

Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article was published in an Elsevier journal. The attached copy is furnished to the author for non-commercial research and education use, including for instruction at the author's institution, sharing with colleagues and providing to institution administration.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



ELSEVIER

Available online at www.sciencedirect.com

Computers & Operations Research 35 (2008) 1465–1482

computers &
operations
research

www.elsevier.com/locate/cor

A two-phase relaxation-based heuristic for the maximum feasible subsystem problem

Edoardo Amaldi*, Maurizio Bruglieri¹, Giuliano Casale²

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Available online 14 April 2007

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 identify all other inequalities that can be added to the current feasible subsystem while preserving feasibility. Computational results, reported for several classes of instances, arising from classification and telecommunication applications, indicate that our method compares well with one of the best available heuristics and with state-of-the-art exact algorithms.

© 2007 Elsevier Ltd. All rights reserved.

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 $Ax \geq b$ with a real matrix $A \in \mathbb{R}^{m \times n}$ and a real vector $b \in \mathbb{R}^m$, find a feasible subsystem containing the largest number of inequalities [18]. 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 [1]. 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 [2], image and signal processing [3,4], linear programming [5–8], radiation therapy [9], political science [10], statistical discriminant analysis, telecommunications [11] and machine learning, see e.g. [12–15].

MAX FS is a difficult problem: it is NP-hard not only to solve optimally [16,17], but also to approximate [18]. 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

* Corresponding author.

E-mail addresses: amaldi@elet.polimi.it (E. Amaldi), bruglier@elet.polimi.it, maurizio.bruglieri@polimi.it (M. Bruglieri), casale@elet.polimi.it, casale@cs.wm.edu (G. Casale).

¹ Current address: Dipartimento INDACO, Politecnico di Milano, Via Durando 38/a, 20158 Milano, Italy.

² Current address: Department of Computer Science, College of William and Mary, 23187-8795 Williamsburg, VA, USA.

$P = NP$ [18]. 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. The complexity results mentioned above for the case with unbounded variables are still valid for the case with bounded variables [18]. Since any variable can be expressed as the difference of two nonnegative variables, we can assume without loss of generality that all variables are nonnegative 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 [11] with the problem of selecting the emission power of a set of transmitters so as 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 [20].

Besides a series of linear programming approximate formulations (see e.g. [21,22]) and some mixed-integer exact formulations (see e.g. [23,24]), 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 [12,25,26], filtering heuristics based on a greedy-like strategy and linear programming [27,5], and bilinear and concave formulations that are tackled with Frank–Wolfe-type methods [13,28,14]. In the filtering heuristic described in [27,5], 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 [13], which was among the best methods for designing optimal linear classifiers.

Most exact solution approaches to MAX FS rely on the concept of the 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 determining 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 [15,8], a first Branch-and-Cut algorithm [29,30] and a polyhedral method using combinatorial Benders' cuts (CBC) [31]. For a survey of work on the MAX FS problem up to the end of 2002 the reader is referred to [32].

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. Bilinear formulation and linearization

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

$$\begin{aligned} \max \quad & \sum_{i=1}^m y_i, \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij}x_j + M(1 - y_i) \geq b_i, \quad i = 1, \dots, m, \\ & y_i \in \{0, 1\}, \quad i = 1, \dots, m, \end{aligned} \tag{1}$$

where a_{ij} denotes the entry (i, j) of matrix A , b_i denotes the i th component of vector \mathbf{b} and M is a large enough constant. 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 0.

Note that if all variables are bounded, that is $l_j \leq x_j \leq u_j$ for all j , and $l_j = 0$ a reasonable choice for M is

$$M = \max_{i=1, \dots, m} \left\{ b_i - \sum_{j: a_{ij} < 0} a_{ij} u_j \right\}.$$

In Section 2.2 we will show that one can assume without loss of generality that $l_j = 0$ for $j = 1, \dots, n$.

2.1. Bilinear formulation

As observed in [32], 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:

$$\begin{aligned} \max \quad & \sum_{i=1}^m y_i, \\ \text{s.t.} \quad & y_i \sum_{j=1}^n a_{ij} x_j \geq y_i b_i, \quad i = 1, \dots, m, \\ & 0 \leq y_i \leq 1, \quad i = 1, \dots, m. \end{aligned} \tag{2}$$

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.

For MAX FS with bounded variables we just have to add the bounds

$$l_j \leq x_j \leq u_j, \quad j = 1, \dots, n$$

to (2). We now present a linearization of this nonlinear continuous exact formulation which will be used in Section 3 to devise an efficient heuristic for MAX FS with mandatory bounds on the variables.

2.2. Linearization

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

$$\max \quad \sum_{i=1}^m y_i \tag{3}$$

$$\text{s.t.} \quad \sum_{j=1}^n a_{ij} z_{ij} \geq y_i b_i, \quad i = 1, \dots, m, \tag{4}$$

$$l_j \leq x_j \leq u_j, \quad j = 1, \dots, n, \tag{5}$$

$$0 \leq y_i \leq 1, \quad i = 1, \dots, m, \tag{6}$$

$$z_{ij} \geq 0, \quad j = 1, \dots, n, \tag{7}$$

which needs to be further constrained so that the variables z_{ij} are 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 $i = 1, \dots, m$:

$$y_i = 0 \implies z_{ij} = 0 \quad \text{for all } j = 1, \dots, n, \tag{C1}$$

$$y_i = 1 \implies z_{ij} = x_j \quad \text{for all } j = 1, \dots, n. \tag{C2}$$

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 $j = 1, \dots, n$. This can be seen via the following transformations. 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 (5) certainly helps to satisfy the inequalities, independently on the values of y_i , since $x_j \geq 0$. Thus, variables z_{ij} are not defined when $a_{ij} \geq 0$ and constraints (5) are replaced by

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

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 $i = 1, \dots, m, j = 1, \dots, n$. While conditions (C1) can be clearly expressed by the following group of at most nm constraints:

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

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

$$z_{ij} \leq x_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0 \tag{10}$$

$$x_j - u_j(1 - y_i) \leq z_{ij}, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0. \tag{11}$$

Constraints (10) hold 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) are redundant since $x_j \leq u_j$ and $z_{ij} \geq 0$. Thus adding (9)–(11) to (3)–(7) and replacing (4) with (8), we obtain the following linearization of the bilinear formulation (2):

$$\max \quad \sum_{i=1}^m y_i, \tag{12}$$

$$\text{s.t.} \quad \sum_{j:a_{ij}<0} a_{ij}z_{ij} + \sum_{j:a_{ij}\geq 0} a_{ij}x_j \geq y_i b_i, \quad i = 1, \dots, m, \tag{13}$$

$$z_{ij} \leq u_j y_i, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0, \tag{14}$$

$$z_{ij} \leq x_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0, \tag{15}$$

$$x_j - u_j(1 - y_i) \leq z_{ij}, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0, \tag{16}$$

$$l_j \leq x_j \leq u_j, \quad j = 1, \dots, n, \tag{17}$$

$$0 \leq y_i \leq 1, \quad i = 1, \dots, m, \tag{18}$$

$$z_{ij} \geq 0, \quad i = 1, \dots, m, \quad j = 1, \dots, n, \quad \text{s.t. } a_{ij} < 0. \tag{19}$$

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, one has to impose that $y_i \in \{0, 1\}$ for $i = 1, \dots, m$.

This formulation can also be derived via the McCormick convex relaxation [33] of the nonconvex constraints $z_{ij} = y_i x_j$ for $i = 1, \dots, m$ and $j = 1, \dots, n$:

$$z_{ij} \geq x_j^L y_i + y_i^L x_j - x_j^L y_i^L, \tag{20}$$

$$z_{ij} \geq x_j^U y_i + y_i^U x_j - x_j^U y_i^U, \tag{21}$$

$$z_{ij} \leq x_j^U y_i + y_i^L x_j - x_j^U y_i^L, \tag{22}$$

$$z_{ij} \leq x_j^L y_i + y_i^U x_j - x_j^L y_i^U, \tag{23}$$

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 [34]). 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 instead of (9):

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

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, see the Appendix for more details.

3. A two-phase relaxation-based heuristic

In this section we propose a simple but efficient heuristic for tackling MAX FS instances with mandatory bounds on the 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 identify all other inequalities that can be added to the above feasible subsystem while preserving feasibility.

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

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 $i \in I_1$.

For both the relaxation (12)–(19) of the bilinear formulation (2) and the linear relaxation of the big-M formulation (1), we observe 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;
- the inequalities indexed by I_1 can also contain the indices of inequalities that do not belong to any *maximum* feasible subsystem.

In spite of the last observation, we propose to tackle this NP-hard problem by considering the subsystem $S1$ composed of all inequalities indexed by I_1 and by looking for a maximum number of remaining inequalities (that is with indices in $\{1, \dots, m\} \setminus I_1$) that added to $S1$ yield 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 $I_1 = \{i : \tilde{y}_i = 1, i = 1, \dots, m\}$.

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 $\tilde{y}_i \geq 1 - \varepsilon$, 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 practice it is not difficult to select an appropriate value for ε .

In Step 3 we can use an exact algorithm for MAX FS [31,30] or solve the exact MIP formulation (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 a subset of the y_i variables.

4. Computational experiments

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 [30]. 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 nonzero elements of each group consisting of three instances. For comparison purposes, the variables take values in $[0, 1]$.

CBC-ML: A set of linear classification problems from the UCI Machine Learning repository [35]. In linear discriminant analysis, the goal is to partition a group of observations into two classes using a separating hyperplane in order to maximize the number of observations that are correctly classified. In the MAX FS formulation of this problem, the variables are the normal vector coefficients of the separating hyperplane and the threshold (the constant term of the hyperplane) [12,13,28,15]. As observed in [20], 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 a hyperplane is unequivocally 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 is artificially bounded in $[-10^5, +10^5]$. The instances mentioned in Table 2 were modified in [20] and used in [31] for the experimental evaluation of the CBC method. The actual values of the big-M constants are the same as in [31].

Table 1
Random instances: characteristics

Group	Avg. non-zeros	Group	Avg. non-zeros
random10 × 20	198	random20 × 50	989
random10 × 30	299	random20 × 60	1185
random10 × 40	396	random20 × 70	1384
random10 × 50	495	random20 × 80	1583
random10 × 60	594	random5 × 100	494
random10 × 70	693	random5 × 10	49
random10 × 80	792	random5 × 20	99
random15 × 30	447	random5 × 30	149
random15 × 40	594	random5 × 40	198
random15 × 50	741	random5 × 50	248
random15 × 60	890	random5 × 60	298
random15 × 70	1038	random5 × 70	346
random15 × 80	1186	random5 × 80	395
random20 × 40	792	random5 × 90	445

Each random $m \times n$ group is composed of three instances with m columns and n rows.

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

ML: A different group of instances from the UCI repository. These linear classification instances were used in previous works (see e.g. [14,5,13]) as a testbed for comparing exact and heuristic methods for MAX FS. Their characteristics are summarized in Table 3. The variables are bounded as 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 [11]. All variables are naturally bounded. 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 searched 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 [5] faced fatal errors on these instances.

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	19 089
mfs_UHF_P4_3	1717	487	22 023
P4_60ofP4	6345	487	65 538
P4_89ofP4	10 338	487	90 210
P4_60_19916	18 035	487	154 845
P4_280ofP4	14 678	487	248 536
P4	15 426	487	304 605
P4_487_19916	19 916	487	547 796

4.2. Experimental campaign

In the computational experiments we consider the two-phase algorithm using the linearization (12)–(19) of the bilinear formulation for Phase 1 and the big-M exact formulation (1) of MAX FS for Phase 2. For the sake of comparison, we also consider a Phase 1 with the linear relaxation of (1) as well as the substitution of our Phase 1 with the ordinary first phase of linear programming. The results obtained with our two-phase relaxation-based algorithm are compared with those provided by the *Filtering* heuristic (Algorithm 1 in [5]), the exact big-M formulation (1) tackled with Cplex 8.1 MIP solver, the Branch-and-Cut algorithm of [29,30] and a recent exact polyhedral method based on CBC [31].

Chinneck's Algorithm 1 [27,5] 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.

The Branch-and-Cut algorithm was tested using the C++ implementation provided by Marc Pfetsch.

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.

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. We used the original *.lp* files only for solving the CBC-ML instances with the exact big-M formulation; 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`. The last parameter asks for the solver to discriminate on the variable integrality more accurately than the default. 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 10 000 seconds. The actual feasibility of all solutions (subsystems found) was double checked with default Cplex settings.

4.3. Computational results

The computational results for the instances of Section 4.1 are reported in Tables 5–8. The following notations are used across the tables:

- *exact-bigM* stands for Cplex 8.1 MIP solver applied to the exact big-M formulation (1).
- *Branch-and-cut* for the branch-and-cut algorithm of [30].
- *CBC* for the polyhedral method based on the CBC [31].
- *Phase1-bigM* for the two-phase algorithm with the linearization of the big-M formulation (1) in Phase 1.
- *Phase1-bilinear* for the two-phase algorithm with the linearization (12)–(19) in Phase 1.
- *Phase1-LP* for the two-phase algorithm with the ordinary LP first phase in Phase 1.
- *Filtering* for the heuristic Algorithm 1 of [5].
- *FS* denotes the number of inequalities in the feasible subsystem found by the algorithm.
- *CPU* the computing time in seconds.
- $V(ub: U)$ indicates the number V of inequalities in the largest feasible subsystem found by Cplex for an exact big-M formulation that cannot be solved to optimality together with the upper bound U on the optimal value.
- $V(< B)$ denotes 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.

4.3.1. Random instances

According to Table 5, the branch-and-cut algorithm of [30] solves all random instances to optimality within 1 h. For the heuristics, solution quality is measured in terms of relative gap, i.e., the difference between the number of inequalities in a largest feasible subsystem and that in the subsystem found by the algorithm, divided by the former optimal value. The *Phase1-bilinear* variant yields on average the best results. As indicated in the last row of the table, the average relative gap for *Phase1-bilinear* is 1.29% against 2.16% for *Phase1-BigM*, 4.68% for *Phase1-LP* and 2.25% for the *Filtering* heuristic. In particular, if we focus on the most difficult instances, that is on those with at least 70 rows and 10 columns, *Phase1-bilinear* always yields the largest feasible subsystem among the three compared heuristics. It is worth pointing out that the number of variables that are fixed by *Phase1-bilinear* at the end of the first phase is always smaller than that fixed by *Phase1-bigM*. Although one may suspect 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 . As to computing time, *Phase1-bilinear* is competitive with *Filtering* but *Phase1-bigM* is the best choice for obtaining good quality solutions rapidly. Furthermore, the computing times required by *Phase1-bilinear* grow rather slowly when only a few columns are present.

4.3.2. CBC-ML instances

According to Table 6, some of the CBC-ML instances are challenging to solve to optimality. Indeed the exact method *CBC* does not find any solution for *Flags-169*, *Horse-colic-253*, *Horse-colic-185* as well as *Solar-flare-1066*. For the sake of comparison, we report the original results for the last four instances presented in [31]. The difference in the results should be accounted for by the tuning of *CBC* parameters. Note that *exact-bigM* provides an optimal solution only for the first 24 instances. But for most of the remaining instances, the best feasible solution found is very close to the optimal solution. When *CBC* does not yield any solution, the best feasible solution found with *exact-bigM* differs from the upper bound by at most eight inequalities, except for the difficult *Solar-flare-1066*.

Both *Phase1-bigM* and *Phase1-bilinear* provide an 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 *Phase1-bigM*, with lower computing times.

Table 5
Random instances: computational results

Instance	Branch-and-Cut		Phase1-bilinear				Phase1-bigM				Phase1-LP				Filtering	
	FS	CPU	FS		CPU		FS		CPU		FS		CPU		FS	CPU
			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		
random5 × 10	22	0.1	17	22	0.1	0.1	17	21	0.1	0.1	16	22	0.1	0.1	22	0.1
random5 × 20	45	1	37	44	0.1	0.1	40	44	0.1	0.1	39	43	0.1	0.1	44	0.1
random5 × 30	62	0.1	43	62	0.1	0.1	57	62	0.1	0.1	52	62	0.1	0.1	61	2
random5 × 40	82	1	69	80	0.1	0.1	74	82	0.1	0.1	75	80	0.1	0.1	80	3
random5 × 50	104	5	89	101	0.1	0.1	95	103	0.1	0.1	94	103	0.1	0.1	103	4
random5 × 60	122	7	102	120	0.1	0.1	108	120	0.1	0.1	105	115	0.1	0.1	117	6
random5 × 70	139	20	115	137	0.1	0.1	125	134	0.1	0.1	118	129	0.1	0.1	130	10
random5 × 80	160	22	131	156	0.1	0.1	141	154	0.2	0.2	135	150	0.1	0.1	151	19
random5 × 90	174	36	131	156	0.1	0.1	154	167	0.2	0.2	148	167	0.1	0.1	169	21
random5 × 100	191	74	162	183	0.2	0.2	172	181	0.2	0.2	167	184	0.1	0.1	182	19
random10 × 20	52	0.1	47	52	0.1	0.1	51	52	0.1	0.1	50	52	0.1	0.1	52	0.3
random10 × 30	69	0.1	57	69	0.1	0.1	59	68	0.1	0.1	60	66	0.1	0.1	69	1
random10 × 40	92	3	73	91	0.1	0.1	78	90	0.1	0.1	79	90	0.1	0.1	89	2
random10 × 50	110	21	88	109	0.1	0.1	95	107	0.1	0.1	93	107	0.1	0.1	109	5
random10 × 60	126	89	97	123	0.1	0.1	103	121	0.1	0.1	96	121	0.1	0.1	117	11
random10 × 70	143	443	110	142	0.1	1	117	141	0.1	0.1	114	136	0.1	0.1	137	14
random10 × 80	162	2584	122	159	0.1	2	125	156	0.1	0.1	125	144	0.1	0.1	156	21
random15 × 30	77	2	64	77	0.1	0.1	67	75	0.1	0.1	64	74	0.1	0.1	76	2
random15 × 40	101	4	78	100	0.1	0.1	86	100	0.1	0.1	83	99	0.1	0.1	101	3
random15 × 50	126	14	104	123	0.1	1	109	123	0.1	0.1	108	118	0.1	0.1	121	5
random15 × 60	148	52	118	147	0.2	2	128	145	0.1	1.1	123	142	0.1	0.1	147	6
random15 × 70	168	301	132	167	0.2	4	143	162	0.2	2	134	163	0.1	0.1	165	12
random15 × 80	190	1782	151	188	0.3	10	165	188	0.2	0.2	156	183	0.1	0.1	185	12
random20 × 40	106	2	84	106	0.1	0.1	103	106	0.1	0.1	92	104	0.1	0.1	106	2
random20 × 50	131	7	100	131	0.2	0.2	112	130	0.1	0.1	108	130	0.1	0.1	130	5
random20 × 60	153	37	117	152	0.3	0.3	123	151	0.2	0.2	121	149	0.1	0.1	153	6
random20 × 70	174	277	131	174	0.3	3	142	172	0.2	2	138	170	0.1	0.1	174	8
random20 × 80	192	2267	136	190	0.4	18	156	186	0.3	18	149	187	0.1	0.1	189	47
Average gap			20.19%	1.29%			13.45%	2.16%			20.68%	4.68%			2.25%	

Variables bounded in [0, 1].

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

Instance	<i>exact-bigM</i>		<i>CBC</i>		<i>Phase1-bilinear</i>				<i>Phase1-bigM</i>				<i>Phase1-LP</i>				<i>Filtering</i>	
	FS	CPU	FS	CPU	FS	CPU		FS	CPU		FS	CPU		FS	CPU		FS	CPU
						ph. 1	ph. 2		ph. 1	ph. 1 + 2		ph. 1	ph. 2		ph. 1	ph. 1 + 2		
Chorales-116	92	3559	92	550	58	92	3	11	46	92	0.1	22	46	92	0.1	16	92	14
Balloons76	66	7	66	0.1	52	66	0.1	1	52	66	0.1	0.1	52	66	0.1	0.1	66	4
BCW-367	359	365	359	1	338	359	93	93	333	359	0.1	0.3	334	359	0.1	1	358	5
BCW-683	673	6750	673	10	649	673	675	679	643	673	0.1	2	643	673	0.1	2	672	10
WPBC-194	189	2279	189	299	166	189	1025	1030	161	189	0.1	4	162	189	0.1	8	189	3
Breast-Cancer-400	376	71	376	0.1	374	376	0.1	3	374	376	0.1	0.2	374	376	0.1	0.1	374	13
Glass-163	150	3849	150	3	146	149	8	9	102	150	0.1	0.1	102	150	0.1	0.1	150	10
Horse-colic-151	146	592	146	12	130	146	82	84	128	146	0.1	0.1	126	146	126	146	146	2
Iris-150	132	463	132	56	107	132	2	3	105	132	0.1	1	104	132	0.1	14	130	9
Credit-300	292	1836	292	2	163	292	80	82	258	292	0.1	0.1	259	292	0.1	1	292	5
Lymphography-142	137	8	137	0.4	123	137	21	21	120	137	0.1	0.3	119	137	0.1	1	137	2
Mech-analysis-107	100	10	100	0.1	78	100	3	3	77	100	0.1	0.1	76	100	0.1	0.3	100	2
Mech-analysis-137	119	775	119	6	93	119	3	4	84	119	0.1	2	84	119	3	11	119	10
Monks-tr-122	109	212	109	0.1	78	109	3	4	76	109	0.1	1	75	109	0.1	2	109	6
Pb-gr-txt-198	187	330	187	3	168	187	0.4	11	170	187	0.1	0.1	170	187	0.1	0.3	186	8
Pb-pict-txt-444	437	81	437	0.1	407	437	37	38	422	437	0.1	0.4	423	437	0.1	0.9	434	5
Pb-hl-txt-277	267	193	267	4	247	267	21	22	248	266	0.1	0.1	246	266	0.1	0.8	267	3
Postoperative-88	72	698	72	0.1	64	72	2	2	64	72	0.1	0.1	64	72	0.1	0.1	66	8
Bv-os-282	276 ^a	210	276 ^a	4	264	277 ^a	255	2	261	276 ^a	0.1	10.1 57	259	276 ^a	0.1	1	277 ^a	2
Opel-Saab-80	77	43	77	11	51	75	15	16	50	74	0.1	0.1	51	74	0.1	5	75	2
Bus-Van-437	431	363	431	5	409	431	806	820	410	431	0.1	6	410	431	0.1	10	431	3
HouseVotes84-435	429	207	429	0.1	409	429	63	65	405	429	0.1	0.1	405	429	0.1	3	429	3
Water-treat-206	202	43	202	4	177	202	1946	1953	175	202	0.1	2	175	202	0.1	8	202	4
Water-treat-213	208	621	208	21	171	208	582	626	175	208	0.1	5	175	208	0.1	25	208	4

Table 6 (Continued)

Instance	<i>exact-bigM</i>		<i>CBC</i>		<i>Phase1-bilinear</i>				<i>Phase1-bigM</i>				<i>Phase1-LP</i>				<i>Filtering</i>	
	FS	CPU	FS	CPU	FS		CPU		FS		CPU		FS		CPU		FS	CPU
					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		
Chorales-134	103(ub : 113)	^b	104	727	50	104	2	33	39	104	0.3	46	40	104	0.1	36	104	27
Chorales-107	80(ub : 85)	^b	80	67	36	80	1	19	31	80	0.2	22	36	80	0.1	22	79	19
Bridges-132	108(ub : 121)	^b	109	136	74	109	33	430	67	109	0.1	58	67	109	0.1	136	109	14
Mech-analysis-152	130(ub : 136)	^b	131	139	117	131	6	6	86	131	0.2	12	88	131	0.1	10	128	16
Monks-tr-124	100(ub : 104)	^b	100	56	55	100	3	24	50	100	0.1	22	55	100	0.1	38	97	17
Monks-tr-115	88(ub : 96)	^b	88	487	49	87	2	56	25	88	0.1	61	49	81	0.1	85	88	24
Solar-flare-323	282(ub : 300)	^b	285	3	254	284	94	96	241	284	0.1	4	247	284	0.1	5	281	45
Bv-os-376	367(ub : 369)	^b	368	125	341	368	494	505	340	367	0.1	5	341	367	0.1	9	367	6
BusVan445	436(ub : 438)	^b	437	102	412	437	320	363	411	437	0.1	4	410	437	0.1	11	437	5
Flags-169	159(ub : 163)	^b	(ub:163) ^c	–	130	160	43	135	118	160	0.2	78	102	150	0.1	0.3	159	6
Horse-colic-253	240(ub : 248)	^b	(ub:244) ^c	–	196	240	221	1275	188	240	0.4	654	190	240	0.1	1640	240	15
Horse-colic-185	173(ub : 177)	^b	(ub:178) ^c	–	145	173	128	272	137	173	0.1	42	135	173	0.1	141	172	9
Solar-flare-1066	796(ub : 1058)	^b	(ub:864) ^d	^b	525	816(< 925)	1	^b	607	814(< 938)	309	^b	526	808(< 976)	1	^b	808	8378
Average gap					18.76%	0.33%			34.50%	0.34%			121.16%	0.56%			0.86%	

^aNumerical instability.

^bTime limit exceeded.

^cMemory limit.

^dNo feasible subsystem found.

Similarly, the two-phase relaxation-based algorithm compares favourably with *Filtering* in terms of solution quality: the average relative gap with respect to the optimal solution value (or to the upper bound, when no optimal solution has been obtained within the time limit) is 0.34% against 0.86% of the latter, excluding *Solar-flare-1066*. However, the computing times of the first phase of *Phase1-bilinear* suggest that the linearization of the bilinear formulation can be much more time-consuming than both *Phase1-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, *Phase1-bilinear* and *Phase1-bigM* achieve similar results, in spite of the fact that the number of variables fixed by *Phase1-bigM* is in general smaller than those fixed by *Phase1-bilinear* (the average relative gap of Phase 1 is 34.50% for the former and 18.76% for the latter). We notice that for this set of instances *Phase1-LP* yields results of intermediate quality with respect to *Phase1-bilinear* and *Filtering* but with a low computational effort, comparable with that one of *Filtering*. The results obtained for the *Bv-os-282* instance call for an explanation. Both *CBC* and *Phase1-bigM* provide a feasible subsystem with 276 inequalities whereas *Filtering* and *Phase1-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.

4.3.3. ML instances

The results obtained for the ML group are summarized in Table 7. Note that *exact-bigM* is able to find an optimal solution within the 10 000 seconds time limit for only three instances, whereas the branch-and-cut algorithm of [30] for only five instances. The results for the ML and CBC-ML instances indicate that, even though the presence of a free variable may in principle affect the performance of our two-phase heuristic, it does not happen in practice. The solution quality of our three variants are indeed comparable to that of *Filtering*, which yields the best average relative gap. Observe that in *Phase1-bilinear* the linearization can require large computing times (see instances *ionosphere*, *pima* and *wdbc*) and, if a small number of variables are fixed in Phase 1, Phase 2 restricted problem can be too hard to be solved to optimality within the computing time limit (see instances *bupa*, *pima* and *wdbc*). Although *Phase1-bilinear* requires higher computing times, it is useful on two hard instances such as *bupa* and *pima*, since it provides better solutions than *Phase1-bigM*. Finally, notice that the *Phase1-LP* algorithm yields slightly worse results than the other methods and it is not able to end within the 10 000 s time limit for two more instances than the other two-phase variants.

4.3.4. DVB instances

The results for the challenging DVB instances are reported in Table 8. Since the variables are all bounded in $[0, 1]$, the linearization of the bilinear formulation is particularly adequate. In fact, *Phase1-bilinear* yields the smallest average relative gap obtained in Phase 1, namely 7.55%. For simplicity of exposition, we divided the instances into three groups according to the instance size. *Phase1-bigM* and *Phase1-bilinear* provide comparable results which are better than those of the *Filtering* method for all instances. *Filtering* has higher computing time requirements than our heuristics and these requirements grow much faster with the instance size, since the number of *Filtering* steps depends on the number of inequalities to be deleted from the original infeasible system in order to achieve feasibility. However, the refinements of the *Filtering* heuristic described in [5] may at least partially offset this drawback but at the price of slightly affecting the solution quality. 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 *Phase1-bigM* finds a slightly better feasible solution in just 5 s.

As to the second group of instances, the advantage of *Phase1-bilinear* with respect to all other methods is remarkable, both in terms of the size of the feasible subsystems and of the cumulative computing time. For instance *P4_60_19916*, *Phase1-bilinear* finds a feasible subsystem with more than 350 additional inequalities compared to the feasible subsystem provided by *Phase1-bigM*. This suggests that for sparse medium-sized instances *Phase1-bilinear* should be preferred to *Phase1-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 *Phase1-bilinear* and *Phase1-bigM* is the only viable method. Note 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*.

Finally, the *Phase1-LP* method performs poorly on this set of instance: it yields an average relative gap of 15.74% against 6.40% for *Phase1-bilinear*.

Table 7
ML instances: computational results

Instance	Branch-and-Cut		exact-bigM		Phase1-bilinear				Phase1-bigM				Phase1-LP				Filtering	
	FS	CPU	FS	CPU	FS		CPU		FS		CPU		FS		CPU		FS	CPU
					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		
breast-cancer-wisconsin	672	43	671(<i>ub</i> : 676)	^a	650	672	370	371	660	672	0.1	1	643	672	0	8	672	15
bupa	227(<i>ub</i> : 263)	^a	248(<i>ub</i> : 331)	^a	103	261(≤294)	71	^a	166	260(≤266)	0.3	^a	101	259(≤298)	0	^a	259	331
glass	178(<i>ub</i> : 193)	^a	176(<i>ub</i> : 205)	^a	79	168	33	133	112	172	0.2	375	79	167(≤183)	0	^a	172	143
ionosphere	345	2215	345(<i>ub</i> : 347)	^a	319	345	2238	2263	322	345	0.1	4	310	345	0	36	345	36
iris.1	125	1735	125	7630	79	124	2	4	78	124	0.1	1	69	124	0	5	125	30
iris.2	149	0.1	149	0.4	147	149	0.1	1	147	149	0.1	0.2	147	149	0.1	0.2	149	0.1
new-thyroid	204	17	204	46	179	204	9	9	181	204	0.1	0.2	167	204	0.1	0.4	204	10
pima	614(<i>ub</i> : 718)	^a	588(<i>ub</i> : 761)	^a	419	617(≤703)	1195	^a	508	615(≤667)	0.4	^a	377	615(≤743)	0.1	^a	619	1977
wdbc	181(<i>ub</i> : 187)	^a	175(<i>ub</i> : 190)	^a	133	178	956	^a	146	178	0.2	1863	132	178(≤181)	0.1	^a	178	91
Average gap					29.73%	7.38%			23.49%	7.22%			28.87%	7.53%			7.11%	

Hyperplane coefficients bounded in $[-1, 1]$.

^aTime limit exceeded.

Table 8
DVB instances: computational results

	<i>exact-bigM</i>		<i>Phase1-bilinear</i>				<i>Phase1-bigM</i>				<i>Phase1-LP</i>		<i>Filtering</i>			
	FS	CPU	FS	CPU	FS	CPU	FS	CPU	FS	CPU	FS	CPU				
mfs_UHF_P4_1	538	0.4	534	537	0.1	0.1	533	537	0.1	0.1	491	493	0.1	0.1	532	127
mfs_UHF_P4_4	1050(<i>ub</i> : 1065)	^a	1028	1044	11	12	1039	1051	4	5	775	1021(≤1024)	0.2	^a	1037	3089
mfs_UHF_P4_3	1532(<i>ub</i> : 1535)	^a	1508	1525	59	61	1509	1523	2	2	1395	1477	0.6	3	1523	5583
P4_60ofP4_15426_naz	3257(<i>ub</i> : 3526)	^a	3071	3130	217	466	2849	3115(≤3235)	10	^a	711	3276(≤3520)	3	^a	≤5926	^a
P4_89ofP4_15426_naz	5505(<i>ub</i> : 6158)	^a	5422	5514(≤5602)	755	^a	5189	5408(≤5456)	4	^a	1499	3834(≤3951)	4	^a	≥9695	^a
P4_60_19916	6805(<i>ub</i> : 8037)	^a	6838	6902	1191	1513	6308	6546(≤6583)	147	^a	1574	5743(≤7059)	7	^a	≥16494	^a
P4_280ofP4_15426_naz	9338(<i>ub</i> : 10708)	^a	9339	9551	7184	9874	9161	9434(≤9713)	24	^a	5105	7184(≤9153)	18	^a	≥14604	^a
P4_15426_naz	12189(<i>ub</i> : 13384)	^a	^b	–	^a	–	12089	12374(≤12692)	52	^a	8170	10700(≤12038)	15	^a	≥15425	^a
P4_487_19916	15700(<i>ub</i> : 19650)	^a	^b	–	^a	–	15584	15940(≤16628)	234	^a	7449	11842(≤15056)	29	^a	≥19915	^a
average gap ^c			1.96%	1.14%			1.73%	0.95%							1.33%	
average gap ^d			7.55%	6.40%			9.73%	7.26%			44.04%	15.74%			–	

Variables bounded in [0, 1].
^aTime limit exceeded.
^bNo feasible subsystem found.
^cInstances solved by all methods.
^dInstances solved by two-phase algorithms.

Table 9
Summary of average percent gaps and average absolute gaps

Testbed	<i>Phase1-bigM</i>		<i>Phase1-bilinear</i>		<i>Filtering</i>	<i>Phase1-LP</i>	
	ph.1	ph.2	ph.1	ph.2		ph.1	ph.2
Random	13.45%	2.16%	20.19%	1.29%	2.25%	16.40%	3.47%
	17.00	2.86	25.11	1.79	3.07	20.68	4.68
CBC-ML	34.50%	0.34%	18.76%	0.33%	0.86%	21.16%	0.56%
	67.50	0.61	28.75	0.55	1.31	30.67	0.78
ML	23.49%	7.22%	29.73%	7.38%	7.11%	28.87%	7.53%
	66.8	29.89	97.78	30.00	29.44	106.44	30.56
DVB ^a	1.73%	0.95%	1.96%	1.14%	1.33%	16.02%	6.12%
	18.25	10	21	11.75	13.25	160.25	54.5
DVB ^b	9.73%	7.26%	7.55%	6.40%	–	44.04%	15.74%
	1294.5	1085.75	480.38	422.38	–	4699.5	2316.75

^aInstances solved by all methods.

^bInstances solved by two-phase algorithms.

4.3.5. Overall comparison

Table 9 summarizes, for the four sets of instances, the average percent relative gaps and the average absolute gaps between the number of inequalities in the feasible subsystem provided by the heuristics and that of an optimal solution (or of 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, *Phase1-bilinear* provides on average better quality solutions than *Phase1-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 on 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 computational results indicate 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 exact method based on CBC. We leave as an open question if the two-phase relaxation-based approach can also be successfully extended to tackle instances of MAX FS in which all variables are unbounded.

Acknowledgements

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 Branch-and-Cut code and the random instances, Fabrizio Rossi and Stefano Smriglio for the DVB instances.

Appendix

We show that the relaxation (linearization) obtained imposing conditions (C1) through constraints (24) leads to a weaker formulation than (12)–(19). 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{i} 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 $0 < \alpha < u_{\hat{j}}$ satisfying the following inequality

$$\alpha a_{\hat{i}\hat{j}} + \sum_{j:a_{\hat{i}j} \geq 0} a_{\hat{i}j} u_j \geq \frac{\alpha b_{\hat{i}}}{\sum_{j:a_{\hat{i}j} < 0} u_j}, \quad (25)$$

where \hat{j} is such that $a_{\hat{i}\hat{j}} < 0$. Therefore the point $(\mathbf{z}, \mathbf{y}, \mathbf{x})$ defined as

$$z_{ij} = \begin{cases} \alpha & \text{if } i = \hat{i} \text{ and } j = \hat{j}, \\ 0 & \text{otherwise,} \end{cases} \quad (26)$$

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

and

$$x_j = u_j, \quad j = 1, \dots, n \quad (28)$$

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

References

- [1] Amaldi E, Kann V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science* 1998;209:237–60.
- [2] Meller J, Wagner M, Elber R. Solving huge linear programming problems for the design of protein folding potentials. *Mathematical Programming B* 2004;101:301–18.
- [3] Amaldi E, Mattavelli M. The MIN PFS problem and piecewise linear model estimation. *Discrete Applied Mathematics* 2002;118:115–43.
- [4] Mattavelli M, Noel V, Amaldi E. Fast line detection algorithms based on combinatorial optimization. In: *Proceedings of IWVF, LCNS 2059*. Berlin: Springer; 2001. p. 410–9.
- [5] Chinneck J. Fast heuristics for the maximum feasible subsystem problem. *INFORMS Journal on Computing* 2001;13:210–3.
- [6] Chinneck JW. Computer codes for the analysis of infeasible linear programs. *Journal of Operational Research Society* 1996;47:61–72.
- [7] Greenberg HJ, Murphy FH. Approaches to diagnosing infeasible linear programs. *ORSA Journal on Computing* 1991;3:253–61.
- [8] Parker M, Ryan J. Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence* 1996;17:107–26.
- [9] Lee EK, Gallagher RJ, Zaider M. Planning implants of radionuclides for the treatment of prostate cancer: an application of MIP. *Optima* 1999;61:1–7.
- [10] Johnson DS, Preparata FP. The densest hemisphere problem. *Theoretical Computer Science* 1978;6:93–107.
- [11] Rossi F, Sassano A, Smriglio S. Models and algorithms for terrestrial digital broadcasting. *Annals of Operational Research* 2001 (107):267–83.
- [12] Amaldi E. From finding maximum feasible subsystems of linear systems to feedforward neural network design. PhD thesis, Dep. of Mathematics, EPF-Lausanne; 1994.
- [13] Bennett KP, Bredensteiner E. A parametric optimization method for machine learning. *INFORMS Journal on Computing* 1997;9:311–8.
- [14] Mangasarian OL. Machine learning via polyhedral concave minimization. In: Fischer H et al., editor. *Applied Mathematics and Parallel Computing*. Wurzburg: Physica-Verlag; 1996. p. 175–88.
- [15] Parker M. A set covering approach to infeasibility analysis of LP problems and related issues. PhD thesis, Dep. of Mathematics, University of Colorado at Denver; 1995.
- [16] Chakravarti N. Some results concerning post-infeasibility analysis. *European Journal of Operational Research* 1994;73:139–43.
- [17] Sankaran J. A note on resolving infeasibility in linear programs by constraint relaxation. *Operational Research Letters* 1993;13:19–20.
- [18] Amaldi E, Kann V. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science* 1995;147:181–210.

- [19] Garey MR, Johnson DS. *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: W.H. Freeman and Company; 1979.
- [20] Rubin PA. Solving mixed integer classification problems by decomposition. *Annals of Operations Research* 1997;74:51–64.
- [21] Bennett KP, Mangasarian OL. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software* 1992;1:23–34.
- [22] Freed N, Glover F. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research* 1981;7:44–60.
- [23] Glover F. Improved linear and integer programming models for discriminant analysis. *Decision Sciences* 1990;21:771–85.
- [24] Stam A, Joachimsthaler EA. A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. *European Journal of Operational Research* 1990;46:113–22.
- [25] Amaldi E, Belotti P, Hauser R. Randomized relaxation methods for the maximum feasible subsystem problem. In: *Integer programming and combinatorial optimization*, 11th international IPCO conference, proceedings, vol. 3509 of LCNS. Berlin: Springer; 2005. p. 249–64.
- [26] Frean M. A “thermal” perceptron learning rule. *Neural Computation* 1992;4(6):946–57.
- [27] Chinneck J. An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem. *Annals of Mathematics and Artificial Intelligence* 1996;17:127–44.
- [28] Mangasarian OL. Misclassification minimization. *Journal of Global Optimization* 1994;5(4):309–23.
- [29] Amaldi E, Pfetsch ME, Trotter Jr LE. On the maximum feasible subsystem problem IISs and IIS-hypergraphs. *Mathematical Programming A* 2003;95:533–54.
- [30] Pfetsch M. The maximum feasible subsystem problem and vertex–facet incidences of polyhedra. PhD thesis, Technische Universität Berlin; 2002.
- [31] Codato G, Fischetti M. Combinatorial Benders’ cuts. In: Nemhauser GL, Bienstock D, editors. *Integer programming and combinatorial optimization*, 10th international IPCO conference, proceedings, vol. 3064 of LNCS. Berlin: Springer; 2004. p. 178–95.
- [32] Amaldi E. The maximum feasible subsystems problem and some applications. In: Agnetis A, Di Pillo G, editors. *Modelli e Algoritmi per l’ottimizzazione di sistemi complessi*. Pitagora Editrice Bologna; 2003.
- [33] McCormick GP. Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. *Mathematical Programming* (10) 1976; 146–75.
- [34] Liberti L. Linearity embedded in nonconvex problems. *Journal of Global Optimization* 2005;33(2):157–96.
- [35] Blake CL, Merz CJ. UCI repository of machine learning databases, 1998. Available at (<http://www.ics.uci.edu/~mllearn/MLRepository.html>).