10
100	25	0.5	2.0	0	0	10	400	100	0.5	2.2	n/a	61	10
100	25	0.2	4.5	0	0	10	400	100	0.2	5.8 (*)	n/a	NF	8
100	50	0.8	2.0	0	0	10	400	200	0.8	3.0	n/a	817	10
100	50	0.5	3.0	0	0	10	400	200	0.5	NA	n/a	NA	NA
100	50	0.2	6.7	10	0	10	400	200	0.2	9.3 (*)	n/a	NF	0
100	100	0.8	3.0	0	2	10	400	400	0.8	-	n/a	NF	0
100	100	0.5	4.7	2	9	10	400	400	0.5	6.2 (*)	n/a	NF	0
100	100	0.2	9.7	NF	6	10	400	400	0.2	14.6 (*)	n/a	NF	0
100	125	0.8	4.0	0	17	10	400	500	0.8	-	n/a	NF	0
100	125	0.5	5.2	180	11	10	400	500	0.5	7.3 (*)	n/a	NF	0
100	125	0.2	11.0	NF	12	10	400	500	0.2	17.1 (*)	n/a	NF	0
200	50	0.8	2.0	0	3	10	500	125	0.8	2.0	0	157	10
200	50	0.5	2.2	0	2	10	500	125	0.5	2.6	0	196	10
200	50	0.2	5.2	5	10	10	500	125	0.2	6.3 (*)	NF	NF	2
200	100	0.8	2.6	0	28	10	500	250	0.8	3.0	5	2192	10
200	100	0.5	3.4	0	19	10	500	250	0.5	4.3 (*)	NF	NF	1
200	100	0.2	7.9	NF	191	10	500	250	0.2	10.3 (*)	NF	NF	0
200	200	0.8	4.0	23	911	10	500	500	0.8	4.8 (*)	NF	NF	0
200	200	0.5	-	NF	NF	9	500	500	0.5	6.9 (*)	NF	NF	0
200	200	0.2	-	NF	NF	7	500	500	0.2	16.4 (*)	NF	NF	0
200	250	0.8	4.0	21	301	10	500	625	0.8	5.1 (*)	NF	NF	0
200	250	0.5	-	NF	NF	9	500	625	0.5	8.4 (*)	NF	NF	0
200	250	0.2	-	NF	NF	3	500	625	0.2	19.0 (*)	NF	NF	0