RUTCOR
RESEARCH
REPORT

An Improved Branch-and-Bound Method for Maximum Monomial Agreement

Jonathan Eckstein[a] Noam Goldberg[b]

RRR 14, July 2009
Revised: October 2009

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]MSIS Department and RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854; jeckstei@rci.rutgers.edu
[b]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854; ngoldberg@rutcor.rutgers.edu

Rutcor Research Report
RRR 14, July 2009 Revised: October 2009

An Improved Branch-and-Bound Method for Maximum Monomial Agreement

Jonathan Eckstein Noam Goldberg

Abstract. The $\mathcal{NP}$-hard *maximum monomial agreement* (MMA) problem consists of finding a single logical conjunction that best fits a weighted dataset of "positive" and "negative" binary vectors. Computing classifiers using boosting methods involves a maximum agreement subproblem at each iteration, although such subproblems are typically solved by heuristic methods. Here, we describe an exact branch and bound method for maximum agreement over Boolean monomials, improving on the earlier work of Goldberg and Shan [13]. In particular, we develop a tighter upper bounding function and an improved branching procedure that exploits knowledge of the bound and the dataset, while having a lower branching factor. Experimental results show that the new method is able to solve larger problem instances and runs faster within a linear programming boosting procedure applied to medium-sized datasets from the UCI repository.

Acknowledgements: We would like to thank Chung-chieh Shan and Endre Boros for their contributions to our development process.

1 Problem Statement and Introduction

The *maximum monomial agreement* (MMA) problem has applications to machine learning, data mining [14, 13], as well as computational geometry and computer graphics [7], where it is known as the *maximum bi-chromatic discrepancy* problem. As input, we will assume a given set of M binary N-vectors in the form of a 0/1 matrix $A \in \{0,1\}^{M\times N}$, along with a partition of its rows into positive observations $\Omega^+ \subset \{1,...,M\}$ and negative observations $\Omega^- = \{1,...,M\} \setminus \Omega^+$. We are also given a function $w : \{1,\ldots,M\} \to [0,\infty)$; $w(i)$ specifies the weight of observation i. A *monomial* on $\{0,1\}^N$ is simply a function $p : \{0,1\}^N \to \{0,1\}$ of the form

$$m_{J,C}(x) = \prod_{j\in J} x_j \prod_{c\in C}(1-x_c), \tag{1}$$

where J and C are (usually disjoint) subsets of $\{1,\ldots,N\}$. The monomial $m_{J,C}$ is said to *cover* a vector $x \in \{0,1\}^N$ if $m_{J,C}(x) = 1$. We define the *coverage* of a monomial $m_{J,C}$ to be

$$\mathrm{Cover}(J,C) = \{i \in \{1,...,M\} \mid m_{J,C}(A_i) = 1\},$$

where A_i denotes the i^{th} row of A. Defining the weight of a set $S \subseteq \{1,\ldots,M\}$ to be $w(S) = \sum_{i\in S} w(i)$, the MMA problem is to find disjoint sets $J, C \subseteq \{1,\ldots,N\}$ maximizing

$$f(J,C) = \left|w(\mathrm{Cover}(J,C)\cap\Omega^+) - w(\mathrm{Cover}(J,C)\cap\Omega^-)\right|. \tag{2}$$

When the problem dimension N is part of the input, this problem is known to be $\mathcal{NP}$-hard [14]. Furthermore, it has been shown [10] that even if the weights $w(i)$ are all equal, a related problem with the slightly different maximization objective

$$\begin{aligned}\tilde{f}(J,C) &= \left(w(\mathrm{Cover}(J,C)\cap\Omega^+) + w(\Omega^- \setminus \mathrm{Cover}(J,C))\right)/M \\ &= \left(w(\mathrm{Cover}(J,C)\cap\Omega^+) - w(\mathrm{Cover}(J,C)\cap\Omega^-) + w(\Omega^-)\right)/M,\end{aligned}$$

is $\mathcal{NP}$-hard to approximate to any factor less than 2. Approximation of the latter objective to a factor of 2 is trivial, however, by simply considering only the constant monomials such that either $m_{J,C}(A_i) = 0$ for all $i \in \{1,\ldots,M\}$, or $m_{J,C}(A_i) = 1$ for all $i \in \{1,\ldots,M\}$.

Dobkin *et al.* consider the maximum bi-chromatic discrepancy problem for axis-aligned rectangles in $\mathbb{R}^2$ and propose an $O(M^2 \log M)$ algorithm. They also suggest a generalization of the algorithm for $\mathbb{R}^n$, which results in $O(M^{2n-2}\log M)$ running time. When the input data of the problem consists of points in $\{0,1\}^N$, then any axis-aligned rectangle corresponds to a monomial $m_{J,C}$. Another related (but not equivalent) problem for real-valued data is the *maximum box* problem [8]. For the special case of input data in $\{0,1\}^N$, the (weighted) maximum box problem can be stated as the problem of finding a box or subcube (J,C) that maximizes $w(\Omega^+ \cap \mathrm{Cover}(J,C))$ subject to $\Omega^- \cap \mathrm{Cover}(J,C) = \emptyset$. In this paper, we consider only binary datasets; however, for general datasets in $\mathbb{R}^N$, we note that there is a corresponding "binarization" with dimension that is at most polynomially larger than M, and in practice does not tend to be much larger than N [3, 13].

The MMA problem can be motivated by applications to supervised learning and classification. For example, we may have M patients, with Ω^+ corresponding to those having a particular disease, and Ω^- to those who do not. Each patient has N binary attributes corresponding to personal traits or results of medical tests, and we would like to find the interaction of attributes which best predicts presence or absence of the disease. For example, a monomial might represent a rule or hypothesis such as ('age' ≥ 50) $\wedge$ ('blood pressure' = 'high') $\Rightarrow$ 'heart disease'.

In classification problems, monomial hypotheses have been found especially useful when used in ensembles, typically by thresholding linear combinations of monomials. In the machine learning community, iterative algorithms that linearly combine classifiers into a "stronger" ensemble classifier originated with Schapire and Freund [11] and are referred to as *boosting algorithms*. In particular, boosting of monomial hypotheses has been suggested by several authors [5, 12, 13]. The results in [13] show that boosting optimal monomial hypotheses, as opposed to heuristically generated monomials (*e.g.* as in SLIPPER [5]), can improve the classification performance when using a sufficiently robust boosting algorithm, such as Servedio's SmoothBoost [17]. Demiriz, Bennett and Shawe-Taylor [6] suggest a boosting method for general weak learner hypotheses (not necessarily monomials), based on column generation linear programming techniques, using a robust reformulation of an LP model for finding a separating hyperplane; see also [16]. In their column generation subproblem, the objective is precisely that of maximum agreement; if the space of possible weak learners consists of all monomials, the subproblem is precisely to maximize (2). Monomial hypotheses (also called logical patterns) are also a basic building block in the *logical analysis of data* (LAD) methodology [4], where linear programming techniques are also used to compute the discriminant function.

2 Branch and bound

One possible approach to exactly solving the MMA problem is to formulate it as an equivalent integer linear program, and solve it with a standard integer programming solver. However, we have found it more efficient to use a specialized branch-and-bound algorithm. The key elements of a branch-and-bound algorithm are the definition of subproblems representing sets of possible solutions, a method for computing a bound on the objective value of all solutions represented by a subproblem, and a branching procedure for subdividing subproblems into smaller ones.

For the MMA problem, we characterize each subproblem as a partition (J, C, E, F) of $\{1, \ldots, N\}$. As in (1), J and C respectively represent the literals which must be in the monomial, and whose complements must be in the monomial. E indicates a set of "excluded" literals: neither they nor their complements may appear in the monomial. Finally, $F = \{1, \ldots, N\} \backslash (J \cup C \cup E)$ is the set of "free", undetermined literals. The set of monomials corresponding to subproblem (J, C, E, F) is given by

$$P(J, C, E) = \{(J', C') \mid J' \supseteq J,\ C' \supseteq C, \text{ and } J', C', E \text{ are disjoint}\}. \tag{3}$$

The search begins with a priority queue Q containing the single root subproblem $(\emptyset, \emptyset, \emptyset, \{1 \ldots, N\})$, which corresponds to the set of all possible monomials. At each iteration, we remove a subproblem (J, C, E, F) from Q and, unless it is fathomed, that is, its upper bound $b(J, C, E) \leq f(J^*, C^*)$ for some known J^*, C^*, we further subdivide (branch) it into smaller subproblems. For the fathoming test to be valid, the upper bound function u should have the property

$$b(J, C, E) \geq f(J', C') \quad \forall\, (J', C') \in P(J, C, E). \tag{4}$$

2.1 The upper bound function

In the special case that $F = \emptyset$, the set $P(J, C, E)$ is a singleton consisting only of (J, C), and we can take $b(J, C, E) = f(J, C)$. If $F \neq \emptyset$, we must use some other function satisfying (4). Goldberg and Shan [13] suggested the simple bound $b(J, C, E) = b_{\text{gs}}(J, C)$, where

$$b_{\text{gs}}(J, C) = \max\{w(\text{Cover}(J, C) \cap \Omega^+), w(\text{Cover}(J, C) \cap \Omega^-)\} \tag{5}$$

Theorem 2.1. *The bound* (5) *satisfies property* (4).

Proof. Let $(J', C') \in P(J, C, E)$. Then by definition $J' \supseteq J$ and $C' \supseteq C'$, and clearly $\text{Cover}(J', C') \subseteq \text{Cover}(J, C)$. Using this inclusion and the nonnegativity of the weight function w, we have

$$\begin{aligned} b_{\text{gs}}(J, C) &= \max\{w(\text{Cover}(J, C) \cap \Omega^+), w(\text{Cover}(J, C) \cap \Omega^-)\} \\ &\geq \max\{w(\text{Cover}(J', C') \cap \Omega^+), w(\text{Cover}(J', C') \cap \Omega^-)\} \\ &\geq \max \left\{ \begin{array}{l} w(\text{Cover}(J', C') \cap \Omega^+) - w(\text{Cover}(J', C') \cap \Omega^-), \\ w(\text{Cover}(J', C') \cap \Omega^-) - w(\text{Cover}(J', C') \cap \Omega^+) \end{array} \right\} \\ &= f(J', C'). \end{aligned}$$

□

However, this bound essentially ignores the information in the set E. To obtain a stronger bound, we now exploit the information in E:

Definition 2.2. *Two binary vectors* $x, y \in \{0, 1\}^N$ *are* inseparable *with respect to a set* $E \subseteq \{1, ..., N\}$*, if, for all* $j \in \{1, ..., N\} \setminus E$*, one has* $x_j = y_j$*.*

Inseparability is an equivalence relation: any set $E \subseteq \{1, \ldots, N\}$ partitions $\{1, ..., M\}$ into equivalence classes, i and i' being in the same class when the observation A_i is inseparable from the observation $A_{i'}$. Let us denote these classes $V_1^E, V_2^E, ..., V_{q(E)}^E$, where $1 \leq q(E) \leq m$; Figure 1 gives an example of such a partitioning. Let

$$\begin{aligned} w_\eta^+(J, C, E) &= w(V_\eta^E \cap \text{Cover}(J, C) \cap \Omega^+) \qquad & w^+(J, C) &= w(\text{Cover}(J, C) \cap \Omega^+) \\ w_\eta^-(J, C, E) &= w(V_\eta^E \cap \text{Cover}(J, C) \cap \Omega^-) \qquad & w^-(J, C) &= w(\text{Cover}(J, C) \cap \Omega^-) \end{aligned}$$

for $\eta = 1, \ldots, q(E)$, and note that

$$f(J,C) = \left|w^+(J,C) - w^-(J,C)\right| = \left|\sum_{\eta=1}^{q(E)} (w_\eta^+(J,C,E) - w_\eta^-(J,C,E))\right|. \tag{6}$$

We will call a monomial *positive* if $w^+(J,C) \geq w^-(J,C)$, and *negative* if $w^+(J,C) < w^-(J,C)$. Positive monomials cover more weight of positive than negative observations, and negative monomials the reverse; we include ties in the positive class.

Now consider some $(J',C') \in P(J,C,E)$, assume it is positive, and take $\eta \in \{1,\ldots,q(E)\}$. Because the observations in the equivalence class V_η^E are inseparable with respect to E, the monomial $m_{J',C'}(x)$ must either cover all of V_η^E, or cover none of it. Considering the η^{th} term in the second expression in (6), we see that the largest value of $f(J',C')$ would result if $m_{J',C'}(x)$ covers the entire class V_η^E whenever $w_\eta^+(J,C,E) \geq w_\eta^-(J,C,E)$, obtaining a value of $w_\eta^+(J,C,E) - w_\eta^-(J,C,E)$, and otherwise does not cover the entire class V_η^E, obtaining the value 0. Thus, if (J',C') is positive, we have

$$f(J',C') \leq \sum_{\eta=1}^{q(E)} \Big(w_\eta^+(J,C,E) - w_\eta^-(J,C,E)\Big)_+.$$

By nearly identical reasoning, we can obtain a similar relation for the case that (J',C') is negative, and combining the two cases, we derive the following upper bound function satisfying (4):

$$b(J,C,E) = \max\left\{ \begin{array}{l} \sum_{\eta=1}^{q(E)} \big(w_\eta^+(J,C,E) - w_\eta^-(J,C,E)\big)_+, \\ \sum_{\eta=1}^{q(E)} \big(w_\eta^-(J,C,E) - w_\eta^+(J,C,E)\big)_+ \end{array} \right\}. \tag{7}$$

A similar use of equivalence classes to obtain tightened subproblem bounds has been used by De Bontridder *et al.* [2], in the context of the minimum test cover problem. Furthermore, we can also rewrite the upper bound $b(J,C,E)$ in terms of the simple upper bound (5).

Theorem 2.3. *The upper bound* (7) *can be equivalently written as,*

$$b(J,C,E) = b_{\text{gs}}(J,C) - \sum_{\eta=1}^{q(E)} \min\{w_\eta^+(J,C,E), w_\eta^-(J,C,E)\}. \tag{8}$$

Proof. Setting $w_\eta^+ = w_\eta^+(J,C,E)$, $w_\eta^- = w_\eta^-(J,C,E)$, and $q = q(E)$ for brevity,

$$\begin{aligned} &\max\left\{\sum_{\eta=1}^{q}(w_\eta^+ - w_\eta^-)_+,\ \sum_{\eta=1}^{q}(w_\eta^- - w_\eta^+)_+\right\} \ +\ \sum_{\eta=1}^{q}\min\{w_\eta^+, w_\eta^-\} \\ &= \max\left\{\sum_{\eta=1}^{q}(w_\eta^+ - w_\eta^-)_+ + \sum_{\eta=1}^{q}\min\{w_\eta^+, w_\eta^-\},\ \sum_{\eta=1}^{q}(w_\eta^- - w_\eta^+)_+ + \sum_{\eta=1}^{q}\min\{w_\eta^+, w_\eta^-\}\right\} \end{aligned}$$

i \ j	1	2	3	4
1	0	0	1	1
2	1	0	0	1
3	1	0	1	1
4	0	1	1	0
5	1	0	0	0

				E	
		j	1	3	2	4
η	i					
1	1		0	1	0	1
	4		0	1	1	0
2	2		1	0	0	1
	5		1	0	0	0
3	3		1	1	0	1

Figure 1: Illustration of inseparability and equivalence classes. Suppose $M = 4$, $N = 5$, $E = \{2, 4\}$, and A is the matrix shown on the left. The table on the right shows A's rows sorted lexicographically by the contents of columns $\{1, \ldots, N\} \setminus E = \{1, 3\}$. Inseparability with respect to E, partitions the rows of A into the three equivalence classes shown.

$$
\begin{aligned}
&= \max \left\{ \begin{array}{l} \sum_{\eta:\, w_\eta^+ \geq w_\eta^-} (w_\eta^+ - w_\eta^- + w_\eta^-) \;+\; \sum_{\eta:\, w_\eta^+ < w_\eta^-} w_\eta^+, \\ \sum_{\eta:\, w_\eta^- \geq w_\eta^+} (w_\eta^- - w_\eta^+ + w_\eta^+) \;+\; \sum_{\eta:\, w_\eta^- < w_\eta^+} w_\eta^- \end{array} \right\} \\
&= \max\{w^+(J, C), w^-(J, C)\} = b_{\text{gs}}(J, C). \qquad \square
\end{aligned}
$$

We define $\phi(J, C, E) = \sum_{\eta=1}^{q(E)} \min\{w_\eta^+(J, C, E), w_\eta^-(J, C, E)\}$, the second term in (8), to be the *inseparability* of E with respect to J and C. Unless $\phi(J, C, E) = 0$, the bound $b(J, C, E)$ is strictly tighter than $b_{\text{gs}}(J, C)$. If $\phi(J, C, E) = 0$, then all the sets $\text{Cover}(J, C) \cap V_\eta^E$, for $\eta = 1, \ldots, q(E)$, are homogeneous with respect to observations with nonzero weight, meaning that among the observations $i \in V_\eta^E \cap \text{Cover}(J, C)$ with $w(i) > 0$, all are either in Ω^+, or all are in Ω^-.

Both of the bounds (5) and (7) may be computed in $O(MN)$ time in the worst case. For (7), the equivalence classes $\{V_\eta^E\}$ need to be computed, which can be done in $O(MN)$ time, by applying a radix sort to the rows of the submatrix of A with columns $\{1, \ldots, N\} \setminus E$. A numerical example of the procedure for identifying the equivalence classes is shown in Figure 1. The radix sort algorithm, in fact, has $\Theta(MN)$ running time. Computing $b_{\text{gs}}(J, C)$, on the other hand, requires only the computation of the set $\text{Cover}(J, C)$, which involves $|J \cup C| \leq N$ set intersection operations, each of which tends to be much faster than $O(M)$ in practice. Thus, it may be more practically efficient to first compute $b_{\text{gs}}(J, C)$, and if that does not fathom the subproblem (J, C, E, F), then compute $\phi(J, C, E)$ and thus (7). Further, it is also possible to use any lower bound for $\phi(J, C, E)$ in order to obtain a tightening of the upper bound b_{gs}. Lower bounds of $\phi(J, C, E)$ can be iteratively computed as $\sum_{\eta=1}^{\bar{\eta}} \min\{w_\eta^+(J, C, E), w_\eta^-(J, C, E)\}$, for $\bar{\eta} = 1, \ldots, q(E)$, until either the subproblem (J, C, E, F) is fathomed, or $\bar{\eta} = q(E)$.

2.2 The branching procedure

Our branching procedure works as follows: given a subproblem (J, C, E, F), we select k distinct elements $j_1, \ldots, j_k$ of F, where $1 \leq k \leq |F|$. In our branching step, we create $2k + 1$

smaller subproblems respectively representing the following subsets of $P(J, C, E)$:

- The subset of monomials in which none of $x_{j_1}, \ldots, x_{j_k}$ appear.
- For $t = 1, \ldots, k$, the subset of monomials in which x_{j_t} is the first in the sequence $x_{j_1}, \ldots, x_{j_k}$ to appear, in the uncomplemented form; $x_{j_1}, \ldots, x_{j_{t-1}}$ are excluded from further consideration.
- For $t = 1, \ldots, k$, the subset of monomials in which x_{j_t} is the first in the sequence $x_{j_1}, \ldots, x_{j_k}$ to be used, and appears in the complemented form $(1 - x_{j_t})$; $x_{j_1}, \ldots, x_{j_{t-1}}$ are excluded from further consideration.

These subsets clearly form a partition of $P(J, C, E)$. In our notation, the corresponding subproblems are represented by the respective 4-tuples in lines 13, 15, and 16 of Algorithm 1 on page 9. We have significant latitude in choosing k for each subproblem. We have experimented with choosing $\alpha \in \{1/4, 1/2, 3/4, 1\}$, and setting $k = \lceil \alpha |F| \rceil$ for each subproblem; we have also experimented with fixing $k = 1$.

Motivated by (8), we attempt for a given k to choose $j_1, \ldots, j_k$ to maximize the inseparability $\phi(J, C, E \cup \{j_1, \ldots, j_k\})$. We now show that exactly maximizing $\phi(J, C, E \cup \{j_1, \ldots, j_k\})$ is itself $\mathcal{NP}$-hard. We will show that even the case $J = C = \emptyset$ and $w(i) = 1$ for all $i \in \{1, \ldots, M\}$ is $\mathcal{NP}$-hard.

Theorem 2.4. *The problem of finding a set $S \subseteq \{1, \ldots, N\}$, of size k, that maximizes $\hat{\phi}(S) = \phi(\emptyset, \emptyset, S)$ is $\mathcal{NP}$-hard.*

Proof. The proof follows by reduction of the clique problem in a graph. Given a graph $G = (V, E)$ and an integer parameter $k \leq |V|$, the clique problem is to find whether there exists a subset $S \subseteq V$ such that $|S| \geq k$ and every two vertices in S are joined by an edge in E. The dense k-subgraph problem is the problem of finding $S \subseteq V$, such that $|S| = k$ and that the maximum number of edges join vertices in S. The clique decision problem can be solved by the dense k-subgraph optimization problem by simply checking whether the optimal objective value equals $\binom{k}{2}$.

Let $N = |V|$. For each edge $(u, v) \in E$, we create one element $i \in \Omega^+$ and a vector A_i such that

$$A_{ij} = \begin{cases} 1, & j = u \text{ or } j = v \\ 0, & \text{otherwise} \end{cases},$$

and another element $i' \in \Omega^-$ such that $A_{i'} = 0$, letting $w(i) = w(i') = 1$.

Let $q(S)$ denote the number of equivalence classes induced by the set $S \subseteq \{1, \ldots, N\}$. Note that since $A_i = 0$ for all $i \in \Omega^-$, then either $w_\eta^-(\emptyset, \emptyset, S) = 0$ or $w_\eta^-(\emptyset, \emptyset, S) = |E|$. Without loss of generality, we assume that $\eta = 1$ denotes the class index corresponding to the zero vector, and hence $w_1^- = |E|$.

$$\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{\hat{\phi}(S)\} = \max_{S \subseteq \{1,\ldots,N\}:|S|=k} \left\{ \sum_{\eta=1}^{q(S)} \min\{w_\eta^+(\emptyset, \emptyset, S), w_\eta^-(\emptyset, \emptyset, S)\} \right\}$$

$$= \max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{w_1^+(\emptyset,\emptyset,S)\}$$

The last equality follows because $w_\eta^-(\emptyset,\emptyset,S) = 0$ for $2 \leq \eta \leq q(S)$, and $w_1^+(J,C,S) \leq w_1^-(J,C,S) = |E|$. Now,

$$\begin{aligned} w_1^+(\emptyset,\emptyset,S) &= \left|\left\{i \in \Omega^+ \mid \forall j \in \{1,\ldots,N\} \setminus S : A_{ij} = 0\right\}\right| \\ &= \left|\{(u,v) \in E \mid u \in S \wedge v \in S\}\right|. \end{aligned}$$

The equivalence class 1 consists of vectors indistinguishable from 0 when excluding S. For the observations in Ω^+, these are precisely the vectors with both their 1's in the vector positions with indices in S, that is, those corresponding to edges (u,v) with $u,v \in S$. Thus, maximizing $\hat{\phi}(S)$ subject to $|S| = k$ solves the dense k-subgraph problem, and

$$\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{w_1^+(\emptyset,\emptyset,S)\} = \binom{k}{2}$$

if and only if there exists a clique of size k in G. □

In the reduction of our proof, $\hat{\phi}(S)$ equals the number of edges joining the set of vertices S, implying that the reduction of the dense k-subgraph is approximation-preserving. Thus, based on an inapproximability result of Khot for the dense k-subgraph problem [15], we can also make the following claim:

Theorem 2.5. *There is no polynomial time approximation scheme (PTAS) for the problem* $\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \hat{\phi}(S)$*, unless* $\mathcal{NP} \subseteq \bigcap_{\epsilon>0} BPTIME(2^{N^\epsilon})$.[1]

Given that the problem of maximizing $\phi(J,C,E)$ subject to a cardinality constraint on E is $\mathcal{NP}$-hard, we consider selecting the k features to add to E by some form of greedy heuristic. For any sets $S \subset E \subset \{1,\ldots,N\}$,

$$\begin{aligned} & \left|\left\{(i,i') \in \Omega^+ \times \Omega^- \mid A_{ij} = A_{i'j}\ \forall j \in \{1,\ldots,N\} \setminus (S \cup \{j'\})\right\}\right| \\ \leq\ & \left|\left\{(i,i') \in \Omega^+ \times \Omega^- \mid A_{ij} = A_{i'j}\ \forall j \in \{1,\ldots,N\} \setminus (E \cup \{j'\})\right\}\right|. \end{aligned}$$

That is, excluding an additional variable j' may induce fewer inseparable observation pairs when only a subset of the variables E are excluded to begin with. In fact, when $E = \emptyset$, selecting any single variable to add to E is unlikely to make many pairs inseparable or to give any indication about the number of pairs that can be made inseparable with the "proper" combination with additional variables to include in E. Intuitively, this observation

[1] BPTIME(t) is the class of decision problems solvable by an t-time probabilistic Turing machine such that:

1. If the answer is "yes" then at least 2/3 of the computation paths accept.
2. If the answer is "no" then at most 1/3 of the computation paths accept.

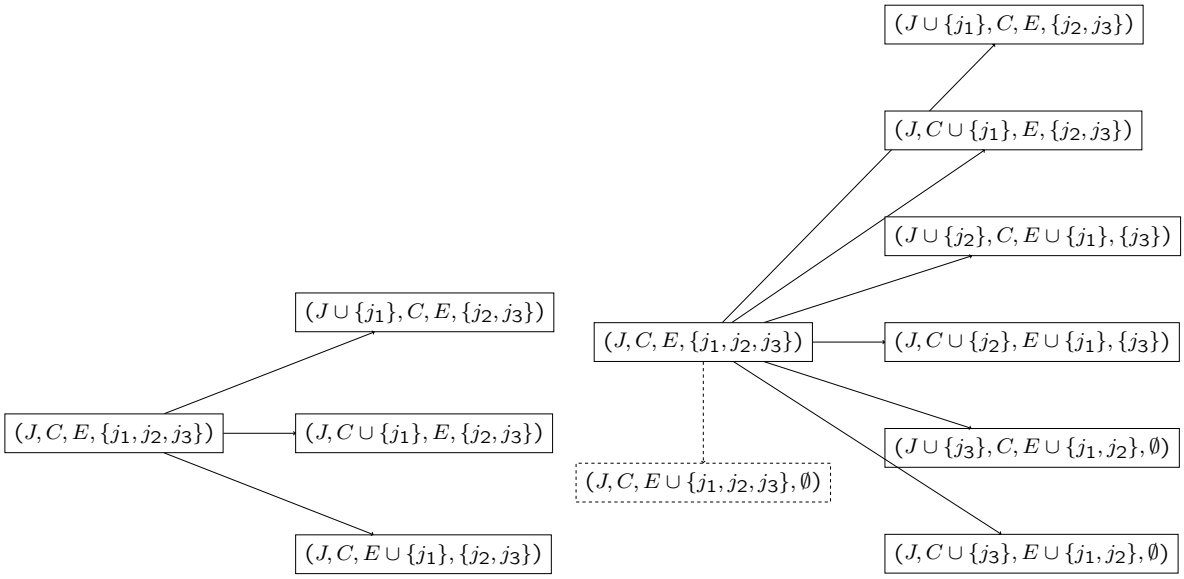
Algorithm 1

Input: Sets Ω^+, Ω^-, and $M \times N$ matrix A.
Output: Best solution value l and corresponding monomial given by (J^*, C^*).
$Q \leftarrow \{(\emptyset, \emptyset, \emptyset, \{1 \ldots, N\})\}$
$l \leftarrow -\infty$
while $Q \neq \emptyset$ **do**
  Remove subproblem $S = (J, C, E, F)$ from Q
  if $b(J, C, E) > l$ **then**
    if $f(J, C) > l$ **then**
      $(J^*, C^*) \leftarrow (J, C)$
      $l \leftarrow f(J^*, C^*)$
    end if
    $(j_1, ..., j_k) \leftarrow exclude(J, C, F)$
    Insert $(J, C, E \cup \{j_1, ..., j_k\}, F \setminus \{j_1, ..., j_k\})$ into Q
    for $t \in \{1, \ldots, k\}$ **do**
      Insert $(J \cup \{j_t\}, C, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ into Q
      Insert $(J, C \cup \{j_t\}, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ into Q
    end for
  end if
end while

motivates our use of a reverse (also known as worst-out) greedy heuristic, instead of the forward greedy algorithm. Conceptually, we set $E' \leftarrow F$, and then remove elements one-by-one from E', greedily with respect to maximizing $\phi(J, C, E \cup E')$, until $|E'| = k$. We then take $\{j_1, \ldots, j_k\} = E'$, and determine the ordering $j_1, \ldots, j_k$ by a similar reverse greedy procedure. We found this algorithm much more effective in practice than the forward greedy algorithm.

We now consider the case $k = 1$, in which case our branching scheme generates three children. In this case, we implement an alternative, "strong" branching method: for each possible branching feature $j \in F$, we calculate the bounds $b(J \cup \{j\}, C, E)$, $b(J, C \cup \{j\}, E)$, and $b(J, C, E \cup \{j\})$ of the three resulting subproblems. For each j, we place these bounds in a triple $\mathbf{b}_j$ with elements sorted in descending order, and branch on some feature j that lexically minimizes $\mathbf{b}_j$. This approach is not only faster than the reverse greedy method, but also more practically effective at pruning search nodes. One reason for the efficiency of this alternative branching strategy is that it considers the bounds of all three prospective children, and not just the bound of the child $(J, C, E \cup \{j\}, F \setminus \{j\})$. Algorithm 1 outlines our entire branch-and-bound algorithm; there, our procedure for choosing $j_1, \ldots, j_k$ is called *exclude*.

The two most extreme branching strategies, with $k = 1$ and with $k = |F|$, are illustrated in Figure 2. We note that in the earlier branch-and-bound method of [13], the branching procedure is essentially the special case $k = |F|$, as illustrated in Figure 2(b), with $j_1, \ldots, j_k$



(a) Ternary branching with $k = 1$.

(b) Branching with $k = |F|$ implies a $2K + 1$ branching factor. In the previous algorithm [13] the ordering j_1, j_2, j_3 is statically fixed to satisfy $j_1 < j_2 < j_3$.

Figure 2: Illustration of branching for a subproblem (J, C, E, F) with $|F| = 3$, that is, three variables that are "free", and the two most extreme branching strategies: $k = 1$ and $k = |F|$.

ordered so that $j_1 < j_2 < \cdots < j_k$. When $k = |F|$, the $(J, C, E \cup \{j_1, \ldots, j_k\}, F \backslash \{j_1, \ldots, j_k\})$ child always represents exactly one monomial, allowing it to be immediately evaluated and implicitly dropped from the search tree. With this branching strategy, it can be shown that the maximum size of the search tree that can evolve, assuming no fathoming, is 3^N, which is exactly the number of possible monomials. Conceptually, this feature is a possible advantage of such a branching procedure; on the other hand, this approach results in a very high branching factor, especially near the root. When one also uses a predetermined ordering of $j_1, \ldots, j_k$, as in [13], the branching scheme is essentially static: the set of children developed from a given subproblem does not depend on the contents of the data matrix A, but only on its number of columns N.

Here, we suggest branching decisions that make use the input data and are designed to tighten the bounds computed for the resulting child subproblems. Particularly, for $k = 1$, illustrated in Figure 2(a), the maximum tree size is $(3^{N+1} - 1)/2 > 3^N$, but the benefits from the smaller branching factor and flexibility of variable selection will become clear in the experiments below.

In lines 15 and 16 of Algorithm 1, we apply a few simple pruning rules that provide further improvement in practical performance: first, we need not insert a subproblem into Q if it is already fathomed. Second, if $\text{Cover}(J \cup \{j_t\}, C) = \text{Cover}(J, C)$, we have that $m_{J \cup \{j_t\}, C}(A_i) = A_{ij_t} m_{J,C}(A_i) = m_{J,C}(A_i)$ for all $i \in \text{Cover}(J, C)$. Thus, $A_{ij_t} = 1$ for all $i \in$

Cover(J, C). Consider any $(J', C') \in P(J, C, E)$. Now, since Cover$(J', C') \subseteq$ Cover(J, C), we have $f(J', C') = f(J' \cup \{j_t\}, C')$. Thus, such a subproblem may be dropped from consideration without insertion into Q. Finally, a similar result holds for the subproblem $(J, C \cup \{j_t\}, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ when Cover$(J, C \cup \{j_t\}) =$ Cover(J, C).

3 Experimental study

We implemented Algorithm 1 in C++ using the open-source PEBBL branch-and-bound library [9]. We ran our experiments on a workstation with 3.00 GHz Intel x5400 Xeon processors and 800MHz memory (running only in serial, although PEBBL is capable of parallelism).

Table 1 and Figures 4-8 show our experimental results using Algorithm 1 as a weak learning algorithm inside the LP-Boost boosting procedure [6], applied to the UCI [1] binary dataset SPECTHRT and binarized versions of additional UCI datasets. We configured our binarization procedure to obtain a larger number of variables (as indicated by N in the table) than is customary for most binary feature selection methods (*e.g.* [3]).

The LP-Boost algorithm finds a separating hyperplane that maximizes the "soft margin" by solving the large-scale LP formulation:

$$\begin{array}{lll} \max & \rho - D\sum_{i=1}^{M} \xi_i & \\ \text{s.t.} & y_i H_i \lambda + \xi_i \geq \rho & i = 1, ..., M \\ & \sum_{u=1}^{U} \lambda_u = 1 & \\ & \xi_i, \lambda_u \geq 0 & i = 1, ..., M,\ u = 1, ..., U \end{array}$$

where $H \in \{-1, 0, 1\}^{M \times U}$ is a matrix whose rows correspond to observations and, in our case, whose columns correspond to monomial classifiers of the form $m_{J,C}(\cdot)$ or $-m_{J,C}(\cdot)$, so that $U = 2 \cdot 3^N$. Also, H_i is the i^{th} row of H. For the formulation's penalty parameter, we selected $D = 3/M$, which in our computational experience provided good classification performance. We compared four different algorithm configurations:

- An algorithm equivalent to [13], in which $k = |F|$, using only the b_{gs} upper bound function
- Algorithm 1, with $k = |F|$ and the bound (7)
- Algorithm 1, with $k = \lceil |F| / 2 \rceil$ and the bound (7).
- Algorithm 1, with $k = 1$, the bound (7), and the "strong" lexical $k = 1$ branching rule described above. Note that $k = 1$ corresponds to a ternary search tree.

t

Dataset	LP-Boost Iters	b_{gs} bound $k = \|F\|$ CPU Sec	b_{gs} bound $k = \|F\|$ BB Nodes	$k = \|F\|$ CPU Sec	$k = \|F\|$ BB Nodes	$k = \lceil \|F\|/2 \rceil$ CPU Sec	$k = \lceil \|F\|/2 \rceil$ BB Nodes	lexical strong branch $k = 1$ CPU Sec	lexical strong branch $k = 1$ BB Nodes
SPECTHRT	1-15	0.6	7791.5	0.2	21.5	0.2	22.4	0.1	50.5
$N = 22$	16-30	1.8	22751.1	0.4	74.6	0.4	78.5	0.3	162.9
CLVHEART	1-15	29.7	89551.2	16.9	626.2	17.3	654.7	9.5	1638.3
$N = 35$	16-30	99.8	317537.9	40.7	1872.3	41.2	1941.6	23.6	4831.6
HEPATITIS	1-15	17.3	83365.6	7.2	442.5	7.3	464.7	3.2	917.1
$N = 37$	16-30	Q LIMIT		13.5	970.9	13.8	1021.5	7.3	2650.3
PIMA	1-15	Q LIMIT		89.2	1290.5	90.0	1323.2	66.0	4606.3
$N = 33$	16-30	Q LIMIT		269.9	5499.3	269.6	5582.7	224.7	20907.1
CMC	1-15	Q LIMIT		1161.0	969.3	1158.6	972.7	496.9	3647.3
$N = 58$	16-30	Q LIMIT		LIMIT		LIMIT		1738.6	19437.3
HUHEART	1-15	Q LIMIT		LIMIT		LIMIT		264.0	14169.3
$N = 72$	16-30	Q LIMIT		LIMIT		LIMIT		763.6	47657.2

Table 1: Runtime and node averages over the specified LP-Boost [6] iterations, applying our algorithm to binarized UCI datasets [1]. "Q LIMIT" indicates an iteration encountered the 500,000-node queue limit, and "LIMIT" indicates an iteration encountered the one-hour time limit.

We used a best-first queueing discipline with the queue size limited to at most 500,000 subproblems. We also limited the CPU time of each branch-and-bound search to 60 minutes. LP-Boost starts with the weights $w(i)$ equal for all i, but subsequently adjusts the observation weights based on the dual variables of its LP formulation's separation constraints. We ran each case for 30 iterations, or until an iteration encountering the queue size or branch-and-bound time limit. As is well-known in practice, boosting algorithms tend to focus their later iterations on observations that are more difficult to classify. In our case, the later iterations produce longer monomials whose $|\text{Cover}(J, C)|$ is smaller. The later iterations also tend to have larger search trees and longer running times.

Table 1 shows the average CPU time and number of search nodes over iterations 1–15 and 16–30, for $k = 1, \lceil |F|/2 \rceil$ and $k = |F|$ with the simple bound (5) and the bound (7). Figures 4–8 show a plot of the actual CPU time and number of search nodes over all iterations, and also display the performance for $k = \lceil |F|/4 \rceil$ and $k = \lceil 3|F|/4 \rceil$. Two conclusions appear to follow from the results in Table 1 and Figures 4–8. First, compared with [13], both our tighter bound and our new reverse greedy branching scheme significantly decrease the number of search nodes required; they allow larger problems to be solved, and improve running time in all but the easiest cases involving SPECTHRT. Second, choosing an intermediate number of branching features $k = \lceil |F|/2 \rceil$ performs better in terms of search nodes than $k = 1$. In Figures 4–8, it is also evident that $k \geq \lceil |F|/2 \rceil$ yields smaller search trees and faster running times than $k = \lceil |F|/4 \rceil$. There does not seem to be any material difference in

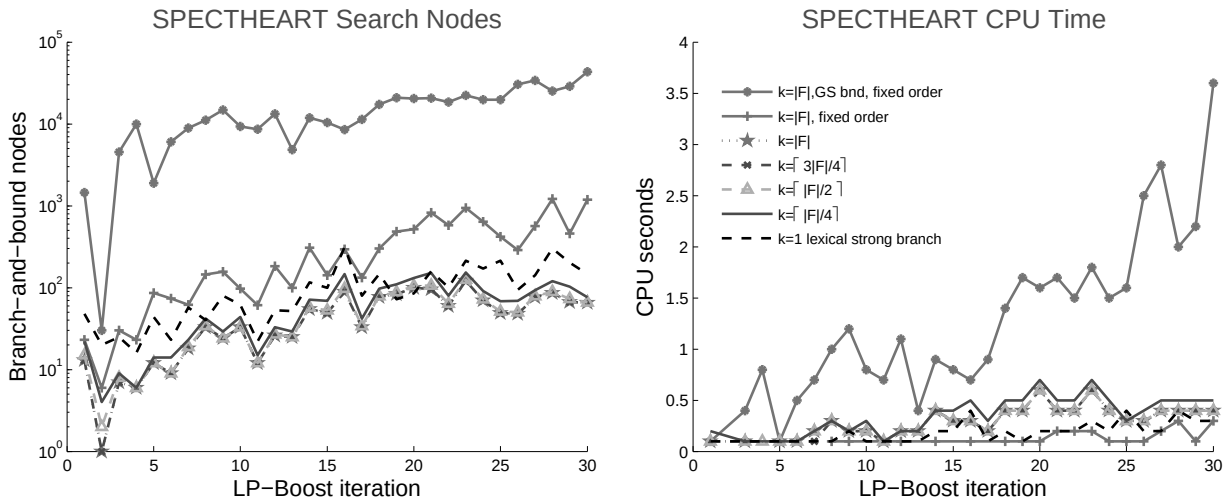


Figure 3: MMA computational performance on the UCI SPECTHEART dataset.

the performance of algorithm variants with $k = |F|$, $k = 3\lceil |F|/4 \rceil$ and $k = \lceil |F|/2 \rceil$, when ordering the subproblems using the reverse greedy procedure. Finally, although taking $k = 1$ is not the best strategy in terms of search tree size, its specialized, faster branching procedure performs well enough that $k = 1$ is clearly best in terms of runtime. Thus, our new bound coupled with the $k = 1$ ternary branching scheme significantly outperform the method of [13] for larger datasets, and in later boosting iterations, when the weights become more "difficult".

References

[1] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007.

[2] Koen M. J. De Bontridder, B. J. Lageweg, Jan K. Lenstra, James B. Orlin, and Leen Stougie. Branch-and-bound algorithms for the test cover problem. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pages 223–233, London, UK, 2002. Springer-Verlag.

[3] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[4] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, 2000.

[5] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 335–342, 1999.

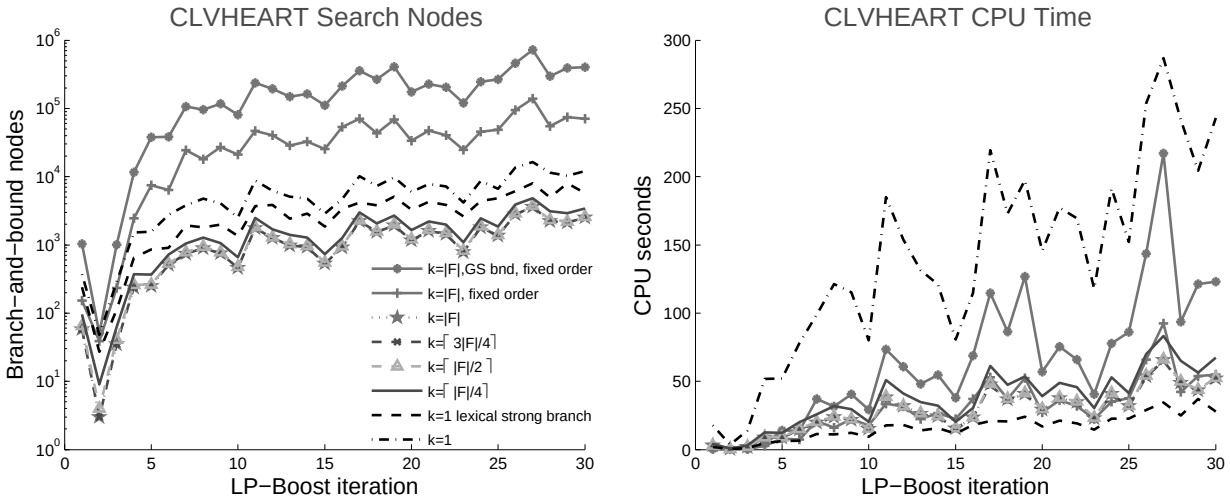


Figure 4: MMA computational performance on the UCI CLVHEART dataset.

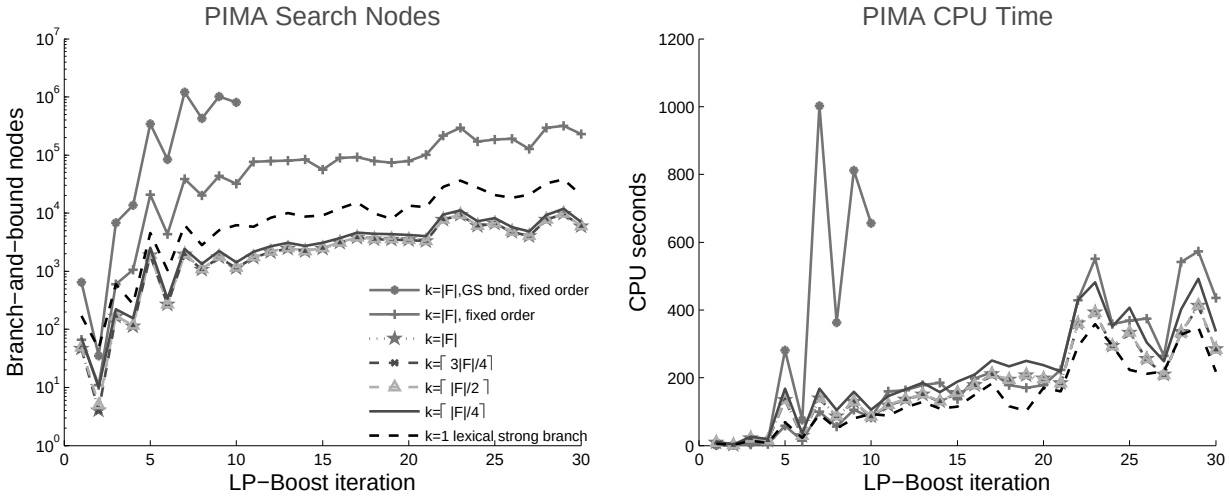


Figure 5: MMA computational performance on the UCI PIMA dataset.

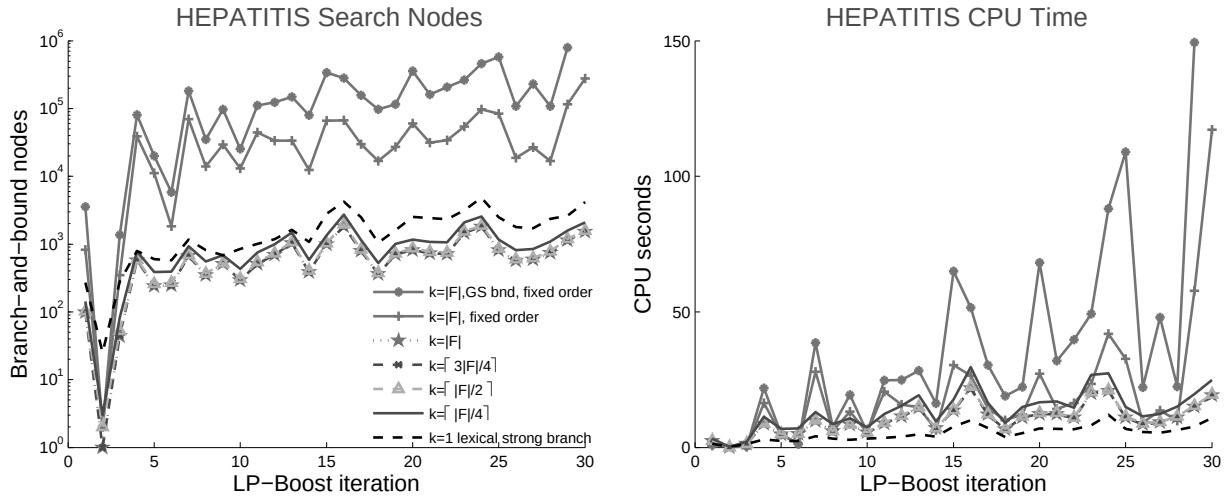


Figure 6: MMA computational performance on the UCI HEPATITIS dataset.

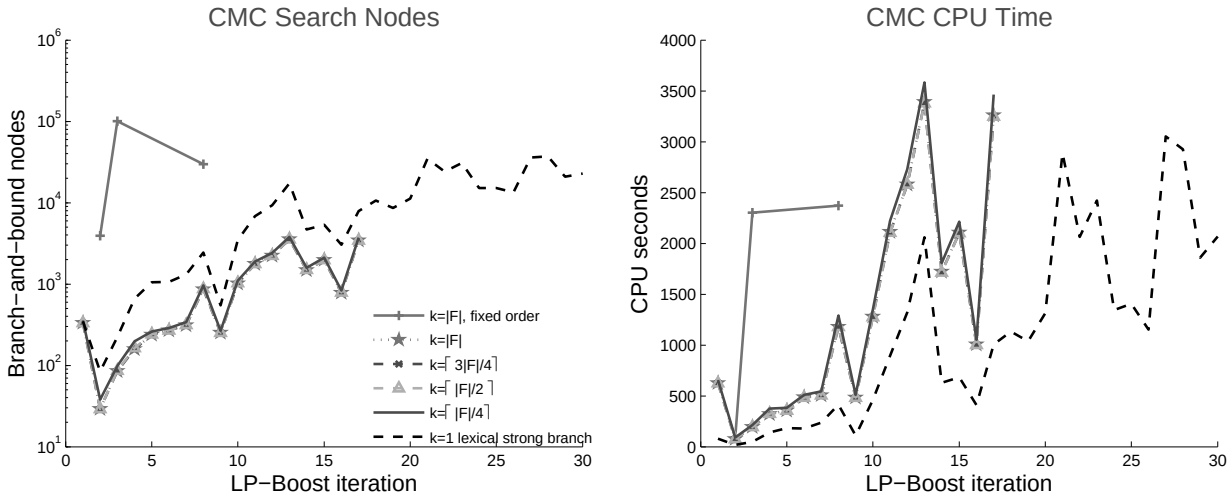


Figure 7: MMA computational performance on the UCI CMC dataset.

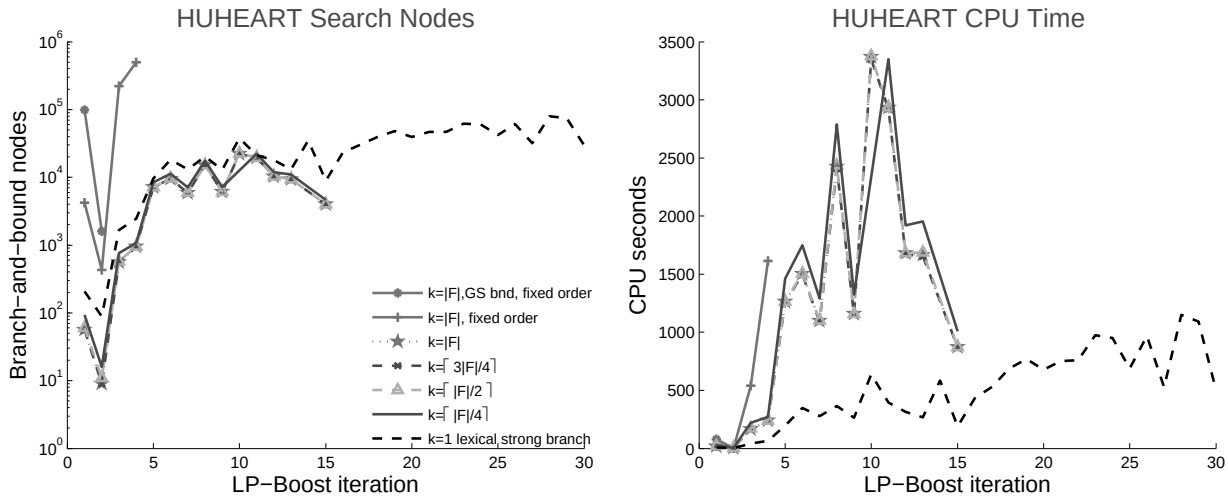


Figure 8: MMA computational performance on the UCI HUHEART dataset.

[6] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[7] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and Systems Sciences*, 52(3):453–470, 1996.

[8] Jonathan Eckstein, Peter L. Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

[9] Jonathan Eckstein, Cynthia A. Phillips, and William E. Hart. PEBBL 1.0 user guide. RUTCOR Research Report RRR 19-2006, RUTCOR, Rutgers University, 2006.

[10] Vitaly Feldman. Optimal hardness results for maximizing agreements with monomials. In *Proceedings of the Annual IEEE Conference on Computational Complexity*, pages 226–236, 2006.

[11] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[12] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–954, 2008.

[13] Noam Goldberg and Chung-chieh Shan. Boosting optimal logical patterns. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, 2007.

[14] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[15] Subhash Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[16] Gunnar Rätsch, Bernhard Schölkopf, Alex J. Smola, Sebastian Mika, Takashi Onoda, and Klaus-Robert Müller. Robust ensemble learning. In Alex J. Smola, Peter J. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000.

[17] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.