A Two-Phase Relaxation-Based Heuristic for the Maximum Feasible Subsystem Problem

Edoardo Amaldi, Maurizio Bruglieri, Giuliano Casale
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 Milano, Italy
{amaldi, bruglier, casale}@elet.polimi.it

Abstract

We consider the maximum feasible subsystem problem in which, given an infeasible system of linear inequalities, one wishes to determine a largest feasible subsystem. The focus is on the version with bounded variables that naturally arises in several fields of application. To tackle this NP-hard problem, we propose a simple but efficient two-phase relaxation-based heuristic. First a feasible subsystem is derived from a relaxation (linearization) of an exact continuous bilinear formulation, and then a smaller subproblem is solved to optimality in order to establish whether other inequalities can be added to the current feasible subsystem while preserving feasibility. Computational results, reported for several classes of instances arising, among others, from classification and telecommunication applications, indicate that our method compares well with one of the best available heuristic and with a state-of-the-art exact algorithm.

Keywords: Linear inequality system, maximum feasible subsystem, bilinear formulation, linearization, variable fixing

1 Introduction

We consider the Maximum Feasible Subsystem problem (Max FS): for an infeasible linear system $A\mathbf{x} \geq \mathbf{b}$ with a real matrix $A \in \mathbb{R}^{m \times n}$ and a real vector $\mathbf{b} \in \mathbb{R}^m$, find a feasible subsystem containing the largest number of inequalities. If the focus is on violated inequalities, one may alternatively minimize the number of inequalities that must be deleted to make the resulting subsystem feasible [5]. These two complementary versions of the problem are clearly equivalent as far as optimal solutions are concerned.

The Max FS problem has a number of relevant applications in a variety of fields including computational biology [30], image and signal processing [6, 28], operations research [13, 14, 22, 32], radiation therapy [24], political science [23], statistical discriminant analysis and machine learning, see e.g. [1, 8, 27, 31]. Interest in a special version of Max FS can be traced back to early work on linear discriminant analysis [38]. During the past twenty years substantial attention has been devoted to the problem of designing an optimal linear classifier (i.e, given two groups of observations represented by points in a feature space, find a hyperplane

that correctly separates the maximum number of points) in machine learning and operations research. Another significant line of research on MAX FS is related to the problem of coping with infeasibility in linear programming, see e.g. [13, 14, 15, 20, 22, 32]. When a linear program is infeasible due to errors in the data or in the automatic constraint generation process, it is indeed valuable to identify a maximum feasible subset of constraints. During the past few years several challenging problems arising in other interesting fields of application have been formulated in terms of MAX FS: the fundamental problem of detecting lines in digital images [6, 28], the problem of selecting transmitters emission powers when planning digital video broadcasting [34], and that of modeling the energy function underlying the folding of proteins [30].

MAX FS is a difficult problem: it is NP-hard not only to solve optimally [11, 36] but also to approximate [4]. While a simple algorithm is guaranteed to provide a feasible subsystem with at least half the number of inequalities contained in a largest feasible subsystem, the problem does not admit a polynomial-time approximation scheme, unless P = NP [4]. It is worth pointing out that MAX FS plays for linear inequality systems a similar role as the well-known problem of MAX SAT (given a set of Boolean clauses, find a truth assignment for the Boolean variables which satisfies a maximum number of clauses [19]) for systems of Boolean clauses. Note however that, since linear system feasibility can be checked in polynomial time, the structure of MAX FS differs substantially from that of MAX SAT.

The focus in this paper is on the version of MAX FS in which all variables are bounded and these bounds are mandatory. Since any variable can be expressed as the difference of two non-negative variables, we can assume without loss of generality that all variables are non-negative and bounded above. Instances of MAX FS with bounded variables naturally arise in several fields of application such as, for example, in planning terrestrial video broadcasts [34] with the problem of deciding the emission power of a set of transmitters in order to maximize territory (population) coverage. But the approach and algorithm we propose can also be applied when all but one of the variables are bounded. In discriminant analysis, for instance, the problem of designing an optimal linear classifier can be formulated in terms of MAX FS with a single unbounded variable [35].

Besides a series of linear programming approximate formulations (see e.g. [9, 18]) and some mixed-integer exact formulations (see e.g. [21, 37]), several heuristics and exact algorithms have been proposed for tackling versions of MAX FS. This includes, for instance, variants of the relaxation method for solving feasible systems of linear inequalities [1, 3, 17] also studied in the machine learning literature, filtering heuristics based on a greedy-like strategy and linear programming [12, 13], and bilinear and concave formulations that are tackled with Frank-Wolfe-type methods [8, 26, 27]. In the filtering heuristic described in [13], at each iteration a single relation is permanently deleted from the current infeasible system until the remaining subsystem is feasible. The relations to be dropped are selected according to information obtained by solving elastic programming formulations, which are closely related to phase I in linear programming. This method compares favourably with the parametric method proposed in [8], which was among the best methods for designing optimal linear classifiers.

Most exact solution approaches to MAX FS rely on the concept of Irreducible Infeasible Subsystem (IIS). An IIS is a minimal infeasible subsystem, i.e., a subsystem such that all its proper subsystems are feasible. Since an infeasible linear system may have an exponential number of IISs, finding a maximum feasible subsystem amounts to determine a minimum number of inequalities to be deleted so as to make the resulting system feasible (at least one inequality from each IIS). The exact algorithms that have been proposed for MAX FS include a method based on a partial set covering formulation where IISs are dynamically

generated [31, 32], a first Branch-and-Cut algorithm [7, 33] and a polyhedral method using combinatorial Benders' cuts [16]. For a survey of work on the MAX FS problem up to the end of 2002 the reader is referred to [2].

The paper is organized as follows. In the next section, we consider an exact formulation of MAX FS as a continuous nonlinear program with a linear objective function subject to bilinear constraints and propose a relaxation (linearization) for it. In Section 3 we present a simple two-phase heuristic in which a first subsystem is derived from an optimal solution of the above relaxation and then a subproblem is solved to optimality to establish whether other inequalities can be added to the current feasible subsystem while preserving feasibility. In Section 4 we report computational results obtained for randomly generated and structured instances arising from the above-mentioned classification and telecommunication applications.

2 Formulation and linear relaxation

By introducing a binary variable y_i, with $1 \leq i \leq m$, for each inequality of the infeasible system under consideration, MAX FS with bounded variables clearly admits the following Mixed Integer Programming (MIP) formulation:

$$\begin{array}{lrl} \max & \sum_{i=1}^{m} y_i & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij}x_j + M(1-y_i) \geq b_i & i = 1,\ldots,m \\ & l_j \leq x_j \leq u_j & j = 1,\ldots,n \\ & y_i \in \{0,1\} & i = 1,\ldots,m, \end{array} \tag{1}$$

where M is a large enough constant, and l_j and u_j are the lower and upper bound for x_j. Each binary variable y_i is equal to 1 if the corresponding inequality is satisfied and to 0 otherwise.

Unfortunately such big-M formulations are often beyond the reach of state-of-the-art MIP solvers even for some medium size instances. Numerical difficulties frequently occur due to the badly conditioned subproblems obtained by linear relaxation. Choosing the value of the parameter M can be very delicate even when all variables are bounded. Too large values of M may lead to a highly ill-conditioned linear program, while too small values may not guarantee that the solution $\mathbf{x}$ found actually satisfies all inequalities for which the corresponding binary variable y_i is equal to 1.

2.1 Bilinear formulation

As observed in [2], the MAX FS problem can be formulated as a nonlinear mathematical program with a linear objective function, bilinear constraints and real variables, a so-called linear program with equilibrium constraints (LPEC). The problem can be written as:

$$\begin{array}{lrl} \max & \sum_{i=1}^{m} y_i & \\ \text{s.t.} & y_i \sum_{j=1}^{n} a_{ij}x_j \geq y_i b_i & i = 1,\ldots,m \\ & l_j \leq x_j \leq u_j & j = 1,\ldots,n \\ & 0 \leq y_i \leq 1 & i = 1,\ldots,m. \end{array} \tag{2}$$

Note that, even though the variables y_i are continuous, in any optimal solution they can only take $0-1$ values. Indeed, for any nonzero (strictly positive) variable y_i the corresponding i-th inequality is satisfied

and, since the sum of the y_i's is maximized, each nonzero variable y_i is as large as possible, that is, is equal to 1.

We now present a linearization of this nonlinear continuous exact formulation which will be used in Section 3 to devise an efficient heuristic algorithm for MAX FS with bounded variables.

2.2 Linearization of the bilinear formulation

By substituting in (2) each bilinear term $y_i x_j$ with a new variable z_{ij}, we have the following linear program:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^{m} y_i && && (3)\\
\text{s.t.} \quad & \textstyle\sum_{j=1}^{n} a_{ij} z_{ij} \geq y_i b_i && i = 1, \ldots, m && (4)\\
& l_j \leq x_j \leq u_j && j = 1, \ldots, n && (5)\\
& 0 \leq y_i \leq 1 && i = 1, \ldots, m && (6)\\
& z_{ij} \geq 0 && j = 1, \ldots, n, && (7)
\end{aligned}$$

which needs to be further constrained so that the variables z_{ij} be really equivalent to the bilinear terms $y_i x_j$. Clearly, if $y_i \in \{0, 1\}$ then we have $z_{ij} = y_i x_j$ if and only if the following two conditions hold for all $i = 1, \ldots, m$:

$$y_i = 0 \Longrightarrow z_{ij} = 0 \qquad \text{for all } j = 1, \ldots, n \tag{$C1$}$$

$$y_i = 1 \Longrightarrow z_{ij} = x_j \qquad \text{for all } j = 1, \ldots, n. \tag{$C2$}$$

Since each variable x_j belongs to an interval $[l_j, u_j]$, we can assume without loss of generality that we have $l_j = 0$ for $j = 1, \ldots, n$. If $u_j < 0$ we can indeed replace x_j with $-x_j$ which takes values in $[-u_j, -l_j]$. If $u_j > 0$ and l_j is nonzero, it suffices to apply the following simple variable substitution. If $l_j < 0$ then $x_j = x_j^+ - x_j^-$, where the positive and negative parts of x_j satisfy $0 \leq x_j^+ \leq u_j$ and, respectively, $0 \leq x_j^- \leq -l_j$. If $l_j > 0$ then $x_j = x_j^+ + l_j$ with $0 \leq x_j^+ \leq u_j - l_j$.

Under this nonnegativity assumption, conditions $(C1)$ and $(C2)$ need to be imposed only for the pair of indices i and j such that $a_{ij} < 0$. For all i and j such that $a_{ij} \geq 0$, replacing z_{ij} with x_j in constraints (4) certainly helps to satisfy the inequalities since $x_j \geq 0$. Thus, variables z_{ij} are not defined when $a_{ij} \geq 0$ and constraints (4) are replaced by:

$$\sum_{j : a_{ij} < 0} a_{ij} z_{ij} + \sum_{j : a_{ij} \geq 0} a_{ij} x_j \geq y_i b_i \qquad i = 1, \ldots, m. \tag{8}$$

Since we want $z_{ij} = y_i x_j$ and both the variables y_i and x_j are nonnegative, the variables z_{ij} can be bounded by imposing $z_{ij} \geq 0$ for all $i = 1, \ldots, m,\ j = 1, \ldots, n$.

While conditions $(C1)$ can be clearly expressed by the following group of at most nm constraints:

$$z_{ij} \leq u_j y_i, \qquad i = 1, \ldots, m,\ j = 1, \ldots, n,\ \text{s.t. } a_{ij} < 0, \tag{9}$$

conditions $(C2)$ can be enforced by the set of linear constraints:

$$\begin{aligned}
z_{ij} &\leq x_j && i = 1, \ldots, m,\ j = 1, \ldots, n,\ \text{s.t. } a_{ij} < 0 && (10)\\
x_j - u_j (1 - y_i) &\leq z_{ij} && i = 1, \ldots, m,\ j = 1, \ldots, n,\ \text{s.t. } a_{ij} < 0. && (11)
\end{aligned}$$

Constraints (10) holds since we want $z_{ij} = y_i x_j$ and we have $y_i \leq 1$. Moreover, when $y_i = 1$ constraints (11) become $z_{ij} \geq x_j$ and together with constraints (10) they ensure $z_{ij} = x_j$. Whereas when $y_i = 0$, constraints (11) is redundant since $x_j \leq u_j$ and $z_{ij} \geq 0$.

Thus adding (9), (10)-(11) to (3)-(7) and replacing (4) with (8), we obtain the following linearization of the bilinear formulation (2):

$$\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^{m} y_i & & (12)\\
\text{s.t.} \quad & \textstyle\sum_{j:a_{ij}<0} a_{ij} z_{ij} + \sum_{j:a_{ij}\geq 0} a_{ij} x_j \geq y_i b_i & i = 1,\dots,m & (13)\\
& z_{ij} \leq u_j y_i, & i = 1,\dots,m,\ j = 1,\dots,n,\ \text{s.t. } a_{ij} < 0 & (14)\\
& z_{ij} \leq x_j & i = 1,\dots,m,\ j = 1,\dots,n,\ \text{s.t. } a_{ij} < 0 & (15)\\
& x_j - u_j (1 - y_i) \leq z_{ij} & i = 1,\dots,m,\ j = 1,\dots,n,\ \text{s.t. } a_{ij} < 0 & (16)\\
& l_j \leq x_j \leq u_j & j = 1,\dots,n & (17)\\
& 0 \leq y_i \leq 1 & i = 1,\dots,m & (18)\\
& z_{ij} \geq 0 & i = 1,\dots,m,\ j = 1,\dots,n,\ \text{s.t. } a_{ij} < 0. & (19)
\end{aligned}$$

Note that, unlike for (2), an optimal solution of this formulation is not guaranteed to have all variables y_i with $0-1$ values. Therefore (12)-(19) is not an exact formulation of the MAX FS problem with bounded variables but a relaxation. In order to ensure exactness, it suffices to impose that $y_i \in \{0,1\}$ for $i = 1,\dots,m$.

In fact, the same formulation (12)-(19) can also be derived by considering the McCormick convex relaxation [29] of the nonconvex constraints $z_{ij} = y_i x_j$ for $i = 1,\dots,m$ and $j = 1,\dots,n$:

$$\begin{aligned}
z_{ij} &\geq x_j^L y_i + y_i^L x_j - x_j^L y_i^L & (20)\\
z_{ij} &\geq x_j^U y_i + y_i^U x_j - x_j^U y_i^U & (21)\\
z_{ij} &\leq x_j^U y_i + y_i^L x_j - x_j^U y_i^L & (22)\\
z_{ij} &\leq x_j^L y_i + y_i^U x_j - x_j^L y_i^U, & (23)
\end{aligned}$$

where x_j^L, x_j^U, y_i^L and y_i^U respectively denote a lower and upper bound on variables x_j and y_i (see for example also [25]). In our case, since we have $x_j^L = y_i^L = 0$, $x_j^U = u_j$, $y_i^U = 1$, constraint (20) is equivalent to (19), (21) to (16), (22) to (14) and (23) to (15).

It is worth noting that conditions $(C1)$ can alternatively be imposed by the following set of m linear constraints:

$$\sum_{j:a_{ij}<0} z_{ij} \leq \sum_{j:a_{ij}<0} u_j y_i, \qquad i = 1,\dots,m, \tag{24}$$

because, for any given i, all variables z_{ij} may simultaneously take the corresponding upper bound value u_j. Although the resulting relaxation (linearization) is more compact than (12)-(19), it turns out to be a weaker formulation. This can be verified by observing that for each $i = 1,\dots,m$ the constraint (24) is obtained by summing over j the corresponding constraints (14), and that there exists at least a point which satisfies (13)-(19) with (14) replaced by (24) but does not satisfy (13)-(19). Without loss of generality we can assume that there exists a row $\hat{\imath}$ of matrix A with at least two negative elements and at least one positive element. If in every row of A all elements are either positive or negative then the MAX FS problem can be solved without introducing variables z_{ij}. If in each row of A there is only one negative element then constraints (14) and (24) are equivalent. Now it is easy to see that there always exists a value α with $0 < \alpha < u_{\hat{\jmath}}$ satisfying the

following inequality

$$\alpha a_{\hat{\imath}\hat{\jmath}} + \sum_{j:a_{\hat{\imath}j}\geq 0} a_{\hat{\imath}j}u_j \geq \frac{\alpha b_{\hat{\imath}}}{\sum_{j:a_{\hat{\imath}j}<0} u_j}. \tag{25}$$

Therefore the point $(\mathbf{z}, \mathbf{y}, \mathbf{x})$ defined as

$$z_{ij} = \begin{cases} \alpha & \text{if } i = \hat{\imath} \text{ and } j = \hat{\jmath}, \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

$$y_i = \begin{cases} \frac{\alpha}{\sum_{j:a_{\hat{\imath}j}<0} u_j} & \text{if } i = \hat{\imath}, \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

and

$$x_j = u_j,\ j = 1, \ldots, n \tag{28}$$

violates constraint (14) for $i = \hat{\imath}$ and $j = \hat{\jmath}$ whereas satisfies constraints (24) for each $i = 1, \ldots, m$ and every remaining constraint of (13)-(19).

3 A two-phase relaxation-based heuristic

In this section we propose a simple but efficient heuristic for tackling Max FS instances with bounded variables. The idea is to exploit the optimal solution of a relaxation of the problem, like the linearization (12)-(19), to determine a first feasible subsystem. Then a smaller subproblem is solved optimally in order to establish whether other inequalities can be added to the above feasible subsystem while preserving feasibility.

For $i = 1, \ldots, m$, let $\tilde{y}_i \in [0, 1]$ be the value of variable y_i in an optimal solution of the linear relaxation and let $I_1 := \{i : \tilde{y}_i = 1, i = 1, \ldots, m\}$.

Observation: The linear subsystem composed of the inequalities $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ with $i \in I_1$ is feasible.

Indeed, when $\tilde{y}_i = 1$ the inequality $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ is imposed in the linear relaxation, and hence the corresponding vector $\tilde{\mathbf{x}}$ satisfies all the inequalities with $i \in I_1$.

Experimentally we have observed for both the relaxation (12)-(19) of the bilinear formulation (2) and the linear relaxation of the big-M formulation (1) that:

- the inequalities corresponding to $\tilde{y}_i < 1$ are not always inconsistent with the subsystem indexed by I_1, that is this feasible subsystem is not *maximal* with respect to inclusion;
- since in the optimal solution of the relaxation under consideration, the variables y_i can achieve any value $\tilde{y}_i \in [0, 1]$ independently on the value they assume in the exact formulation, I_1 could also contain the indices of inequalities that do not belong to any *maximum* feasible subsystem of Max FS.

In spite of the last observation, we propose to tackle this $\mathcal{NP}$-hard problem by considering the subsystem $(S1)$ composed of all inequalities indexed by I_1 and by trying to identify the maximum number of remaining inequalities (that is with indices in $\{1, \ldots, m\} \setminus I_1$) that added to $(S1)$ yields a larger feasible subsystem.

Two-phase algorithm
Phase 1:
1. Solve a relaxation of the MAX FS obtaining a solution $\tilde{\mathbf{y}}$;
2. Determine $I_1 = \{i : \tilde{y}_i = 1, i = 1, \ldots, m\}$;
Phase 2:
3. Solve an exact formulation of MAX FS fixing $y_i = 1$ for all $i \in I_1$.

Notice that in Step 2 all the tests $\tilde{y}_i = 1$ should be performed as $\tilde{y}_i \geq 1 - \epsilon$, where ϵ is a small enough constant that reflects machine precision. If ϵ is overestimated, the inequalities indexed by I_1 may correspond to an infeasible subsystem. Clearly, the closer ϵ is to 0 the smaller the feasible subsystem indexed by I_1 will be.

In Step 3 we can use an exact formulation of MAX FS such as (1). Indeed, after fixing all variables y_i with $i \in I_1$ to 1, the number of free variables y_i in this formulation is generally moderate and a state-of-the-art commercial MIP solver almost always yields an optimal solution of the resulting linear MIP subproblem in reasonable computing time. Since the subproblem amounts to identifying the maximum number of inequalities that can be added to the inequalities indexed by I_1 so as to obtain a larger feasible subsystem, our two-phase algorithm can be viewed as a *variable fixing* approach where an optimal solution of a relaxation of the bilinear exact formulation (2) of MAX FS is used to fix part of the y_i variables.

4 Experimental results

In the computational experiments we consider the two-phase algorithm with in Phase 1 the linearization (12)-(19) of the bilinear formulation and in Phase 2 the big-M exact formulation (1) of MAX FS. For the sake of comparison, we also consider a Phase 1 with the linear relaxation of (1). The results obtained with our two-phase relaxation-based algorithm are compared with those provided by the *Filtering* heuristic (Algorithm 1 in [13]), the exact big-M formulation (1) tackled with Cplex 8.1 MIP solver and a recent exact polyhedral method based on combinatorial Benders' cuts (*CBC*) [16].

4.1 Instances

We tested the performance of the two-phase algorithm on the following four groups of instances.

Random: A set of random instances presented in the doctoral thesis [33]. Three random instances are generated, with different random seeds, for each choice of the number of inequalities (rows) and variables (columns). The matrix A and the vector $\mathbf{b}$ have almost full density, and all their elements are integer values in the interval $[-100, 100]$. Table 1 indicates the average size and the number of non-zero elements of each group consisting of three instances. For comparison purposes we bounded the variables in $[0, 1]$.

CBC-ML: A group of linear classification problems from the UCI Machine Learning repository [10]. In linear discriminant analysis, the goal is to partition a group of observations into two classes using a separating hyperplane (a linear classifier) so as to maximize the number of observations that are correctly classified. In the MAX FS formulation of this problem, the variables are the normal vector

Table 1: Random Instances: characteristics. Each random$m$xn group is composed of three instances with m columns and n rows.

Group	Avg. Non-zeros	Group	Avg. Non-zeros
random10x20	198	random20x50	989
random10x30	299	random20x60	1185
random10x40	396	random20x70	1384
random10x50	495	random20x80	1583
random10x60	594	random5x100	494
random10x70	693	random5x10	49
random10x80	792	random5x20	99
random15x30	447	random5x30	149
random15x40	594	random5x40	198
random15x50	741	random5x50	248
random15x60	890	random5x60	298
random15x70	1038	random5x70	346
random15x80	1186	random5x80	395
random20x40	792	random5x90	445

coefficients of the separating hyperplane and the threshold (the constant term of the hyperplane) [1, 8, 26, 31]. As observed in [35], we can assume without loss of generality that all the coefficients of the hyperplane are bounded in $[-1, 1]$ except for the threshold, since the direction of an hyperplane is univocally determined up to a multiplicative factor. Applying the variable substitution described in Section 2.2, we have bounded all the variables in $[0, 1]$, except the free variable that was artificially bounded in $[-10^5, +10^5]$. The instances mentioned in Table 2 were modified in [35] and used in [16] for the experimental evaluation of the CBC method. The actual values of the big-M constants are the same as in [16].

ML: A different group of instances from the UCI Machine Learning repository. These *machine learning* instances were used in previous works as a testbed for comparing exact and heuristic methods for Max FS. Their characteristics are summarized in Table 3. The variables were bounded like for the CBC-ML instances.

DVB: A set of instances arising in planning *Digital Video Broadcasts*, in particular when selecting the emission power of a set of transmitters in order to maximize the territory (or population) coverage [34]. Compared to the other testbeds, the DVB instances have a larger number of rows and columns, but very low density (average 3% and standard deviation 1.4%). For all instances listed in Table 4 we looked for a maximum feasible subsystem. The large difference between the values of the coefficients, ranging between 10^{-11} and 10^{11}, causes serious numerical difficulties. This accounts for the fact that the implementation of the *Filtering* heuristic based on MINOS [13] faced fatal errors on these instances. The variables for this testbed were bounded in $[0, 1]$ as for the Random instances.

4.2 Implementation and parameter settings

The computational experiments were conducted on a dual-processor PC with two Intel Xeon 2.80 Ghz CPUs with a 512 KB L2 cache, 2 GB of physical memory and 4 GB of virtual memory limit. The system was running Linux 2.4.18-14smp and was configured with four virtual processors. We used the commercial solver ILOG-Cplex 8.1 and the AMPL ver. 200220528 modelling environment. Only for solving the CBC-ML instances with the exact big-M formulation we used the original *.lp* files; in all other cases the Cplex solver

was called from the AMPL interface. The algorithms were run on the testbeds ML, CBC-ML, and Random with default settings for both Cplex and AMPL, except for the Cplex option `integrality=1e-09`. For the numerically unstable DVB instances, the primal simplex with the Cplex options `presolve=0` and `scale=1` performed in a stable way on most instances. The maximum CPU time limit was always set to 10000 seconds. The actual feasibility of all solutions (subsystems found) was double checked with default Cplex settings.

Chinneck's Algorithm 1 [13] was implemented in AMPL like our two-phase relaxation-based method. Some of the reported CPU times may thus be slightly higher than those obtained with more efficient implementations. For the exact CBC method, we used a C++ implementation provided by Codato and Fischetti. Since the actual code runs only on the CBC-ML instances, this method could not be tested on the other groups. Moreover, the code being based on Cplex callable library, the CPU times reported for the CBC method are not directly comparable with those reported for the other methods implemented in AMPL.

4.3 Computational results

The experimental results for the instances of Section 4.1 are reported in Tables 5, 6, 7 and 8. The following conventions are used across the tables:

- *exact-bigM* stands for Cplex 8.1 MIP solver applied to the exact big-M formulation (1),
- *CBC* for the polyhedral method based on the Combinatorial Benders' Cuts [16],
- *2-ph-bigM* for the two-phase algorithm with the big-M formulation (1) in Phase 1,
- *2-ph-bilinear* for the two-phase algorithm with the linearization (12)-(19) in Phase 1,
- *Filtering* for the heuristic Algorithm 1 of [12],
- *FS* denotes the number of inequalities in the selected feasible subsystem,
- *CPU* represents the computing time in seconds;
- *V (ub: U)* indicate for an exact big-M formulation that cannot be solved to optimality, the number V of inequalities in the largest feasible subsystem found by Cplex and the upper bound U on the optimal value,
- *V (< B)* denote for the two-phase algorithm the number V of inequalities in the largest feasible subsystem and the upper bound B found by Cplex when solving the exact big-M formulation of Phase 2, where the y variables are fixed according to the optimal solution of the relaxation in Phase 1.

Note that in the last item B is not an upper bound on the optimal value of MAX FS but only on the cardinality of the maximal feasible subsystem containing all the inequalities indexed by I_1. In all tables, the optimal solutions are emphasized in boldface.

According to Table 5 the Random instances are quite easy, since the Cplex 8.1 MIP solver applied to the exact big-M formulation always finds an optimal solution (a largest feasible subsystem) within 1 hour and for most of them even within a few tens of seconds. Note that the solutions provided by *2-ph-bilinear* are at least as good as those yielded by *2-ph-bigM*, except for *random10x30*, and for many instances they are significantly better than the solutions obtained with *Filtering*. In particular, if we focus on the most difficult

instances, that is on those with at least 70 rows and 10 columns, *2-ph-bilinear* always yields the largest feasible subsystem. Remarkably, the best results are achieved on the largest groups of instances, *random20x70* and *random20x80*, for which it improves *Filtering* results and outperforms *2-ph-bigM*. As reported in the last row of the table, the relative average gap for *2-ph-bilinear* (i.e., the average ratio between the number of inequalities in the feasible subsystem found by the algorithm and that in a largest feasible subsystem) is 1.12% against 2.31% for the *Filtering* heuristic.

It is worth pointing out that the number of variables that are fixed by *2-ph-bilinear* at the end of the first phase is always smaller than that fixed by *2-ph-bigM*. Although one may argue that fixing an excessive number of variables could prevent the second phase from selecting a good feasible subsystem, this is not always the case as we shall see for the remaining testbeds. The experimental results indicate that the linearization of the bilinear formulation, when compared to a Phase 1 with the linear relaxation of the big-M formulation, leads to a "smarter" choice of the initial feasible subsystem, namely of the index set I_1. Before considering the remaining testbeds, we observe that the CPU time of *2-ph-bilinear* is competitive with that of *Filtering*, while *2-ph-bigM* is the best choice for obtaining a quick solutions. Furthermore, the computing times required by *2-ph-bilinear* grow rather slowly when only a few columns are present.

Let us now consider the CBC-ML instances. According to Table 6 this set is considerably more challenging than the previous one, since an optimal solution is obtained using the big-M formulation only for the first 24 instances. For most of the remaining instances, the best feasible solution yielded by *exact-bigM* is very close to the optimal solution obtained with *CBC*. Moreover, when *CBC* cannot find any solution since the memory limit is exceeded (*Flags-169*, *Horse-colic-253*, *Horse-colic-185*), Cplex MIP solver finds a best feasible solution which differs from the upper bound by at most 8 inequalities, except for the difficult *Solar-flare-1066*. For the sake of comparison, we report the original results for the last 4 instances presented in [16]. The difference in the results should be accounted for by the tuning of *CBC* parameters. Both *2-ph-bigM* and *2-ph-bilinear* find the optimal solution for most of the instances, but the computing times of the latter are higher. It is remarkable that, even though *CBC* is based on a sophisticated polyhedral method, our simple two-phase approach yields solutions with comparable quality and, for *2-ph-bigM*, within lower computing times. Similarly, the two-phase relaxation-based algorithm compares favourably with *Filtering* in terms of solution quality: the relative average gap 0.34% against 0.87% of the latter, excluding *Solar-flare-1066*. However, the computing times of the first phase of *2-ph-bilinear* suggest that the linearization of the bilinear formulation can be much more time-consuming than both *2-ph-bigM* and *Filtering*. As for the Random instances, the number of variables fixed after the first phase of the two-phase algorithm does not affect the quality of the final solution. For these instances, *2-ph-bilinear* and *2-ph-bigM* achieve similar results, in spite of the fact that the number of variables fixed by *2-ph-bigM* is in general smaller than those fixed by *2-ph-bilinear* (the relative average gap of Phase 1 is 34.50% for the former and 18.76% for the latter).

The results obtained for the *Bv-os-282* instance call for an explanation. Both *CBC* and *2-ph-bigM* provide a feasible subsystem with 276 inequalities whereas *Filtering* and *2-ph-bilinear* find a solution with 277 inequalities. This discrepancy is due to the limited machine accuracy, which can also affect the double-check that the selected subset of inequalities is actually feasible.

We now turn our attention to the ML group, see Table 7. Note that for only three instances *exact-bigM* is able to find the optimal solution within the 10000 seconds time limit. The interest behind this testbed and the CBC-ML set is that, while the presence of a free variable may in principle affect the final results of our two-phase heuristic, it does not happen in practice, as the solution quality is comparable to that of the

Filtering method (both have a relative average gap of approximately 7%). Although *2-ph-bilinear* requires higher computing times, it is useful on two hard instances such as *bupa* and *pima* since it provides better solutions than *2-ph-bigM*.

Finally, we consider the results for the challenging DVB instances reported in Table 8. Since the variables are all bounded in $[0, 1]$, the linearization of the bilinear formulation is particularly adequate. In fact, *2-ph-bilinear* yields the smallest relative average gap obtained in Phase 1, namely 7.55%. For simplicity of exposition, we divided the instances into three groups according to the size of the instances. Although *2-ph-bigM* and *2-ph-bilinear* provide comparable results, the two-phase approach dominates the *Filtering* method for all instances except for *dvb1*. Note that, for instance *mfs_UHF_P4_4*, Cplex 8.1 MIP solver is not able to find an optimal solution of the *exact-bigM* formulation within the time limit but the *2-ph-bigM* finds a slightly better feasible solution just in 5 seconds. While the CPU time requirements of our heuristics are moderate, those of *Filtering* turn out to grow fast with the instance size, since the number of steps of the greedy procedure depends on the number of inequalities to be deleted from the original infeasible system to achieve feasibility. For this set of instances several tens or even thousands of inequalities have to be deleted. Note, however, that the refinements of the *Filtering* heuristic described in [13] may at least partially contrast this drawback but at the price of slightly affecting the solution quality.

Let us now focus on the second group of instances. The advantage of *2-ph-bilinear* with respect to all other methods is remarkable, both in terms of the size of the feasible subsystem and of the cumulative CPU time. For instance *P4_60_19916*, *2-ph-bilinear* finds a feasible subsystem with more than 350 additional inequalities compared to the feasible subsystem provided by *2-ph-bigM*. This suggests that for sparse medium-sized instances *2-ph-bilinear* should be preferred to *2-ph-bigM*, and that the higher computational requirements for the first phase are indeed justified. The instances in the last group are far too large for *2-ph-bilinear* and *2-ph-bigM* is the only viable method. It is noteworthy that the first phase with the linear relaxation of the big-M formulation gives in a few seconds a solution that compares well with the best integer solution found with *exact-bigM*.

Table 9 summarizes, for the four sets of instances, the relative average gaps between the value achieved by the heuristics and the value of an optimal solution (or an upper bound, when no optimal solution has been obtained within the time limit). The two-phase relaxation-based method turns out to compare favourably with the *Filtering* heuristic on all the classes of instances except for the ML instances where the solution quality is comparable. Moreover, *2-ph-bilinear* provides in average better quality solutions than *2-ph-bigM*.

5 Conclusions

We have presented a new two-phase heuristic for the Max FS problem with bounded variables. A feasible subsystem is derived from an optimal solution of a relaxation (linearization) of an exact continuous bilinear formulation and then a smaller instance of Max FS is solved optimally to identify all the other inequalities that are consistent with the above feasible subsystem. The computational complexity of the method does not depend on the number of inequalities that need to be deleted to achieve a feasible subsystem.

Our simple two-phase approach, which can be easily extended to infeasible systems of linear equations, provides in average better quality solutions than the *Filtering* heuristic for several types of Max FS instances in which all variables are bounded. Although the proposed relaxation of the bilinear formulation leads to very

good results, other relaxations can be considered to further improve solution quality or reduce computing times. In particular, we have seen that when a relaxation of the big-M formulation is used in the first phase, execution times are dramatically reduced and the solution quality is not substantially affected. This is of course interesting for large instances that cannot be tackled with other state-of-the-art methods.

The experimental results suggest that our two-phase approach also performs very well for Max FS instances with a single unbounded variable. Indeed, it yields an optimal solution for most linear classification instances of the CBC-ML testbed and it compares well with the recent exact method based on combinatorial Benders' cuts. We leave as an open question if the two-phase relaxation-based approach can also be successfully adapted to tackle instances of Max FS in which all variables are unbounded.

Acknowledgments
The authors would like to thank Gianni Codato and Matteo Fischetti for providing the code of their polyhedral method based on combinatorial Benders' cuts, John Chinneck for useful discussions concerning the AMPL implementation of the filtering heuristic and for trying to run his implementation of *Filtering* on the DVB instances, Marc Pfetsch for the Random instances, Fabrizio Rossi and Stefano Smriglio for the DVB instances.

References

[1] E. Amaldi. *From finding maximum feasible subsystems of linear systems to feedforward neural network design*. PhD thesis, Dep. of Mathematics, EPF-Lausanne, 1994.

[2] E. Amaldi. The maximum feasible subsystems problem and some applications. In A. Agnetis and G. Di Pillo, editors, *Modelli e Algoritmi per l'ottimizzazione di sistemi complessi*. Pitagora Editrice Bologna, 2003.

[3] E. Amaldi, P. Belotti, and R. Hauser. Randomized relaxation methods for the maximum feasible subsystem problem. In *Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, Proceedings*, LCNS. Springer, 2005. to appear.

[4] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147:181–210, 1995.

[5] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[6] E. Amaldi and M. Mattavelli. The MIN PFS problem and piecewise linear model estimation. *Discrete Appl. Math.*, 118:115–143, 2002.

[7] E. Amaldi, M. E. Pfetsch, and L. E. Trotter Jr. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. *Math. Programming A*, 95:533–554, 2003.

[8] K. P. Bennett and E. Bredensteiner. A parametric optimization method for machine learning. *INFORMS Journal on Computing*, 9:311–318, 1997.

[9] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

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

[11] N. Chakravarti. Some results concerning post-infeasibility analysis. *Eur. J. Oper. Res.*, 73:139–143, 1994.

[12] J. Chinneck. An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem. *Annals of Mathematics and Artificial Intelligence*, 17:127–144, 1996.

[13] J. Chinneck. Fast heuristics for the maximum feasible subsystem problem. *INFORMS Journal on Computing*, 13:210–213, 2001.

[14] J. W. Chinneck. Computer codes for the analysis of infeasible linear programs. *J. Oper. Res. Soc.*, 47:61–72, 1996.

[15] J. W. Chinneck. Finding a useful subset of constraints for analysis in an infeasible linear program. *INFORMS J. Comput.*, 9(2):164–174, 1997.

[16] G. Codato and M. Fischetti. Combinatorial Benders' cuts. In G. L. Nemhauser and D. Bienstock, editors, *Integer Programming and Combinatorial Optimization, 10th International IPCO Conference, New York, NY, USA, June 7-11, 2004, Proceedings*, volume 3064 of *LNCS*, pages 178–195. Springer, 2004.

[17] M. Frean. A "thermal" perceptron learning rule. *Neural Comp.*, 4(6):946–957, 1992.

[18] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. *Eur. J. Oper. Res.*, 7:44–60, 1981.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.

[20] J. Gleeson and J. Ryan. Identifying minimally infeasible subsystems of inequalities. *ORSA J. Comput.*, 2(1):61–63, 1990.

[21] F. Glover. Improved linear and integer programming models for discriminant analysis. *Decision Sciences*, 21:771–785, 1990.

[22] H. J. Greenberg and F. H. Murphy. Approaches to diagnosing infeasible linear programs. *ORSA Journal on Computing*, 3:253–261, 1991.

[23] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoret. Comput. Sci.*, 6:93–107, 1978.

[24] E. K. Lee, R. J. Gallagher, and M. Zaider. Planning implants of radionuclides for the treatment of prostate cancer: An application of MIP. *Optima*, 61:1–7, 1999.

[25] L. Liberti. Linearity embedded in nonconvex problems. *Journal of Global Optimization*, To appear.

[26] O. L. Mangasarian. Misclassification minimization. *J. of Global Optimization*, 5(4):309–323, 1994.

[27] O. L. Mangasarian. Machine learning via polyhedral concave minimization. In H. Fischer et al., editor, *Applied Mathematics and Parallel Computing*, pages 175 – 188. Physica-Verlag, 1996.

[28] M. Mattavelli, V. Noel, and E. Amaldi. Fast line detection algorithms based on combinatorial optimization. In *Proceedings of IWVF*, LCNS 2059, pages 410–419. Springer, 2001.

[29] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i – convex underestimating problems. *Mathematical Programming*, (10):146–175, 1976.

[30] J. Meller, M. Wagner, and R. Elber. Solving huge linear programming problems for the design of protein folding potentials. *Math. Programming B*, 101:301–318, 2004.

[31] M. Parker. *A set covering approach to infeasibility analysis of LP problems and related issues.* PhD thesis, Dep. of Mathematics, University of Colorado at Denver, 1995.

[32] M. Parker and J. Ryan. Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 17:107–126, 1996.

[33] M. Pfetsch. *The Maximum Feasible Subsystem Problem and Vertex-Facet Incidences of Polyhedra.* PhD thesis, Technischen Universität Berlin, 2002.

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

[35] P. A. Rubin. Solving mixed integer classification problems by decomposition. *Annals of Operations Research*, 74:51–64, 1997.

[36] J. Sankaran. A note on resolving infeasibility in linear programs by constraint relaxation. *Oper. Res. Lett.*, 13:19–20, 1993.

[37] A. Stam and E. A. Joachimsthaler. A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. *Eur. J. Oper. Res.*, 46:113–122, 1990.

[38] R. E. Warmack and R. C. Gonzalez. An algorithm for optimal solution of linear inequalities and its application to pattern recognition. *IEEE Transactions on Computers*, 22:1065–1075, 1973.

Table 2: CBC-ML Instances: characteristics.

Instance	Rows	Columns	Non-zeros
Chorales-116	116	6	380
Balloons76	76	5	106
BCW-367	367	10	3656
BCW-683	683	10	6830
WPBC194	194	34	6509
Breast-Cancer-400	400	18	495
Glass-163	163	10	1362
Horse-colic-151	151	27	3735
Iris-150	150	5	700
Credit-300	300	15	4047
Lymphography-142	142	18	2556
Mech-analysis-107	107	8	739
Mech-analysis-137	137	7	757
Monks-tr-122	122	6	732
Pb-gr-txt-198	198	10	1980
Pb-pict-txt-444	444	10	4440
Pb-hl-txt-277	277	10	2770
Postoperative-88	88	8	659
Bv-os-282	282	18	5039
Opel-Saab-80	80	18	1433
Bus-Van-437	437	18	7811
HouseVotes84-435	435	16	3813
Water-treat-206	206	38	7751
Water-treat-213	213	38	8016
Chorales-134	134	6	637
Chorales-107	107	6	539
Bridges-132	132	12	1584
Mech-analysis-152	152	8	1033
Monks-tr-124	122	6	744
Monks-tr-115	115	6	690
Solar-flare-323	323	12	2982
BV-OS-376	376	18	6722
BusVan445	445	18	7955
Flags-169	169	29	2530
Horse-colic-253	253	26	6285
Horse-colic-185	183	26	4532
Solar-Flare-1066	1066	12	9817

Table 3: ML Instances: characteristics.

Instance	Rows	Columns	Non-zeros
breast-cancer-wisconsin	683	9	6147
bupa	345	6	2061
glass	214	9	1534
ionosphere	351	34	10513
iris.1	150	4	600
iris.2	150	4	600
new-thyroid	215	5	1071
pima	768	8	5381
wpbc	194	32	6121

Table 4: DVB Instances: characteristics.

Instance	Rows	Columns	Non-zeros
mfs_UHF_P4_1	642	487	3603
mfs_UHF_P4_4	1174	487	19089
mfs_UHF_P4_3	1717	487	22023
P4_60ofP4	6345	487	65538
P4_89ofP4	10338	487	90210
P4_60_19916	18035	487	154845
P4_280ofP4	14678	487	248536
P4	15426	487	304605
P4_487_19916	19916	487	547796

	Branch-and-Cut		*2-ph-bigM*				*2-ph-bilinear*				*2-ph-artificial*				*Filtering*	
	FS	CPU	FS		CPU		FS		CPU		FS		CPU		FS	CPU
Instance			ph.1	ph 2	ph.1	ph.1+2	ph.1	ph.2	ph.1	ph.1+2	ph.1	ph.2	ph.1	ph.1+2		
random5x10	**22**	0.1	17	21	0.1	0.1	17	**22**	0.1	0.1	16	**22**	0.1	0.1	**22**	0.1
random5x20	**45**	1	40	44	0.1	0.1	37	44	0.1	0.1	39	43	0.1	0.1	44	0.1
random5x30	**62**	0.1	57	**62**	0.1	0.1	43	**62**	0.1	0.1	52	**62**	0.1	0.1	61	2
random5x40	**82**	1	74	**82**	0.1	0.1	69	80	0.1	0.1	75	80	0.1	0.1	80	3
random5x50	**104**	5	95	103	0.1	0.1	89	101	0.1	0.1	94	103	0.1	0.1	103	4
random5x60	**122**	7	108	120	0.1	0.1	102	120	0.1	0.1	105	115	0.1	0.1	117	6
random5x70	**139**	20	125	134	0.1	0.1	115	137	0.1	0.1	118	129	0.1	0.1	130	10
random5x80	**160**	22	141	154	0.2	0.2	131	156	0.1	0.1	135	150	0.1	0.1	151	19
random5x90	**174**	36	154	167	0.2	0.2	144	166	0.1	0.1	148	167	0.1	0.1	169	21
random5x100	**191**	74	172	181	0.2	0.2	162	183	0.2	0.2	167	184	0.1	0.1	182	19
random10x20	**52**	0.1	51	**52**	0.1	0.1	47	**52**	0.1	0.1	50	**52**	0.1	0.1	**52**	0.3
random10x30	**69**	0.1	59	68	0.1	0.1	57	**69**	0.1	0.1	60	66	0.1	0.1	**69**	1
random10x40	**92**	3	78	90	0.1	0.1	73	91	0.1	0.1	79	90	0.1	0.1	89	2
random10x50	**110**	21	95	107	0.1	0.1	88	109	0.1	0.1	93	107	0.1	0.1	109	5
random10x60	**126**	89	103	121	0.1	0.1	97	123	0.1	0.1	96	121	0.1	0.1	117	11
random10x70	**143**	443	117	141	0.1	0.1	110	142	0.1	1	114	136	0.1	0.1	137	14
random10x80	**162**	2584	125	156	0.1	0.1	122	159	0.1	2	125	144	0.1	0.1	156	21
random15x30	**77**	2	67	75	0.1	0.1	64	**77**	0.1	0.1	64	74	0.1	0.1	76	2
random15x40	**101**	4	86	100	0.1	0.1	78	100	0.1	0.1	83	99	0.1	0.1	**101**	3
random15x50	**126**	14	109	123	0.1	0.1	104	123	0.1	1	108	118	0.1	0.1	121	5
random15x60	**148**	52	128	145	0.1	1.1	118	147	0.2	2	123	142	0.1	0.1	147	6
random15x70	**168**	301	143	162	0.2	2	132	167	0.2	4	134	163	0.1	0.1	165	12
random15x80	**190**	1782	165	188	0.2	0.2	151	188	0.3	10	156	183	0.1	0.1	185	12
random20x40	**106**	2	103	**106**	0.1	0.1	84	**106**	0.1	0.1	92	104	0.1	0.1	**106**	2
random20x50	**131**	7	112	130	0.1	0.1	100	**131**	0.2	0.2	108	130	0.1	0.1	130	5
random20x60	**153**	37	123	151	0.2	0.2	117	152	0.3	0.3	121	149	0.1	0.1	**153**	6
random20x70	**174**	277	142	172	0.2	2	131	**174**	0.3	3	138	170	0.1	0.1	**174**	8
random20x80	**192**	2267	156	186	0.3	18	136	190	0.4	18	149	187	0.1	0.1	189	47

Table 5: Random instances: computational results. Variables bounded in $[0, 1]$.

	exact-bigM		*CBC*		*2-ph-bigM*		
	FS	CPU	FS	CPU	FS		
Instance					ph.1	ph 2	p
Chorales-116	**92**	3559	**92**	550	46	**92**	0
Balloons76	**66**	7	**66**	0.1	52	**66**	0
BCW-367	**359**	365	**359**	1	333	**359**	0
BCW-683	**673**	6750	**673**	10	643	**673**	0
WPBC-194	**189**	2279	**189**	299	161	**189**	0
Breast-Cancer-400	**376**	71	**376**	0.1	374	**376**	0
Glass-163	**150**	3849	**150**	3	102	**150**	0
Horse-colic-151	**146**	592	**146**	12	128	**146**	0
Iris-150	**132**	463	**132**	56	105	**132**	0
Credit-300	**292**	1836	**292**	2	258	**292**	0
Lymphography-142	**137**	8	**137**	0.4	120	**137**	0
Mech-analysis-107	**100**	10	**100**	0.1	77	**100**	0
Mech-analysis-137	**119**	775	**119**	6	84	**119**	0
Monks-tr-122	**109**	212	**109**	0.1	76	**109**	0
Pb-gr-txt-198	**187**	330	**187**	3	170	**187**	0
Pb-pict-txt-444	**437**	81	**437**	0.1	422	**437**	0
Pb-hl-txt-277	**267**	193	**267**	4	248	266	0
Postoperative-88	**72**	698	**72**	0.1	64	**72**	0
Bv-os-282	276◇	210	276◇	4	261	276◇	0
Opel-Saab-80	**77**	43	**77**	11	50	74	0
Bus-Van-437	**431**	363	**431**	5	410	**431**	0
House-Votes-441	**429**	207	**429**	0.1	405	**429**	0
Water-treat-206	**202**	43	**202**	4	175	**202**	0
Water-treat-213	**208**	621	**208**	21	175	**208**	0
Chorales-134	103(ub : 113)	†	**104**	727	39	**104**	0
Chorales-107	80(ub : 85)	†	**80**	67	31	**80**	0
Bridges-132	108(ub : 121)	†	**109**	136	67	**109**	0
Mech-analysis-152	130(ub : 136)	†	**131**	139	86	**131**	0
Monks-tr-124	100(ub : 104)	†	**100**	56	50	**100**	0
Monks-tr-115	88(ub : 96)	†	**88**	487	25	**88**	0
Solar-flare-323	282(ub : 300)	†	**285**	3	241	284	0
Bv-os-376	367(ub : 369)	†	**368**	125	340	367	0

Table 6: CBC-ML instances: computational results. Hyperplane coefficients bounded in $[-1, 1]$.

	Branch-and-Cut		*exact-bigM*		*2-ph*	
	FS	CPU	FS	CPU	FS	
Instance					ph.1	ph 2
breast-cancer-wisconsin	**672**	43	671(ub : 676)	†	660	**672**
bupa	227(ub : 263)	†	248(ub : 331)	†	166	260($\leq$ 266
glass	178(ub : 193)	†	176(ub : 205)	†	112	172
ionosphere	**345**	2215	345(ub : 347)	†	322	345
iris.1	**125**	1735	**125**	7630	78	124
iris.2	**149**	0.1	**149**	0.4	147	**149**
new-thyroid	**204**	47	**204**	46	181	**204**
pima	614(ub : 718)	†	588(ub : 761)	†	508	615($\leq$ 667
wpbc	181(ub : 187)	†	175(ub : 190)	†	146	178

Legend: †=time limit exceeded)

Table 7: ML instances: computational results. Hyperplane coefficients bounded in $[-1, 1]$

	exact-bigM		*2-ph-bigM*			
	FS	CPU	FS		CPU	
mfs_UHF_P4_1	**538**	0.4	533	537	0.1	0.1
mfs_UHF_P4_4	1050(ub : 1065)	†	1039	1051	4	5
mfs_UHF_P4_3	1532(ub : 1535)	†	1509	1523	2	2
P4_60ofP4_15426_naz	3257(ub : 3526)	†	2849	3115($\leq$ 3235)	10	†
P4_89ofP4_15426_naz	5505(ub : 6158)	†	5189	5408($\leq$ 5456)	4	†
P4_60_19916	6805(ub : 8037)	†	6308	6546($\leq$ 6583)	147	†
P4_280ofP4_15426_naz	9338(ub : 10708)	†	9161	9434($\leq$ 9713)	24	†
P4_15426_naz	12189(ub : 13384)	†	12089	12374($\leq$ 12692)	52	†
P4_48714916	15700(ub : 19650)	†	15584	15940($\leq$ 16628)	234	†
average gap$^{\natural}$			1.73%	**0.95**%		
average gap$^{\natural\natural}$			9.73%	7.26%		

Legend: †= time limit exceeded ♭= no feasible subsystem found
♮=instances solved by acc methods ♮♮=instances solved by two-phase algorithms

Table 8: DVB instances: computational results. Variables bounded in $[0, 1]$.

Testbed	*2-ph-bigM*		*2-ph-bilinear*		*Filtering*
	ph.1	ph. 2	ph.1	ph. 2	
Random	15.54%	3.46%	26.97%	**1.12**%	2.31%
CBC-ML	34.50%	0.35%	18.76%	**0.34%**	0.87%
ML	23.49%	7.22%	29.73%	7.38%	**7.11%**
DVB$^{\flat}$	1.73%	**0.95**%	1.96%	1.14%	1.33%
DVB$^{\natural}$	9.73%	7.26%	7.55%	**6.40**%	-

Legend: $\flat$=instances solved by all methods

$\natural$=instances solved by two-phase algorithms

Table 9: Summary of average gaps