Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

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)

BRANCH-AND-CUT FOR THE MAXIMUM FEASIBLE SUBSYSTEM PROBLEM

MARC E. PFETSCH

ABSTRACT. We present a branch-and-cut algorithm for the $\mathcal{NP}$-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. The main issue of a branch-and-cut algorithm for MAX FS is to efficiently find such infeasible subsystems. We present three heuristics for the corresponding $\mathcal{NP}$-hard separation problem and discuss further cutting planes. This paper contains an extensive computational study of our implementation 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 $\mathcal{NP}$-hard combinatorial optimization problem has a number of interesting applications in a wide range of fields, for instance, in linear programming [26, 28, 33], statistical discriminant analysis and machine learning [4, 17, 39], telecommunications [48], and computational biology [55]. 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" in Berlin.

Since the number of IISs can be exponential in the size of the system Σ (see Chakravarti [25] and Pfetsch [47]), IISs have to be generated dynamically in order to solve this formulation of Min IIS Cover.

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 $\mathcal{NP}$-hard, see Johnson and Preparata [35], Sankaran [52], and Chakravarti [25]. In terms of approximability, however, they differ: Max FS does not admit a polynomial-time approximation scheme, unless $\mathcal{P} = \mathcal{NP}$, but there exists a 2-approximation, see Amaldi and Kann [9]. Min IIS Cover is harder to approximate: Unless $\mathcal{P} = \mathcal{NP}$, it cannot be approximated 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 $\mathcal{NP}$-hard and we present three heuristics for it (see Section 3.2). Two of these methods either generate a feasible solution for Min IIS Cover or (hopefully violated) IIS-inequalities. As long as no feasible solution has been generated, the process is iterated, which often produces many useful IIS-inequalities. The additional benefit are 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 [16] for set covering problems and Gomory cuts (see e.g. Balas et al. [15]).

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 inequalities 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 [47].

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 solution for Min IIS Cover, three methods to separate IIS-inequalities, the primal heuristic, preprocessing, branching, and the inequalities by Balas and Ng. In Section 4 we extensively test the implementation on the above mentioned 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 ones 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 [33]; Chinneck [27]; Chinneck and Dravnieks [29]). 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 [26, 28] 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 [17]; Bennett and Mangasarian [18]; Mangasarian [39]).

An exact integer programming approach for Min IIS Cover appeared in Parker [45] and Parker and Ryan [46]. 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 to generate 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 instances within one hour, 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 [45]. 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 [48] and Codato and Fischetti [30]. In fact, Codato and Fischetti proposed a general way of removing the "big-M" from this type of formulations 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 [41] 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 CPLEX 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 [42], Padberg and Rinaldi [44], Thienel [54], and Caprara and Fischetti [23]. A description and computational study of Gomory cuts is given in Balas, Ceria, Cornuéjols, and Natraj [15].

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, 47] 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 the inequalities of Balas and Ng.

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 [32]). *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 [53]. 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 [46]). *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. □

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 [34]. 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 $\mathcal{NP}$-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

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

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 to separate an IIS-inequality, already used by Parker and Ryan [46], is to approximate (2) by the following 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 only generates 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.

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 there exists one. If the LP is feasible, the result gives us a vertex which corresponds to an IIS, otherwise we 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. This procedure yields an IIS-cover at termination and different IISs in each step (except in the last). It is related to a primal heuristic proposed by Ryan [51].

The IISs found by this approach have several nice properties. First, 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. Second, the generated IISs are pairwise different. To see this, consider the set $\mathcal{I}(S)$ of all IISs which are covered by S, i.e., $I \cap S \neq \varnothing$ for each $I \in \mathcal{I}(S)$. By definition, the polyhedron $P_S(\Sigma)$ only has vertices whose components indexed by S are zero, which excludes all vertices corresponding to IISs in $\mathcal{I}(S)$ by Theorem 1. Hence, each new IIS is not contained in the set $\mathcal{I}(S)$ of the previous step. Third, one can ensure that the computed IISs are different from the previously known IISs: Take $\boldsymbol{y}^*$ to be the optimal solution of the LP-relaxation in some node of the tree. Clearly, $\boldsymbol{y}^*$ satisfies $y^*(I) \geq 1$ for each IIS I whose inequality is included in this LP-relaxation. Therefore, for each such IIS I there exists $i \in I$ such that $y^*_i > 0$. In particular, we have $I \in \mathcal{I}(S)$ for $S = \operatorname{supp}(\boldsymbol{y}^*)$. Hence, the new IISs are not included in the current LP-relaxation.

This method turns out to be quite effective for generating many violated IIS-inequalities. Furthermore, we obtain a primal solution in each turn, 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. We choose $\alpha \in [0,1]$ and initially let $S := \{i : y_i^* \geq \alpha\}$. Then we proceed greedily as above. 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.

In the implementation we start with $\alpha = 0.1$ and then increase α by 0.1 until 0.6 is exceeded or S is not an IIS-cover (in this case the above procedure is started).

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 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 may IISs, which we store in a pool. It turned 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. Therfore, the pool is equipped with an aging mechanism which removes IISs whose inequality has not been active for some time. Nevertheless, all IISs are stored in a second global pool, which is checked every 20th level of the tree for violated inequalities.

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

3.4. **Primal Heuristics.** We implemented a simple primal heuristic, which 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 a 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 pseudo costs during branching.

We also experimented with constraint branching rules. For instance, we used the well-known rule of Ryan and Foster [50]. 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 [24] and Borndörfer [20]. 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 $\mathcal{NP}$-hard.

We experimented with the aggregated cycle cuts of Borndörfer and Weismantel [21, 22]. Unfortunately, on our test problems their separation heuristic almost never found a violated inequality. It remains as an interesting open problem to identify problem specific inequalities for Min IIS Cover.

3.7.1. *The Inequalities of Balas and Ng.* A class of inequalities for set covering that we use in our implementation were proposed by Balas and Ng [16]. 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 $\boldsymbol{a}\boldsymbol{y} \geq \beta$, 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 $\boldsymbol{a} \geq \mathbf{0}$, and if $\beta = 1$ then $\boldsymbol{a}$ is a row of D (see, e.g., [16]).

Balas and Ng show that for every facet defining inequality $\boldsymbol{a}\boldsymbol{y} \geq 2$ 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 $\boldsymbol{a}^S\boldsymbol{y} \geq 2$ defines a facet of $P_{SC}(D)$.

The separation problem for these inequalities is $\mathcal{NP}$-hard, see Amaldi and Pfetsch [11]. However, when the size of S is fixed, the separation problem can be solved in time $\mathcal{O}(m^{|S|})$ by enumeration. In our implementation we enumerate sets S of *cardinality three* and check if the inequalities $\boldsymbol{a}^S\boldsymbol{y} \geq 2$ are violated by the current LP-solution. Note that the sets S of cardinality two are uninteresting, since in this case $\boldsymbol{a}^S\boldsymbol{x} \geq 2$ 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.

4. Computational Results

In this section we present and discuss computational results of our branch-and-cut implementation for Min IIS Cover. The algorithm was implemented in C++ and uses the framework SCIP (Solving Constraint Integer Programs) by Achterberg [1]. We used CPLEX 9.1 as the basic LP solver. The computations were performed on a 3.4 GHz Pentium 4 machine with 2 GB of main memory and 512 KB cache running Linux. All instances used in the following can be obtained from the web page [40].

We use best-first search, the branching rule explained in Section 3.6, and run the primal heuristic of Section 3.4 every tenth level. Similarly, new IIS-inequalities, Balas and Ng cuts, and (mixed integer) Gomory cuts are separated only every tenth level of the tree. Recall that the pool of IIS-inequalities is separated in every node of the tree.

We also experimented with strengthened Chvátal-Gomory cuts, see Letchford and Lodi [38]. The results were, however, always inferior to the ones obtained by Gomory cuts and we therefore do not present them here.

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 only has a negligible effect on the objective function value, see Koch [37]. 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 [13] or cdd [31] is computationally too expensive.

4.1. **The Netlib Problems.**

The Netlib library [43] contains a well known set of 29 infeasible linear inequality systems. We do not report results on these data since they all can be solved within seconds, except for numerical difficulties with the problem `gran`. They were also solved to optimality by Parker [45], Parker and Ryan [46]; for more computational results on these problems see Chinneck [28] and [47].

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 one hour of computation time. In contrast, the instances discussed in the following sections vary highly in size and complexity: often they are either solved within seconds or cannot be solved to optimality in reasonable time.

The infeasible random inequality systems are generated 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 with 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 $m \geq 2 \cdot n$) and is almost completely dense.

Note that the alternative polyhedra in Theorem 1 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 $\mathcal{NP}$-hard.

Table 1 gives computational results for the following combinations of cuts:

- Method 1 "Single" (Section 3.2.1),
- Method 2 "Extend" (Section 3.2.2),
- Method 3 "Round" (Section 3.2.3),
- Method 3 "Round" and Balas/Ng cuts (see Section 3.7.1),
- Method 3 "Round" and Gomory cuts,
- Method 3 "Round", Balas/Ng cuts, and Gomory cuts.

The columns correspond to the following: "nodes" gives the total number of nodes in the branch-and-bound tree, "D" lists the maximum depth of the tree, "time" is the CPU time in seconds, "IISs" gives the number of IISs found during the optimization, "ptime" is the percentage of the total time needed until the optimal solution was found, "root" denotes the dual bound in the root node, and "opt" is the optimal value.

Our primary goal of this investigation is to find the combination of cuts that is the fastest, *together* with the primal heuristic. Hence, a combination of cuts may be slow because the heuristic finds the solution very late during the branch-and-bound process.

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 only test combinations of method "round" with additional cuts. The studied combinations consistently reduce the number of nodes with respect to the method "round" alone, but there are three exceptions with the variants involving Gomory cuts, see below.

Adding Balas/Ng cuts generates the fewest total number of nodes of all variants, but is quite time consuming. The variants involving these cuts also generate the most IISs in total; this might happen because adding other cuts increases the number of rounds per node, which increases the chance that method "round" finds additional IISs.

Applying Gomory cuts in addition to method "round" results in fewer nodes and yields the lowest total computation time among all variants. Using Gomory cuts and Balas/Ng cuts further reduces the number of nodes. An exception is instance $(20, 90)$ where it is only second in the fewest number of nodes; here the optimal solution is found very late. The total running time of this variant is very large.

Among the studied variants there is no clear winner. The total computation times of the best variants (method "round" and adding Gomory cuts) are quite close together. Adding cuts usually reduces the number of nodes, but also increases the computation time. Furthermore, the optimal solution is often found later than in method "round". Based on this observations it seems fair to say that neither Balas and Ng cuts nor Gomory cuts significantly help to solve random problems in addition to method "Round".

Table 1: Results of the branch-and-cut algorithm on *random inequality systems* for different cut generation strategies. For each variant the sum of the values for "nodes", "D", "time", and "IISs" is given. The column "ptime" gives the percentage of the total time needed until the optimal solution was found.

n	m	nodes	D	time	IISs	ptime	root	opt
Single:								
5	100	174646	30	2543.9	16400	4.5	14.96	33
10	80	214449	26	2049.5	8420	0.3	6.97	17
15	80	47439	24	317.1	6029	1.8	3.73	11
20	90	86145	28	1448.2	21545	79.5	3.23	10
25	90	97724	32	1995.8	27782	0.2	2.75	9
		620403	140	8354.6	80176			
Extend:								
5	100	432005	34	5736.9	5360	41.1	13.61	33
10	80	277142	27	2872.8	4627	18.5	6.64	17
15	80	56885	25	506.2	4382	85.8	3.43	11
20	90	115630	32	1293.2	6346	4.5	2.96	10
25	90	158434	36	2313.4	9743	1.2	2.65	9
		1040096	154	12722.4	30458			
Round:								
5	100	43750	23	1446.8	34779	4.2	15.55	33
10	80	73281	21	1808.8	79124	0.5	7.27	17
15	80	20894	22	378.9	25851	48.1	4.24	11
20	90	44584	30	1501.7	50336	67.9	3.37	10
25	90	48502	30	2513.7	79234	0.1	3.02	9
		231011	126	7649.8	269324			
Round & Balas/Ng:								
5	100	29898	21	1349.7	56643	0.9	16.35	33
10	80	59986	20	2331.1	190094	17.9	7.43	17
15	80	19968	20	749.8	102102	99.8	4.58	11
20	90	34924	27	2422.2	105621	64.3	3.55	10
25	90	43078	29	5125.9	150153	0.2	3.11	9
		187854	117	11978.8	604613			
Round & Gomory:								
5	100	31432	23	1063.9	32404	2.3	16.34	33
10	80	74826	21	2600.7	78443	92.5	7.19	17
15	80	19136	21	347.5	25794	60.0	4.31	11
20	90	45542	26	1031.6	29711	4.1	3.16	10
25	90	47932	30	2387.2	69971	0.6	2.90	9
		218868	121	7430.9	236323			
Round & Balas/Ng & Gomory:								
5	100	27726	21	1197.5	54703	4.2	16.97	33
10	80	40047	20	2011.7	198066	17.8	7.55	17
15	80	13859	20	611.6	86918	86.8	4.49	11
20	90	72499	25	6732.6	181541	99.9	3.45	10
25	90	42776	30	5091.4	138554	1.5	3.08	9
		196907	116	15644.8	659782			

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	17	1986
5	40	204	44816
5	50	520	204833
5	60	407	614853
5	70	2767	1818718

It can be observed that the optimal values of the random problems decrease when increasing the dimension. This often makes the problems more tractable, but the solution of the intermediate LPs over the alternative polyhedron becomes more time consuming.

Finally, Table 2 shows the total number of IISs and the number 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 [13]. Since the alternative polyhedra are nondegenerate, the IISs can be generated in time polynomial in the input and output size, see Avis and Fukuda [14]. Note that for general polyhedra this is not possible, unless $\mathcal{P} = \mathcal{NP}$, see Khachiyan et al. [36].

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

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 [48]. Here, to plan 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 maximize 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

Table 3: 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 et al. [6].

name	[6]	m	nodes	D	time	IISs	root	dual	best	gap
dvb1	dvb2	1044	1798	22	579.99	2877	165.3	174.0	174	0.0
mfs_UHF_P4_1	dvb1	642	1	0	0.54	90	103.4	104.0	104	0.0
mfs_UHF_P4_3	dvb3	1717	199176	22	8805.44	4708	174.7	183.0	183	0.0
mfs_UHF_P4_4	–	1174	500339	37	346476.36	18796	88.9	96.1	125	23.1

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, Bruglieri, and Casale [7] and Amaldi, Belotti, and Hauser [6], see Table 3. The dimension of this instances is always 487 and the variable bounds ($0 \leq \boldsymbol{x} \leq 1$) are mandatory. 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 [39], Bennett and Bredensteiner [17], and Rubin [49].

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 $\boldsymbol{ax} \leq \beta$, with $\boldsymbol{a} \in \mathbb{R}^N$ and $\beta \in \mathbb{R}$. The points in P_1 should satisfy the inequality $\boldsymbol{ax} < \beta$ and the points in P_2 should satisfy $\boldsymbol{ax} > \beta$. Hence, we are looking for $(\boldsymbol{a}, \beta) \in \mathbb{R}^n$, with $n := N + 1$ so that the number of misclassified points

$$|\{\boldsymbol{p} \in P_1 \,:\, \boldsymbol{ap} \geq \beta\}| + |\{\boldsymbol{p} \in P_2 \,:\, \boldsymbol{ap} \leq \beta\}|$$

is minimized. This minimization is performed in order to maximize the chance that a new point can be correctly classified.

In the following we will discuss two equivalent ways to model this problem via Min IIS Cover and present computational results for different datasets. The main difference is that 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

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

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

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

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 [49]. We use variables $\boldsymbol{a} \in \mathbb{R}^N$, β, $\gamma \in \mathbb{R}$ and the following system:

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

Table 4: Characteristics of the *classification instances* of Section 4.4. Column "N" lists the number of attributes. The column labeled $m^\star$ gives the number of original data sets and m the number of data sets remaining after removing incomplete ones. The right most 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 5: Results of the branch-and-cut algorithm for the *classification instances* in Section 4.4 with method "Round" (Section 3.2.3).

name	nodes	D	time	IISs	root	dual	best	gap	Chi
breast-cancer	123	10	2.15	319	7.3	11.0	11	0.0	11
bupa	7509	25	18010.72	455222	43.7	58.9	83	29.0	83
echo	2	1	0.10	141	4.7	6.0	6	0.0	–
glass	37497	22	18000.69	353143	18.5	32.5	36	9.7	39
heart	42390	22	18000.08	249320	13.0	23.2	29	20.0	–
ionosphere	1571	24	47.14	5696	2.3	6.0	6	0.0	6
iris.1	588	10	12.81	821	19.1	25.0	25	0.0	25
iris.2	1	0	0.00	2	1.0	1.0	1	0.0	1
new-thyroid	2	1	0.09	119	9.5	11.0	11	0.0	11
pima	1415	21	18001.41	136183	68.9	75.1	148	49.2	148
tic-tac-toe	44771	35	5314.37	72247	59.8	86.0	86	0.0	–
wpbc	24791	30	18000.01	298204	3.7	8.3	13	36.2	10

Hence, the coefficients of the normal vector $\boldsymbol{a}$ are bounded to lie within the interval $[-1, 1]$, while β is unbounded. For instances arising from this model the variable bounds are mandatory.

Note that in both models it might happen that this final system is feasible, i.e., the points are completely separable (in which case we only need to 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 [19]). The problem characteristics are given in Table 4. For some instances we had to remove incomplete data sets. A complete description of the particular contexts in which these instances arise is available at the UCI Repository. Most of these twelve instances are also used by Chinneck [28] for testing his heuristic for Max FS/Min IIS Cover.

Table 5 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 Section 4.2, where additionally "dual" gives the final lower bound,

Table 6: *Classification problems*: Comparison of the gaps of different variants of cutting planes. Only instances for which a positive gap after five hours remains are shown. Column "B&G" stands for the variant using Balas/Ng and Gomory cuts.

name	single	extend	round	Balas/Ng	Gomory	B&G
bupa	29.3	36.0	29.0	30.3	30.3	30.0
glass	13.8	20.0	9.7	9.8	10.0	9.9
heart	21.2	23.5	20.0	19.7	20.9	22.4
pima	54.0	59.0	49.2	50.4	50.2	50.4
tic-tac-toe	9.4	16.1	0.0	0.0	0.0	0.0
wpbc	37.3	29.3	36.2	42.2	36.3	37.0

"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)/best $\cdot$ 100.0. The column "Chi" gives results obtained by the heuristic of Chinneck [28].

The values show that most of these 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. The heuristic of Chinneck generates very good solutions and is usually done in a few minutes. The primal solutions that are generated by our branch-and-cut implementation are relatively good and in one case better.

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 5 and we therefore do not present them here.

Table 6 compares the gaps of the different variants considered in Section 4.2. The table only displays instances for which the optimal solutions could not be found within five hours. We see that the gaps of method "round" are only beaten in two cases: for instance `heart` slightly by variant "Balas/Ng" and for instance `wpbc` by variant "extend" by a larger amount. For the latter instance, method "extend" could compute much more nodes and therefore improve the dual bound to 9.2 compared to the bound of 8.3 by method "round". Altogether, however, method "round" seems to be the best variant for solving these classification problems.

Our second test set consists of data from Codato and Fischetti [30] and uses the second model. The data again originate from the UCI Repository of Machine Learning Databases, but are preprocessed in way we could not reconstruct. Hence, the results for these instances and the instances of Table 4 may not be comparable (there are three instances which seem to arise from the same original data: `breast-cancer` $\leftrightarrow$ `breast-cancer-2`, `iris.1` $\leftrightarrow$ `iris-150`, `wpbc` $\leftrightarrow$ `WPBC194`). Instances `Breast-Cancer-2` and `Breast-Cancer-400` seem to be different to the ones used in Codato and Fischetti [30].

Table 7 shows the results of method "Round" on these instances. The notation is as in Table 1. 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

Table 7: *Classification Problems*: Results of the branch-and-cut algorithm for the problems of Codato and Fischetti with method "Round"(Section 3.2.3).

name	n	m	nodes	D	time	IISs	root	opt
Balloons-76	7	76	1	0	0.03	104	9.7	10
BCW-367	12	367	33	6	0.44	137	5.0	8
BCW-683	12	683	244	10	3.65	392	6.8	10
Breast-Cancer-2	11	683	155	10	2.47	319	7.2	11
Breast-Cancer-400	20	400	1	0	0.03	117	24.0	24
Bridges-132	14	132	218	10	3.71	1702	19.7	23
BusVan-437	20	437	123	10	1.60	342	2.9	6
BusVan-445	20	445	1698	20	19.12	2983	4.0	8
BusVan-447	20	447	2738	20	59.26	9361	4.5	10
BV-OS-282	20	282	93	9	1.03	289	3.0	6
BV-OS-376	20	376	855	16	12.28	1429	4.4	9
Chorales-107	8	107	641	13	10.08	1819	20.0	27
Chorales-116	8	116	685	13	17.28	2811	17.5	24
Chorales-134	8	134	1079	12	58.07	5399	22.5	30
Credit-300	17	300	55	8	1.06	169	5.3	8
Flags-169	31	169	4804	20	225.44	20960	3.6	9
Glass-163	12	163	13	4	0.89	216	10.8	13
Horse-colic-151	28	151	79	14	1.93	460	2.3	5
Horse-colic-185	28	183	46611	25	1313.08	51074	3.5	10
Horse-colic-253	28	253	135063	25	8948.58	80624	4.7	13
HouseVotes84-435	18	435	74	8	0.90	284	4.2	6
Iris-150	7	150	1037	12	6.97	1121	11.2	18
Lymphography-142	20	142	17	7	0.22	114	3.0	5
Mech-analysis-107	10	107	1	0	0.05	93	6.0	7
Mech-analysis-137	9	137	877	12	7.67	1021	11.9	18
Mech-analysis-152	10	152	1212	13	59.63	6073	14.9	21
Monks-tr-115	8	115	815	13	18.16	1849	21.6	27
Monks-tr-122	8	122	17	5	0.50	197	10.8	13
Monks-tr-124	8	124	572	13	10.24	2154	17.7	24
Opel-Saab-76	20	76	805	16	10.82	2913	3.0	7
Opel-Saab-80	20	80	93	10	1.32	360	2.9	6
Opel-Saab-83	20	83	1703	19	29.03	7382	3.1	8
Opel-Saab-84	20	84	527	16	7.16	1572	3.2	7
Pb-gr-txt-198	12	198	60	10	0.82	266	8.0	11
Pb-hl-pict-277	12	277	174	10	1.98	325	6.7	10
Pb-pict-txt-444	12	444	24	7	0.29	136	6.1	7
Postoperative-88	10	88	2	1	0.12	294	14.8	16
Solar-flare-323	14	323	4	2	0.30	364	37.2	38
Solar-flare-1066	14	1066	3308	30	1220.64	20174	224.2	243
Water-treat-206	40	206	136	20	6.19	840	2.1	4
Water-treat-213	40	213	120	19	6.83	794	2.3	5
WPBC-194	36	194	289	16	10.35	1454	2.5	5

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 seems therefore fair to say that our code considerably improves upon the results of Codato and Fischetti [30].

5. Conclusions

In this paper we described a branch-and-cut implementation for the Max FS/Min IIS Cover problem and provided extensive computational results on several types of instances. The findings can roughly be 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 cuts and Gomory cuts does not significantly help to improve the running time, but usually helps to reduce the number of nodes for instances that can be solved to optimality. They also do not help to decrease the gaps between the best solution and the lower bound for instances not solved to optimality.

With respect to the problem data, the considered instances vary highly 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 be helpful 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, ZIB, 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*, Can. 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, 2003, pp. 31–69.

[6] E. Amaldi, P. Belotti, and R. Hauser, *Randomized relaxation methods for the maximum feasible subsystem problem*, in Integer Programming and Combinatorial Optimizations, Proceedings of the 11th IPCO Conference, M. Jünger and V. Kaibel, eds., Lecture Notes in Computer Science 3509, Springer, 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*, Computers and Operations Research (2005). Under revision.

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

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

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

[16] 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.

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

[18] 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.

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

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

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

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

[23] 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, 1997, ch. 4, pp. 45–63.

[24] 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, 1997, ch. 23, pp. 415–428.

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

[26] 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.

[27] 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.

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

[29] 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.

[30] 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., Lecture Notes in Computer Science 3064, Springer, Berlin Heidelberg, 2004, pp. 178–195.

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

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

[33] H. J. Greenberg and F. H. Murphy, *Approaches to diagnosing infeasible linear programs*, ORSA J. Comput. **3**, no. 3 (1991), pp. 253–261.
[34] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics 2, Springer-Verlag, Heidelberg, 2nd ed., 1993.
[35] D. S. Johnson and F. P. Preparata, *The densest hemisphere problem*, Theor. Comput. Sci. **6** (1978), pp. 93–107.
[36] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich, *Generating all vertices of a polyhedron is hard*, Tech. Report RRR 18, RUTCOR, 2005.
[37] T. Koch, *The final* Netlib-*LP results*, Oper. Res. Lett. **32**, no. 2 (2004), pp. 138–142.
[38] A. N. Letchford and A. Lodi, *Strengthening Chvátal-Gomory cuts and Gomory fractional cuts*, Oper. Res. Lett. **30**, no. 2 (2002), pp. 74–82.
[39] O. L. Mangasarian, *Misclassification minimization*, J. Glob. Optim. **5**, no. 4 (1994), pp. 309–323.
[40] Maximum Feasible Subsystem Home Page. `http://www.elet.polimi.it/res/maxfs/`.
[41] T. S. Motzkin and I. J. Schoenberg, *The relaxation method for linear inequalities*, Can. J. Math. **6** (1954), pp. 393–404.
[42] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.
[43] Netlib. `http://www.netlib.org`.
[44] 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.
[45] M. Parker, *A Set Covering Approach to Infeasibility Analysis of Linear Programming Problems and Related Issues*, PhD thesis, University of Colorado at Denver, 1995.
[46] 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.
[47] M. E. Pfetsch, *The Maximum Feasible Subsystem Problem and Vertex-Facet Incidence of Polyhedra*, PhD thesis, TU Berlin, 2002.
[48] F. Rossi, A. Sassano, and S. Smriglio, *Models and algorithms for terrestrial digital broadcasting*, Annals of Operations Research **107** (2001), pp. 267–283.
[49] Rubin, *Solving mixed integer classification problems by decomposition*, Ann. Oper. Res. **74** (1997), pp. 51–64.
[50] 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.
[51] J. Ryan, *Transversals of IIS-hypergraphs*, in Proc. 22nd Southeast Conf. on Combinatorics, Graph Theory, and Computing, Baton Rouge, Congr. Numerantium 81, 1991, pp. 17–22.
[52] J. K. Sankaran, *A note on resolving infeasibility in linear programs by constraint relaxation*, Oper. Res. Letters **13** (1993), pp. 19–20.
[53] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, 1986.
[54] S. Thienel, *ABACUS – A Branch-And-CUt System*, PhD thesis, Universität zu Köln, 1995.
[55] 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`