DIMACS Technical Report 2000-10
March 2000

Finding Small Sets of Essential Attributes in Binary Data

by

Endre Boros[1] Takashi Horiyama[2] Toshihide Ibaraki[3]
Kazuhisa Makino[4] Mutsunori Yagiura[5]

[1]Permanent Member; RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway New Jersey 08854. (boros@rutcor.rutgers.edu)
[2]Graduate School of Information Science, Nara Institute of Science and Technology, Nara 630-0101, Japan. (horiyama@is.aist-nara.ac.jp)
[3]Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan. (ibaraki@i.kyoto-u.ac.jp)
[4]Department of Systems and Human Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka, 560-8531, Japan. (makino@sys.es.osaka-u.ac.jp)
[5]Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan. (yagiura@amp.i.kyoto-u.ac.jp)

*This work was partially supported by the Grants in Aid by the Ministry of Education, Science, Sports and Culture of Japan (Grants 09044160 and 10205211). The visit of the first author to Kyoto University (January to March, 1999) was also supported by this grant (Grant 09044160). The research of the first and third authors were supported in part by the Office of Naval Research (Grant N00014-92-J-1375). The first author thanks also the National Science Foundation (Grant DMS 98-06389) for partial support.

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We consider the problem of finding support sets (i.e., sets of essential attributes) in a given data set, which consists of n-dimensional binary vectors of positive examples and negative examples. A set of attributes is a support set if positive examples and negative examples can be separated by using only the attributes in the set. Finding small support sets is an important topic in such fields as knowledge discovery, data mining, learning theory and logical analysis of data. Based on several measures of separation, we discuss why finding small support sets is important, and how to find such sets, together with results of some computational experiment. Theoretical analysis of the approximation ratios of the proposed algorithms is also provided.

1 Introduction

We consider the problem of analyzing a data set consisting of positive and negative examples for an unknown phenomenon. We denote by T the set of positive examples, and by F the set of negative examples. It is further assumed that each example is given as a binary n-dimensional vector. This is a typical problem setting studied in such fields as knowledge discovery, data mining, learning theory and logical analysis of data [3, 1, 7, 13, 15, 26, 30, 31].

Let $\mathbb{B} = \{0, 1\}^n$. A pair $(T, F), T, F \in \mathbb{B}^n$, is called a *partially defined Boolean function* (pdBf), where $a \in T$ (resp., $b \in F$) are called true vectors (resp., false vectors). Usually the sets T and F are assumed to be disjoint, i.e., $T \cap F = \emptyset$, and fully specified Boolean functions f that separate T and F (i.e., for which $f(a) = 1$ for all $a \in T$ and $f(b) = 0$ for all $b \in F$) are sought [8, 9, 13], as such functions provide logical explanations of the phenomenon represented by the data set (T, F).

For instance, a data vector x may represent the symptoms to diagnose a disease, e.g., x_1 denotes whether temperature is high ($x_1 = 1$) or not ($x_1 = 0$), and x_2 denotes whether blood pressure is high ($x_2 = 1$) or not ($x_2 = 0$), etc. Establishing an *extension* f, which is consistent with the given data set, then amounts to finding a diagnostic explanation.

In general, a Boolean function f separating T and F can depend on all the n attributes that correspond to the n elements of data vectors. However, it is quite common that the data set contains attributes which are *irrelevant* to the phenomenon under consideration. Such attributes may only play the role of noise in the data, and can be removed from consideration. There may also be attributes which are redundant in the sense that their values are *dependent* on other attributes in the data set, and hence they can also be removed without losing the information of the data set. In this respect, it is an interesting problem to find a minimal set of *essential attributes*, which still can explain the given data set (T, F) without using the rest of nonessential attributes. Such a set of essential attributes then gives a compact Boolean function f that explains pdBf (T, F), and is in the spirit of Occam's razor [6]. In some applications, such a set will be helpful to reduce the cost of collecting the data. Finding a small subset of the variables, which explains the input data as much as possible, is a frequent subject of studies in computational learning theory. Models, like the so called "attribute-efficient learning" consider learning problems in which the target concept depend only on a small number of the attributes, and the objective is to find learning algorithms, the complexity of which depends only at a very low degree on the (large) number of irrelevant attributes (see e.g. [23, 10, 5, 14]).

Finding the relevant (or essential) attributes in the presence of large number of irrelevant and/or dependent attributes is the main problem we address in this paper.

Let $V = \{1, 2, \ldots, n\}$ denote the set of indices of all attributes, and let S be a subset of V. For a vector $a \in \mathbb{B}^n$, $a[S]$ denotes the projection of a on S, and, for a set of vectors X, we denote

$$X[S] = \{a[S] \mid a \in X\}.$$

A set $S \subseteq V$ is called a *support set* of a pdBf (T, F) if

$$T[S] \cap F[S] = \emptyset \tag{1}$$

holds (see [13].) Simply stated, the goal of this paper is to find one of the support sets of the smallest size (or small sizes, if obtaining the smallest is difficult). However, as there may be many support sets of a given size, it becomes evident that we need more refined measures of separation between $T[S]$ and $F[S]$, which can be used as secondary criteria. Such measures will also be useful in the process of computing small support sets in some algorithms. For these purposes, we introduce a few definitions of separation on the basis of Hamming distances between pairs of vectors taken from T and F, and formulate optimization problems to find best support sets according to respective measures.

Although the problem of finding a support set that minimizes such a measure as described above is NP-hard, it turns out that the continuous relaxations of the minimization problem for some measures become convex programming problems, which can be solved efficiently. Based on these observations, we propose in Section 5 of this paper several heuristic algorithms to find support sets of small sizes. We also discuss how to deal with irrelevant and/or dependent variables in these algorithms. Experimental results show that these heuristic algorithms are quite effective in finding small sets of essential attributes that support given data sets, as well as in removing irrelevant and/or dependent attributes.

From the theoretical side, we present a probabilistic argument showing that a support set of large size will need a large number of data vectors (i.e., $|T|$ and $|F|$ should be large) in order to distinguish it from those randomly existing support sets. In other words, for a given data set (T, F), a small support set (if any) has a better chance of being a real support set, which is inherent in the represented phenomenon. This is another observation that supports our goal of finding small support sets. We also carry out theoretical analysis of the performance of the proposed heuristic algorithms, and show that several of them have an approximation ratio of $1 - \frac{1}{e}$.

2 Measures of separation

We consider the problem of finding a support set of minimum size. However, we note that such minimum support sets may not be unique, and many may exist. To compare the quality among support sets, we now introduce various measures of separation.

For two vectors $a, b \in \mathbb{B}^n$, let

$$d(a, b) = \sum_{j \in a\Delta b} 1 \tag{2}$$

denote the *Hamming distance* between a and b, where

$$a\Delta b = \{j | a_j \neq b_j\}.$$

For subsets $X, Y \subseteq \mathbb{B}^n$, let

$$\rho(X, Y) = \min_{a \in X, b \in Y} d(a, b),$$

and define

$$\rho(S) = \rho(T[S], F[S]) \tag{3}$$

for a given pdBf (T, F) and a set $S \subseteq V$, which is a first measure of separation between sets $T[S]$ and $F[S]$. A set S is a support set if and only if $\rho(T[S], F[S]) \geq 1$ holds.

Instead of the minimum Hamming distance, we may consider the average distance between sets X and Y:

$$\begin{aligned} \sigma(X, Y) &= \frac{1}{|X||Y|} \sum_{a \in X, b \in Y} d(a, b), \\ \sigma(S) &= \sigma(T[S], F[S]), \end{aligned} \tag{4}$$

or the number of separated pairs:

$$\begin{aligned} \theta(X, Y) &= \sum_{a \in X, b \in Y} \min\{d(a, b), 1\} \\ \theta(S) &= \theta(T[S], F[S]). \end{aligned} \tag{5}$$

Another class of measures can be constructed by considering the *Hamming distance vector* between X and Y:

$$h(X, Y) = (h_0, h_1, ..., h_n),$$

where

$$h_k = h_k(X, Y) = \sum_{a \in X,\ b \in Y\ :\ d(a,b)=k} 1$$

denotes the number of pairs $(a, b) \in X \times Y$ with Hamming distance k. As in the above cases, we define

$$h(S) = (h_0(S), h_1(S), ..., h_n(S)), \tag{6}$$

where $h_k(S) = h_k(T[S], F[S]), k \in V$.

To rank subsets S according to their power of separation, we may use the lexicographic order $\leq_L$ of their Hamming distance vectors $h(S)$. We then observe that $h = (h_0, h_1, ..., h_n) \leq_L h' = (h'_0, h'_1, ..., h'_n)$ holds if and only if

$$\sum_{k=0}^{n} h_k \alpha^k \leq \sum_{k=0}^{n} h'_k \alpha^k$$

holds for some sufficiently small $\alpha > 0$. Thus, using the above expression, we can associate a potential ϕ_α to any set S by

$$\phi_\alpha(S) = \sum_{k=0}^{n} h_k(S)(1 - \alpha^k) = \sum_{a \in T} \sum_{b \in F} (1 - \alpha^{d(a[S], b[S])}),$$

where $0 \leq \alpha < 1$ is a fixed constant. Here ϕ_α is defined as the weighted sum of $(1 - \alpha^k)$, but not α^k, since we are interested in the separation measure ϕ_α such that $\phi_\alpha(\emptyset) = 0$ holds and a larger value $\phi_\alpha(S)$ implies a better S. We also note that $\phi_\alpha(S)$ is a generalization of the lexicographic order since α may be set to any nonnegative number. In particular, if

$\alpha = 0$, then $\phi_0(S) = |T||F| - h_0(S)$ counts the number of pairs $(a, b) \in T[S] \times F[S]$ with Hamming distance at least 1, and hence $\phi_0(S) = \theta(S)$. The case of $\alpha = 1$ is excluded, because $\phi_1(S) = 0$ is constant for all S. In general case, it represents a weighted sum of the components of the Hamming distance vector. All these measures will be meaningful in the following discussion.

Let us introduce a general notation $\psi(S)$ to represent the measure of separation; it may stand for $\rho(S), \sigma(S), \theta(S)$ or $\phi_\alpha(S)$.

Let K^* denote the size of a minimum support set. The main problem considered in this paper is to find out a support set S that optimizes $\psi(S)$ among those satisfying $|S| = K^*$. To determine K^* itself is an NP-hard problem in general [13], because, as shown below, it contains the minimum set covering problem, which is known to be NP-hard (e.g., [17]).

Given an underlying set $M = \{1, 2, \ldots, m\}$ and a family of its subsets $C_j \subseteq M, j = 1, 2, \ldots, n$, the minimum set covering problem asks to find a subfamily $C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ such that $\cup_{l=1}^{k} C_{j_l} = M$ holds and k is minimum. For a given instance of the minimum set covering problem, we consider the pdBf (T, F) defined by $F = \{(00\cdots0)\}$ and $T = \{v^{(i)} \mid i = 1, 2, \ldots, m\}$, where $v_j^{(i)} = 1$ if $i \in C_j$, and 0 if $i \notin C_j$. Then $S = \{j_1, j_2, \ldots, j_k\}$ is a support set of (T, F) if and only if $\cup_{l=1}^{k} C_{j_l} = M$ holds; hence the minimum set covering problem can be solved by finding a minimum support set of the corresponding pdBf.

To avoid the difficulty of knowing K^*, we sometimes introduce a parameter K (positive integer), and try to find a support set S whose size is at most K and which minimizes the measure of separation $\psi(S)$. This problem is described as follows for a given pdBf (T, F).

$$\begin{array}{ll} \text{maximize} & \psi(S) \\ \text{subject to} & |S| \leq K \\ & S \subseteq V. \end{array} \tag{7}$$

If K is set to be too small, the optimal solution S of this problem may not be a support set. Thus, if small support sets are required, we search the minimum K for which the optimal solution of (7) provides a support set.

An alternative approach to guarantee a support set is to add the constraint,

$$\rho(S) \geq 1 \tag{8}$$

to (7). In this case, if K is too small, the problem becomes infeasible, and the smallest K, for which the problem is feasible, gives a smallest support set.

Based on these formulations, we shall consider heuristic algorithms of finding small support sets with good separation measures, later, after presenting a probabilistic argument in the next section to justify our goal of finding small support sets.

3 Probabilistic analysis

In this section, we evaluate how many support sets of size K exist on average if T and F are randomly generated. The results show that, given K, many support sets usually exist unless the numbers of data vectors $|T|$ and $|F|$ are larger than certain values (dependent on K). Interpreting this in the reverse direction, this says that, given a data set (T, F) with its $|T|$ and $|F|$, the support sets of size K found for (T, F) may be those just randomly existent, not those inherent in the phenomenon from which the data set is taken, unless K is smaller than a certain threshold. Therefore, if our efforts to find small support sets succeed and the size of the obtained support sets is smaller than the threshold, then, with high probability, they are concluded to be real support sets inherent in the phenomenon under consideration.

For the analysis, suppose that each vector in T and F is randomly chosen from $\mathbb{B}^n$ uniformly and independently. The number of trials for T and F are m_T and m_F, respectively. Note that $|T| < m_T$ or $|F| < m_F$ may occur, since we allow duplicate vectors. For a subset $S \subseteq V$, we use notations

$$m_T^S = |T[S]| \text{ and } m_F^S = |F[S]|.$$

Let X_S be the indicator random variable defined by

$$X_S = \begin{cases} 1, & T[S] \cap F[S] = \emptyset \text{ (i.e., } S \text{ is a support set)} \\ 0, & \text{otherwise,} \end{cases}$$

and let

$$X(K) = \sum_{S \subseteq V, |S|=K} X_S,$$

where K is a given constant. Then the expectation of $X(K)$, $\mathbf{E}(X(K))$, gives the expected number of support sets of size K. As $\mathbf{E}(X_S)$ is the same for all $S \subseteq V$ of size K,

$$\begin{aligned} \mathbf{E}(X(K)) &= \sum_{S \subseteq V, |S|=K} \mathbf{E}(X_S) \\ &= \binom{n}{K} \mathbf{E}(X_S) \end{aligned} \tag{9}$$

holds by the linearity of expectation. Given $m = m_T^S$ and an $S \subseteq V$ of size K, the probability that each random vector in $F[S]$ does not appear in $T[S]$ is $(2^K - m)/2^K$. Therefore, we have

$$\mathbf{Prob}(X_S = 1 \mid m_T^S = m) = \left(\frac{2^K - m}{2^K} \right)^{m_F}. \tag{10}$$

Based on these, we first consider a lower bound on $\mathbf{E}(X(K))$. Since

$$\begin{aligned} \mathbf{E}(X_S) &= \mathbf{Prob}(X_S = 1) \qquad (11) \\ &= \sum_{m=1}^{m_T} \mathbf{Prob}(m_T^S = m) \mathbf{Prob}(X_S = 1 \mid m_T^S = m), \end{aligned}$$

the following theorem is immediate.

Theorem 1 *For any $S \subseteq V$ of size K, $\mathbf{E}(X_S) \geq ((2^K - m_T)/2^K)^{m_F}$ holds. Hence we have*

$$\mathbf{E}(X(K)) \geq \binom{n}{K} \left(\frac{2^K - m_T}{2^K} \right)^{m_F}.$$

Next we derive an upper bound on the expectation. For this purpose, we first estimate the number of different vectors in $T[S]$ for a fixed $S \subseteq V$ of size K. Let $T = \{a^{(1)}, a^{(2)}, \ldots, a^{(m_T)}\}$ and let Y_i be the indicator random variable taking value 1 if $a^{(i)} \neq a^{(j)}$ holds for all $j < i$, and 0 otherwise. Then $m_T^S = \sum_{i=1}^{m_T} Y_i$ holds.

Now we present as a lemma the Chernoff bound [2], which holds for the sum of random variables, and then a key lemma used to derive our upper bound (Theorem 2). The proof of Lemma 2 will be given in Appendix.

Lemma 1 (Chernoff bound) *For any $p_1, p_2, \ldots, p_k \in [0,1]$, let $Z_1, Z_2, \ldots, Z_k$ be mutually independent random variables satisfying $\mathbf{Prob}(Z_i = 1 - p_i) = p_i$ and $\mathbf{Prob}(Z_i = -p_i) = 1 - p_i$, and let $Z = \sum_{i=1}^{k} Z_i$. Then, for any $\delta > 0$, $\mathbf{Prob}(Z > \delta) < e^{-2\delta^2/k}$ and $\mathbf{Prob}(Z < -\delta) < e^{-2\delta^2/k}$ hold.*

Lemma 2 *For any δ (> 0), we have*

$$\mathbf{Prob}[m_T^S < \mu - \delta] \leq e^{-2\delta^2/m_T},$$

where $\mu = m_T \left(1 - \frac{m_T - 1}{2^{K+1}}\right)$.

Theorem 2 *For any $S \subseteq V$ of size K,*

$$\mathbf{E}(X_S) \leq \min_{\delta > 0} \left(e^{-(m_T(1-(m_T-1)/2^{K+1})-\delta)m_F/2^K} + e^{-2\delta^2/m_T} \right) \tag{12}$$

holds. Hence we have

$$\mathbf{E}(X(K)) \leq \binom{n}{K} \min_{\delta > 0} \left(e^{-(m_T(1-(m_T-1)/2^{K+1})-\delta)m_F/2^K} + e^{-2\delta^2/m_T} \right).$$

Proof: By (10), we have

$$\begin{aligned}
\mathbf{Prob}(X_S = 1 \mid m_T^S = m) &= \left(\frac{2^K - m}{2^K} \right)^{m_F} \\
&= \left\{ \left(1 - \frac{m}{2^K} \right)^{\frac{2^K}{m}} \right\}^{\frac{m m_F}{2^K}} \\
&\leq e^{-\frac{m m_F}{2^K}}.
\end{aligned}$$

Therefore, by (11) and Lemma 2,

$$\begin{aligned}\mathbf{E}(X_S) &\leq \mathbf{Prob}(m_T^S \geq \mu - \delta) \cdot e^{-\frac{(\mu-\delta)m_F}{2^K}} + \mathbf{Prob}(m_T^S < \mu - \delta) \cdot 1 \\ &\leq e^{-\frac{(\mu-\delta)m_F}{2^K}} + e^{-2\delta^2/m_T}\end{aligned}$$

holds for any $\delta > 0$, where μ is given in Lemma 2. ■

To get a good upper bound, we must minimize (12) over δ. This is not difficult numerically, because (12) is a convex function of δ and its derivative is easily computed.

We show in Table 1 some numerical results to see the accuracy of the lower and upper bounds on $\mathbf{E}(X(K))$ of Theorems 1 and 2, respectively. We assume $m_T = m_F$ and the figures in the row LB (resp., UB) are the maximum (resp. minimum) m_T such that the lower (resp., upper) bound on $\mathbf{E}(X(K))$ of Theorem 1 (resp., 2) is at least (resp., at most) 1.0, where parameter δ in Theorem 2 is optimized. In other words, $\mathbf{E}(X(K)) \geq 1$ holds if $m_T = m_F \leq LB$, and $\mathbf{E}(X(K)) \leq 1$ holds if $m_T = m_F \geq UB$. In the table, the mark '∞' means that the upper bound on $\mathbf{E}(X(K))$ is always more than 1. The meaning of LB and UB are explained as follows. If $m_T = m_F \leq LB$, the number of vectors in T and F is not sufficient to avoid randomly existent support sets, but if $m_T = m_F \geq UB$, random support sets rarely appear. From the table, we can observe that the two bounds are close together especially for larger K.

Next we examine the asymptotic behavior of $\mathbf{E}(X_S)$ when K is sufficiently large. For this, we first derive the following two lemmas from Theorem 1 and Theorem 2, respectively. The proofs will be given in Appendix.

Lemma 3 *For any constant $c > 1$, suppose $m_T \leq 2^K/c$ and let $c' = (1 - 1/c)^c$. Then $\mathbf{E}(X_S) \geq (c')^{m_T m_F/2^K}$ holds for any $S \subseteq V$ of size K.*

Lemma 4 *For any S of size K, if $m_T = o(2^{3K/5})$ and $m_F = o(2^{3K/5})$, then $\mathbf{E}(X_S) \leq (2 + o(1))e^{-m_T m_F/2^K}$ holds, where $o(1) \to 0$ as $K \to \infty$.*

Note that $0 < c' < 1/e$ and $c' \to 1/e$ $(c \to \infty)$ hold, whose convergence speed is quite fast. Lemmas 3 and 4 together with (9) then give the following theorem.

Theorem 3 *Suppose $m_T = o(2^{3K/5})$ and $m_F = o(2^{3K/5})$. Then $\mathbf{E}(X(K)) = 1$ implies $m_T m_F = 2^K \ln \binom{n}{K}(1 + o(1))$, where $o(1) \to 0$ as $K \to \infty$.*

Proof: By Lemma 3 and (9), $\mathbf{E}(X(K)) > 1$ holds if $m_T m_F < 2^K \ln \binom{n}{K}/(-\ln c')$. By the assumption, we can choose $c = \Theta(2^{2K/5})$, which implies $-\ln c' = 1 + o(1)$. While by Lemma 4 and (9), $\mathbf{E}(X(K)) < 1$ holds if $m_T m_F > 2^K(\ln \binom{n}{K} + \ln(2 + o(1)))$. ■

Note that there exist m_T and m_F satisfying all conditions $m_T = o(2^{3K/5})$, $m_F = o(2^{3K/5})$ and $m_T m_F \geq \Omega(2^K \ln \binom{n}{K})$, if $n = o(e^{(2^{K/5})/K})$ holds. It might be interesting to note that

$$m_T = m_F = \sqrt{2^K \ln \binom{n}{K}} \tag{13}$$

lies between LB and UB of Table 1 for any K and n, since

$$\left(\frac{2^K - m_T}{2^K}\right)^{m_F} \leq e^{-m_T m_F/2^K} \leq e^{-(m_T(1-(m_T-1)/2^{K+1})-\delta)m_F/2^K} + e^{-2\delta^2/m_T}$$

holds for any $\delta > 0$. In this sense, (13) may be used as a good estimate of $m_T = m_F$ for which the condition $\mathbf{E}(X(K)) = 1$ holds.

4 Optimization problems and their relaxations

We first formulate our main problem (7) as a mathematical programming problem by introducing variables $y_j, j \in V$, which correspond to the attributes of the data set. This formulation enables us to consider its relaxation, which can be efficiently solved by mathematical programming algorithms. These will then be used in some of the heuristic algorithms to be discussed in Section 5.

Let $\mathbb{U} = [0, 1]$, the interval of real numbers between 0.0 and 1.0. We consider the above vector of variables y to satisfy $y \in \mathbb{U}^n$, and generalize the Hamming distance (2) as follows.

$$d_y(a, b) = \sum_{j \in a \Delta b} y_j. \tag{14}$$

The vector y is used to represent a subset $S \subseteq V$ in the subsequent discussion. If y is a 0-1 vector (i.e., $y \in \mathbb{B}^n$), then $y_j = 1$ (resp., 0) says that $j \in S$ (resp., $j \notin S$) holds. In general, $y_j \in \mathbb{U}$ describes how much each attribute j belongs to S. As a result of this generalization, the measures of separation defined in Section 2 for a pdBf (T, F) are also extended as follows.

$$\begin{aligned}
\rho(y) &= \min_{a \in T, b \in F} d_y(a, b) \\
\sigma(y) &= \sum_{a \in T, b \in F} d_y(a, b) \\
\theta(y) &= \sum_{a \in T, b \in F} \min\{d_y(a, b), 1\} \\
\phi_\alpha(y) &= \sum_{a \in T, b \in F} (1 - \alpha^{d_y(a,b)}).
\end{aligned}$$

The general measure to be maximized is also written as $\psi(y)$, which represents $\rho(y), \sigma(y), \theta(y)$ or $\phi_\alpha(y)$.

Using these notations, the main problem (7) can be written as the following integer program.

$$\begin{aligned}
\text{maximize} \quad & \psi(y) \\
\text{subject to} \quad & \sum_{j \in V} y_j \leq K \\
& y \in \mathbb{B}^n.
\end{aligned} \tag{15}$$

The continuous relaxation of this problem is then defined by

$$\begin{array}{ll} \text{maximize} & \psi(y) \\ \text{subject to} & \sum_{j\in V} y_j \le K \\ & y \in \mathbb{U}^n. \end{array} \tag{16}$$

If the condition (8) to guarantee a support set is required, we add the following constraints to the above formulations.

$$\sum_{j\in a\Delta b} y_j \ge 1, \quad a \in T, b \in F \tag{17}$$

This ensures that the solution y separates T and F in the sense of attaining $d_y(a,b) \ge 1$ for all pairs of $a \in T$ and $b \in F$.

Note that problem (16) becomes a linear programming problem for the objective function $\psi(y) = \rho(y), \sigma(y)$ or $\theta(y)$. In the first case, the objective function can be written as

$$\text{maximize } z$$

after adding the following constraints to (16).

$$z \le \sum_{j\in a\Delta b} y_j, \quad a \in T, b \in F.$$

In the second case, $\sigma(y)$ is already a linear function of y. In the third case, we introduce z_{ab} for each pair $a \in T$ and $b \in F$, and solve

$$\text{maximize} \sum_{a\in T, b\in F} z_{ab}$$

after adding the following constraints to (16).

$$z_{ab} \le d_y(a,b), \quad z_{ab} \le 1, \quad a \in T, b \in F \tag{18}$$

If $\psi(y) = \phi_\alpha(y)$, problem (16) is no longer a linear programming problem, but is a nonlinear programming problem. However, it is not difficult to see that, for $\alpha > 0$, $\phi_\alpha(y)$ is a concave function of y, and various efficient algorithms are available to maximize a concave function over a convex polyhedral region, defined by a set of linear inequalities (e.g., [29]). Although such algorithms do not have finite convergence, except for some special cases, some algorithms run in polynomial time in the input size of the problem and $1/\varepsilon$, where $\varepsilon > 0$ is the error bounding the difference between the attained objective value and the exact optimum.

Before concluding this section, let us remark that the constraints (17) need not be explicitly added to (16), whenever $\psi = \rho$, $\psi = \theta$ or $\psi = \phi_\alpha$ with a sufficiently small $\alpha \ge 0$ (i.e., lexicographic minimization of $h(S)$) is used, because an optimal solution y^* of (16) without (17) is also an optimal solution of (16)-(17) if and only if y^* satisfies (17). In the remaining case of $\psi = \sigma$, this property may not hold true. In our computational experiments reported later, we always use the formulation without constraint (17), since it reduces the size of problem significantly. The obtained solutions tend to satisfy (17), however, even in the case of $\psi = \sigma$.

5 Heuristic algorithms

We present here three types of heuristic algorithms to find small support sets S for a given pdBf (T, F), which also attain a good separation measure $\psi(S)$. Some of these algorithms are based on problem (7) (i.e., (15)) and its continuous relaxation (16). We shall report some computational results in Section 6.

5.1 Rounding algorithms

This type of algorithms first solve the continuous relaxation (16), and then round the obtained continuous solution to a 0-1 solution. The parameter K is first set to 1, and then increased one by one, until a feasible solution (i.e., a support set) is found.

ψ-ROUNDING

Input: A pdBf (T, F).

Step 0: Set $K := 1$.

Step 1: Solve problem (16), and let $y^* \in \mathbb{U}^n$ denote its optimal solution.

Step 2: Round the continuous solution y^* to a 0-1 solution y'. If y' is feasible (i.e., gives a support set of size K), then output y' and halt. Otherwise, let $K := K + 1$ and return to Step 1.

If ψ is specified to ρ, etc., for example, we call it ρ-ROUNDING, etc. The search for the minimum K with a feasible solution may be done by using binary search, rather than increasing K one by one; such modification will be effective if the final K is relatively large. There are various ideas how to conduct the rounding operation in Step 2. A simple one is based on the greedy idea; i.e., round up the first K largest variables y_j^* to one, and set the rest to zero. Some probabilistic ideas may also be employed; e.g., round up each y_j^* to one with probability y_j^*. Even the iteration of incrementing K may be simplified into one iteration. That is, we solve problem (16) only once with $K = 1$, and round up y_j^* to one in the nonincreasing order of y_j^* until a support set is obtained.

We do not go into details, however, since other approaches to be described in Subsections 5.2 and 5.3 appear to be more effective.

5.2 Greedy algorithms

Greedy algorithms are perhaps most commonly found in heuristic approaches to solve various combinatorial optimization problems. Starting from the empty solution, a greedy algorithm constructs a feasible solution step by step, each time selecting a change which is considered to be the best according to the current local information. Applied to our problem, a standard greedy algorithm may be described as follows.

> ψ-GREEDY
>
> **Input:** A pdBf (T, F).
>
> **Step 0:** Set $S := \emptyset$.
>
> **Step 1:** Choose a j for which $\psi(S \cup \{j\})$ is maximum among all $j \in V \setminus S$.
>
> **Step 2:** Set $S := S \cup \{j\}$. If S is a support set, then output S and halt. Otherwise, return to Step 1.

Analogous to the previous algorithm, the ψ in the name stands for ρ, ϕ_α and so forth. A most popular criterion may be $\psi = \phi_0 = \theta = |T||F| - h_0$, as it is easy to compute and is directly related to the definition of support sets (i.e., S is a support set if and only if $\phi_0(S) = |T||F|$). Of course, general ϕ_α with a positive α is also a good candidate. This algorithm always halts after at most n iterations, provided that the input pdBf (T, F) satisfies $T \cap F = \emptyset$.

It should be emphasized that the optimization problem (16) need not be solved in ψ-GREEDY. Therefore each iteration of Steps 1 and 2 can be easily implemented and efficiently executed. This simplicity is a great advantage of this type of algorithms.

5.3 Stingy algorithms

A stingy algorithm may be stated as a reverse greedy algorithm, as it starts from $S = V$ and then removes elements from S successively until a minimal support set is obtained. It may be simply implemented by executing ψ-GREEDY in the reverse manner. However we propose here a more sophisticated stingy algorithm by employing the optimization problem (16) in each iteration. In this algorithm, more than one variable y_j may be set to 0 (i.e., the corresponding attributes are removed from S) in one iteration, and the size K of the attempted support set is increased from 1 successively. Upon termination, the resulting (16) has an optimal solution y^* that is binary, and it gives a minimal support set of the given pdBf (T, F).

In the algorithm, we maintain a set $Z \subseteq V$ to keep the attributes removed from S; i.e., the constraint

$$y_j = 0, \quad j \in Z \tag{19}$$

is added to the optimization problem (16). The set Z is successively enlarged during execution of the algorithm.

ψ-STINGY

Input: A pdBf (T, F).

Step 0: Set $K := 1$ and $Z := \emptyset$.

Step 1: Solve the optimization problem (16) extended with the constraints (17) and (19), and let $y^* \in \mathbb{U}^n$ denote the obtained optimal solution. If this problem has no feasible solution, then set $K := K + 1$, and return to Step 1. Otherwise, let

$$z^* = \rho(y^*) \; (= \min_{a \in T,\, b \in F} d_{y^*}(a, b)).$$

Step 2: If $K + |Z| = n$, then output y^* and STOP. Otherwise, sort the components of y^*, such that $y^*_{i_1} \leq y^*_{i_2} \leq \cdots \leq y^*_{i_n}$, and set k to be the largest integer satisfying

$$\sum_{j=1}^{k} y^*_{i_j} < z^*.$$

Step 3: If $k \leq |Z|$, then set $K := K + 1$; otherwise set $Z := \{i_1, i_2, ..., i_k\}$. Return to Step 1.

Assume that the optimization problem (16) can be solved in polynomial time. Then this algorithm has the following nice feature.

Theorem 4 *Algorithm* ψ-STINGY *halts in polynomial time, and returns a binary vector* y^* *such that the set* $S^* = \{j \mid y^*_j = 1\}$ *is a minimal support set of pdBf* (T, F).

Proof: Let us note first that in every iteration either the size of Z increases, or the value of K is incremented by one; i.e. $K + |Z|$ increases at least by one. Hence, the algorithm terminates in at most n iterations.

Let us note next that the set $V \setminus Z$ remains a support set after every iterations, because the selection rule in Step 2 guarantees that

$$\sum_{i \in Z} y^*_i < z^* = \min_{a \in T, b \in F} d_{y^*}(a, b) = \min_{a \in T, b \in F} \sum_{i \in a \Delta b} y^*_i$$

and hence $\rho(V \setminus Z) \geq 1$ holds. Upon termination, we must have $K = |V \setminus Z|$, due to the stopping criterion in Step 2. We must also have $\sum_{i \in V \setminus Z} y^*_i = K$, which is implied by the monotonicity of ψ. Hence, $y_i^* = 1$ holds for all $i \in V \setminus Z$, and we have S^* $(= \{j \mid y_j^* = 1\}) = V \setminus Z$.

To verify that S^* is a minimal support set at termination, we assume otherwise, and let $S^* \setminus \{i^*\}$ be a support set. Consider the last iteration, in which the value of K was increased. In that iteration we have, on one hand, $V \setminus Z \supseteq S^* \supset S^* \setminus \{i^*\}$, and $K = |S^* \setminus \{i^*\}|$; thus problem (16) – (19) was feasible, and hence $z^* \geq 1$. On the other hand, $|V \setminus Z| > K$, and thus $\min_{i \in V \setminus Z} y_i^* < 1$. Hence the sum of the smallest y_i^* in $V \setminus Z$ together with $y_i^* = 0$ for all $i \in Z$ is still smaller than z^*, implying $k > |Z|$ in this iteration. Therefore, the algorithm must have increased the set Z, instead of incrementing the value K; a contradiction, proving our claim. Thus S^* is a minimal support set. ■

Let us remark that we needed in our proof the constrains (17) only for proving the minimality of S^* Also, as remarked at the end of Section 4, constraints (17) are not explicitly needed if the objective function is $\psi = \rho, \theta$, or ϕ_α with a sufficiently small $\alpha \geq 0$.

6 Computational results

6.1 Generation of problem instances

In our first experiment, we generate pdBfs (T, F) with known support sets, and compare the performance of heuristic algorithms of Section 5 on the basis of how much they can recover the original support sets.

In the generation, we assume that T and F are the results of random samplings from some real phenomenon that depends on an (unknown) subset of the observed attributes. Thus, in particular, we assume in our experiment the existence of a Boolean function $f : \mathbb{B}^H \mapsto \mathbb{B}$ for some subset $H \subseteq V$, such that $f(a[H]) = 1$ for $a \in T$ and $f(b[H]) = 0$ for $b \in F$. We shall say that f is the *hidden logic* behind the pdBf (T, F), and H is the set of *hidden attributes*. We shall also assume that the rest of the variables is partitioned into *dependent* attributes $D \subseteq V$ and *random* (or *irrelevant*) attributes $R \subseteq V$, i.e.,

$$V = H \cup D \cup R$$

is a partition, where for every $k \in D$ there exists a Boolean function $g_k : \mathbb{B}^H \mapsto \mathbb{B}$ such that $x_k = g_k(x[H])$ holds for all $x \in T \cup F$, while for indices $k \in R$ such a mapping does not exist (assuming that $T \cup F$ is large enough).

Clearly, H is a support set of any pdBf (T, F) generated by the above model. In the following, we compare heuristic algorithms according as how close are the obtained support sets to the original set H. We shall study the effect of different measures of separation, and the problem of how large the data set $|T \cup F|$ should be in order to have H as one of the "best" support sets.

The construction of hidden logic f and dependency functions g_k is done by the following two methods, where all random drawings are from uniform distribution.

1. Random functions: Given H, the truth table of f is randomly constructed by assigning the output 0 or 1 to each input vector with equal probabilities. Each g_k is dependent on a variable set $H' \subset H$ of size $\lceil |H|/2 \rceil$, randomly chosen, and its output values are determined in a manner similar to f.

2. Functions represented by a DNF: Terms are randomly generated from the variable set H, by choosing a positive literal with probability 1/4, a negative literal with probability 1/4 and no literal with probability 1/2, for each variable in H. The disjunction of an appropriate number of such terms is then used to define the hidden logic f (the generation of terms is terminated when their disjunction covers at least half of the whole space $\mathbb{B}^H$). Dependence functions g_k are also generated analogously after choosing H' in the same way as in 1.

Based on these two models, each instance (T, F) is constructed by repeating $|T| + |F|$ times the following: First generate a random binary vector of dimension $|H|$, determine whether it belongs to T or F by applying f, append $|D|$ more components by using the dependence functions g_k, $k \in D$, and finally append the last $|R|$ components generated randomly.

6.2 Comparisons of heuristic algorithms

Our experimental results for the instances of Section 6.1 are now summarized. The sizes $|H|, |D|$ and $|R|$ of six types of generated instances are listed in Table 2.

(i) We first present overall performance of nine representative heuristic algorithms in Section 5: ρ-ROUNDING, σ-GREEDY, ρ-STINGY, θ-ROUNDING, θ-GREEDY, θ-STINGY, ϕ_α-ROUNDING, ϕ_α-GREEDY and ϕ_α-STINGY, where $\alpha = 0.001$ is used for the last three. Recall that θ is equal to ϕ_α with $\alpha = 0$. Table 3 is for random functions and Table 4 is for DNF functions. They show the average behavior of $|S|$ and $r_H = |S \cap H|/|S|, r_D = |S \cap D|/|S|, r_R = |S \cap R|/|S|$, for 40 instances each, where S are the computed support sets. It is desired to have $|S|$ as small as $|H|$, and to have large r_H and small r_D, r_R.

From the tables we see that θ-GREEDY, ϕ_α-GREEDY and ψ-STINGY perform reasonably well, where ψ stands for all of the ρ, θ and ϕ_α, while ψ-ROUNDING and σ-GREEDY are no good. There are not big differences among the first five, but they may be ranked in the order of ϕ_α-STINGY, θ-STINGY, ρ-STINGY, ϕ_α-GREEDY and θ-GREEDY from the best.

In most results of Tables 3 and 4, r_H is much larger than r_D and r_R. This says that these heuristic algorithms are able to distinguish essential attributes from others, though their power varies according to the algorithms. There are tendencies for types A, B and C, as to which of dependent variables D and random variables R are more confused with essential variables H. However, there is no apparent rule to tell which of D and R is more confusable.

(ii) It may be interesting to apply the probabilistic argument of Section 3 to these results, even though these instances do not satisfy the assumptions of Section 3. Note that $|T| \approx |F|$ holds in our problem instances. For type A instances with $|H| = 10$ and $n = 20$, Table 1 says that $|T| + |F| = 200$ is below the threshold to attain $\mathbf{E}(X(K)) \leq 1$ for $K = 10$ but $|T| + |F| = 400$ is above the threshold. This may be observed from the fact that, when support sets S with $|S| \approx 10$ are obtained (which are attained by some methods), r_H is close to 1.0 for type A2 but is smaller for type A1. For type B and C instances with $|H| = 5$, Table 1 does not tell much, but indicates that both $|T| + |F| = 200, 400$ are sufficient to guarantee $\mathbf{E}(X(K)) \leq 1$ for $K = 5$. In fact, if $|S| \approx 5$ is realized, the corresponding r_H in the tables are always close to 1.0. However our heuristic algorithms may not be powerful enough to obtain such small support sets S for type B instances.

(iii) The effect of parameter α can be seen in Table 5, in which average $|S|$ are reported with various values of α, for ϕ_α-GREEDY and ϕ_α-STINGY applied to problem instances of random functions. Recall that ϕ_α-GREEDY (resp., ϕ_α-STINGY) with $\alpha = 0.0$ is equivalent to θ-GREEDY (resp., θ-STINGY). The table tells that parameter α has little influence on the performance; but a small positive value like $\alpha = 0.001$ may be recommended if we have to choose one. The influence of α becomes even smaller for the instances of DNF functions, though we omit the details.

(iv) The computation time to execute the above heuristic algorithms varies greatly depending on whether the linear or convex programs (16) need be solved or not. As discussed before, ψ-ROUNDING and ψ-STINGY require to solve linear programs or convex programs repeatedly. We used in our experiment the CPLEX code for linear programming and the SQP (successive quadratic programming) code developed in [20] for convex programming; see, e.g., [4] for the method of SQP. For example, using a machine of Sun Ultra 60 (360MHz, 2GB memory), ϕ_α-STINGY with $\alpha = 0.001$ (using convex programming) took more than 1000 seconds on the average to solve one A2 instance of random functions, and ρ-STINGY (using linear programming) took about 60 seconds for the same case. The algorithm θ-STINGY is computationally much heavier, even if it is based on linear programming, because the numbers of variables and constraints in (18) are very large. Therefore, we did not complete the experiment of θ-STINGY for the instances of A2, B2 and C2. On the other hand, algorithms ψ-GREEDY are very fast. In the above case of type A2, for example, σ-GREEDY, θ-GREEDY and ϕ_α-GREEDY took only 0.99, 1.4 and 6.1 seconds, respectively. In view of these, our conclusion is to recommend θ-GREEDY or ϕ_α-GREEDY as a practical tool to find reasonably small support sets.

6.3 Computational results for real world data

To see how our algorithms perform on real world data, we took four data sets from Irvine repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). As some attributes of these data sets were numerically or nominally valued, we transformed them into binary attributes, as described respectively, before computing support sets. Also all data vectors containing missing bits were omitted from consideration in our experiments.

(1) Breast Cancer: There are 9 original attributes, each taking an integer value from $\{1, 2, \ldots, 10\}$. To transform this into binary data set, a cut-point is placed between every

two consecutive integers of each attribute; e.g., binary attribute 2/2.5 denotes that it takes value 1 if the value of original attribute 2 is more than 2.5 (i.e., 3 or larger), and 0 if the value of original attribute 2 is less than 2.5. The resulting binary data set has $9 \times 9 = 81$ binary attributes. After removing data vectors containing missing bits, this data set has $|T| = 239$ and $|F| = 444$ data vectors.

Some runs of θ-GREEDY (as it breaks ties randomly, each run may produce a different solution), and all runs of ϕ_α-GREEDY and ϕ_α-STINGY (where $\alpha = 0.001$ was used) obtained support sets of 12 binary attributes. (As ϕ_α-STINGY takes too much computation time on the original binary data set, the data set was reduced by choosing 27 promising binary attributes before applying ϕ_α-STINGY.) We list two typical support sets for the purpose of future comparisons.

(a) 1/4.5, 1/6.5, 2/2.5, 3/3.5, 3/4.5, 4/2.5, 5/4.5, 6/3.5, 6/8.5, 7/4.5, 8/2.5, 8/4.5.
(b) 1/4.5, 1/6.5, 2/2.5, 3/3.5, 3/4.5, 4/1.5, 4/2.5, 5/3.5, 6/3.5, 6/7.5, 7/4.5, 8/2.5.

(2) Mushrooms: Original data set contains 22 nominally valued attributes. Each nominal value of one attribute was transformed into one binary attribute that takes value 1 if and only if the original attribute takes the chosen nominal value. In this way, 126 binary attributes were generated. The data set has $|T| = 2156$ and $|F| = 3488$ data vectors.

Applications of θ-GREEDY and ϕ_α-GREEDY could find the following support set of 6 binary attributes (ϕ_α-STINGY was not tested as the data set was too large):

bruises=bruises (or its negation), odor=none, gill-size=broad,
stalk-root=bulbous, spore-print-color=white, habitat=woods.

This may be an interesting result, because the rules cited in the Irvine repository contains 8 binary attributes. It should be further checked, however, whether this smaller support set can produce better rules for Mushrooms.

(3) Hepatitis: The original data set has 13 binary attributes and 6 numerical attributes. We introduced 4 to 6 cut points of appropriate locations for each numerical attribute. The resulting binary data set has 47 binary attributes, and $|T| = 67$, $|F| = 13$.

All three methods were applied to the resulting data set; θ-GREEDY and ϕ_α-GREEDY obtained the following support set of 7 attributes:

MALAISE=(yes/no), SPIDERS=(yes/no) (or AGE>50 or SPLEEN PALPABLE=(yes/no) or ASCITES=(yes/no)), VARICES=(yes/no), ALK PHOSPHATE>120, ALK PHOSPHATE>200, PROTIME>50, HISTOLOGY=(yes/no).

On the other hand, ϕ_α-STINGY obtained a slightly different support set, though the number of attributes is the same.

AGE>40, SPIDERS=(yes/no), BILIRUBIN>1.20, ALK PHOSPHATE>120
PROTIME>40, PROTIME>50, HISTOLOGY=(yes/no).

(4) Voting Records: The original data set is already binary. It has 16 attributes and $|T| = 124$, $|F| = 108$. All three methods obtained the following support set of 8 attributes.

Attributes 1, 2, 3, 4, 11, 13, 15, 16.

The computation time required for these experiments varies according to the sizes of data sets. θ-GREEDY and ϕ_α-GREEDY took about 30 seconds for Breast Cancer, more than 1000 seconds for Mushrooms, and less than 1 second for Hepatitis and Voting Records. ϕ_α-STINGY is much slower, and took 11 seconds for Hepatitis, and 360 seconds for Voting Records.

7 Approximation ratios of heuristic algorithms

In this section we show that, based on the results of submodular maximization, several of the proposed greedy algorithms perform with a constant approximation ratio.

7.1 Formulation as a set cover

Let us recall first (see e.g. [13]) that finding a smallest support set can be formulated as a set covering problem.

For a given pdBf (T, F) with $T, F \subseteq \mathbb{B}^n$, let $m = |T||F|$ and define an $m \times n$ matrix $A_{(T,F)} = \{a_{rj}\}$ by

$$a_{rj} = \begin{cases} 1 & \text{if } j \in a \,\Delta\, b \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

for all $r = (a, b)$, $a \in T$ and $b \in F$, and $j \in V$. From the arguments given earlier, it is clear that the problem of finding the smallest support set can be formulated as the following special form of integer program, which is known as the *minimum set covering problem*,

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{n} y_j \\ \text{subject to} & A_{(T,F)} y \geq \mathbf{1} \\ & y \in \mathbb{B}^n. \end{array} \tag{21}$$

Here the set $S^* = \{j \mid y_j^* = 1\}$ corresponding to an optimal solution y^* is a minimum size support set.

A greedy heuristic algorithm for the set covering problem is known to have the worst-case approximation ratio

$$H(m) = 1 + 1/2 + \cdots + 1/m \tag{22}$$

(i.e., the m-th harmonic number), which lies between $\ln m$ and $1+\ln m$ (see e.g. [11, 21, 24]). Furthermore, it is also known that no approximation algorithm can have a ratio $(1-\epsilon)\ln m$, for any $\epsilon > 0$, under some mild assumption of complexity theory (see [16]). It can easily be seen that this standard greedy heuristic is essentially the same as θ-GREEDY in Section 5.2, when it is applied to the above formulation (21).

7.2 Formulation as a submodular maximization

Our starting point in Section 2 was not only finding a support set S but also optimizing a separation measure $\psi(S)$, as formulated in (15). We derive in this section an approximation ratio of the heuristic algorithm(s) ψ-GREEDY.

Let us consider the case when the objective function ψ is a separable function, i.e.

$$\begin{array}{lll} \text{maximize} & \psi = \displaystyle\sum_{a\in T, b\in F} \psi_{a,b}(w_{a,b}) & \\ \text{subject to} & w = A_{(T,F)}y & (23) \\ & \displaystyle\sum_{j=1}^{n} y_j \le K & \\ & y_j \in \{0,1\},\ j = 1,2,\ldots,n, & \end{array}$$

where the matrix $A_{(T,F)}$ is defined by (20), w is an m-dimensional vector of the components indexed by pairs of $a \in T$ and $b \in F$, and $\psi_{a,b} : \mathbb{R} \mapsto \mathbb{R}$ are defined for $a \in T$ and $b \in F$.

Note that $w_{a,b} = d_y(a,b)$ holds in (23) for all $a \in T$ and $b \in F$, and except for ρ, all the separation measures $\psi = \sigma, \theta, \phi_\alpha$ introduced in Section 2 can be represented as separable functions $\psi = \sum_{a\in T, b\in F} \psi_{a,b}(w_{a,b})$ by using the following:

$$\begin{array}{rll} \sigma: & \psi_{a,b}(w_{a,b}) = w_{a,b} & \\ \theta: & \psi_{a,b}(w_{a,b}) = \min\{w_{a,b}, 1\} & (24) \\ \phi_\alpha: & \psi_{a,b}(w_{a,b}) = 1 - \alpha^{w_{a,b}} & \end{array}$$

for $a \in T$ and $b \in F$.

In formulation (23), variables $w_{a,b}$ are implicitly used only to represent the objective function ψ. We may regard ψ as a function of y, after substituting $w_{a,b} = d_y(a,b)$. Then, it is easy to see that problem (23) is equivalent to problem (15). Therefore, the heuristic algorithm ψ-GREEDY developed for (23) can be interpreted as an algorithm to solve problem (15) with a fixed K (where the stopping condition in Step 2 is replaced by $|S| = K$, since we are asked to obtain a set $S = \{j \mid y_j = 1\}$ that satisfies $|S| \le K$ and maximizes a nondecreasing function ψ).

Let us note that all functions $\psi_{a,b} : \mathbb{R} \mapsto \mathbb{R}$ in (24) have the following properties:

(i) $\psi_{a,b}(0) = 0$,

(ii) $\psi_{a,b}$ is nondecreasing,

(iii) $\psi_{a,b}$ is concave.

Recall that a function ψ is *concave* if $\psi(\alpha x + (1-\alpha)x') \geq \alpha\psi(x) + (1-\alpha)\psi(x')$ holds for all $x, x' \in \mathbb{R}$ and $0 \leq \alpha \leq 1$. It is also well-known that a function ψ is concave if and only if the inequality

$$\psi(x + \Delta) - \psi(x) \geq \psi(x' + \Delta) - \psi(x') \tag{25}$$

holds for all $x \leq x'$ and $\Delta \geq 0$.

Let us next recall (see e.g. [28]) that a function $\psi : \mathbb{B}^n \mapsto \mathbb{R}$ is called *submodular* if

$$\psi(y) + \psi(y') \geq \psi(y \vee y') + \psi(y \wedge y')$$

holds for all pairs of $y, y' \in \mathbb{B}^n$, where $(y \vee y')_j = y_j \vee y'_j$ and $(y \wedge y')_j = y_j \wedge y'_j$ for all $j = 1, 2, \ldots, n$. Furthermore, it is called *nondecreasing* if $\psi(y) \leq \psi(y')$ holds for all pairs of y and y' with $y \leq y'$ (i.e., $y_j \leq y'_j$ for all $j = 1, 2, \ldots, n$).

The approximation ratio we shall establish for the (modified) ψ-GREEDY algorithm is based on the results of [12, 28], in which problem (15) is considered with a nondecreasing submodular function ψ such that $\psi(0, 0, \ldots, 0) = 0$ (as a generalization of the maximum coverage problem discussed in more details in [19] for example), and it is shown that the greedy algorithm attains an approximation ratio of $(1 - \frac{1}{e})$, where e is the base of natural logarithm.

For our goal, therefore, it is enough to show that all functions $\psi : \mathbb{B}^n \mapsto \mathbb{R}$ considered above are submodular and nondecreasing, and satisfy $\psi(0, 0, \ldots, 0) = 0$.

Lemma 5 *The separable function $\psi(w) = \sum_{a \in T, b \in F} \psi_{a,b}(w_{a,b})$ of (23) is submodular and nondecreasing in the variables $y \in \mathbb{B}^n$, if all functions $\psi_{a,b}$ are nondecreasing and concave over $\mathbb{R}$.*

Proof: For vectors $y, y' \in \mathbb{B}^n$, let $y'' = y \vee y'$ and $y''' = y \wedge y'$, and let $w = A_{(T,F)}y$, $w' = A_{(T,F)}y'$, $w'' = A_{(T,F)}y''$ and $w''' = A_{(T,F)}y'''$. Then we can easily see that $w_{a,b} + w'_{a,b} = w''_{a,b} + w'''_{a,b}$, and hence

$$w_{a,b} - w'''_{a,b} = w''_{a,b} - w'_{a,b} \quad (\geq 0) \tag{26}$$

holds for all $a \in T$, $b \in F$. Since $\psi_{a,b}$ is assumed to be concave and $w'''_{a,b} \leq w'_{a,b}$, (26) and (25) imply

$$\psi_{a,b}(w_{a,b}) - \psi_{a,b}(w'''_{a,b}) \geq \psi_{a,b}(w''_{a,b}) - \psi_{a,b}(w'_{a,b})$$

and thus

$$\psi_{a,b}(w_{a,b}) + \psi_{a,b}(w'_{a,b}) \geq \psi_{a,b}(w''_{a,b}) + \psi_{a,b}(w'''_{a,b})$$

for all $a \in T$ and $b \in F$. Summing up these inequalities, we obtain $\psi(w) + \psi(w') \geq \psi(w'') + \psi(w''')$, showing that ψ is indeed submodular over $\mathbb{B}^n$.

Next, to see that ψ is nondecreasing in y, we only observe that $w_{a,b} = d_y(a, b)$ is nondecreasing in y and each $\psi_{a,b}$ is assumed to be nondecreasing over $\mathbb{R}$. ∎

Since $\psi(0, 0, \ldots, 0) = 0$ is clear from that all functions $\psi_{a,b}$ satisfy $\psi_{a,b}(0) = 0$, we have the following result.

Theorem 5 *Problem* (15) *is approximable within a factor of* $(1 - \frac{1}{e})$ *by the modified* ψ*-GREEDY, if all functions* $\psi_{a,b}$ *in problem* (23) *are nondecreasing and concave over* $\mathbb{R}$*, and satisfy* $\psi_{a,b}(0) = 0$.

Note that this approximation ratio applies readily to $\psi = \sigma, \theta$ and ϕ_α. However it cannot apply to ρ. The next section shows a hardness result for ρ.

7.3 Non-approximability result for ρ

This section gives a hardness result for problem (15) in case of $\psi = \rho$.

Theorem 6 *Unless P=NP, there exists no polynomial-time* $f(N)$*-approximation algorithm for problem* (15) *with* $\psi = \rho$*, where* N *denotes the problem size and* $0 < f(N) \leq 1$ *for all* N.

Proof: Assuming that there exists a polynomial-time $f(N)$-approximation algorithm A, we construct a polynomial algorithm B for the set covering problem, which is known to be NP-hard [17]. This will prove the theorem statement. Now note that the set covering problem is the same as finding the smallest K for which the objective function $\psi = \rho$ in (15) is at least 1. Therefore, to solve the set covering problem, we apply A to problem (15) with $\psi = \rho$ for $K = 1, 2, \ldots, n$. Let $y^{(K)} \in \mathbb{B}^n$ denote the corresponding solutions. For each K, algorithm A outputs a solution $y^{(K)}$ with $\rho(y^{(K)}) \geq f > 0$ (i.e., $\rho(y^{(K)}) = 1$ since ρ is integer valued), whenever the optimum value of (15) is 1; otherwise (i.e., if the optimum value of is 0), it outputs a solution $y^{(K)}$ with $\rho(y^{(K)}) = 0$. Letting K^* be the smallest K such that $\rho(y^{(K)}) = 1$, this implies that $y^{(K^*)}$ is the optimum solution of the set covering problem. ■

The above theorem shows that no constant-factor approximation algorithm exists for $\psi = \rho$. This contrasts with the approximability results for $\psi = \sigma, \theta$ and ϕ_α.

8 Conclusion and discussion

In this paper, we studied the problem of finding small support sets S for a given data set (T, F), capturing the essential attributes of the phenomenon under consideration. Although such data set can have many "random" support sets, our probabilistic argument shows that essential ones can be distinguished from random ones if their sizes are smaller than a certain threshold (determined by the sizes $|T|$ and $|F|$). After introducing various measures of separation between T and F, we showed that the problem of finding a best support set can be formulated as an integer programming problem, which can be relaxed to a certain linear or nonlinear programming problem. Based on these, we proposed several heuristic algorithms, and compared their performance by computational experiments on artificial and real-world data sets. Some of the heuristic algorithms turned out to be quite effective to find small support sets consisting of mostly essential attributes.

This direction of research is very important, since finding essential attributes in large data sets is a core issue in such areas as data mining, knowledge discovery and logical analysis of data. There remain however, many possible improvements for future research:

1. To distinguish dependent variables from essential variables, a special routine may be included in the proposed heuristic algorithms. Suppose that a set of attributes S' has been selected so far, which will then be enlarged to a support set S. To check whether an attribute $j \in V \setminus S'$ is dependent on S', we partition set $T[S'] \cup F[S']$ into $T_j[S']$ and $F_j[S']$ such that $x[S'] \in T[S'] \cup F[S']$ belongs to $T_j[S']$ (resp., $F_j[S']$) if $x_j = 1$ (resp., $x_j = 0$) holds. By applying separation measures to pdBf $(T_j[S'], F_j[S'])$, we can evaluate how much x_j depends on S'. In other words, if $T_j[S']$ and $F_j[S']$ are completely separated, then there is a function that determines the value of x_j from $x[S'] \in T[S'] \cup F[S']$; i.e., x_j is dependent on S'. This information may be utilized in a special routine so that highly dependent attributes are excluded from the candidate list for the next selection step.

2. The input data set (T, F) may contain errors, in the sense that some elements of vectors in $T \cup F$ are different from the true values, and that classification into T and F may also be erroneous. As a result of errors, even the property $T \cap F = \emptyset$ may not hold true. In this situation, measures such as σ, θ and in particular ϕ_α still convey useful information of separation, and can be used in the algorithms for solving (15). Note that condition (8) is not meaningful in this case.

3. Data vectors in real applications may contain missing bits $*$, implying that the value is not known for some reason (c.f. [9].) To deal with such cases, the definitions of separation measures have to be extended. Some attempts and experimental studies are under way in our group.

4. Many of the real-world data sets contain numerical vectors, rather than binary ones. To deal with numerical data, further study is necessary in order to generalize meaningfully the separation measures and the algorithms for computing support sets. We are directing some of our efforts also in this direction.

5. The sizes of pdBfs (T, F) encountered in real life applications can be very large. Although the experiments in Section 6 were done for rather small data sets, this was mainly due to the large computation time needed for solving linear and nonlinear programming problems (16). Those heuristic algorithms not relying on (16) (e.g., θ-GREEDY and ϕ_α-GREEDY) can be successfully applied to much larger data sets. However, further efforts would be necessary to make them practically useful. For example, evaluating a separation measure for a pdBf (T, F) takes $O(n|T||F|)$ time if it is implemented naively. For some separation measures, this can be improved to $O(n(|T| + |F|)\log(|T| + |F|))$. Even for other separation measures, an approximate evaluation of its value may be done much faster than $O(n|T||F|)$. There are several other issues related to the implementation of practical codes.

References

[1] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large databases, International Conference on Management of Data (SIGMOD 93), (1993) 207-216.

[2] N. Alon, J.H. Spencer and P. Erdős, *The Probabilistic Method*, John Wiley & Sons, New York, 1992.

[3] D. Angluin, Queries and concept learning, *Machine Learning*, 2 (1988) 319-342.

[4] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, 1982.

[5] A. Blum, L. Hellerstein, and N. Littlestone, Learning in the presence of finitely or infintely many irrelevant attributes. *Journal of Computer and System Sciences* **50** (1995) pp. 32-40.

[6] A. Blumer, A. Ehrenfeucht, D. Haussler and M. K. Warmuth, Occam's razor, *Information Processing Letters*, 24 (1987) 377-380.

[7] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz and I. Muchnik, An implementation of logical analysis of data, RUTCOR Research Report RRR 22-96, Rutgers University, 1996; *IEEE Trans. on Knowledge and Data Engineering*, to appear.

[8] E. Boros, T. Ibaraki and K. Makino, Error-free and best-fit extensions of a partially defined Boolean function, *Information and Computation*, 140 (1998) 254-283.

[9] E. Boros, T. Ibaraki and K. Makino, Logical analysis of binary data with missing bits, *Artificial Intelligence*, 107 (1999) 219–264.

[10] N. Bshouty and L. Hellerstein, Attribute-efficient learning in query and mistake-bound models, *J. of Comput. Syst. Sci.* 56 (1998) 310-319.

[11] V. Chvátal, A greedy heuristic for the set-covering problem, *Mathematics of Operations Research* **4**(3) (1979) 233–235.

[12] M. Conforti and G. Cornuejols, Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem, *Discrete Applied Mathematics*, 7 (1984) 251-274.

[13] Y. Crama, P. L. Hammer and T. Ibaraki, Cause-effect relationships and partially defined Boolean functions, *Annals of Operations Research*, 16 (1988) 299-326.

[14] P. Damaschke, Adaptive versus nonadaptive attribute-efficient learning, *Proceedings of the* 30^{th} *Annual ACM Symposium on Theory of Computing*, 1998, pp. 590-596.

[15] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, 1996.

[16] U. Feige, A threshold of $\ln n$ for approximating set cover, ACM Symposium on Theory of Computing, 1996.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, New York, 1979.

[18] R. Greiner, A. Grove and A. Kogan, Exploiting the omission of irrelevant data, *Artificial Intelligence*, 97 (1997) 345-380.

[19] D. S. Hochbaum and A. Pathria, Analysis of the greedy approach in covering problems, *Naval Research Quarterly*, 45 (1998) 615-627.

[20] T. Ibaraki and M. Fukushima, *FORTRAN 77 Optimization Programming* (in Japanese), Iwanami, 1991.

[21] D. S. Johnson, Approximation algorithms for combinatorial problems, *J. Computer and System Sciences*, 9 (1974) 256-278.

[22] S. Khuller, A. Moss and J. Naor, The budgeted maximum coverage problem, *Information Processing Letters*, **70** (1999) 39-45.

[23] N. Littlestone, Learning quickly when irreleveant attributes abound: a new linear-threshold algorithm, *Machine Learning* **2** (1988) pp. 285-318.

[24] L. Lovász, On the ratio of optimal integral and fractional covers, *Discrete Mathematics*, **13**(4) (1975) 383–390.

[25] O.L. Mangasarian, R. Setiono and W.H. Wolberg, Pattern recognition via linear programming: Theory and Applications to medical diagnosis, in *Large-Scale Numerical Optimization*, edited by T.F. Coleman and Y. Li, SIAM Publications, Philadelphia, (1990) 22-30.

[26] H. Mannila, H. Toivonen and A.I. Verkamo, Efficient algorithms for discovering association rules, AAAI Workshop on Knowledge Discovery in Database, edited by U.M. Fayyad and R. Uthurusamy, (1994) 181-192.

[27] P. M. Murphy and D. W. Aha, UCI Repository of machine learning databases [http://www.ics.uci.edu/˜mlearn/MLRepository.html], University of California, Department of Information and Computer Science, 1994.

[28] G. L. Nemhauser and L. Wolsey, Maximizing submodular set functions: formulations and analysis of algorithms, in *Studies of Graphs and Discrete Programming*, North-Holland, Amsterdam, 1981, pp. 279-301.

[29] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, 1994.

[30] J. R. Quinlan, Induction of decision trees, *Machine Learning*, 1 (1986) 81-106.

[31] L. G. Valiant, A theory of the learnable, *Communications of the ACM*, 27 (1984) 1134-1142.

Appendix: Proofs of lemmas

Lemma 2 For any δ (> 0), we have

$$\mathbf{Prob}[m_T^S < \mu - \delta] \leq e^{-2\delta^2/m_T},$$

where $\mu = m_T\left(1 - \frac{m_T - 1}{2^{K+1}}\right)$.

Proof: Let D_i be a set such that $D_i \subseteq \mathbb{B}^K \setminus \{a_1[S], a_2[S], \ldots, a_{i-1}[S]\}$ and $|D_i| = 2^K - i + 1$ hold (recall that some of $a_1[S], a_2[S], \ldots, a_{i-1}[S]$ may be identical). Let $\tilde{Y}_i$ be the indicator random variable taking value 1 if $a_i \in D_i$, and 0 otherwise. Let $\tilde{m}_T^S = \sum_{i=1}^{m_T} \tilde{Y}_i$. Note that we can always specify such set D_i before choosing a_i, although the choice of D_i depends on $\{a_1, a_2, \ldots, a_{i-1}\}$. As $\tilde{Y}_i = 1$ implies $Y_i = 1$, $\mathbf{Prob}(Y_i = 1) \geq \mathbf{Prob}(\tilde{Y}_i = 1)$ holds. Let q_i be

$$q_i = \mathbf{Prob}(\tilde{Y}_i = 1) = 1 - \frac{i-1}{2^K},$$

and let $\mu = \mathbf{E}(\tilde{m}_T^S)$. Then, by the linearity of expectation, we have

$$\begin{aligned} \mu &= \sum_{i=1}^{m_T} \mathbf{E}(\tilde{Y}_i) \\ &= \sum_{i=1}^{m_T} q_i \\ &= m_T\left(1 - \frac{m_T - 1}{2^{K+1}}\right). \end{aligned} \tag{27}$$

Let $\tilde{Z}_i = \tilde{Y}_i - q_i$ and $\tilde{Z} = \sum_{i=1}^{m_T} \tilde{Z}_i$; then $\tilde{Z}_i$ is the random variable satisfying

$$\tilde{Z}_i = \begin{cases} 1 - q_i, & \text{with probability } q_i \\ -q_i, & \text{otherwise} \end{cases}$$

and $\tilde{Z} = \tilde{m}_T^S - \mu$. For any $\delta > 0$, as $\mathbf{Prob}(\tilde{Z} < -\delta) \leq e^{-2\delta^2/m_T}$ holds by using Lemma 1, we have

$$\begin{aligned} \mathbf{Prob}(m_T^S < \mu - \delta) &\leq \mathbf{Prob}(\tilde{m}_T^S < \mu - \delta) \\ &= \mathbf{Prob}(\tilde{Z} < -\delta) \\ &\leq e^{-2\delta^2/m_T}. \end{aligned}$$

■

Lemma 3 For any constant $c > 1$, suppose $m_T \leq 2^K/c$ and let $c' = (1 - 1/c)^c$. Then $\mathbf{E}(X_S) \geq (c')^{m_T m_F/2^K}$ holds for any $S \subseteq V$ of size K.

Proof: By the assumption,

$$\begin{aligned}
\mathbf{E}(X_S) &\geq \left(\frac{2^K - m_T}{2^K}\right)^{m_F} \\
&= \left\{\left(1 - \frac{m_T}{2^K}\right)^{\frac{2^K}{m_T}}\right\}^{\frac{m_T m_F}{2^K}} \\
&\geq (c')^{\frac{m_T m_F}{2^K}}
\end{aligned}$$

holds, since $(1 - 1/x)^x$ is monotonically increasing over $x \geq 1$. ■

Lemma 4 For any S of size K, if $m_T = o(2^{3K/5})$ and $m_F = o(2^{3K/5})$, $\mathbf{E}(X_S) \leq (2 + o(1))e^{-m_T m_F/2^K}$ holds, where $o(1) \to 0$ if $K \to \infty$.

Proof: By substituting δ in (12) with $m_T\sqrt{m_F/2^{K+1}}$, we have

$$\begin{aligned}
\mathbf{E}(X_S) &\leq e^{-\frac{m_T m_F}{2^K} + \frac{m_T(m_T-1)m_F}{2^{2K+1}} + \frac{m_T m_F \sqrt{m_F}}{2^K\sqrt{2^{K+1}}}} + e^{-\frac{m_T m_F}{2^K}} \\
&= e^{-\frac{m_T m_F}{2^K} + o(1)} + e^{-\frac{m_T m_F}{2^K}} \\
&= (2 + o(1))e^{-\frac{m_T m_F}{2^K}}.
\end{aligned}$$

■

Table 1: Lower and upper bounds on $m_T = m_F$ to satisfy $\mathbf{E}(X(K)) = 1$.

	K	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$n = 10$	LB	11	17	23	30	33											
	UB	30	29	34	40	42											
$n = 15$	LB	13	21	31	45	63	88	119	156	194	209						
	UB	∞	41	49	63	82	107	139	176	213	227						
$n = 20$	LB	15	23	35	51	75	108	153	216	300	413	559	743	958	1171	1252	
	UB	∞	51	58	75	99	134	182	247	334	449	599	785	1002	1215	1292	
$n = 40$	LB	17	27	42	63	94	139	204	297	430	619	888	1270	1811	2576	3657	5179
	UB	∞	∞	78	98	131	179	249	347	487	684	963	1356	1911	2693	3793	5338
$n = 100$	LB	19	31	48	74	113	168	250	367	537	782	1135	1643	2372	3418	4917	7062
	UB	∞	∞	118	126	165	225	312	438	619	876	1245	1771	2522	3595	5126	7310
$n = 1000$	LB	23	38	61	95	146	221	331	491	724	1063	1555	2265	3293	4775	6912	9989
	UB	∞	∞	∞	210	240	318	437	610	860	1219	1736	2478	3543	5071	7263	10408

Table 2: Sizes of six types of problem instances.

Types of instances	$\|H\|$	$\|D\|$	$\|R\|$	$\|T\| + \|F\|$
A1	10	5	5	200
A2	10	5	5	400
B1	5	10	10	200
B2	5	10	10	400
C1	5	0	40	200
C2	5	0	40	400

Table 3: Performance summary for random functions.

Types	ρ-ROUNDING				σ-GREEDY				ρ-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	12.55	0.579	0.141	0.279	14.83	0.586	0.131	0.283	11.05	0.538	0.155	0.307
A2	13.78	0.674	0.114	0.212	15.88	0.614	0.114	0.272	11.40	0.791	0.083	0.126
B1	5.85	0.619	0.381	0.000	11.30	0.404	0.297	0.299	5.22	0.738	0.263	0.000
B2	6.17	0.581	0.419	0.000	10.81	0.462	0.303	0.236	5.38	0.716	0.284	0.000
C1	12.78	0.281	0.000	0.719	12.92	0.317	0.000	0.684	7.80	0.623	0.000	0.377
C2	12.57	0.394	0.000	0.606	13.23	0.366	0.000	0.634	6.88	0.812	0.000	0.188

Types	θ-ROUNDING				θ-GREEDY				θ-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	13.35	0.547	0.105	0.348	11.18	0.602	0.131	0.267	10.75	0.699	0.082	0.219
A2	15.07	0.580	0.090	0.330	12.14	0.756	0.076	0.168	—	—	—	—
B1	5.42	0.713	0.287	0.000	6.14	0.567	0.400	0.032	6.00	0.550	0.372	0.078
B2	5.53	0.728	0.273	0.000	6.01	0.591	0.392	0.018	—	—	—	—
C1	11.25	0.319	0.000	0.681	5.84	0.866	0.000	0.134	6.15	0.848	0.000	0.152
C2	9.82	0.608	0.000	0.392	5.24	0.976	0.000	0.024	—	—	—	—

Types	ϕ_α-ROUNDING				ϕ_α-GREEDY				ϕ_α-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	14.00	0.580	0.068	0.352	11.18	0.615	0.109	0.275	10.75	0.740	0.038	0.222
A2	15.25	0.694	0.066	0.330	12.12	0.771	0.062	0.167	10.78	0.913	0.010	0.078
B1	10.62	0.400	0.213	0.387	6.10	0.595	0.377	0.029	5.80	0.704	0.223	0.073
B2	11.68	0.416	0.191	0.458	6.00	0.596	0.389	0.016	5.70	0.730	0.216	0.053
C1	8.30	0.652	0.000	0.348	5.78	0.885	0.000	0.115	5.67	0.923	0.000	0.077
C2	6.20	0.901	0.000	0.099	5.22	0.976	0.000	0.024	5.00	1.000	0.000	0.000

$\alpha = 0.001$

$r_H = |S \cap H|/|S|,\ r_D = |S \cap D|/|S|,\ r_R = |S \cap R|/|S|$

Table 4: Performance summary for DNF functions.

Types	ρ-ROUNDING				σ-GREEDY				ρ-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	11.88	0.610	0.116	0.273	14.45	0.557	0.240	0.203	10.60	0.570	0.128	0.302
A2	13.18	0.668	0.102	0.229	16.04	0.558	0.241	0.201	11.57	0.740	0.065	0.195
B1	6.67	0.451	0.525	0.024	10.10	0.308	0.563	0.129	5.40	0.546	0.454	0.000
B2	6.28	0.492	0.502	0.006	9.35	0.342	0.595	0.064	5.28	0.605	0.395	0.000
C1	11.35	0.360	0.000	0.640	11.75	0.367	0.000	0.633	7.33	0.679	0.000	0.321
C2	12.43	0.377	0.000	0.623	12.51	0.402	0.000	0.598	6.60	0.821	0.000	0.180

Types	θ-ROUNDING				θ-GREEDY				θ-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	12.00	0.623	0.087	0.290	10.08	0.806	0.061	0.134	10.22	0.773	0.050	0.177
A2	13.97	0.650	0.047	0.303	10.20	0.970	0.008	0.022	—	—	—	—
B1	5.67	0.603	0.393	0.004	5.72	0.512	0.463	0.025	5.65	0.483	0.479	0.039
B2	5.53	0.631	0.369	0.000	5.47	0.547	0.453	0.000	—	—	—	—
C1	9.50	0.492	0.000	0.509	5.14	0.968	0.000	0.032	5.83	0.889	0.000	0.111
C2	7.55	0.772	0.000	0.228	5.00	1.000	0.000	0.000	—	—	—	—

Types	ϕ_α-ROUNDING				ϕ_α-GREEDY				ϕ_α-STINGY			
	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R	$\|S\|$	r_H	r_D	r_R
A1	12.18	0.643	0.053	0.304	10.03	0.836	0.040	0.124	9.82	0.897	0.010	0.094
A2	13.95	0.647	0.027	0.326	12.12	0.771	0.062	0.167	10.35	0.939	0.004	0.057
B1	8.75	0.422	0.416	0.162	5.75	0.498	0.474	0.028	5.72	0.503	0.454	0.043
B2	8.97	0.407	0.429	0.164	5.47	0.560	0.440	0.000	5.47	0.558	0.430	0.013
C1	6.95	0.771	0.000	0.229	5.13	0.975	0.000	0.025	5.20	0.964	0.000	0.036
C2	5.47	0.949	0.000	0.051	5.00	1.000	0.000	0.000	5.00	1.000	0.000	0.000

$\alpha = 0.001$

$r_H = |S \cap H|/|S|,\ r_D = |S \cap D|/|S|,\ r_R = |S \cap R|/|S|$

Table 5: Average sizes of $|S|$ with different α (random functions).

Instances	ϕ_α-GREEDY				ϕ_α-STINGY			
	$\alpha = 0.1$	0.01	0.001	0.0	$\alpha = 0.1$	0.01	0.001	0.0
A1	11.43	11.18	11.18	11.18	10.95	10.82	10.75	10.75
A2	12.10	12.12	12.12	12.14	11.40	11.05	10.78	—
B1	6.67	6.13	6.10	6.14	6.53	6.05	5.80	6.00
B2	6.53	6.00	6.00	6.01	6.03	5.63	5.70	—
C1	5.60	5.78	5.78	5.84	6.53	5.80	5.67	6.15
C2	5.28	5.22	5.22	5.24	5.58	5.60	5.00	—