MARC E. PFETSCH

Branch-And-Cut for the Maximum Feasible Subsystem Problem

Supported by the DFG Research Center MATHEON “Mathematics for key technologies” in Berlin

ZIB-Report 05-46 (November 2005, revised May 2008)

BRANCH-AND-CUT FOR THE MAXIMUM FEASIBLE SUBSYSTEM PROBLEM

MARC E. PFETSCH

ABSTRACT. This paper presents a branch-and-cut algorithm for the NP-hard maximum feasible subsystem problem: For a given infeasible linear inequality system, determine a feasible subsystem containing as many inequalities as possible. The complementary problem, where one has to remove as few inequalities as possible in order to make the system feasible, can be formulated as a set covering problem. The rows of this formulation correspond to irreducible infeasible subsystems, which can be exponentially many. It turns out that the main issue of a branch-and-cut algorithm for the maximum feasible subsystem problem is to efficiently find such infeasible subsystems. We present three heuristics for the corresponding NP-hard separation problem and discuss cutting planes from the literature, such as set covering cuts of Balas and Ng, Gomory cuts, and $\{0, \frac{1}{2}\}$-cuts. Furthermore, we compare a heuristic of Chinneck and a simple greedy algorithm. The main contribution of this paper is an extensive computational study on a variety of instances arising in a number of applications.

1. INTRODUCTION

In the *maximum feasible subsystem problem* (MAX FS), we are given an infeasible linear inequality system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, with $A \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$, and have to find a feasible subsystem containing as many inequalities as possible. This NP-hard combinatorial optimization problem has a number of interesting applications in a wide range of fields, for instance, in linear programming [29, 31, 36], statistical discriminant analysis and machine learning [4, 19, 43], telecommunications [54], and computational biology [61]. Additional applications and a survey can be found in [4] and [5], respectively.

The complementary problem of MAX FS amounts to removing as few inequalities of Σ as possible so that the resulting system is feasible. To achieve feasibility, one has to remove at least one inequality from each *irreducible infeasible subsystem* (IIS), i.e., an infeasible subsystem of Σ for which every proper subsystem is feasible. Introducing a binary variable y_i for each inequality of Σ, the complementary problem can be formulated as a set covering problem and is therefore called MIN IIS COVER:

$$\begin{array}{lll} \min & \sum_{i=1}^{m} y_i & \\ \text{s.t.} & \sum_{i \in I} y_i \geq 1 & \text{for all IISs } I \\ & \boldsymbol{y} \in \{0,1\}^m. & \end{array} \tag{1}$$

2000 *Mathematics Subject Classification.* 90C27.

Key words and phrases. infeasible linear inequality system, irreducible infeasible subsystem (IIS), maximum feasible subsystem problem, minimum IIS-cover, branch-and-cut.

Supported by the DFG Research Center MATHEON "Mathematics for key technologies".

Since the number of IISs can be exponential in the size of the system Σ (see Chakravarti [28] and Pfetsch [53]), IISs have to be generated dynamically in order to solve this formulation of Min IIS Cover efficiently.

Clearly, the set of all inequalities not contained in a solution of Max FS form a solution of Min IIS Cover and vice versa. Hence, these two problems are equivalent when solving to optimality and are both strongly NP-hard, see Johnson and Preparata [39], Sankaran [58], and Chakravarti [28]. In terms of approximability, however, they differ: Max FS does not admit a polynomial-time approximation scheme, unless P = NP, but there exists a 2-approximation, see Amaldi and Kann [9]. Min IIS Cover is harder to approximate: Unless P = NP, it cannot be approximated in polynomial time within any constant factor, see Amaldi and Kann [10].

In this paper, we present a branch-and-cut approach for Max FS via formulation (1) for Min IIS Cover. A key issue of this approach is to find violated *IIS-inequalities*, i.e., the inequalities arising from IISs in (1). The corresponding separation problem is NP-hard, and we present three heuristics for it (see Section 3.2). Two of these methods generate either a feasible solution for Min IIS Cover or a (hopefully violated) IIS-inequality. As long as no feasible solution has been generated, the process is iterated, which often produces many useful IIS-inequalities. The additional benefit is reasonably good primal solutions, which can be improved by a simple greedy algorithm. This combination leads to an effective primal heuristic. Additionally, we examine the application of inequalities of Balas and Ng [18] for set covering problems, $\{0, \frac{1}{2}\}$-cuts, and Gomory cuts.

The emphasis of this paper is on an extensive computational study of the branch-and-cut implementation. Our aim is to show the potential and the limits of such an approach by performing tests on three problem sets: random infeasible inequality systems (Section 4.2), problems arising in digital video broadcasting (Section 4.3), and classification problems (Section 4.4).

The theoretical foundation for our approach appears in Amaldi, Pfetsch, and Trotter [12], where algorithmic and geometric questions concerning IISs are studied and the feasible subsystem polytope is investigated. (The polyhedral results carry over to the polytope for Min IIS Cover by a simple affine transformation.) The work presented here is an improved version of part of the author's Ph.D. thesis [53].

In the literature to date, only two exact approaches towards Min IIS Cover have appeared. Parker and Ryan [52] discuss an iterative approach that generates IISs in each step and then solves an integer program. This approach turns out to be impractical for harder instances. Codato and Fischetti [33] present a branch-and-cut algorithm for Min IIS Cover in a more general context. We discuss these approaches in more detail in the next section. Our algorithm improves upon both methods and is currently the best available exact approach (see Section 4).

The outline of this paper is as follows. In Section 2 we review solution approaches for Max FS. In Section 3 we describe the main ingredients of our branch-and-cut implementation. We discuss a way to check the feasibility of solutions for Min IIS Cover, three methods to separate IIS-inequalities, primal heuristics, preprocessing, branching, inequalities by Balas and Ng, and other used cutting planes. In Section 4 we extensively test the implementation on the abovementioned problem sets. We close with some conclusions in Section 5.

We use the following notation. We define $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$ and typeset vectors in bold font. For a set $S \subseteq [n]$ and a vector $\boldsymbol{x} \in \mathbb{R}^n$, define

$$\boldsymbol{x}(S) = \sum_{i \in S} x_i \,.$$

The *support* of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is $\operatorname{supp}(\boldsymbol{x}) := \{i \in [n] \,:\, x_i \neq 0\}$. By $\mathbb{1}$ we denote a vector of all 1's of appropriate dimension.

2. Alternative Solution Approaches

In this section we give a short overview of solution approaches for Max FS and Min IIS Cover.

In the context of linear programming, attention was first devoted to the problem of identifying IISs with a small and possibly minimum number of inequalities (see Greenberg and Murphy [36], Chinneck [30], and Chinneck and Dravnieks [32]). The goal is to help the modeler resolve infeasibility of large linear programs. Since minimum cardinality covers of IISs reveal essential information about infeasibility of the model and are often smaller than IISs, emphasis has shifted towards their identification. Chinneck [29, 31] developed heuristics for Max FS/Min IIS Cover and provided computational results, see Section 4.4. These heuristics are extended greedy algorithms.

For the application of Min IIS Cover to classification problems (see Section 4.4), several heuristics were proposed, based on nonlinear programming formulations of Max FS (Bennett and Bredensteiner [19], Bennett and Mangasarian [20], and Mangasarian [43]).

An exact integer programming approach for Min IIS Cover appeared in Parker [51] and Parker and Ryan [52]. Their idea is to consider the formulation in (1) with a partial list of IISs. If there exist IISs that are not covered by a solution to this formulation, they are added and the process is iterated. Otherwise, an optimal solution to Min IIS Cover is found. Parker and Ryan discuss several methods of generating IISs at each step and consider heuristics for solving the set covering problem (only the last instance has to be solved exactly).

We reimplemented a basic version of their algorithm, where the set covering problems are solved to optimality. This implementation turned out to be inferior to our branch-and-cut implementation: It could not solve within one hour instances solved by our branch-and-cut approach within a few minutes. We therefore refrained from performing further experiments.

There is a straightforward mixed integer programming formulation for Min IIS Cover containing a binary variable with a “big-M” for each of the inequalities of Σ, so that an inequality is relaxed when the corresponding binary variable is 1. This formulation has the typical numerical problems of big-M formulations and is in general inefficient for Max FS, see Parker [51]. If there are fixed bounds on the variables, however, one can obtain a tight formulation. This leads to a quite efficient approach, see Rossi, Sassano, and Smriglio [54] and Codato and Fischetti [33]. In fact, Codato and Fischetti propose a general way of removing the "big-M" from this type of formulation and apply it to classification instances. In this context, it leads to the formulation (1), and their solution method is, in fact, a branch-and-cut method for Min IIS Cover, independent from our approach. Computational results show that their approach is faster compared to the big-M formulation. In Section 4.4 we compare our implementation with their approach.

Versions of the classical *relaxation method* of Agmon [3] and Motzkin and Schoenberg [47] for solving linear inequality systems can be applied to minimize the sum of violations in infeasible linear inequality systems. Randomized variants of this method were proposed by Amaldi [4] to solve Max FS. Amaldi and Hauser [8] and Amaldi, Belotti, and Hauser [6] establish probabilistic convergence guarantees to an optimal solution of Max FS under appropriate conditions. Computational results for digital video broadcasting data, classification instances, and huge systems arising in computational biology are given in [6].

Amaldi, Bruglieri, and Casale [7] propose a two-step heuristic in which first a linearization of an exact bilinear formulation of Max FS is used to derive a feasible subsystem. In the second step, a reduced problem is solved to optimality in order to identify inequalities that can be added to the first system while preserving feasibility. This turns out to be competitive with respect to the method of Codato and Fischetti and an integer programming solver applied to the "big-M" formulation for the whole system.

3. Ingredients for Branch-and-Cut

In the following we assume that the reader is familiar with the branch-and-cut approach. More information can be found in Nemhauser and Wolsey [48], Padberg and Rinaldi [50], Thienel [60], and Caprara and Fischetti [26]. A description and computational study of Gomory cuts is given in Balas, Ceria, Cornuéjols, and Natraj [17].

Recall that we are given the infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, where $A \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m$. Depending on the application, *mandatory* variable bounds can be present, i.e., these bounds may not be removed for obtaining a feasible system (see Sections 4.3 and 4.4). This can easily be dealt with in the branch-and-cut approach. Furthermore, weighted versions of Min IIS Cover are easy to handle, too.

Without loss of generality we can restrict attention to inequality systems in the form of Σ: Clearly, bounds on variables and "greater or equal" inequalities can be transformed to this format. Equations can be replaced by a pair of opposing inequalities. Since any point satisfies at least one inequality out of each pair, an optimal solution to the new instance contains $m^* + m_E$ inequalities if and only if an optimal solution to the original instance with m^* linear relations exists; here m_E is the number of equations. Thus, from a computational point of view, it suffices to handle systems in the form of Σ. Polyhedral results for the two cases, however, may differ, see [12, 53] for more information.

To simplify notation, we identify an inequality of Σ with its index. Then $S(\Sigma) := [m]$ is the set of constraints of Σ. With this notation, $I \subseteq S(\Sigma)$ is an IIS of Σ if and only if all proper subsets of I are feasible. We call a set $C \subseteq S(\Sigma)$ an *IIS-cover* if it intersects every IIS of Σ.

In the rest of this section we give a more detailed account of the main aspects of our implementation: the recognition problem for IIS-covers, the separation problem of IIS-inequalities, pool handling, primal heuristics, preprocessing, branching, and other cutting planes.

3.1. **Recognition Problem for IIS-Covers.** We consider the following fundamental problem: Given a subset $C \subseteq S(\Sigma)$, check whether it is an IIS-cover and if this is not the case generate a witness, i.e., an IIS which is not covered. Our approach is based on the following theorem.

Theorem 1 (Gleeson and Ryan [35])**.** *Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system. Then the IISs of Σ are in one-to-one correspondence with the supports of the vertices of the polyhedron*

$$P(\Sigma) := \{\, \boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}^{\mathrm{T}} A = \boldsymbol{0},\ \boldsymbol{y}^{\mathrm{T}} \boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0} \,\}.$$

Note that the vertices of $P(\Sigma)$ are uniquely defined by their supports. This theorem is strongly related to the Farkas lemma, which states that $P(\Sigma) \neq \varnothing$ if and only if Σ is infeasible, see, e.g., Schrijver [59]. The polyhedron $P(\Sigma)$ is called the *alternative polyhedron* of Σ.

To apply Theorem 1, we define for $S \subseteq S(\Sigma)$ the polyhedron

$$P_S(\Sigma) := \{\, \boldsymbol{y} \in P(\Sigma) \,:\, y_i = 0,\ i \in S \,\},$$

which might be empty. We need the following fact.

Lemma 2 (Parker and Ryan [52])**.** *The set $C \subseteq S(\Sigma)$ is an IIS-cover if and only if $P_C(\Sigma) = \varnothing$.*

Proof. The system defining $P(\Sigma)$, in which all variables indexed by C are removed, has no solution if and only if $P_C(\Sigma) = \varnothing$. By the Farkas lemma, the former is the case if and only if Σ with inequalities indexed by C removed is feasible, i.e., C is an IIS-cover. $\square$

Recognizing whether $C \subseteq S(\Sigma)$ is an IIS-cover is now easy: If $P_C(\Sigma) = \varnothing$, by Lemma 2, C is an IIS-cover. Otherwise, let $\boldsymbol{v}$ be a vertex of $P_C(\Sigma)$. Then $\operatorname{supp}(\boldsymbol{v}) \cap C = \varnothing$, which shows that $\operatorname{supp}(\boldsymbol{v})$ is an IIS that is uncovered (by Theorem 1). This provides a polynomial-time algorithm for the problem, since finding a vertex of a polyhedron can be done in polynomial time, see Grötschel, Lovász, and Schrijver [37]. Note that by Theorem 1 and Lemma 2, $P_C(\Sigma)$ always has a vertex if it is nonempty.

This recognition test, in fact, suffices for a rudimentary branch-and-cut algorithm, since we can now test feasibility of a vector $\boldsymbol{y} \in \{0,1\}^m$ for (1) by testing whether $\operatorname{supp}(\boldsymbol{y})$ is an IIS-cover.

3.2. **Separation of IIS-Inequalities.** IIS-inequalities play a prominent role in the formulation (1) for Min IIS Cover. In fact, it can be shown that the inequality arising from the IIS I defines a facet of the polytope

$$P_{IISC} = \operatorname{conv}\{\, \boldsymbol{y} \in \{0,1\}^m \,:\, y(S) \geq 1 \text{ for all IISs } S \,\},$$

as long as $|I| > 1$, see Amaldi, Pfetsch, and Trotter [12]. Therefore, the following *separation problem for IIS-inequalities* is of crucial importance:

Given a vector $\boldsymbol{y}^* \in [0,1]^m$, check whether there exists an IIS I so that its corresponding inequality is violated by $\boldsymbol{y}^*$, i.e., $\boldsymbol{y}^*(I) < 1$. The recognition problem for IIS-covers is a special case, where $\boldsymbol{y}^*$ is the incidence vector of the set to be tested. In the general case, however, we have the following.

Proposition 3 (Amaldi, Pfetsch, and Trotter [12])**.** *The separation problem for IIS-inequalities is* NP*-hard.*

In this section, we therefore present three heuristics for the separation problem. All of these heuristics may fail to produce a violated IIS-inequality.

The heuristics build on the following reformulation of the separation problem: Compute

$$\lambda := \min\{\, \boldsymbol{y}^*(S) \,:\, S = \operatorname{supp}(\boldsymbol{v}),\ \boldsymbol{v} \text{ vertex of } P(\Sigma)\,\}. \qquad (2)$$

If $\lambda < 1$, by Theorem 1, $\operatorname{supp}(\boldsymbol{v})$ provides an IIS whose IIS-inequality is violated; otherwise no such IIS exists (we define $\lambda = \infty$ if $P(\Sigma) = \varnothing$).

3.2.1. *Method 1: "Single".* The first quite intuitive idea for separating an IIS-inequality, already used by Parker and Ryan [52], is to approximate (2) by the following linear program (LP):

$$\min\{\, (\boldsymbol{y}^*)^{\mathrm{T}}\boldsymbol{p} \,:\, \boldsymbol{p} \in P(\Sigma)\,\}.$$

A vertex solution provides an IIS, whose corresponding inequality is not necessarily violated, but in practice it often is.

This method generates only one IIS at a time. We also experimented with solving the above LP by the simplex algorithm and then testing whether the support of each vertex on the path to the optimum is an IIS whose inequality is violated. In our experiments this variant was inefficient and will not be considered further.

3.2.2. *Method 2: "Extend".* We extend Method 1 as follows. Let S be the support of $\boldsymbol{y}^*$. Applying Lemma 2, we can check whether S is an IIS-cover by finding a vertex solution of

$$\min\{\, (\boldsymbol{y}^*)^{\mathrm{T}}\boldsymbol{p} \,:\, \boldsymbol{p} \in P_S(\Sigma)\,\}$$

if one exists. If the LP is feasible, the result gives us a vertex which corresponds to an IIS, otherwise we have found an IIS-cover, i.e., a primal solution for Min IIS Cover.

This approach can be iterated when S is not an IIS-cover. Let I be the IIS obtained in this case. We enlarge S greedily by an element of I and iterate. In our implementation, we choose an element of I that is contained in the maximal number of IISs we have found so far. At termination this yields an IIS-cover. This procedure is related to a primal heuristic proposed by Ryan [57].

The IISs found by this approach have several nice properties. First, the new IISs are different from all IISs that were known before the run, if the current solution $\boldsymbol{y}^*$ of the LP-relaxation satisfies $y^*(I) \geq 1$ for each previously found IIS I. This follows since at least one element of each I is contained in S, and hence I cannot be generated again. Second, the corresponding inequalities are always violated, since they have empty intersection with $S \supseteq \operatorname{supp}(\boldsymbol{y}^*)$, i.e., $\boldsymbol{y}^*(I) = 0 < 1$ for each produced IIS I. Third, by construction of the set S, the generated IISs are pairwise different.

This method turns out to be quite effective for generating many violated IIS-inequalities. Furthermore, we obtain a primal solution in each run, which can be improved to very good solutions, see Section 3.4. When the current LP-relaxation contains many cuts, however, the support of $\boldsymbol{y}^*$ tends to be large and often is already an IIS-cover or close to one, and the method cannot produce new IISs; this often happens in the deeper regions of the branch-and-bound tree. This might even be desirable, since this saves time for high depths. Nevertheless, this situation can be changed, as indicated by the next method.

3.2.3. *Method 3: "Round".* The idea of Method 2 can be further extended by using the fact that an arbitrary set S can be used at the start. In the extension, we choose $\alpha \in [0, 1]$ and initially let $S := \{i : y_i^* \geq \alpha\}$. In the implementation we start with $\alpha = 0.1$ and then increase α by 0.1 until S is not an IIS-cover (in this case the above procedure is started). We terminate with a failure if α exceeds 0.6.

The fact that S is smaller for larger α has two effects: First, the number of steps needed to greedily obtain an IIS-cover is larger, and hence the number of generated IISs is increased. Second, the method also computes IISs in the deeper regions of the tree.

Again, in each step an IIS is generated, which is not covered by S, except in the last step where we obtain an IIS-cover. In contrast to the method "extend", the generated IISs are not necessarily new, and their corresponding inequalities may not be violated by $\boldsymbol{y}^*$.

3.3. Pool for IIS-Inequalities.

The above three methods tend to produce many IISs, which we store in a pool. It turns out that the best performance of the algorithm is achieved by checking the pool for violated inequalities in *every* node of the tree. Of course, the pool should be as small as possible without losing important inequalities. Therefore, the pool is equipped with an aging mechanism which removes IISs whose inequality has not been active for some time.

The computational results presented in Section 4 indicate that only a small fraction of the total number of IISs needs to be generated by our branch-and-cut implementation; indeed, for larger problems there are far too many IISs to be enumerated completely, cf. Table 2 in Section 4.2. Hence, the size of the pool can be relatively small.

3.4. Primal Heuristics.

Chinneck [31] proposed a greedy heuristic for Min IIS Cover, which we use as an initial primal heuristic. The basic tool is a so-called *elastic LP* in which the inequalities $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ are relaxed by adding slack variables and the sum of violations is minimized:

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{s} \\ & A\boldsymbol{x} - \boldsymbol{s} \leq \boldsymbol{b} \\ & \boldsymbol{s} \geq 0. \end{aligned}$$

Starting with $S = \varnothing$, in each iteration $S \subseteq S(\Sigma)$ is enlarged by an inequality that yields the largest drop in the elastic LP objective if its objective coefficient is set to 0. The method stops once the objective is 0, i.e., S is a Min IIS Cover. To speed up the solution, in each iteration only inequalities from a candidate set are checked. Chinneck proposes a measure based on the violation and dual variables to generate the candidate set. We refer to [31] for details.

For a heuristic running in the tree, we use a primal heuristic that greedily decreases the size of a given IIS-cover until a minimal one is obtained. We start this heuristic from IIS-covers produced by the separation methods in Section 3.2, if available (otherwise we use a simple rounding heuristic). We start with C being an IIS-cover to be improved. We consider each element from C in the order of increasing fractional value of the current LP-solution $\boldsymbol{y}^*$. We remove an element if the remaining set is an IIS-cover (which is checked by the method in Section 3.1).

3.5. **Preprocessing.** In a preprocessing step we search for small IISs. Such small IISs are of interest since their corresponding IIS-inequalities provide “strong” cuts and are hard to find by other methods.

We first check for IISs of cardinality one, e.g., $\mathbf{0}\boldsymbol{x} \leq -1$. Then we check for IISs that involve one inequality and bounds on the variables (if present). Such IISs often occur when variable bounds are mandatory, see e.g. Section 4.4. In this case, a single inequality might be infeasible with the bounds and counts as an IIS. Furthermore, we look for IISs of cardinality two, which are easy to find by comparing their normal vectors and right-hand sides. Identifying other types of IISs would require higher computational effort.

3.6. **Branching.** As a branching rule, we apply *reliability branching*, introduced by Achterberg, Koch, and Martin [2]. It performs strong branching on a subset of the variables, which are chosen based on their so-called pseudocosts during branching. If in strong branching one of the child nodes turns out to be infeasible, the corresponding variable is fixed to the complementary value; if both children are infeasible, the current node can be pruned.

We also experimented with constraint branching rules. For instance, we used the well-known rule of Ryan and Foster [56]. This rule was superior to a simple variable branching, but inferior to reliability branching both in terms of computation time and the number of branch-and-bound nodes. We therefore selected reliability branching for all tests.

3.7. **Inequalities for Set Covering.** Many facet-defining inequalities for the set covering polytope have been investigated, see Ceria, Nobili, and Sassano [27] and Borndörfer [22]. However, few (problem-specific) polynomial-time separable inequalities for set covering are known. For many classes of inequalities the complexity status is unknown, but is likely to be NP-hard.

We experimented with the aggregated cycle cuts of Borndörfer and Weismantel [23, 24]. Unfortunately, on our test problems their separation heuristic almost never found a violated inequality. Furthermore, it remains an interesting open problem to identify problem-specific inequalities for Min IIS Cover.

A class of inequalities for set covering that we use in our implementation were proposed by Balas and Ng [18]. To describe these inequalities, consider the set covering polytope $P_{SC}(D) = \operatorname{conv}\{\boldsymbol{y} \in \{0,1\}^m : D\boldsymbol{y} \geq \mathbb{1}\}$, where $D = (d_{ij}) \in \{0,1\}^{k\times m}$. Assume $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y} \geq \beta$, with $\boldsymbol{a} \in \mathbb{Z}^m$ and $\beta \in \mathbb{Z}$, defines a facet of $P_{SC}(D)$. It is well known that if $\beta > 0$, then $\boldsymbol{a} \geq \boldsymbol{0}$, and if $\beta = 1$, then $\boldsymbol{a}$ is a row of D (see, e.g., [18]).

Balas and Ng showed that for every facet defining inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{y} \geq 2$ with $\boldsymbol{a} \in \mathbb{Z}^n$, there exists a set $S \subseteq [k]$ such that $\boldsymbol{a} = \boldsymbol{a}^S$, where

$$a_j^S = \begin{cases} 0 & \text{if } d_{ij} = 0 \text{ for all } i \in S, \\ 2 & \text{if } d_{ij} = 1 \text{ for all } i \in S, \\ 1 & \text{otherwise} \end{cases} \qquad \text{for } j = 1, \dots, m.$$

These inequalities can also be obtained by a Chvátal-Gomory rounding procedure. Furthermore, Balas and Ng discuss conditions under which $(\boldsymbol{a}^S)^{\mathrm{T}}\boldsymbol{y} \geq 2$ defines a facet of $P_{SC}(D)$.

The separation problem for these inequalities is NP-hard, see Amaldi and Pfetsch [11]. However, when the size of S is fixed, the separation problem can be solved in polynomial time by enumeration. In our implementation we enumerate sets S of *cardinality three* and check whether the inequalities $(\boldsymbol{a}^S)^{\mathrm{T}}\boldsymbol{y} \geq 2$ are violated by the current LP-solution. Note that sets S of cardinality two are uninteresting, since in this case $(\boldsymbol{a}^S)^{\mathrm{T}}\boldsymbol{y} \geq 2$ is the sum of two IIS-inequalities and hence is never violated if the IIS-inequalities are satisfied.

Additionally, we try to strengthen these cuts: If an inequality is violated, we greedily enlarge the set S as long as the violation of the resulting inequality increases. See Section 4 for computational results.

3.8. **General Purpose Inequalities.** In our computational experiments we used Gomory cuts as implemented in SCIP (see Section 4); see the books of Nemhauser and Wolsey [48] or Schrijver [59] for a description.

We furthermore used $\{0, \frac{1}{2}\}$-cuts introduced by Caprara and Fischetti [25]. Codato and Fischetti [33] identified these cuts as important for solving MIN IIS COVER. We implemented these cuts along the lines of Hansen, Labbé, and Schindl [38]. See also Andreello, Caprara, and Fischetti [13] for a computational study of $\{0, \frac{1}{2}\}$-cuts. Note that in our implementation $\{0, \frac{1}{2}\}$-cuts are produced only for set covering and nonnegativity inequalities; in particular, they do not depend on $\{0, \frac{1}{2}\}$-cuts produced earlier.

We also experimented with mixed integer rounding cuts (CMIR) (see Marchand and Wolsey [44]) and strengthened Chvátal-Gomory cuts (see Letchford and Lodi [42]) as they are implemented in SCIP. The results were, however, discouraging, and we therefore do not present them.

4. Computational Results

In this section we discuss computational results of our branch-and-cut implementation for MIN IIS COVER. The algorithm was implemented in `C++` and uses version 0.90 of the framework SCIP (Solving Constraint Integer Programs) by Achterberg [1]. CPLEX 10.11 is used as the basic LP solver. The computations were performed on a 3.4 GHz Pentium 4 machine with 3 GB of main memory and 1 MB cache running Linux. All instances used in the following can be obtained from the web page [45].

We use best-first search as a node selection scheme and the branching rule explained in Section 3.6. All separation routines are called only every tenth level of the tree, except that the pool of IIS-inequalities is checked in every node of the tree. In nodes in which cuts are separated, we proceed until no more violated cuts can be found. SCIP chooses among the generated cuts according to an orthogonality measure, see, for instance, Andreello, Caprara, and Fischetti [13]. We perform reduced cost fixing at every node of the tree.

Before presenting computational results, we want to discuss the influence of the limited precision used for solving LPs. The basic question that has to be repeatedly answered in our context is whether a given system is infeasible or not. Today's LP solvers are tuned towards quickly finding an optimal solution of a feasible LP. Sometimes their bases are not really optimal, but this has only a negligible effect on the objective function value, see Koch [41]. When checking infeasibility, however, small errors can lead to completely wrong decisions. The answer depends on the particular instance, the solution method of the LP solver, its parameters, e.g., the precision (usually around 10^{-6}), and often also the preprocessing and starting basis. Being aware of the possibility that we might produce wrong results, as a safeguard, we confirmed that the final solution is really an IIS-cover for the original system.

Currently, using exact LP solvers, like the ones included in `lrs` [15] or `cdd` [34] is computationally too expensive. In the future, codes that use dynamically adjusted precision might help, see Applegate, Cook, Dash, and Espinoza [14].

4.1. **The Netlib Problems.**

The Netlib library [49] contains a well-known set of 29 infeasible linear inequality systems. We do not report results on these data since these instances all can be solved within seconds, except for numerical difficulties with the problem `gran`. They were also solved to optimality by Parker [51] and Parker and Ryan [52]; for more computational results on these problems see Chinneck [31] and Pfetsch [53].

4.2. **Random Problems.**

We consider random inequality systems to compare different cut strategies in the branch-and-cut implementation. We used difficult random instances that nevertheless can be solved within approximately one hour of computation time. In contrast, the instances discussed in the following sections vary highly in size and complexity: Most are either solved within seconds or cannot be solved to optimality in reasonable time.

The infeasible random inequality systems are generated as follows: Each coefficient and the right-hand side was chosen to be a random integer in the range -100 to 100. We generated five instances for each of the combinations $(5, 100)$, $(10, 80)$, $(15, 80)$, $(20, 90)$, $(25, 90)$, where the first component is the dimension n of the space and the second one is the number m of inequalities. Each system turned out to be infeasible (this almost always happens as soon as $m > 2 \cdot n$, see Motzkin [46]) and is almost completely dense. Note that all the instances in the following sections are dense as well.

In Theorem 1, the alternative polyhedra of these random systems are nondegenerate with high probability. It is currently unknown whether Max FS and Min IIS Cover restricted to such systems are NP-hard.

Table 1: Results of the branch-and-cut algorithm on *random inequality systems* for different IIS separation strategies. The numbers are averages over five instances of each size. The last line gives the averages over each column.

		single			extend			round		
n	m	nodes	time	IISs	nodes	time	IISs	nodes	time	IISs
5	100	70473.0	1050.64	8781.0	120371.4	1808.71	5281.4	16913.8	564.44	11034.8
10	80	167970.8	1226.45	10298.4	174302.6	1689.26	8450.4	79086.6	996.51	14491.8
15	80	214004.0	1509.72	53419.8	255933.0	1984.60	44825.8	106119.0	1465.16	62151.0
20	90	50029.0	276.05	22354.8	59117.8	337.11	15869.0	28699.0	317.22	23418.0
25	90	169868.2	1185.81	99728.6	243568.6	1534.17	80400.4	77147.0	1235.41	155331.4
	∅:	134469.0	1049.73	38916.5	170658.7	1470.77	30965.4	61593.1	915.75	53285.4

Table 2: The number of IISs found by method “round” for random problems and the total number of IISs.

n	m	found	total
5	30	11	1986
5	40	101	44816
5	50	520	204833
5	60	526	614853
5	70	453	1818718

We first compare the three different strategies to separate IIS-inequalities of Section 3.2. Table 1 provides a comparison of methods “single” (Section 3.2.1), “extend” (Section 3.2.2), and “round” (Section 3.2.3). Columns labeled “nodes” give the average number of nodes in the branch-and-bound tree, those labeled “time” are the average CPU times in seconds, and those labeled “IISs” give the average number of IISs found during the optimization; here averages are taken over the five instances of each size. To eliminate the influence of primal heuristics we initialized all runs with the optimal solution.

Among the three IIS-inequality separation versions, method “round” outperforms methods “single” and “extend” in the number of nodes and in the total computation time, although method “single” is sometimes a bit faster. Method “round” also generates the highest number of IISs. Based on this result, we decided to use method “round” in the following experiments.

Table 2 shows the total number of IISs and the number of IISs found by method “round” for small random instances generated in the same manner as above. By Theorem 1, the IISs correspond to vertices of the alternative polyhedron. We enumerated the vertices with `lrs` [15]. Since the alternative polyhedra are nondegenerate, the IISs can be generated in time polynomial in the input and output size, see Avis and Fukuda [16]. Note that for general polyhedra this is not possible unless P = NP, see Khachiyan et al. [40].

We could not enumerate or count the IISs for larger instances. From Table 2, however, it can be expected that the total number of IISs for the instances used in Table 1 is much higher. We conclude that the branch-and-cut implementation needs only a small part of the total set of IISs (the number of IISs for instance $(5, 70)$ is two orders of magnitudes larger than the average number of IISs found by any of the variants in Table 1).

Table 3: Results of the branch-and-cut algorithm on *random inequality systems* for different cut generation strategies; all variants use method "round" as a basis. Given are the average values over all 25 instances.

type	nodes	time	root	# BaNg	# Gom.	$\#\{0,\frac{1}{2}\}$
round	61593.1	915.75	6.54	0.0	0.0	0.0
BaNg	58796.4	1054.39	6.80	6134.0	0.0	0.0
Gom.	58434.7	1164.56	7.00	0.0	10440.6	0.0
$\{0,\frac{1}{2}\}$	61479.1	957.37	6.54	0.0	0.0	43.0
BaNg & Gom.	57911.9	1298.49	7.22	6955.3	10234.8	0.0
BaNg & $\{0,\frac{1}{2}\}$	60197.0	1080.89	6.78	5738.6	0.0	31.0
Gom. and $\{0,\frac{1}{2}\}$	58852.8	1158.42	7.01	0.0	10441.2	56.8
all	60092.7	1365.63	7.19	6699.5	10335.6	46.2

Table 4: Results of method "round" for random instances with $m = 80$ inequalities. Column "Opt" gives the average optimal solution values. All entries are averages over five instances.

n	nodes	time	IISs	root	opt
5	2029.4	32.26	3527.8	12.23	21.8
10	79086.6	996.51	14491.8	6.88	15.8
15	106119.0	1465.16	62151.0	4.56	11.8
20	7408.0	56.18	5743.4	2.69	5.8
25	16472.6	132.79	20884.0	2.43	6.8
∅:	42223.1	536.58	21359.6	5.76	12.4

Table 3 lists computational results for all combinations of method "round" with Balas/Ng cuts (BaNg), Gomory cuts (Gom.), and $\{0,\frac{1}{2}\}$-cuts. The values are averages over all 25 instances. Column "root" gives the dual bound after the root node. The last three columns list the number of cuts found for the respective methods. Again, we initialize the algorithms with the optimal solution. All cuts are separated every ten levels of the tree.

The studied combinations on average reduce the number of nodes with respect to the method "round" alone; the best combination in this respect are Balas/Ng and Gomory cuts. Furthermore, all combinations, except $\{0,\frac{1}{2}\}$-cuts, improve the root dual bound with respect to the basic version. The studied methods, however, increase the CPU time needed. The main slowdown comes from the fact that the intermediate LPs become harder to solve. The corresponding separation times are acceptable, however: The average separation times for the version that uses all three methods are: 1.8% (BaNg), 17.0% (Gomory), 1.0% ($\{0,\frac{1}{2}\}$). We conclude that the basic version "round" alone is fastest on random systems.

Table 4 shows average results for method "round" on random instances with $m = 80$ inequalities. It can be observed that the optimal values of the random problems tend to decrease when increasing the dimension. This often makes the problems more tractable. But of course, the solution of the intermediate LPs over the alternative polyhedron is more time consuming.

4.3. **Digital Video Broadcasting Problems.** In this section we present results for problems arising in an application of Max FS in telecommunications, which is described by Rossi, Sassano, and Smriglio [54]. Here, to plan

Table 5: Results for the DVB instances in Section 4.3 with method "round". The column labeled "[6]" lists the names of the instances as used in Amaldi, Belotti, and Hauser [6].

name	[6]	m	nodes	time	IISs	root	dual	best	gap
dvb1	dvb2	1044	503	103.6	3064	166.4	174.0	174	0.0
mfs_UHF_P4_1	dvb1	642	1	2.3	86	104.0	104.0	104	0.0
mfs_UHF_P4_3	dvb3	1717	539	599.72	5414	174.2	183.0	183	0.0
mfs_UHF_P4_4	–	1174	68049	196514.41	1002912	90.3	115.2	124	7.6

the digital video broadcasting (DVB) network of Italy, transmitters have to be placed and their emission frequency and power have to be chosen as to maximize the area coverage, subject to quality constraints. A subproblem of this can be modeled as a linear inequality system. Interference of the signals leads to areas where the digital signal cannot be received, resulting in an infeasible system. Maximizing the total weight of satisfied inequalities then amounts to maximizing the area coverage.

Linearizing the model leads to numerically challenging problems. The coefficients take values between 10^{-11} and 10^{11}, and the resulting LPs are very instable. We tackled the problems by scaling the original instances before starting the branch-and-cut algorithm. This helps but nevertheless leaves hard problems. Without scaling, however, the algorithm terminated early with a completely wrong solution.

We could compute optimal solutions for the smallest instances used in Amaldi, Belotti, and Hauser [6] and Amaldi, Bruglieri, and Casale [7], see Table 5. Here, column "dual" gives the final lower bound, "best" denotes the value of the best primal solution obtained (i.e., the primal bound), and "gap" is the gap between the dual bound and primal bound in percent, i.e., (best − dual)/dual · 100.0. The dimension of these instances is always 487, and the variable bounds ($0 \leq \boldsymbol{x} \leq 1$) are mandatory. We separate $\{0, \frac{1}{2}\}$-cuts every 10th level of the tree. Our primal heuristic of Section 3.4 is run every 40th level. Note that these instances can be solved faster using the "big-M" formulation (resulting in the same optimal solution values), see [6, 7].

4.4. Classification Problems.

One of the historically first applications of Min IIS Cover is the design of linear classifiers, see Amaldi [4], Mangasarian [43], Bennett and Bredensteiner [19], and Rubin [55].

In this application, one is given m points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_m$ in $\mathbb{R}^N$, each belonging to one of two possible *classes* P_1 and P_2, i.e., P_1 and P_2 partition the set $\{\boldsymbol{p}_1, \dots, \boldsymbol{p}_m\}$. Each of the N components of the points stores a measurement of an attribute (or feature) relevant for the concrete application. The goal is to strictly separate these points in $\mathbb{R}^N$ by an oriented hyperplane defined by $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq \beta$, with $\boldsymbol{a} \in \mathbb{R}^N$ and $\beta \in \mathbb{R}$. The points in P_1 should satisfy the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} < \beta$, and the points in P_2 should satisfy $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} > \beta$. Hence, we are looking for $(\boldsymbol{a}, \beta) \in \mathbb{R}^n$, with $n := N + 1$, so that the number of misclassified points

$$|\{\boldsymbol{p} \in P_1 \,:\, \boldsymbol{a}^{\mathrm{T}}\boldsymbol{p} \geq \beta\}| + |\{\boldsymbol{p} \in P_2 \,:\, \boldsymbol{a}^{\mathrm{T}}\boldsymbol{p} \leq \beta\}|$$

is minimized. This minimization is performed in order to maximize the chance that a new point can be correctly classified. Note that with this formulation points in $\{\boldsymbol{x} : \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = \beta\}$ are counted twice (the models can be modified to eliminate this).

In the following we will discuss two equivalent ways to model this problem via Min IIS Cover and present computational results for different data sets. In the first model no bounds on the variables are present, while in the second all variables are bounded except one.

For the first model we use variables $(\boldsymbol{a}, \beta) \in \mathbb{R}^n$ and the following inequalities:

$$\boldsymbol{p}^{\mathrm{T}}\boldsymbol{a} - \beta \begin{cases} < 0 & \text{if } \boldsymbol{p} \in P_1 \\ > 0 & \text{if } \boldsymbol{p} \in P_2 \end{cases} \qquad \text{for each } \boldsymbol{p} \in \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_m\}.$$

Since $(\boldsymbol{a}, \beta)$ are unbounded we can scale them to obtain

$$\boldsymbol{p}^{\mathrm{T}}\boldsymbol{a} - \beta \begin{cases} \leq -1 & \text{if } \boldsymbol{p} \in P_1 \\ \geq 1 & \text{if } \boldsymbol{p} \in P_2 \end{cases} \qquad \text{for each } \boldsymbol{p} \in \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_m\}.$$

Of course, any other positive value instead of 1 can be taken in order to obtain a numerically more stable system.

The second model is due to Rubin [55]. It uses variables $\boldsymbol{a} \in \mathbb{R}^N$ and β, $\gamma \in \mathbb{R}$ in the following system:

$$\begin{aligned} \boldsymbol{p}^{\mathrm{T}}\boldsymbol{a} - \beta + \gamma \leq 0 &\quad \text{if } \boldsymbol{p} \in P_1 \\ \boldsymbol{p}^{\mathrm{T}}\boldsymbol{a} - \beta - \gamma \geq 0 &\quad \text{if } \boldsymbol{p} \in P_2 \\ -\mathbb{1} \leq \boldsymbol{a} \leq \mathbb{1}& \\ \gamma \geq 0.001.& \end{aligned}$$

Hence, the coefficients of the normal vector $\boldsymbol{a}$ are bounded to lie within the interval $[-1, 1]$, while β is unbounded. Of course, the lower bound 0.001 for γ can be replaced by any suitably small positive number. For instances arising from this model the variable bounds are mandatory.

Note that in both models it might happen that the systems are feasible, i.e., the points are completely separable (in which case we need only solve one linear program).

In our first test we use the first model and classification data from the UCI Repository of Machine Learning Databases (Blake and Merz [21]). The problem characteristics are given in Table 6. For some instances we had to remove incomplete data sets. A complete description of the instances is available at the UCI Repository. Most of these twelve instances are also used by Chinneck [31] for testing his heuristic for Max FS/Min IIS Cover.

Table 7 lists the results of the branch-and-cut implementation on these instances with method "round" of Section 3.2.3. The computation time was limited to *five hours* (18000 sec.). The columns have the same meaning as in Sections 4.2 and 4.3.

Column "Chi" gives results obtained by the heuristic of Chinneck, see Section 3.4; its running times are negligible and therefore not listed. Our implementation found the same solutions as Chinneck [31], except for the instances `glass` and `wpbc`, for which Chinneck obtained solutions of size 39 and 10, respectively. Our primal heuristic described in Section 3.4 is run every tenth level. It could improve the initial solutions for models `glass`,

Table 6: Characteristics of the *classification instances.* Column "N" lists the number of attributes. The column labeled "$m^\star$" gives the number of original data sets and column "m" gives the number of data sets remaining after removing incomplete ones. The rightmost column gives additional notes, e.g., the name of the instance in the UCI database.

name	N	m	$m^\star$	notes
breast-cancer	9	683	699	breast-cancer-wisconsin
bupa	6	345	345	liver-disorders
echo	8	61	132	echocardiogram
glass	9	214	214	type 2 vs. others
heart	13	297	303	heart-disease (Cleveland)
ionosphere	34	351	351	
iris.1	4	150	150	Versicolor vs. others
iris.2	4	150	150	Virginica vs. others
new-thyroid	5	215	215	normal vs. others
pima	8	768	768	Pima-indians-diabetes
tic-tac-toe	9	958	958	
wpbc	32	194	198	Wisconsin breast-cancer database

Table 7: Results of the branch-and-cut algorithm for the *classification instances.*

name	nodes	time	IISs	root	dual	best	gap	Chi
breast-cancer	313	2.88	359	7.2	11.0	11	0.0	11
bupa	9669	18000.11	179562	43.2	59.6	83	39.3	83
echo	2	0.05	89	6.0	6.0	6	0.0	6
glass	36859	18000.00	99833	18.5	32.7	36	10.0	41
heart	51274	18000.02	122000	12.8	23.5	29	23.6	30
ionosphere	2465	38.59	3967	2.4	6.0	6	0.0	6
iris.1	845	12.45	623	19.1	25.0	25	0.0	25
iris.2	1	0.01	2	0.0	1.0	1	0.0	1
new-thyroid	2	0.09	147	11.0	11.0	11	0.0	11
pima	1522	18000.18	64166	68.2	75.6	148	95.7	148
tic-tac-toe	50691	5167.03	19850	60.9	86.0	86	0.0	93
wpbc	56657	18000.00	739494	3.5	8.7	13	48.7	13

`heart`, and `tic-tac-toe`. We conclude that the heuristic of Chinneck generates very good starting solutions, while our primal heuristic sometimes helps to find better solutions.

The results of Table 7 show that most instances are quite hard to solve and about half of them could not be solved within the time bound of five hours. Because of their size, only few nodes could be processed.

We also conducted experiments with the same data but using the second model instead of the first. Intuitively this should result in better numerical properties of the LPs that have to be solved during the algorithm. The results are, however, comparable to the ones shown in Table 7, and we therefore do not present them here.

Table 8 compares the gaps of the different cut strategies. The table displays only instances for which the optimal solutions could not be found within five hours. It turns out that all variants find the same final primal solutions, although at different times during the computation. Note that this actually compares the interplay of cutting strategies and our primal heuristic. On the average, the smallest gaps are produced by taking Gomory cuts,

Table 8: *Classification problems*: Comparison of the *gaps* of different variants of cutting planes. Only instances for which a positive gap remains after five hours are shown. The notation is as in Table 3. The last line contains the averages over each column.

					BaNg	BaNg	Gom.	
name	round	BaNg	Gom.	$\{0, \frac{1}{2}\}$	Gom.	$\{0, \frac{1}{2}\}$	$\{0, \frac{1}{2}\}$	all
bupa	39.3	46.7	40.0	41.9	44.0	45.0	41.5	45.5
glass	10.0	12.7	10.0	10.6	12.2	12.7	9.8	12.4
heart	23.6	23.8	22.6	24.2	25.4	25.2	27.8	26.0
pima	95.7	101.8	95.0	98.6	103.2	101.4	94.1	105.4
wpbc	48.7	47.8	44.6	49.8	49.0	49.0	45.6	50.5
∅:	43.5	46.6	42.4	45.0	46.7	46.7	43.8	48.0

then method “round”, Gomory and $\{0, \frac{1}{2}\}$-cuts, $\{0, \frac{1}{2}\}$-cuts alone, Balas/Ng cuts, Balas/Ng cuts and Gomory cuts, Balas/Ng cuts and $\{0, \frac{1}{2}\}$-cuts, and finally all cuts together. The main reason why all cuts together produce the worst results (on average) is that this combination could explore the fewest number of nodes. We conclude that the additional cutting planes do not yield a big improvement over method “round” alone. Although Gomory cuts produce the smallest gaps, the studied cutting planes do not seem to be crucial for solving these instances.

Our second test set consists of data from Codato and Fischetti [33] and uses the second model. The data again originate from the UCI Repository of Machine Learning Databases, but are preprocessed in a way we could not reconstruct. Hence, the results for these instances and the instances of Table 6 may not be comparable (there are three instances which seem to arise from the same original data: `breast-cancer` ↔ `breast-cancer-2`, `iris.1` ↔ `iris-150`, `wpbc` ↔ `WPBC194`). Instances `Breast-Cancer-2` and `Breast-Cancer-400` seem to be different from those used in Codato and Fischetti [33].

Table 9 shows the results of method “round” on these instances. The notation is as in Table 3. Note that here the dimension is $n = N + 2$, because we use the second model. Most of the instances could be solved within a few seconds. This is the first time that the complete set could be solved to optimality: no optimal solution to the harder instances (`Flags-169`, `Horse-colic-185`, `Horse-colic-253`, and `Solar-flare-1066`) was previously available. Our implementation solves all instances except these four in under a minute. Although we worked on a faster computer, it nevertheless seems fair to say that our code considerably improves upon the results of Codato and Fischetti [33].

5. Conclusions

In this paper we described a branch-and-cut implementation for the Max FS/Min IIS Cover problem, which is the best exact method currently available. The findings of the extensive computational results can be roughly summarized as follows: With respect to the implementation, the best cutting plane strategy is to find as many (violated) IIS-inequalities as possible. Additionally applying Balas/Ng, Gomory, or $\{0, \frac{1}{2}\}$-cuts does not significantly help to improve the performance: On random instances they do not improve

Table 9: *Classification problems*: Results of the branch-and-cut algorithm for the problems of Codato and Fischetti with method "round".

name	n	m	nodes	time	IISs	root	opt
Balloons-76	7	76	1	0.02	59	10.0	10
BCW-367	12	367	110	0.97	252	5.5	8
BCW-683	12	683	71	1.70	235	6.8	10
Breast-Cancer-2	11	683	352	2.21	322	7.0	11
Breast-Cancer-400	20	400	2	0.08	116	24.0	24
Bridges-132	14	132	299	3.44	1563	20.2	23
BusVan-437	20	437	237	1.72	353	3.0	6
BusVan-445	20	445	605	5.53	750	3.3	8
BusVan-447	20	447	2334	37.65	4187	4.4	10
BV-OS-282	20	282	214	1.39	338	3.0	6
BV-OS-376	20	376	969	12.03	1361	4.2	9
Chorales-107	8	107	951	9.57	1187	21.4	27
Chorales-116	8	116	1022	19.85	1981	17.2	24
Chorales-134	8	134	1198	50.99	4008	20.8	30
Credit-300	17	300	13	0.93	222	5.9	8
Flag-169	31	169	7621	209.63	17276	3.5	9
Glass-163	12	163	15	0.64	158	10.9	13
Horse-Colic-151	28	151	231	2.25	540	2.2	5
Horse-Colic-185	28	183	69155	886.10	61414	3.6	10
Horse-Colic-253	28	253	273389	7938.84	308862	4.8	13
House-Votes84-435	18	435	56	0.68	200	4.0	6
Iris-150	7	150	1017	6.58	1011	11.7	18
Lymphography-142	20	142	21	0.24	131	2.9	5
Mech-analysis-107	10	107	1	0.04	83	7.0	7
Mech-analysis-137	9	137	757	5.83	890	11.6	18
Mech-analysis-152	10	152	900	32.05	3042	13.0	21
Monks-tr-115	8	115	917	16.24	1570	20.9	27
Monks-tr-122	8	122	4	0.45	267	11.2	13
Monks-tr-124	8	124	489	5.91	1187	18.1	24
Opel-Saab-76	20	76	1111	9.28	1756	2.9	7
Opel-Saab-80	20	80	241	2.01	512	3.0	6
Opel-Saab-83	20	83	2113	25.05	3904	3.2	8
Opel-Saab-84	20	84	572	7.06	1318	3.3	7
Pb-gr-txt-198	12	198	147	1.09	267	7.7	11
Pb-hl-pict-277	12	277	178	1.61	314	6.7	10
Pb-pict-txt-444	12	444	2	0.12	79	7.0	7
Postoperative-88	10	88	1	0.12	209	16.0	16
Solar-flare-323	14	323	3	0.71	478	37.2	38
Solar-flare-1066	14	1066	2292	787.64	14960	227.3	243
Water-treat-206	40	206	41	1.43	204	1.7	4
Water-treat-213	40	213	288	8.04	845	2.2	5
WPBC-194	36	194	172	3.21	468	2.2	5

the running time, but usually help to reduce the number of nodes. Gomory cuts only slightly help to reduce the gaps for classification instances, and the other cuts do not improve the gap.

With respect to the problem data, the considered instances vary greatly in their properties and difficulty. Depending on the particular data, quite large instances can be solved to optimality, but there are also relatively small instances which turn out to be extremely hard to solve. As shown by the DVB problems, one has to be careful with numerically instable instances.

An interesting open issue is the existence of problem-specific cutting planes and whether they can be efficiently separated. Another question is whether other valid inequalities for the set covering problem could help to improve the performance of the implementation.

Acknowledgments. The author thanks Tobias Achterberg for help with the SCIP implementation and Edoardo Amaldi and Les Trotter for helpful discussions. Furthermore, he thanks Edoardo Amaldi and Pietro Belotti for providing the DVB instances of Section 4.3, and Gianni Codato and Matteo Fischetti for the data used in Section 4.4.

References

[1] T. Achterberg, *SCIP – A framework to integrate constraint and mixed integer programming*, Report 04–19, Zuse Institute Berlin, 2004. `http://www.zib.de/Publications/abstracts/ZR-04-19/`.

[2] T. Achterberg, T. Koch, and A. Martin, *Branching rules revisited*, Oper. Res. Lett. **33**, no. 1 (2005), pp. 42–54.

[3] S. Agmon, *The relaxation method for linear inequalities*, Canad. J. Math. **6** (1954), pp. 382–392.

[4] E. Amaldi, *From Finding Maximum Feasible Subsystems of Linear Systems to Feedforward Neural Network Design*, PhD thesis, EPF-Lausanne, 1994.

[5] E. Amaldi, *The maximum feasible subsystem problem and some applications*, in Modelli e Algoritmi per l'Ottimizzazione di Sistemi Complessi, A. Agnetis and G. D. Pillo, eds., Pitagora Editrice, Bologna, Italy, 2003, pp. 31–69.

[6] E. Amaldi, P. Belotti, and R. Hauser, *Randomized relaxation methods for the maximum feasible subsystem problem*, in Proc. 11th International Conference on Integer Programming and Combinatorial Optimization (IPCO), Berlin, M. Jünger and V. Kaibel, eds., LNCS 3509, Springer-Verlag, Berlin Heidelberg, 2005, pp. 249–264.

[7] E. Amaldi, M. Bruglieri, and G. Casale, *A two-phase relaxation-based heuristic for the maximum feasible subsystem problem*, Comput. Oper. Res. , no. 5 (2008), pp. 1377–1756.

[8] E. Amaldi and R. Hauser, *Boundedness theorems for the relaxation method*, Math. Oper. Res. **30**, no. 4 (2005), pp. 1–17.

[9] E. Amaldi and V. Kann, *The complexity and approximability of finding maximum feasible subsystems of linear relations*, Theor. Comput. Sci. **147**, no. 1–2 (1995), pp. 181–210.

[10] E. Amaldi and V. Kann, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theor. Comput. Sci. **209**, no. 1–2 (1998), pp. 237–260.

[11] E. Amaldi and M. E. Pfetsch, *Separation problems for set covering*, 2005. Manuscript.

[12] E. Amaldi, M. E. Pfetsch, and L. E. Trotter, Jr., *On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs*, Math. Program. **95**, no. 3 (2003), pp. 533–554.

[13] G. Andreello, A. Caprara, and M. Fischetti, *Embedding $\{0, \frac{1}{2}\}$-cuts in a branch-and-cut framework: A computational study*, INFORMS J. Comput. **19** (2007), pp. 229–238.

[14] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza, *Exact solutions to linear programming problems*, Oper. Res. Lett. **35** (2007), pp. 693–699.

[15] D. Avis, `lrs` *home page*. Available at: `http://cgm.cs.mcgill.ca/~avis/C/lrs.html`.

[16] D. Avis and K. Fukuda, *Reverse search for enumeration*, Discrete Appl. Math. **65**, no. 1–3 (1996), pp. 21–46.

[17] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj, *Gomory cuts revisited*, Oper. Res. Lett. **19**, no. 1 (1996), pp. 1–9.

[18] E. Balas and S. M. Ng, *On the set covering polytope. I. All the facets with coefficients in* $\{0, 1, 2\}$, Math. Program. **43**, no. 1 (1989), pp. 57–69.

[19] K. P. Bennett and E. J. Bredensteiner, *A parametric optimization method for machine learning*, INFORMS J. Comput. **9**, no. 3 (1997), pp. 311–318.

[20] K. P. Bennett and O. L. Mangasarian, *Neural network training via linear programming*, in Advances in optimization and parallel computing, P. M. Pardalos, ed., North-Holland, Amsterdam, 1992, pp. 56–67.

[21] C. L. Blake and C. J. Merz, *UCI repository of machine learning databases*, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[22] R. Borndörfer, *Aspects of Set Packing, Partitioning, and Covering*, PhD thesis, TU Berlin, 1998.

[23] R. Borndörfer and R. Weismantel, *Set packing relaxations of some integer programs*, Math. Program. **88** (2000), pp. 425–450.

[24] R. Borndörfer and R. Weismantel, *Discrete relaxations of combinatorial programs*, Discrete Appl. Math. **112**, no. 1–3 (2001), pp. 11–26.

[25] A. Caprara and M. Fischetti, $\{0, \frac{1}{2}\}$*-Chvátal-Gomory cuts*, Math. Prog. **74**, no. 3 (1996), pp. 221–235.

[26] A. Caprara and M. Fischetti, *Branch-and-cut algorithms*, in Annotated Bibliographies in Combinatorial Optimization, M. Dell'Amico, F. Maffioli, and S. Martello, eds., John Wiley & Sons, Chichester, UK, 1997, ch. 4, pp. 45–63.

[27] S. Ceria, P. Nobili, and A. Sassano, *Set covering problem*, in Annotated Bibliographies in Combinatorial Optimization, M. Dell'Amico, F. Maffioli, and S. Martello, eds., John Wiley & Sons, Chichester, UK, 1997, ch. 23, pp. 415–428.

[28] N. Chakravarti, *Some results concerning post-infeasibility analysis*, Eur. J. Oper. Res. **73** (1994), pp. 139–143.

[29] J. W. Chinneck, *An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem*, Ann. Math. Artif. Intell. **17**, no. 1–2 (1996), pp. 127–144.

[30] J. W. Chinneck, *Finding a useful subset of constraints for analysis in an infeasible linear program*, INFORMS J. Comput. **9**, no. 2 (1997), pp. 164–174.

[31] J. W. Chinneck, *Fast heuristics for the maximum feasible subsystem problem*, INFORMS J. Comput. **13**, no. 3 (2001), pp. 210–223.

[32] J. W. Chinneck and E. W. Dravnieks, *Locating minimal infeasible constraint sets in linear programs*, ORSA J. Comput. **3**, no. 2 (1991), pp. 157–168.

[33] G. Codato and M. Fischetti, *Combinatorial Benders' cuts*, in Proc. 10th International Conference on Integer Programming and Combinatorial Optimization (IPCO), New York, D. Bienstock and G. Nemhauser, eds., LNCS 3064, Springer-Verlag, Berlin Heidelberg, 2004, pp. 178–195.

[34] K. Fukuda, `cdd` *home page*. Available at: `http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`.

[35] J. Gleeson and J. Ryan, *Identifying minimally infeasible subsystems of inequalities*, ORSA J. Comput. **2**, no. 1 (1990), pp. 61–63.

[36] H. J. Greenberg and F. H. Murphy, *Approaches to diagnosing infeasible linear programs*, ORSA J. Comput. **3**, no. 3 (1991), pp. 253–261.

[37] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Heidelberg, 2nd ed., 1993.

[38] P. Hansen, M. Labbé, and D. Schindl, *Set covering and packing formulations of graph coloring: algorithms and first polyhedral results*, tech. report, GERAD, HEC Montréal, Canada, 2005.

[39] D. S. Johnson and F. P. Preparata, *The densest hemisphere problem*, Theor. Comput. Sci. **6** (1978), pp. 93–107.

[40] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich, *Generating all vertices of a polyhedron is hard*, in Proc. of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), ACM Press, New York, 2006, pp. 758–765.

[41] T. Koch, *The final* Netlib-*LP results*, Oper. Res. Lett. **32**, no. 2 (2004), pp. 138–142.

[42] A. N. Letchford and A. Lodi, *Strengthening Chvátal-Gomory cuts and Gomory fractional cuts*, Oper. Res. Lett. **30**, no. 2 (2002), pp. 74–82.

[43] O. L. Mangasarian, *Misclassification minimization*, J. Glob. Optim. **5**, no. 4 (1994), pp. 309–323.

[44] H. Marchand and L. Wolsey, *Aggregation and mixed integer rounding to solve MIPs*, Oper. Res. **49**, no. 3 (2001), pp. 363–371.

[45] Maximum Feasible Subsystem Home Page. http://risorse.dei.polimi.it/maxfs/.

[46] T. S. Motzkin, *The probability of solvability of linear inequalities*, in Selected papers, D. Cantor, B. Gordon, and B. Rothschild, eds., Contemp. Mathematicians, Birkhäuser, Boston, 1983, pp. 116–120.

[47] T. S. Motzkin and I. J. Schoenberg, *The relaxation method for linear inequalities*, Canad. J. Math. **6** (1954), pp. 393–404.

[48] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[49] Netlib. http://www.netlib.org.

[50] M. Padberg and G. Rinaldi, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev. **33**, no. 1 (1991), pp. 60–100.

[51] M. Parker, *A Set Covering Approach to Infeasibility Analysis of Linear Programming Problems and Related Issues*, PhD thesis, University of Colorado at Denver, 1995.

[52] M. Parker and J. Ryan, *Finding the minimum weight IIS cover of an infeasible system of linear inequalities*, Ann. Math. Artif. Intell. **17**, no. 1–2 (1996), pp. 107–126.

[53] M. E. Pfetsch, *The Maximum Feasible Subsystem Problem and Vertex-Facet Incidence of Polyhedra*, PhD thesis, TU Berlin, 2002.

[54] F. Rossi, A. Sassano, and S. Smriglio, *Models and algorithms for terrestrial digital broadcasting*, Ann. Oper. Res. **107** (2001), pp. 267–283.

[55] Rubin, *Solving mixed integer classification problems by decomposition*, Ann. Oper. Res. **74** (1997), pp. 51–64.

[56] D. M. Ryan and B. A. Foster, *An integer programming approach to scheduling*, in Computer scheduling of public transport: Urban passenger vehicle and crew scheduling, A. Wren, ed., North-Holland, Amsterdam, 1981, pp. 269–280.

[57] J. Ryan, *Transversals of IIS-hypergraphs*, in Proc. 22nd Southeast Conf. on Combinatorics, Graph Theory, and Computing (Baton Rouge, 1991), Congr. Numer. 81, 1991, pp. 17–22.

[58] J. K. Sankaran, *A note on resolving infeasibility in linear programs by constraint relaxation*, Oper. Res. Lett. **13** (1993), pp. 19–20.

[59] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, UK, 1986.

[60] S. Thienel, *ABACUS – A Branch-And-CUt System*, PhD thesis, Universität zu Köln, 1995.

[61] M. Wagner, J. Meller, and R. Elber, *Solving huge linear programming problems for the design of protein folding potentials*, Math. Program. **101** (2004), pp. 301–318.

Marc E. Pfetsch, Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
E-mail address: pfetsch@zib.de