Pattern Recognition 73 (2018) 172–188

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

A fast Branch-and-Bound algorithm for U-curve feature selection

Esmaeil Atashpaz-Gargari[a,1], Marcelo S. Reis[b], Ulisses M. Braga-Neto[a,c,*], Junior Barrera[d], Edward R. Dougherty[a,c]

[a] Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA
[b] Center of Toxins, Immune-response and Cell Signaling (CeTICS), LECC, Instituto Butantan, São Paulo, Brazil
[c] Center for Bioinformatics and Genomics Systems Engineering, TEES, College Station, TX, USA
[d] Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil

ARTICLE INFO

Article history:
Received 21 June 2015
Revised 9 August 2017
Accepted 11 August 2017
Available online 18 August 2017

Keywords:
Branch-and-bound algorithm
Feature selection
U-curve assumption
W-operator design

ABSTRACT

We introduce a fast Branch-and-Bound algorithm for optimal feature selection based on a U-curve assumption for the cost function. The U-curve assumption, which is based on the peaking phenomenon of the classification error, postulates that the cost over the chains of the Boolean lattice that represents the search space describes a U-shaped curve. The proposed algorithm is an improvement over the original algorithm for U-curve feature selection introduced recently. Extensive simulation experiments are carried out to assess the performance of the proposed algorithm (IUBB), comparing it to the original algorithm (UBB), as well as exhaustive search and Generalized Sequential Forward Search. The results show that the IUBB algorithm makes fewer evaluations and achieves better solutions under a fixed computational budget. We also show that the IUBB algorithm is robust with respect to violations of the U-curve assumption. We investigate the application of the IUBB algorithm in the design of imaging *W*-operators and in classification feature selection, using the average mean conditional entropy (MCE) as the cost function for the search.

1. Introduction

Feature selection is the problem of finding an optimal subset of a finite set of features that minimizes a cost function that is correlated to the classification error (e.g., the estimated classification error) [8]. Determining the optimal set of features can be a complicated task, since for a problem with n features, an exhaustive search requires considering all 2^n possible feature sets. The Cover–Campenhout theorem [3] stipulates that to be guaranteed to find the optimal feature set, no algorithm can avoid the exponential complexity of exhaustive search, in a worst-case sense, unless there is extra information about the problem.

Algorithms have been proposed that use heuristics to attempt to find the optimal feature set in fewer evaluations than exhaustive search; among them are feature selection algorithms based on the well-known Branch-and-Bound (BB) paradigm for discrete and combinatorial optimization [5,12]. A BB algorithm uses some property of the cost function, such as monotonicity, to accomplish a systematic enumeration of the features sets in the form of a *tree*. At each step of the algorithm, the tree is traversed (*Branch*) and the cost of the best feature set found until that step is recorded (*Bound*). If the cost of a node is smaller than the bound, its successor nodes are explored further and the bound is updated. Otherwise, the successors of that node can be safely discarded or *pruned*, by exploring the monotonicity of the cost. If the tree is organized in such a way that large sections of it can be pruned en masse, then the BB algorithm is successful. Different improvements have been proposed to enhance the performance of the basic BB algorithm [11]. Yu and Yuan [17] suggest avoiding the evaluation of intermediate single-branching nodes by obtaining a "minimum search tree." Also ordering the nodes in the tree based on the significance of the features is used in some of the variants of the BB algorithm [11]. In addition, to minimize the number of cost evaluations, some algorithms use analytical properties of the search space [16].

It is well-known that the optimal classification error is monotonically nonincreasing with an increasing number of features [4], making it a perfect candidate for a cost function for a BB algorithm. However, the optimal classifier and optimal classification error are rarely known in practice, and the criterion used is typically the classification error for a classifier designed using sample data, which does not generally decrease monotonically. Rather, increas-

* Corresponding author at: Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA.
E-mail address: ulisses@ece.tamu.edu (U.M. Braga-Neto).
[1] Present address: School of Engineering and Computing, National University, San Diego, CA, USA.

http://dx.doi.org/10.1016/j.patcog.2017.08.013
0031-3203/

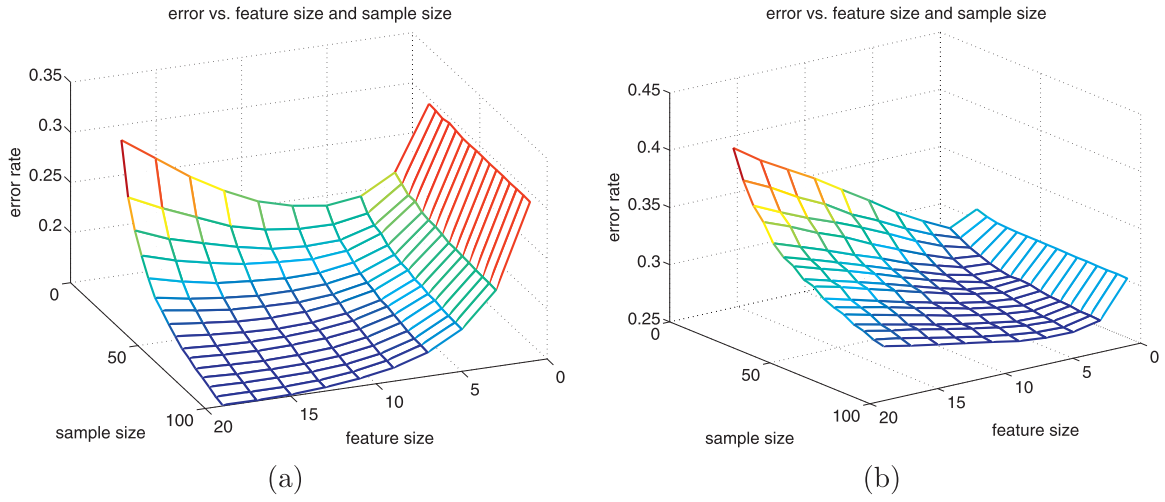

Fig. 1. Peaking phenomenon. (a) Slightly correlated features. $\rho = 0.125$. (b) Highly correlated features. $\rho = 0.5$. Reproduced from [15].

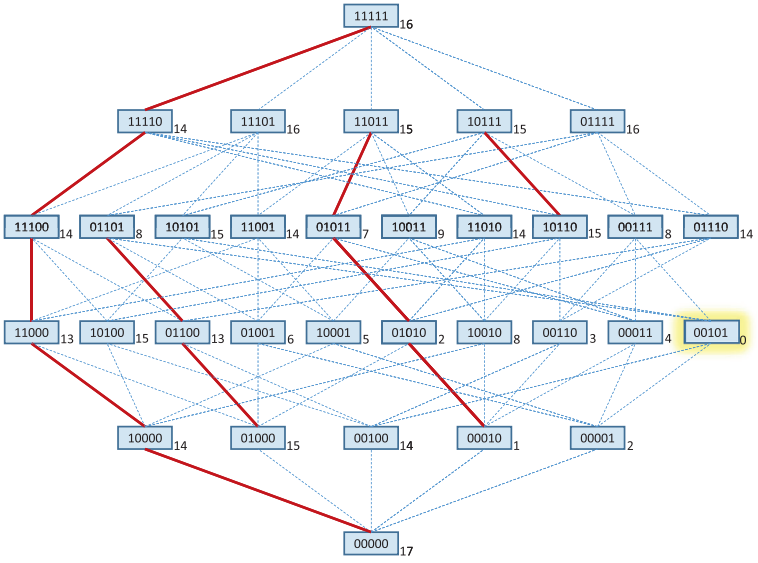

Fig. 2. Lattice for 5 features, with 4 chains highlighted in red. The cost function for this example is decomposable in U-shaped curves. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ing the number of features used to design the classifier, with a fixed sample size, generally makes the expected error of the designed classifier decrease and then increase. This is known as the *peaking phenomenon*, which was first studied in [7].

Fig. 1(a) and (b) shows the peaking phenomenon for the Linear Discriminant Analysis (LDA) classification rule. In Fig. 1(a) the features are slightly correlated. In this case, peaking occurs earlier (i.e., for a smaller number of features) or later depending on the sample size. For example, for sample size 30, peaking occurs with about 6 features, but when sample size increases to 100, peaking occurs at a larger feature size. In Fig. 1(b) the features are highly correlated. As we see in this case, even for a large sample size, peaking occurs early.

Due to the peaking phenomenon, the error of the designed classifier (as opposed to the optimal classification error) is likely to display a U-shaped behavior along a chain of increasing nested feature sets. Thus, it is reasonable to make the assumption that all the chains of the Boolean lattice that represent the search space have U-shaped behavior (U-curve assumption). The U-curve assumption was used by Ris and colleagues to formulate the *U-curve* optimization problem, which in turn can be employed to model the feature selection step of classifier design [14]. To solve this problem, the original BB algorithm, or its variants mentioned previously, are not suitable, as all of these algorithms assume that the cost function is monotone. Hence, the solution found by these algorithms will not necessarily be the globally best possible feature set. A feature selection algorithm based on a U-shaped cost function was proposed in [14]. They also presented some principles for a Branch-and-Bound procedure to tackle the U-curve problem, which were developed by Reis into the U-curve Branch-and-Bound (UBB) algorithm [13].

In this paper, we propose and evaluate a Branch-and-Bound algorithm for the U-curve optimization problem, which outperforms the original UBB algorithm, and investigate its application in the design of imaging *W*-operators and in feature selection for classifier design. Section 2 presents a formal description of the U-curve optimization problem and also reviews the original UBB algorithm. In Section 3, we introduce the Improved UBB (IUBB) algorithm, a

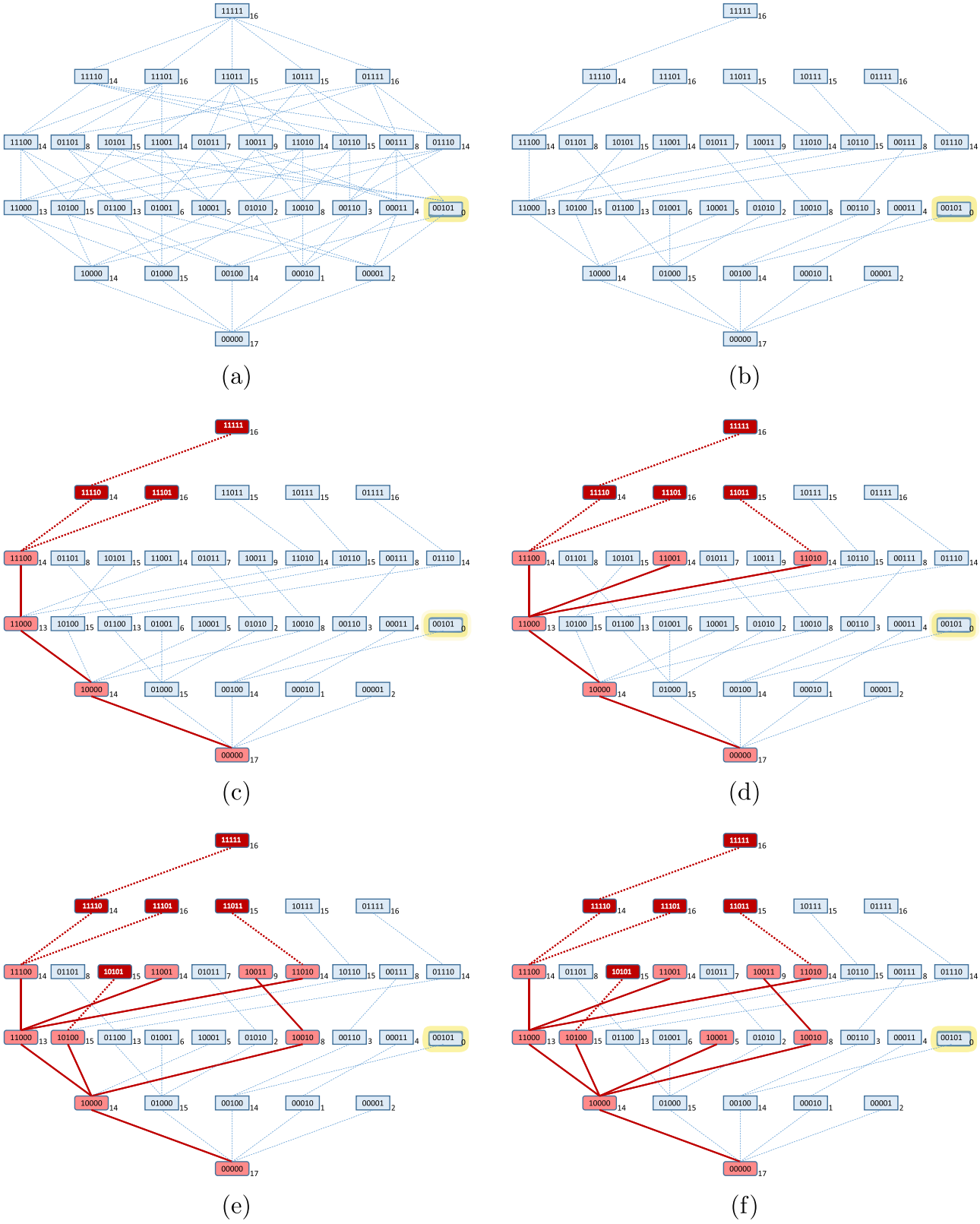

Fig. 3. The UBB algorithm. (a) Search space. (b) The tree produced by enumeration scheme. (c–f) Four iterations of the algorithm. Visited nodes are colored pink, while pruned nodes are colored red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

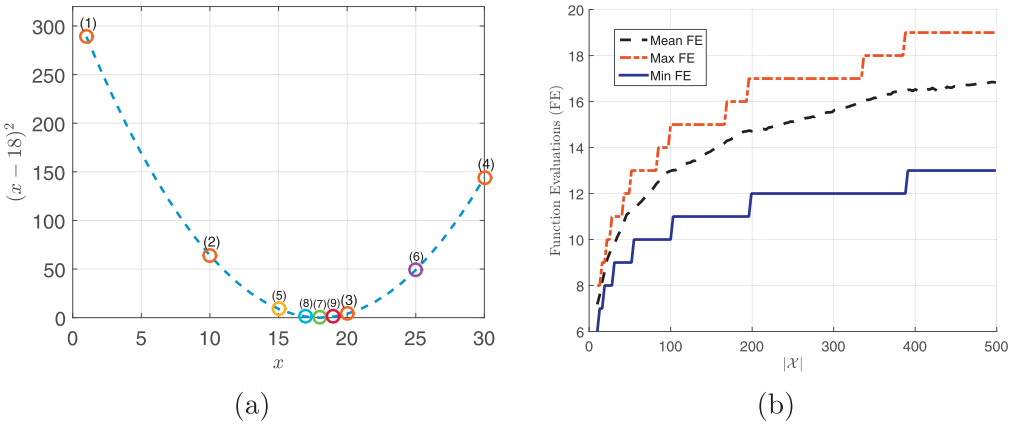

Fig. 4. (a) Illustration of the bisection algorithm steps for a sample chain of size 30 with $i^* = 18$. (b) Number of function evaluations required by bisection to find the minimum-cost feature set F_{i^*} of the chain $\mathcal{F}$ vs. $|\mathcal{F}|$, when i^* is uniformly distributed in the set $2, \ldots, |\mathcal{F}| - 1$ and the cost function is $c(F_i) = (i - i^*)^2$.

fast method to solve the U-curve optimization problem, which has two main innovations in relation to the original UBB algorithm: iterative updating of optimal chains and the use of bisection to find chain minima. In Section 4, we introduce a model for the U-curve feature selection problem and employ it as a synthetic benchmark to assess the performance of the IUBB algorithm, as well as compare it to the original UBB algorithm; the case where the U-curve assumption is violated is also investigated. Section 6 considers the application of the IUBB algorithm in the design of imaging W-operators using the average mean conditional entropy (MCE) as the cost function for the search. Section 7 assesses the performance of the IUBB algorithm in an actual classification problem using synthetic data. In all our experiments, we have observed that IUBB displays a qualitative superior performance to the original UBB algorithm. This and other conclusions and directions for future research are discussed in Section 8.

2. U-curve Branch and Bound algorithm

In this section, we introduce the U-curve optimization problem formally, and review the UBB algorithm. Let S be a set of features, and let $\mathcal{P}(S)$ denote the set of all possible feature sets.

Definition 2.0.1 (Chain)**.** A chain is a collection of feature sets $\mathcal{F} = \{F_1, F_2, \ldots, F_k\} \subseteq \mathcal{P}(S)$, such that $F_1 \subseteq F_2 \subseteq \ldots \subseteq F_k$.

Definition 2.0.2 (U-shaped curve)**.** Let $\mathcal{F} \subseteq \mathcal{P}(S)$ be a chain. A function $f : \mathcal{F} \to \mathbb{R}$ describes a U-shaped curve if $F_1 \subseteq F_2 \subseteq F_3$ implies that $f(F_2) \leq \max\{f(F_1), f(F_3)\}$, for $F_1, F_2, F_3 \in \mathcal{F}$.

Definition 2.0.3 (Decomposability in U-shaped curves)**.** A cost function $c : \mathcal{P}(S) \to \mathbb{R}$ is decomposable in U-shaped curves if, for each chain $\mathcal{F} \subseteq \mathcal{P}(S)$, the restriction of c to $\mathcal{F}$ describes a U-shaped curve.

Definition 2.0.4 (Minimum cost)**.** Let $F^* \in \mathcal{F} \subseteq \mathcal{P}(S)$ and let c be a cost function defined on $\mathcal{P}(S)$. If there does not exist another feature set $F \in \mathcal{F}$ such that $c(F) < c(F^*)$, then F^* is of minimum cost in $\mathcal{F}$. If $\mathcal{F} = \mathcal{P}(S)$, then we say that F^* is of minimum cost.

The previous definitions are illustrated in Fig. 2, which displays a Boolean lattice corresponding to $\mathcal{P}(S)$, where S is a set of 5 features. Each node in the lattice denotes a distinct feature set $F \in \mathcal{P}(S)$, where "0" and "1" indicate whether the corresponding feature is absent or present, respectively. Four different chains are shown in red. The cost $c(F)$ of each feature set $F \in \mathcal{P}(S)$ is indicated next to the corresponding node. Notice that c is decomposable in U-shaped curves. The feature set F^* of minimum cost in this example has cost zero and is highlighted in yellow.

The U-curve problem may be defined as follows: for a cost function $c : \mathcal{P}(S) \to \mathbb{R}$ that is decomposable in U-shaped curves, find a feature set $F^* \in \mathcal{P}(S)$ of minimum cost.

2.1. U-curve Branch and Bound (UBB) algorithm

The UBB algorithm, proposed in [13], uses the U-assumption to find the feature set of minimum cost in $\mathcal{P}(S)$ without evaluating all the elements in $\mathcal{P}(S)$. Through a recursive enumeration scheme, it first constructs a tree and then uses it as the search space. The fact that c is decomposable in U-shaped curves is used to prune the tree during the search: the tree is pruned when the cost of an element in the search chain starts increasing. Although in the early iterations of the UBB algorithm, finding a minimum element in a chain leads to removal of many elements in the tree, in later iterations the search chains are not the best possible chains in the search space and the pruning becomes very slow. UBB iterates over the pruned tree until the search space is exhausted. Fig. 3 illustrates the UBB algorithm, using the lattice in Fig. 2. More details about the UBB algorithm, including pseudocode, can be found in [13].

3. Improved U-curve Branch and Bound algorithm

The results in [13] show that the UBB algorithm requires fewer function calls compared to Exhaustive Search (ES) in finding the global best solution of the U-curve problem. However, the number of function calls in UBB is still high: in the numerical experiments of [13], UBB required about half of the function calls of ES. To tackle the large number of function calls of UBB, an improved algorithm is proposed here. The Improved UBB (IUBB) algorithm is based on two main innovations:

(1) **Iterative updating of optimal chains.**

To improve the pruning process, instead of limiting the search to the tree structure constructed through the enumeration process of UBB, at each step of IUBB, we determine a chain in the search space that leads to pruning the maximum number of elements; such chain is called optimal. In order to update the search after each pruning, at the beginning of each iteration, we need to update a data structure that manages the current state of the search space. This is done as follows. For a feature selection problem with n features, the search space can be represented by a Boolean lattice $\mathcal{L}$ of degree n. Let $\mathcal{L}$ be organized in layers, as represented in Figs. 2 and 3, where L_l denotes the lth layer, for $l = 0, 1, \ldots, n$, that is, let L_l contain all possible feature sets of size l. The proposed data structure contains vectors that store the pruning gain of

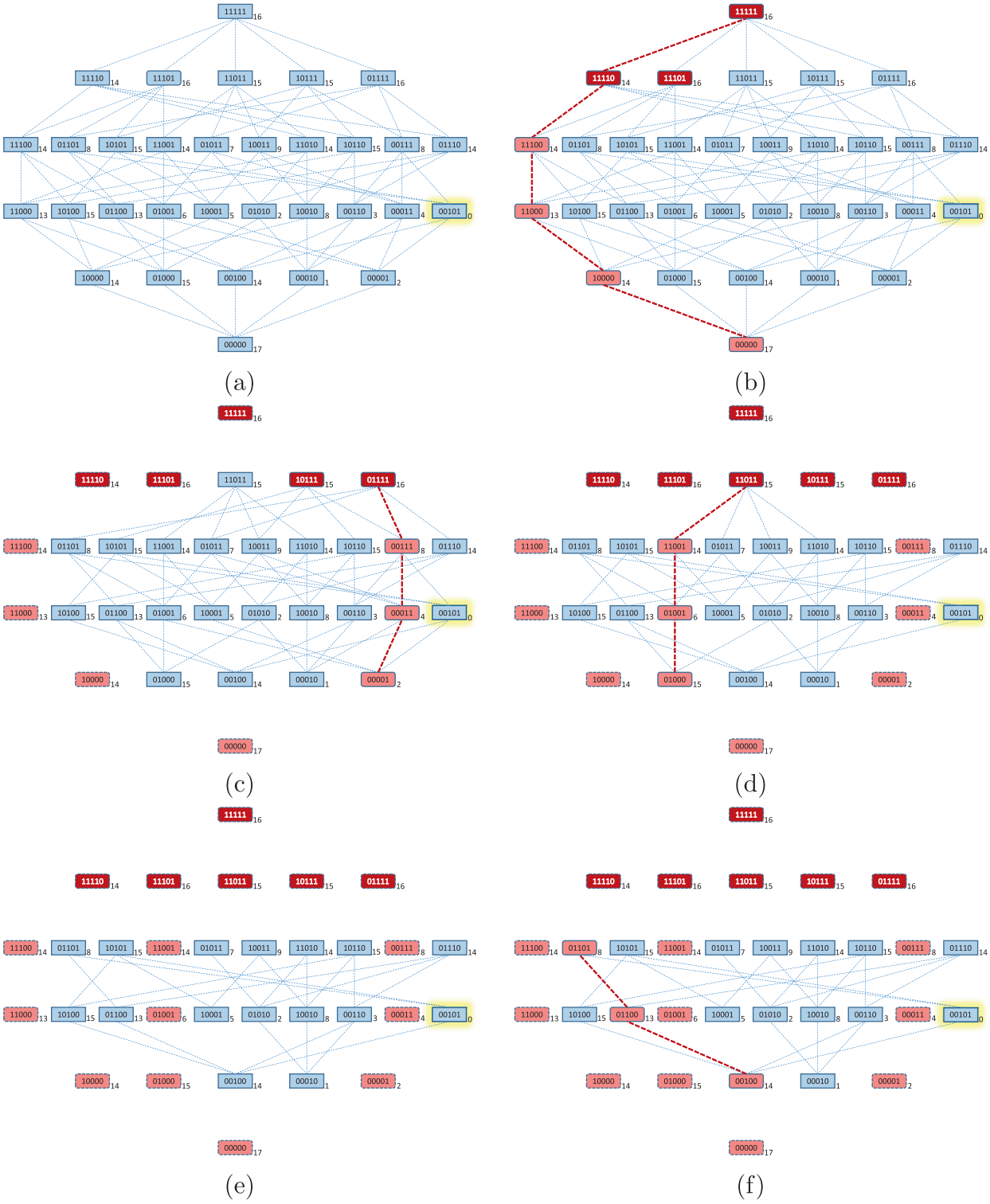

Fig. 5. The IUBB algorithm. (a) The original search space. (b–f) Five iterations of the algorithm. Visited nodes are colored pink, while pruned nodes are colored red. The selected optimal chain in each iteration is shown by a red dashed line in all the diagrams. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

each element in the current search space. Those vectors are computed recursively with the assistance of adjacency matrices. For $l = 0, 1, \ldots, n-1$, let $\mathbf{R}_l = [r_{ij}]$ be a matrix of size $\binom{n}{l} \times \binom{n}{l+1}$, with (i, j)-element given by

$$r_{ij} = \begin{cases} 1, & \text{if the Hamming distance between } F_j^l \text{ and } F_i^{l+1} \\ & \quad \text{is equal to 1,} \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

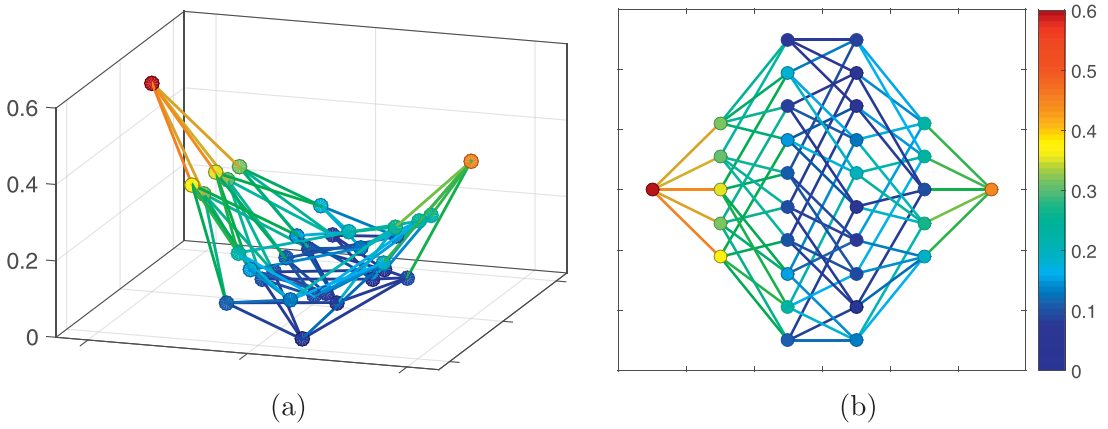

Fig. 6. A sample U-shaped cost function and the U-shaped chains generated using the proposed benchmark function model with $n = 5$ and $\alpha = 0.6$. (a) 3D representation. (b) 2D representation.

where F_i^l and F_j^{l+1} are elements of L_l and L_{l+1}, respectively. Thus, if $r_{ij} = 1$, then the two feature sets F_i^l and F_j^{l+1} are on the same chain. For each $l = 0, 1, \ldots, n$ let $\mathbf{V}_l = [v_{li}]$ be an auxiliary vector such that

$$v_{li} = \sum_{j=1}^{|L_{l+1}|} r_{ij}, \quad i = 1, \ldots, |L_l| . \tag{2}$$

Finally, let the main vectors that control the current search space be constructed recursively as

$$\begin{aligned} \mathbf{T}_n &= 0 , \\ \mathbf{T}_l &= \mathbf{V}_l + \mathbf{R}_l \times \mathbf{T}_{l+1} , \quad l = n-1, n-2, \ldots, 0 . \end{aligned} \tag{3}$$

The ith element of $\mathbf{T}_l$ indicates the pruning gain of its corresponding element in the search space. We use $\mathbf{T}_l$ to find chains for which finding the minimum element results in maximum pruning of the search space. At each step of the algorithm, an optimal chain $\mathcal{F}^*$ is found, and after determining the minimum element of the chain, all the states connected to the optimal element are removed from the search space. Then the matrices $\mathbf{R}_l$, $\mathbf{C}_l$ and $\mathbf{T}_l$ are updated, and the algorithm continues until there are no remaining elements in the search space.

(2) **Use of bisection to find the chain minimum.**

A drawback of UBB is that in finding the minimum element of a chain, it searches all the nodes in the chain before reaching a feature set that shows an increase in cost value. That is, for a chain $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$, if the U-curve cost function has a minimum $F^* = F_{i^*}$, the algorithm evaluates the cost function $(i^* + 1)$ times to find F^*. When dealing with a large number of features, the algorithm will tend to need a large number of unnecessary function calls to find F^*. To use the U-curve assumption efficiently, a faster method based on bisection is proposed to find the minimum element of the chain. Bisection changes the complexity of finding the minimum of the chain $\mathcal{F}$ from $O(|\mathcal{F}|)$ to $O(\log(|\mathcal{F}|))$. The pseudocode for the bisection algorithm is given in Algorithm 1. Fig. 4(a) illustrates the bisection algorithm steps for a sample chain of size 30 with $i^* = 18$. As we see, bisection requires only 9 function calls to find the F^* while 19 function evaluations needed by the linear search in the original UBB algorithm for the same problem. Fig. 4(b) displays the number of function calls required to find the minimum-cost feature set F^* in the chain when i^* is uniformly distributed in the set $2, \ldots, |\mathcal{F}| - 1$ and the cost function is $c(F_i) = (i - i^*)^2$. As we see, at $|\mathcal{F}| = 500$, bisection requires on average 17 function evaluations, while $|\mathcal{F}|/2 = 250$ function evaluations are needed on average by the method in UBB to find the minimum-cost feature set F_i^*, which is 14 times the number of function evaluations made by IUBB. The UBB algorithm uses the U-curve assumption only as the stopping criterion for the search, whereas the IUBB algorithm uses the assumption for optimizing the chain search process, as well as using it as a stopping criterion.

Algorithm 1 Bisection

Initialize the algorithm for the chain $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$.
$i_L \Leftarrow 1, i_{ML} \Leftarrow \lfloor k/3 \rfloor, i_{MH} \Leftarrow \lfloor 2 \times k/3 \rfloor, i_H \Leftarrow k$
$c_L \Leftarrow c(F_{i_L}), c_{ML} \Leftarrow c(F_{i_{ML}}), c_{MH} \Leftarrow c(F_{i_{MH}}), c_H \Leftarrow c(F_{i_H})$
$(i_{Best}, c_{Best}) \Leftarrow \min(c_L, c_{ML}, c_{MH}, c_H)$
while $((i_H - i_{MH} \geq 1) \vee (i_{MH} - i_{ML} \geq 1) \vee (i_{ML} - i_L)$ **do**
  if $((c_L \geq c_{ML}) \wedge (c_{ML} \geq c_{MH})$ **then**
    if $(i_H - i_{MH}) \geq (i_{MH} - i_{ML})$ **then**
      $i_L \Leftarrow i_{ML}, i_{ML} \Leftarrow \lfloor (i_{ML} + i_{MH})/2 \rfloor, i_{MH} \Leftarrow i_{MH}, i_H \Leftarrow i_H$
    else
      $i_L \Leftarrow i_{ML}, i_{ML} \Leftarrow i_{MH}, i_{MH} \Leftarrow \lfloor (i_{MH} + i_H)/2 \rfloor, i_H \Leftarrow i_H$
    end if
  else if $((c_H \geq c_{MH}) \wedge (c_{MH} \geq c_{ML})$ **then**
    if $(i_{MH} - i_{ML}) \geq (i_{ML} - i_L)$ **then**
      $i_H \Leftarrow i_{MH}, i_{MH} \Leftarrow \lfloor (i_{ML} + i_{MH})/2 \rfloor, i_{ML} \Leftarrow i_{ML}, i_L \Leftarrow i_L$
    else
      $i_H \Leftarrow i_{MH}, i_{MH} \Leftarrow i_{ML}, i_{ML} \Leftarrow \lfloor (i_L + i_{ML})/2 \rfloor, i_L \Leftarrow i_L$
    end if
  end if
  $c_L \Leftarrow c(F_{i_L}), c_{ML} \Leftarrow c(F_{i_{ML}}), c_{MH} \Leftarrow c(F_{i_{MH}}), c_H \Leftarrow c(F_{i_H})$
  $(i_{Best}, c_{Best}) \Leftarrow \min(c_L, c_{ML}, c_{MH}, c_H)$
end while
return i_{Best} and c_{Best}

With the aforementioned modifications to increase efficiency, the IUBB pseudocode is displayed below as Algorithm 2.

Compared to the original UBB algorithm, the proposed IUBB algorithm uses the U-assumption efficiently by first using a different search structure which focuses on an optimal chain $\mathcal{F}^*$ in the search space at each step of the algorithm. Then using the U-assumption for not only pruning the search space but also for finding the minimum element F^* of each chain (`Bisection` module). This improved and efficient use of the U-curve assumption enables the proposed algorithm to outperform UBB in most problems. Fig. 5 illustrates the UBB algorithm, using the lattice in Fig. 2. In the next section, feature selection experiments are conducted to assess the performance of both algorithms.

4. Feature selection experiments

In this section, the performances of the proposed IUBB algorithm and the original UBB algorithm are compared in various fea-

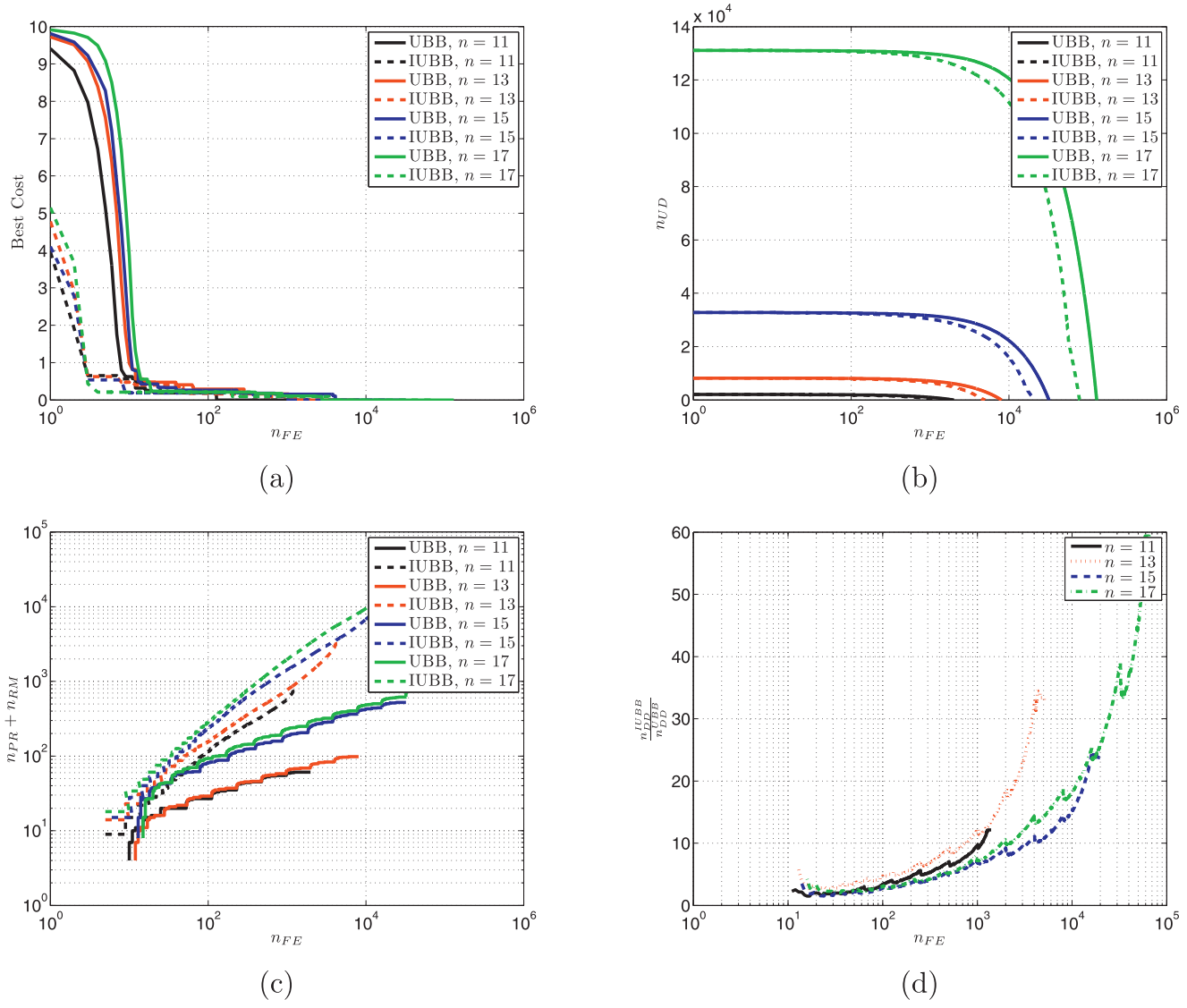

Fig. 7. Performance of the IUBB and UBB algorithms for $\alpha = 0.75$ and different values of n. All quantities are plotted against the number of feature sets visited n_{FE}. (a) Best cost found by each algorithm. (b) Number of feature sets not visited, pruned or removed n_{UD}. (c) Number of feature sets pruned or removed n_{DD}. (d) Ratio of n_{DD} for IUBB over n_{DD} for UBB.

Algorithm 2 IUBB Algorithm

Initialize Boolean lattice $\mathcal{L}$ of degree n.
$N_r \Leftarrow$ Number of elements is the search space that are not visited.
$c^{opt} \Leftarrow \infty$
while $N_r \geq 0$ **do**
  Find the optimal chain $\mathcal{F}^*$
  `Bisection`$(\mathcal{F}^*) \Rightarrow F^*$
  if $c(F^*) < c^{opt}$ **then**
    $c^{opt} \Leftarrow c(F^*)$
    $F^{opt} \Leftarrow F^*$
  end if
  `Prune`$(\mathcal{L}, \mathcal{F}^*, F^*) \Rightarrow (\mathcal{L}, N_r)$
end while
return F^{opt} and c^{opt}

ture selection experiments. The analysis is broken into two different parts. First, the algorithms are compared in a set of synthetic benchmark U-curve problems. The parameters in the synthetic problems allow us to study the effects the U-shape assumption can have on the underlying cost function and their impact on the behavior of the algorithms. Next, the robustness of the algorithms are studied under violation of the U-curve assumption, which would typically happen in wrapper feature selection [9] when the estimated classification error is used as the cost.

4.1. Synthetic benchmark

In this section, we report results of numerical experiments using synthetic data.

4.1.1. Cost function

The model for the cost function is given by:

$$c(F \mid F_0, \mathbf{W}, c_{\max}) = c_{\max}\left[1 - \exp\left(-\frac{1}{2}(F - F_0)^T \mathbf{W}(F - F_0)\right)\right] \quad (4)$$

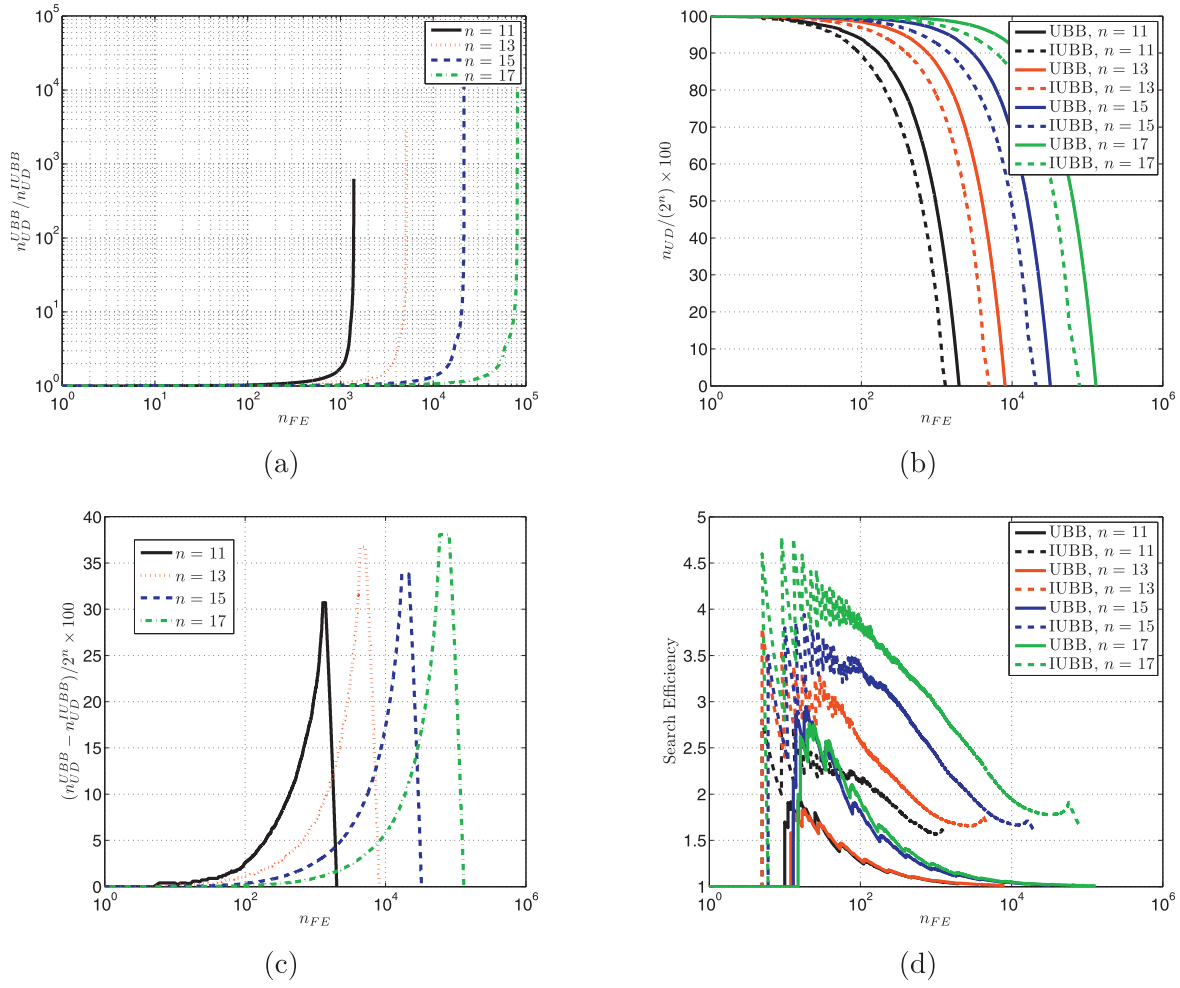

Fig. 8. Performance of the IUBB and UBB algorithms for $\alpha = 0.75$ and different values of n. All quantities are plotted against the number of feature sets visited n_{FE}. (a) Ratio of n_{UD} for IUBB over n_{UD} for UBB. (b) Number of feature sets not visited, pruned or removed n_{UD} as a percentage of the entire search space. (c) Difference between n_{UD} for UBB and n_{UD} for IUBB as a percentage of the entire search space. (d) Search efficiency.

where

$$\begin{aligned} &F \in \{0,1\}^n : \text{Feature set (binary string representation)} \\ &F_0 \in \{0,1\}^n : \text{Global minimum of the cost function} \\ &\mathbf{W} \in \mathbb{R}^{(n\times n)} : \text{Positive-definite weighting matrix (shaping matrix)} \\ &c_{\max} : \text{Cost scale, or ideal maximum value of cost} \end{aligned} \tag{5}$$

The density of 1's in the global minimum,

$$\alpha = \frac{1}{n}\sum_{i=1}^{n} F_0(i), \tag{6}$$

with $0 \le \alpha \le 1$, indicates how late peaking occurs. Note that $n\alpha$ is the number of features in the optimal feature set. Fig. 6 displays a sample U-shaped cost function for $n = 5$ and $\alpha = 0.6$. Fig. 6(a) and (b) display the U-shaped chains of the search space in 2D and 3D, respectively. We can see that the parameter α of the cost function model controls the peaking latency. The proposed benchmark function set and its controlling parameters enable us to compare the performance of the algorithms over different classes of problems. This benchmark can be used in future work to study the peaking phenomenon and assess the capability of any feature selection algorithm that incorporates the U-assumption and deals with U-shaped cost functions.

4.1.2. Performance metrics

Let n_{FE} be the number of function evaluations, i.e., the number of feature sets visited and evaluated by the algorithm, n_{PR} be the number of feature sets pruned by applying the U-curve assumption on the search, and n_{RM} be the number of feature sets removed from the search for reasons other than pruning, e.g., removal of feature sets from a chain using bisection to find the minimum element.

The algorithms will be compared using the following metrics:

- Cost of best feature set found by the algorithm.
- Number of feature sets pruned or removed: $n_{DD} = n_{PR} + n_{RM}$.
- Number of feature sets not visited, pruned or removed: $n_{UD} = 2^n - n_{FE} - n_{PR} - n_{RM}$.
- Number of function evaluations required to find the optimal feature set.
- Search Efficiency (SE):

$$SE = \frac{n_{FE} + n_{DD}}{n_{FE}}. \tag{7}$$

The SE measures how efficient the algorithm is in using the U-curve assumption to discard undesired solutions from the search space. The minimum value $SE = 1$ is achieved by an exhaustive search, when $n_{DD} = 0$. A small value of SE will show that the algorithm uses the assumption inefficiently.

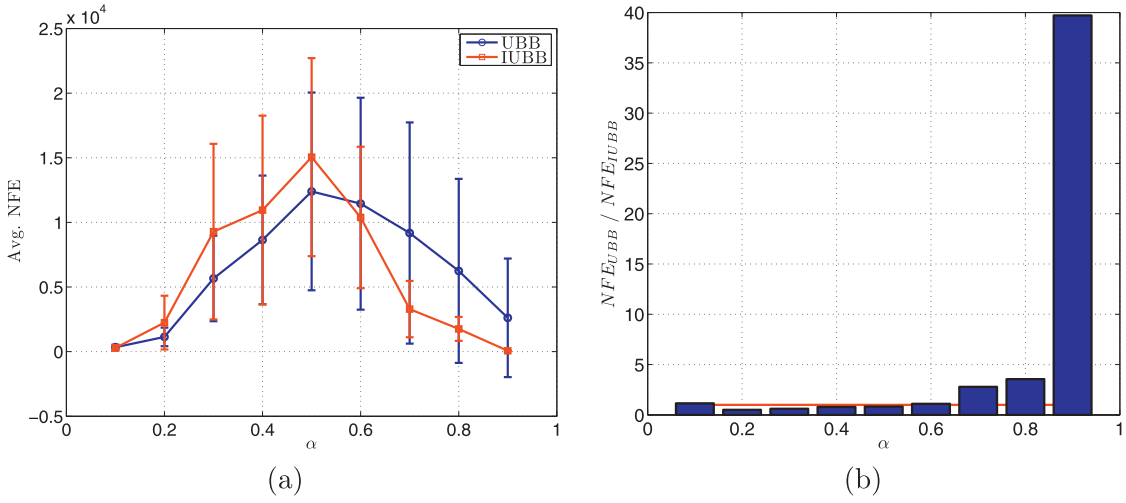

Fig. 9. (a) Plot of the average number of feature set evaluations required by each algorithm to find the optimal feature set, for $n = 15$. (b) Barplot of the average gain in efficiency displayed by IUBB over UBB in terms of feature set evaluations required to find the global best feature set.

4.1.3. Results

Figs. 7 and 8 display plots of several of the metrics discussed in the previous section vs. the number of feature sets visited, for $\alpha = 0.75$ and varying n. One can observe that IUBB presents a large search gain for small n_{FE}, which is significant, as this represents the case where computational resources are limited.

Fig. 9 displays the average number of function evaluations needed to find the best feature set, as a function of α, for $n = 15$. Varying α allows us to observe the relative performance of the algorithms with respect to early and late peaking. We can see in Fig. 9(a) that the increase of α from 0 to 0.5 increases the number of feature set evaluations required by each algorithm. This behavior is expected, as the increase of α delays success in finding the best feature sets. Both algorithms have the worst performance when α is about 0.5. However, increasing α in the interval [0.5, 1] improves the performance of the two algorithms. When α is close to 1, the optimal solution is in the first few selected chains and both algorithms perform well, but with a noticeable superiority of IUBB. Fig. 9 (b) shows that UBB might require about 40 times more feature set evaluations for a problem with late peaking.

Fig. 10 compares the two algorithms using the minimum cost found, search efficiency SE, and SE ratio plots as a function of α, for $n_{FE} = 5\%$ and $n_{FE} = 10\%$ of the search space, and $n = 15$. Limiting the total number of functions evaluations models the realistic scenario of limited computational resources. We can see that IUBB has a better performance in terms of the minimum cost found and search efficiency, over the entire range of α. We also see that the two algorithms display large search efficiency for small values of α and, as α increases, the search efficiency decreases quickly for both algorithms.

Fig. 11 allows us to examine the loss of performance typically displayed by Branch-and-Bound algorithms in the face of increasing dimensionality, by plotting the average minimum cost found (with error bars) as a function of the number of features, with $\alpha = 0.85$. We can see that IUBB outperforms UBB, but that the performance of both algorithms degrade as dimensionality increases.

Fig. 12 displays the number of function calls and the percentage of the search space evaluated by each algorithm in order to find the best feature set, as a function of the dimensionality, for $\alpha = 0.85$. One can observe that, on average, IUBB makes a smaller number of function evaluations and explores a smaller percentage of the search space than UBB in finding the optimal feature set. The difference tends to increase with larger dimensionality. The smaller error bars displayed by IUBB indicate good stability with respect to random changes in the structure of the problem.

4.2. Violation of the U-curve assumption

In order to study the robustness of the algorithms, in this section we allow violation of the U-curve assumption; for instance, this would be the case in wrapper feature selection when classifier error estimators are used to obtain the cost function. This is accomplished by adding a sinusoidal disturbance to the cost model to allow violation of the U-assumption:

$$c(F \mid F_0, \mathbf{W}, c_{\max}) = c_{\max}\left[1 - \exp\left(-\frac{1}{2}(F - F_0)^T \mathbf{W}(F - F_0)\right) + A\cos(2\pi f \beta(F))\right], \tag{8}$$

where

$$\begin{aligned} &\beta(F) = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} F(i), \\ &A : \text{Amplitude of the sinusoidal disturbance}, \\ &f : \text{Frequency of the sinusoidal disturbance}; \end{aligned} \tag{9}$$

all the other parameters are as before. The value of f controls the number (frequency) of local minima and A controls the depth of the local minima. If A is set to 0, then the U-curve assumption is not violated, whereas if $A > 0$, then the problem does not satisfy the U-curve assumption and each chain might have more than one local minimum. A robust Branch-and-Bound algorithm should be able to avoid most of these minima and display a small reduction in its ability to find the global minimum of each chain and pruning the search space, provided that A and f are not too large. In Fig. 13, we display the average cost of the best feature sets found as the value of A increases. As we see, for $A \leq 0.075$ and $f = 2$ the two algorithms are able to tolerate the violation of the U-curve assumption. However, as A becomes greater than 0.075, the performance of UBB degrades suddenly, while IUBB is robust. With $f = 3$, even a small value of deviation from U-assumption is enough for UBB to get stuck in local minima of the function.

5. Comparison with Generalized Sequential Forward Search

In order to assess the performance of the IUBB algorithm in comparison with classical feature selection algorithms, in this section IUBB is compared with Generalized Sequential Forward Search (GSFS). Many classical feature selection algorithms, including GSFS,

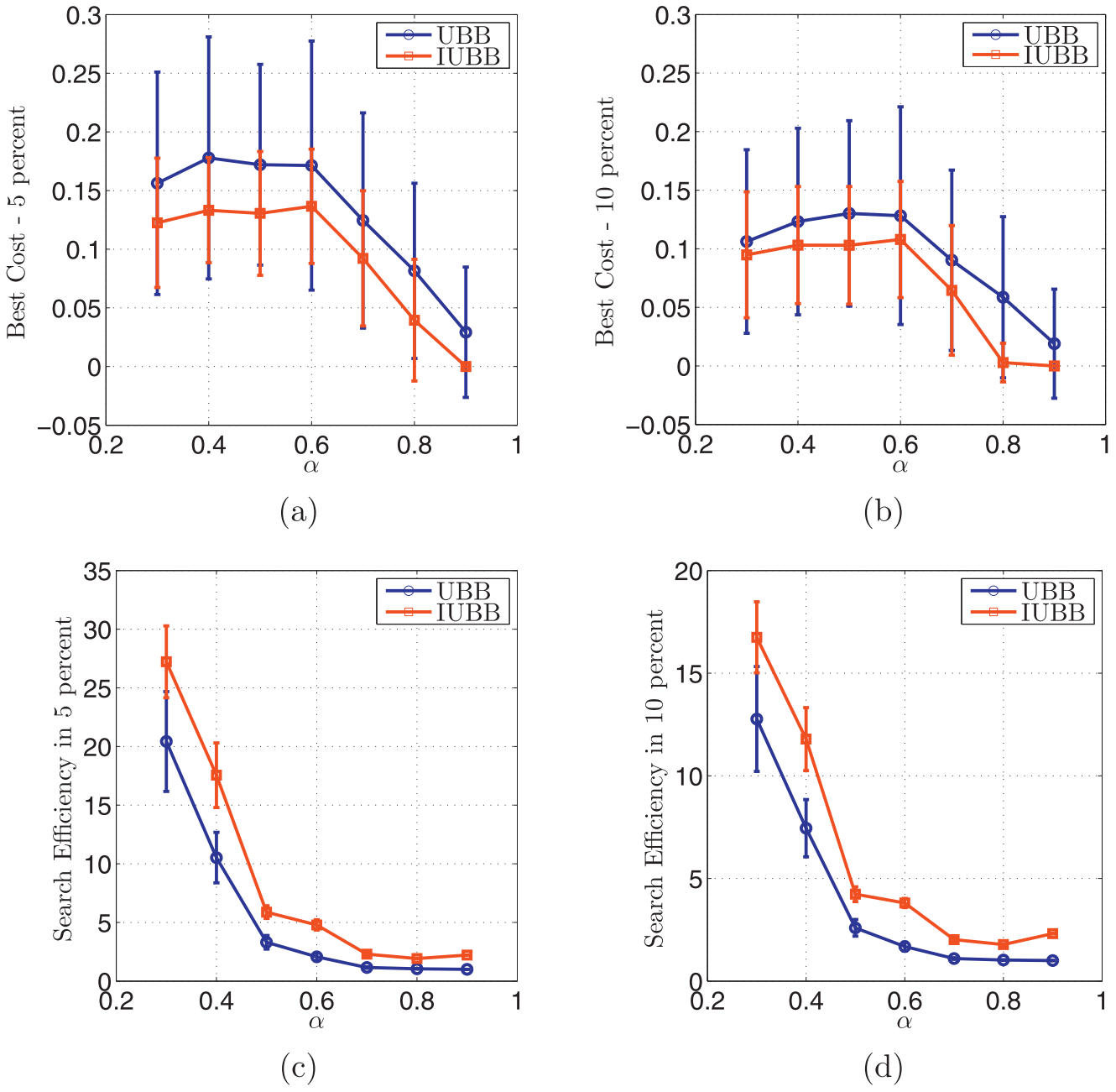

Fig. 10. Performance for fixed number of visited feature sets, $n_{FE} = 5\%$ and $n_{FE} = 10\%$ of the search space, and $n = 15$. All quantities are plotted against α. (a,b) Average minimum cost found. (c,d) Search efficiency.

do not search for the optimal features, but rather try to find a reasonable suboptimal solution in fewer function calls than optimal feature selection. In this case, the main criterion becomes the number of function calls rather than the quality of the solution or finding the optimal set of features. On the other hand. IUBB ensures that for U-curved functions it not only uses a reasonable number of function evaluations but also the set of features found by the algorithm is the global best solution. Hence, comparing the two algorithms based on the quality of the results will not be fair as IUBB will outperform the suboptimal search methods in U-curve problems.

In order to compare the results of the two algorithms, first GSFS is used to find the best suboptimal feature set, and this solution is passed to IUBB as the stopping criterion. Then the number of function evaluations needed by IUBB to reach the same solution as GSFS is used to compare the two algorithms. The result is displayed in Fig. 14 for different values for n and α. As we see, in all the cases IUBB outperforms GSFS. In addition, for larger values of α and the delay in peaking, IUBB reaches the same solution as GSFS with 4 to 10 times fewer function evaluations.

6. Application to the design of imaging W-operators

In this section, we employ the IUBB algorithm in the design of W-operators, a class of morphological imaging operators that are translation-invariant and locally defined inside a window [1]. The design of a W-operator involves a feature selection procedure, which requires the choice of a suitable cost function. Previous results have shown that an effective cost function for W-operator window design is the minimization of the mean conditional entropy [10].

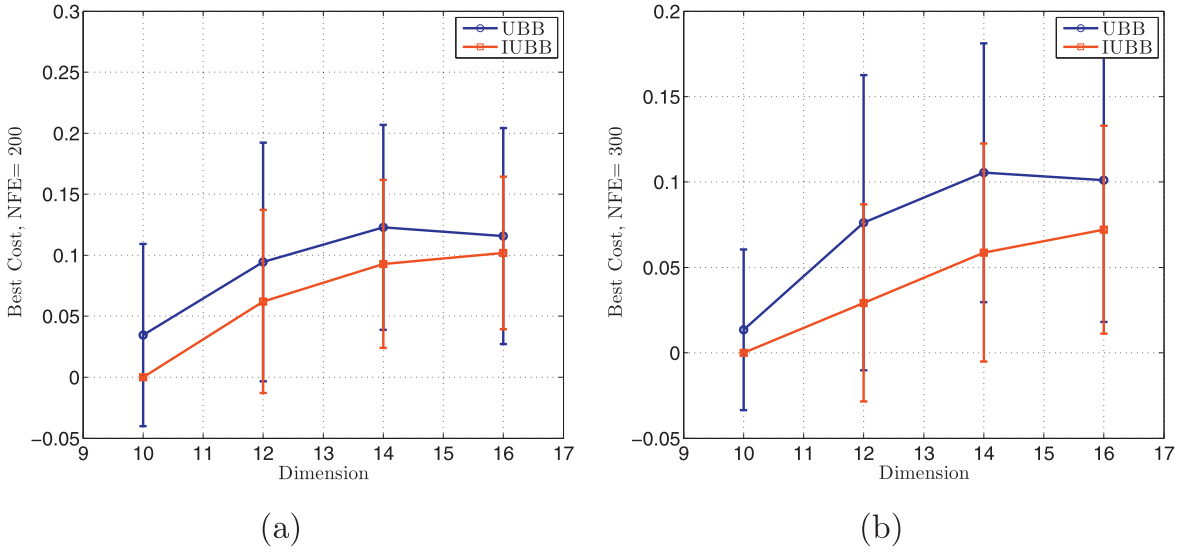

Fig. 11. Average minimum cost vs. dimension for fixed number of function evaluations, (a) $n_{FE} = 200$ and (b) $n_{FE} = 300$.

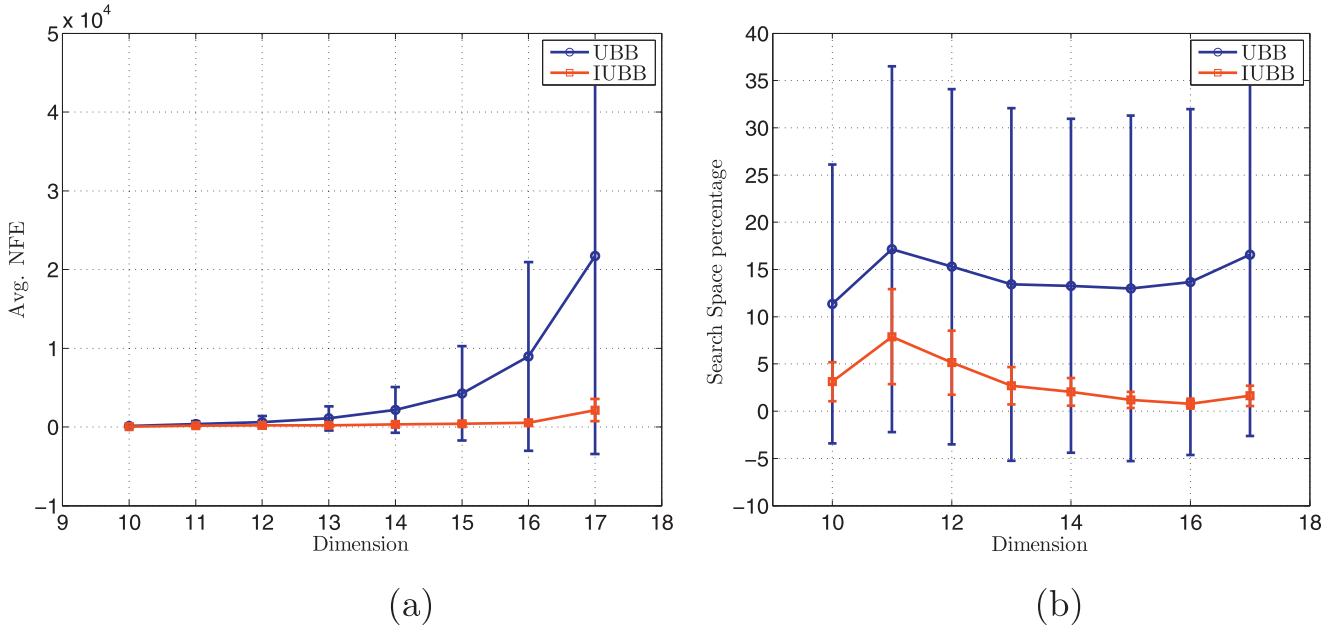

Fig. 12. Number of function calls and the percentage of the search space evaluated by each algorithm in order to find the best feature set, as a function of the dimensionality, for $\alpha = 0.85$. (a) Number of function evaluations used by each algorithm. (b) Percentage of the search space evaluated by each algorithm.

The cost function to be employed is the (estimated) mean conditional entropy [2]. The conditional entropy of a discrete random variable Y taking values in $\mathcal{Y}$ given a realization of another discrete random variable $\mathbf{X}$ taking values in $\mathcal{X}$ is defined by

$$H(Y|\mathbf{X}{=}\mathbf{x}) = -\sum_{y\in\mathcal{Y}} P(Y=y|\mathbf{X}{=}\mathbf{x}) \log P(Y=y|\mathbf{X}=\mathbf{x}), \quad \text{for } \mathbf{x}\in\mathcal{X}, \tag{10}$$

so that the mean conditional entropy (MCE) of Y given $\mathbf{X}$ is expressed by

$$E[H(Y|\mathbf{X})] = \sum_{\mathbf{x}\in\mathcal{X}} H(Y|\mathbf{X}=\mathbf{x})P(\mathbf{X}=\mathbf{x}) \tag{11}$$

$$= -\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{X}=\mathbf{x}) \sum_{y\in\mathcal{Y}} P(Y=y|\mathbf{X}=\mathbf{x}) \log P(Y=y|\mathbf{X}=\mathbf{x}) \tag{12}$$

$$= -\sum_{\mathbf{x}\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(Y=y,\mathbf{X}=\mathbf{x}) \log \frac{P(Y=y,\mathbf{X}=\mathbf{x})}{P(\mathbf{X}=\mathbf{x})}, \tag{13}$$

where $0 \log 0$ is to be interpreted as zero. Since $P(Y=y,\mathbf{X}=\mathbf{x}) \le P(\mathbf{X}=\mathbf{x})$, one has $E[H(Y|\mathbf{X})] \ge 0$. It can be shown that the minimum value of zero is obtained if and only if Y is completely determined by $\mathbf{X}$.

Given t independent and identically distributed (i.i.d.) pairs $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_t, Y_t)$ distributed as $(\mathbf{X}, Y)$, plugging in the standard sample-based estimators for the probabilities in (13) leads to an estimator $\hat{E}[H(Y|\mathbf{X})]$ for the MCE of Y given $\mathbf{X}$. However, since t is usually small, the estimation error is typically large. To minimize this, we penalize pairs that are observed only once by considering such occurrences as following a uniform distribution, which leads to maximum entropy. Given that the probability of a pair $(\mathbf{X}_i, Y_i)$ occurring only once is $1/t$ and the fact that the number of such pairs is m, the total penalty contribution to the MCE estimator is

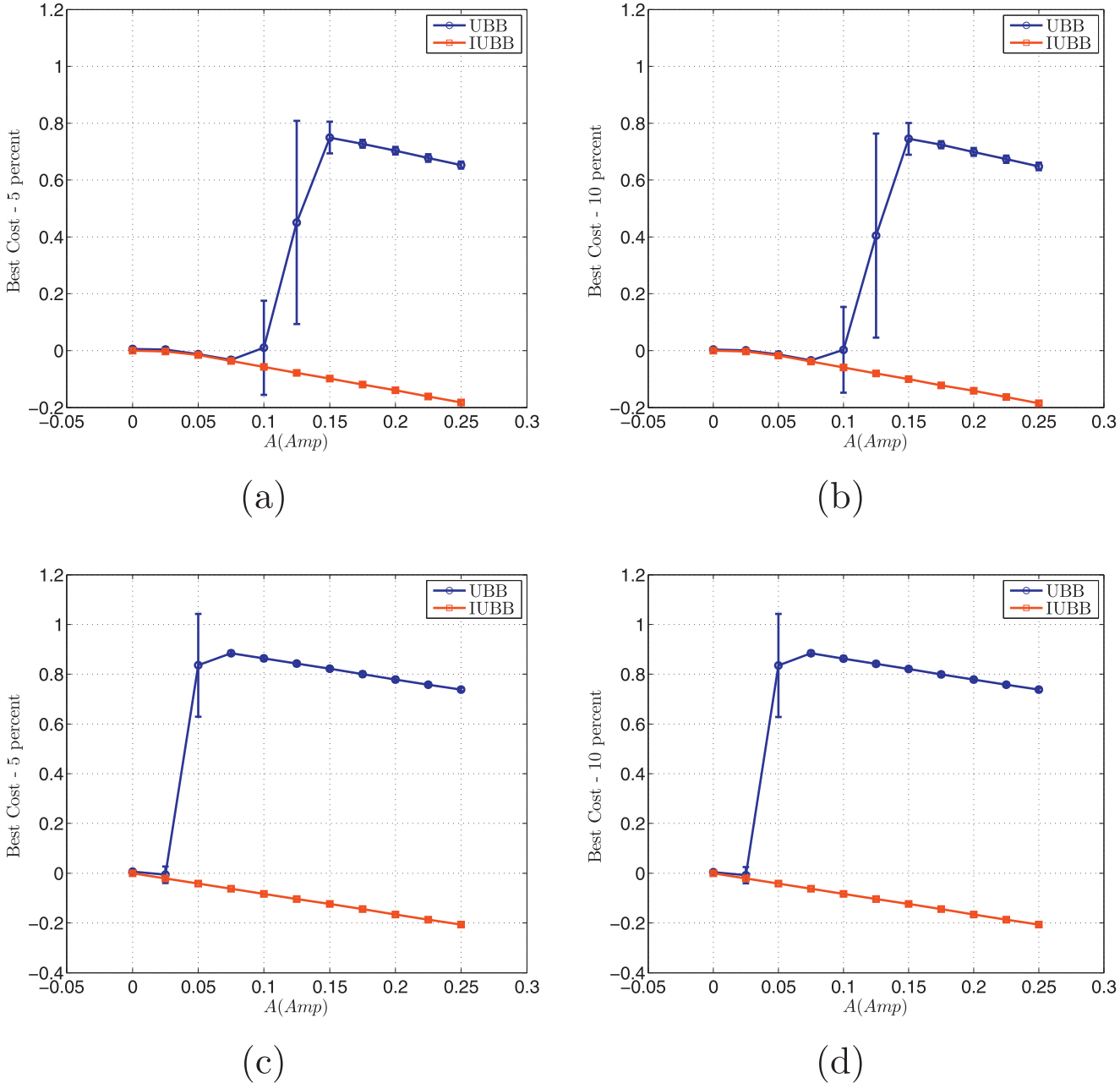

Fig. 13. Average minimum cost for fixed number of function evaluations, $n_{FE} = 5\%$ and $n_{FE} = 10\%$ of the search space, and $n = 15$. All quantities are plotted against A. Top row: $\alpha = 0.75,\ f = 2$. Bottom row: $\alpha = 0.85,\ f = 3$.

m/t. Therefore, the final MCE estimator is given by

$$\hat{E}[H(Y|\mathbf{X})] = \frac{m}{t} - \sum_{\mathbf{x}\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \hat{P}(Y=y,\mathbf{X}=\mathbf{x}) \times \log \frac{\hat{P}(Y=y,\mathbf{X}=\mathbf{x})}{\hat{P}(\mathbf{X}=\mathbf{x})} I_{\hat{P}(Y=y,\mathbf{X}=\mathbf{x}) \geq \frac{2}{t}}. \quad (14)$$

Now, consider an ideal binary image I and a corresponding noisy binary image I_N. Let W be a window centered on a pixel $x \in I$, and let $\mathbf{X}$ be a set of pixels in W. A W-operator ψ: $\mathbf{X} \mapsto \{0, 1\}$ tries to predict the value $Y = I(x) \in \{0, 1\}$ using the values taken by I_N on $\mathbf{X}$. The W-operator is assumed to be translation-invariant, that is, the same set of pixels $\mathbf{X}$ (relative to the translated window) is used for all $x \in I$. The design of a W-operator, for a given window W, involves determining the appropriate subset of pixels $\mathbf{X}$ and the mapping ψ. Now assume that a sample realization of an image pair (I, I_N) is available. By assuming stationarity, one can slide the window W across the images and obtain t approximately i.i.d. realizations of the pair $(\mathbf{X}, Y)$, for a given subset of pixels $\mathbf{X}$, where t is the number of pixels in the image I (ignoring border effects), which allows one to compute the MCE estimator $\hat{E}[H(Y|\mathbf{X})]$ in (14), for the given $\mathbf{X}$. We propose to apply the IUBB algorithm to select the best $\mathbf{X}$ using the MCE estimator as the cost function — once the best $\mathbf{X}$ is selected, the value of $\psi(\mathbf{X})$ is determined by majority voting over the t realizations of $(\mathbf{X}, Y)$.

We conducted an experiment where the noisy image I_N was created by adding 30% salt-and-pepper noise to the original ideal image I — an example is shown in Fig. 15. Such pairs of images were screened considering windows of size ranging from 8 to 16 pixels, some of which are displayed in Fig. 16.

Performance was evaluated by means of the search efficiency and best cost criteria defined previously, averaged over a set of

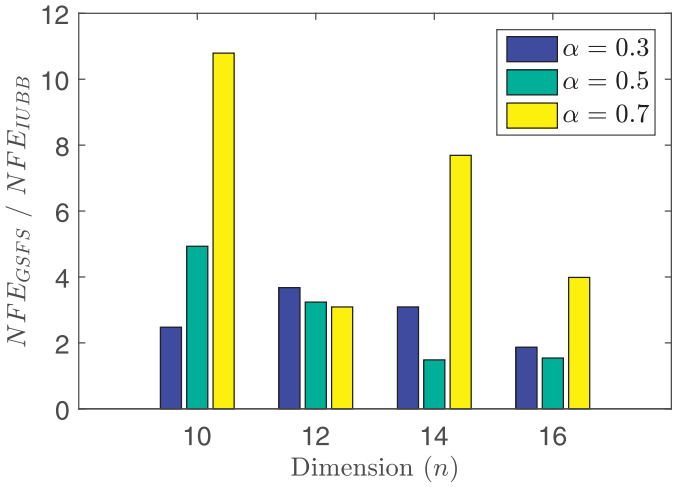

Fig. 14. Comparing IUBB and Generalized Sequential Forward Search (GSFS). For the GSFS the parameter r, which indicates the number of selected features at each step, is set to 2. Each algorithm is run 50 times and the ratio of the average number of function calls is plotted.

fifty pairs of ideal/noisy images. We also computed the results obtained by the original UBB algorithm, for comparison. Fig. 17 displays the values of the performance metrics as a function of the window size, which plays the role of dimensionality in this case. The left and right columns depict results for 5% and 10% of the search space explored, respectively. We can see that both the average search efficiency and the average minimum value of MCE found improve with an increasing window size — there will usually be a point at which increasing window size further will not improve performance and may indeed degrade it, for a fixed image size, but the window sizes employed in the experiment were small enough, compared to the image size, to avoid this peaking phenomenon. We can also see that IUBB outperforms UBB in each case.

Fig. 18 displays the result of applying the W-operator found by each algorithm to the noisy image shown in Fig. 15. Each algorithm is run with $n_{FE} = 2000$ and the best 16-pixel W-operator found by each algorithm is applied twice on the noisy image to enhance the image and reduce the noise. As we see the W-operator found by IUBB achieves a smaller noise level and cleaner image than the one found by UBB.

7. Application to classification feature selection

In this section, we report the results of a feature selection experiment in the classification of synthetic data generated from a Gaussian model, using the Mean Conditional Entropy (MCE) criterion, introduced in the previous section, as the cost function.

We generate synthetic data according to the model proposed in [6]. The individual features are divided into two different groups, *markers* and *non-markers* (i.e. "noise"). A Gaussian block model is used for the abundance of markers and non-markers, with the latter group being divided into two groups, high-variance and low-variance. There are altogether D_{gm} global markers. The class-conditional distributions for the markers are D_{gm}-dimension Gaussian: $N(\mu_0^{gm}, \Sigma_0^{gm})$ for class 0 and $N(\mu_1^{gm}, \Sigma_1^{gm})$ for class 1, where μ_0^{gm} and μ_1^{gm} are the mean vectors of class 0 and 1, respectively, and Σ_0^{gm} and Σ_1^{gm} are the covariance matrices. The means are set to $\mu_0^{gm} = (0, 0, \ldots, 0)$ and $\mu_1^{gm} = (1, 1, \ldots, 1)$, while a block-based structure is used to define the covariance matrices, whereby markers are divided into groups of D_m markers each. Markers from different groups are uncorrelated and markers of the same group possess the same correlation ρ between each other. More specifically, we define Σ_0^{gm} and Σ_1^{gm} as $\Sigma_0^{gm} = \sigma_0^2 \times \Sigma$ and $\Sigma_1^{gm} = \sigma_1^2 \times \Sigma$, with

$$\Sigma = \begin{bmatrix} R_\rho & 0 & \cdots & 0 \\ 0 & R_\rho & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_\rho \end{bmatrix}, \tag{15}$$

where R_ρ is a $D_m \times D_m$ matrix with 1's on the diagonal and ρ's elsewhere. On the other hand, the non-markers are features that provide no discriminating power between two classes. The non-marker features are uncorrelated and their distribution is a mixture of Gaussians, $N(0, \sigma_0^2)$ and $N(1, \sigma_1^2)$, where σ_0^2 and σ_1^2 are the same values used for the markers.

Using this data model and the estimated MCE as the cost function, the IUBB algorithm was used to find the best feature set. Results are also computed for the original UBB algorithm for comparison. The specific parameter values used in the simulation are displayed in Table 1.

Fig. 19 displays the results of the experiment. As we can see, increasing the dimension decreases the search efficiency of UBB, while IUBB is able to exploit the problem assumptions and prunes the search space very fast.

(a) (b)

Fig. 15. An example of a pair of images that was used in the design of imaging W-operators. (a) Ideal image I. (b) Noisy image I_N.

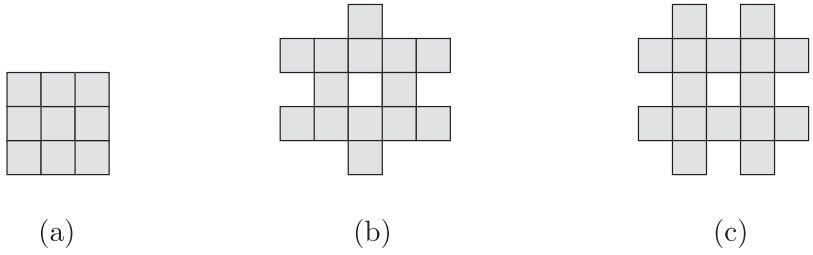

Fig. 16. A few of the *W*-operator windows used in the experiment. (a) 9-pixel window. (b) 14-pixel window. (c) 16-pixel window.

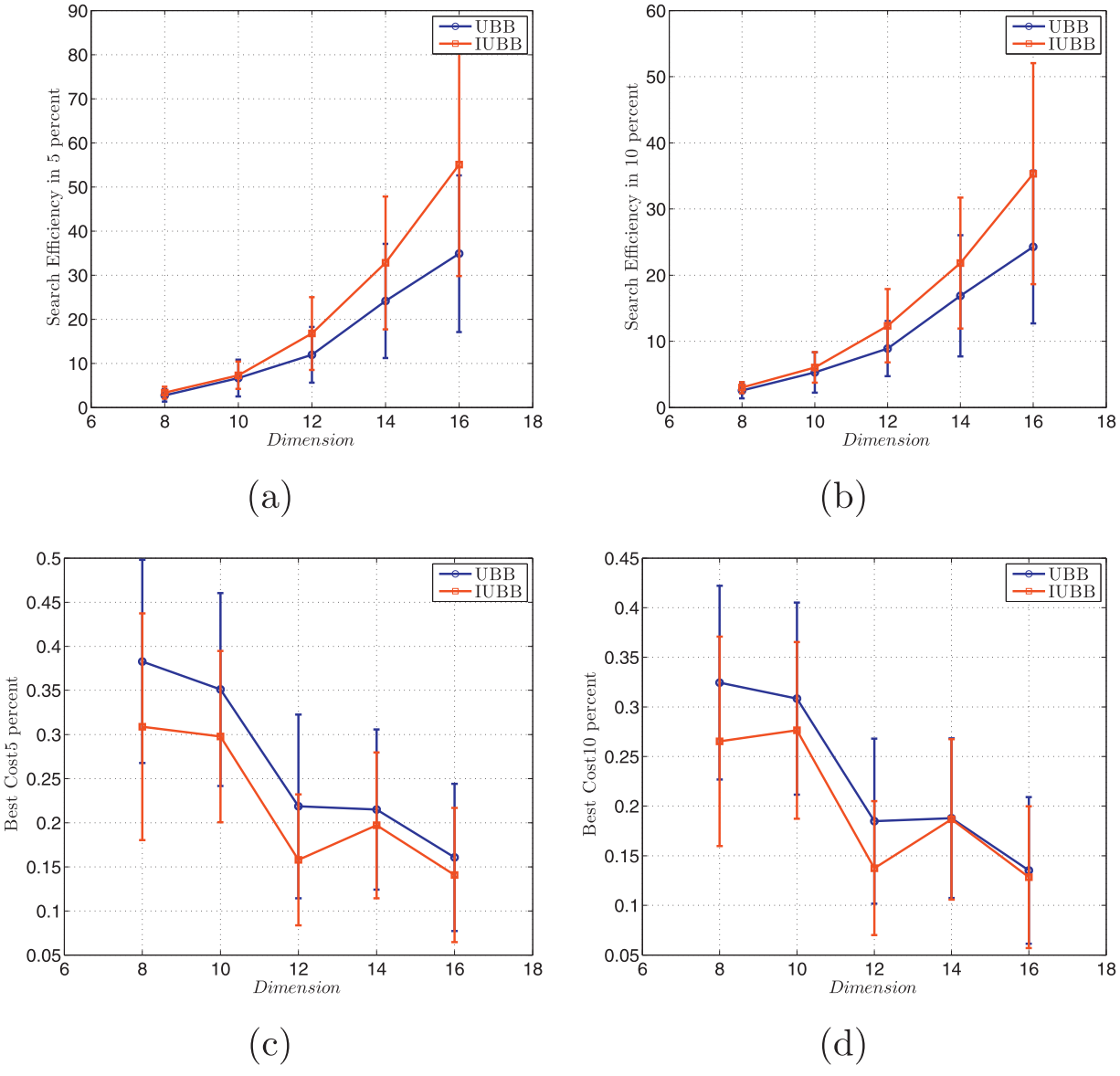

Fig. 17. Performance of *W*-operator design. The left and right columns depict results for 5% and 10% of the search space explored, respectively. All quantities plotted as a function of the window size.

8. Conclusion

In this paper, we proposed a new and improved Branch and Bound algorithm for the U-curve optimization problem, assessed its performance by means of simulated optimization problems, and demonstrated its application in the design of imaging *W*-operators and in feature selection for classification. The proposed algorithm (IUBB) improves on a previous algorithm (UBB), which requires fewer number of function calls compared to exhaustive search. However, the experiments in the present paper show that the UBB algorithm does not use the U-curve assumptions efficiently. The IUBB algorithm uses the U-assumption efficiently by reorganizing the tree structure in an optimal fashion and using bisection to find the minimum along the chains. Different indices were used to evaluate the performance of the proposed algorithm. The number of function calls needed to find the best feature set, the minimum

Fig. 18. Results produced by the 16-pixel W-operators found by each algorithm applied to the noisy image shown in Fig. 15. (a) The result for UBB (b) The result for IUBB.

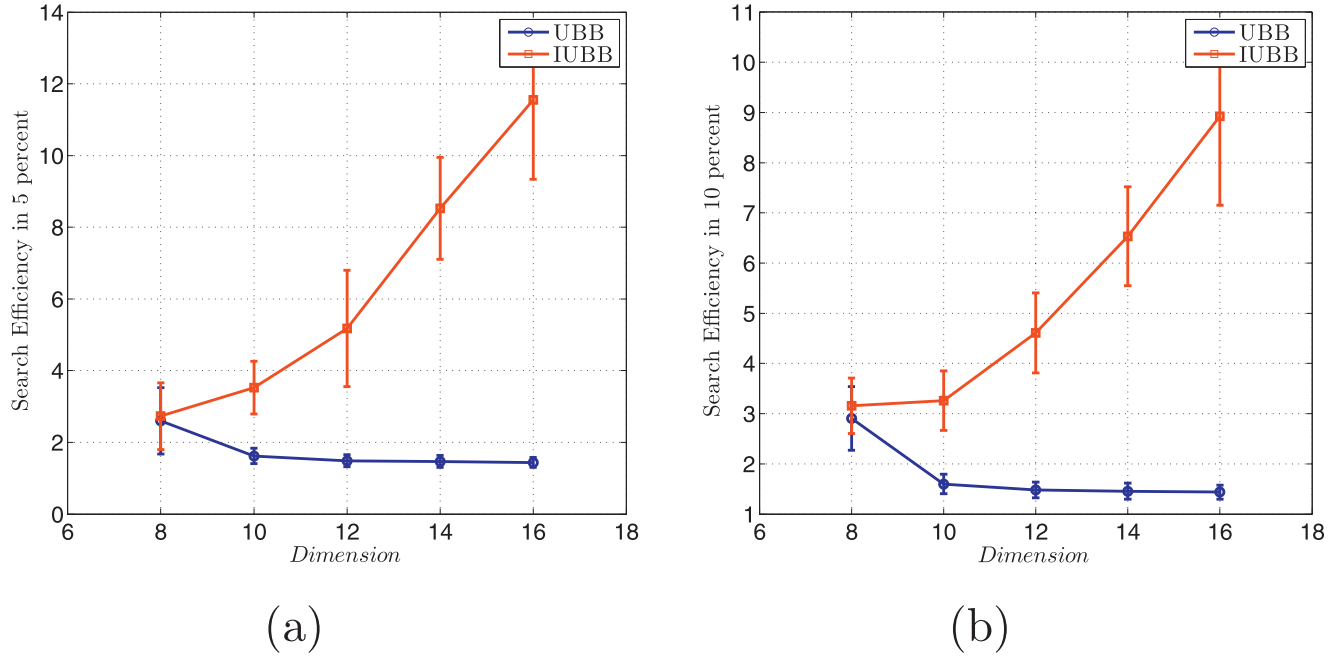

Fig. 19. Search efficiency vs. dimension, (a) *NFE* = 5% of search space, and (b) *NFE* = 10% of search space.

Table 1
Summary of parameters.

Parameter		Value
No. of sample size	nTr	100
Block size	D_m	2
Correlation	ρ	0.5
Variances	$\sigma_0^2 = \sigma_1^2$	0.3^2

cost vs. number of function calls, and the search efficiency were three major indices used to assess the performance. The results showed that the proposed IUBB algorithm requires a qualitatively smaller number of function evaluations to find the optimal solution than UBB. In addition, for a fixed budget, i.e., with the same number of function evaluations, IUBB generally reaches a feature set with lower cost value compared to UBB.

Acknowledgements

Reis and Barrera were supported by grants #2013/07467-1 and #2015/01587-0, São Paulo Research Foundation (FAPESP).

References

[1] J. Barrera, G.P. Salas, Set operations on closed intervals and their applications to the automatic programming of morphological machines, J. Electron. Imaging 5 (3) (1996) 335–352.
[2] T. Cover, Elements of Information Theory, 2nd, Wiley, New York, NY, 2006.
[3] T. Cover, J. van Campenhout, On the possible orderings in the measurement selection problem, IEEE Trans. Syst. Man Cybern. 7 (1977) 657–661.
[4] L. Devroye, L. Gyorfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, New York, 1996.
[5] A. Frank, D. Geiger, Z. Yakhini, A distance-based branch and bound feature selection algorithm, in: Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2002, pp. 241–248.
[6] J. Hua, W.D. Tembe, E.R. Dougherty, Performance of feature-selection methods in the classification of high-dimension data, Pattern Recognit. 42 (3) (2009) 409–424.
[7] G. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inf. Theory 14 (1) (1968) 55–63.
[8] A. Jain, D. Zongker, Feature selection: evaluation, application, and small sample performance, IEEE Trans. Pattern Anal. Mach. Intell. 19 (2) (1997) 153–158.
[9] R. Kohavi, G. John, Wrappers for feature subset selection, Artif. Intell. 97 (1–2) (1997) 273–324.
[10] D. Martins-Jr, R. Cesar-Jr, J. Barrera, W-operator window design by minimization of mean conditional entropy, Pattern Anal. Appl. 9 (2) (2006) 139–153.
[11] S. Nakariyakul, D.P. Casasent, Adaptive branch and bound algorithm for selecting optimal features, Pattern Recognit. Lett. 28 (12) (2007) 1415–1427.

[12] P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Trans. Comput. 100 (9) (1977) 917–922.
[13] M.S. Reis, Minimization of decomposable in U-shaped curves functions defined on poset chains – algorithms and applications, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil, 2012 Ph.D. thesis. (in Portuguese)
[14] M. Ris, J. Barrera, D.C. Martins Jr, U-curve: a Branch-and-Bound optimization algorithm for U-shaped cost functions on boolean lattices applied to the feature selection problem, Pattern Recognit. 43 (3) (2010) 557–568.
[15] C. Sima, E.R. Dougherty, The peaking phenomenon in the presence of feature-selection, Pattern Recognit. Lett. 29 (11) (2008) 1667–1674.
[16] P. Somol, P. Pudil, J. Kittler, Fast branch & bound algorithms for optimal feature selection, IEEE Trans. Pattern Anal. Mach. Intell. 26 (7) (2004) 900–912.
[17] B. Yu, B. Yuan, A more efficient branch and bound algorithm for feature selection, Pattern Recognit. 26 (6) (1993) 883–889.

Esmaeil Atashpaz-Gargari is an Assistant Professor at the Department of Computer Science and Information Systems, National University, San Diego, CA. He received his Ph.D. in Genomic Signal Processing form Department of Electrical and Computer Engineering at Texas A & M University, College Station, TX. His research interests include Artificial Intelligence, Pattern Recognition, Error Estimation and Proteomics.

Marcelo S. Reis is an Associate Investigator at Instituto Butantan, São Paulo, Brazil. He received his Ph.D. in Computer Science at Universidade de São Paulo, São Paulo, Brazil, and held a post-doctoral position at the same university. He also holds a postgraduate specialization in Bioinformatics from Universität zu Köln, Cologne, Germany, and a bachelor in Computer Science from Universidade Estadual de Campinas, Campinas, Brazil. His research interests include Mathematical Morphology, Combinatorial Optimization and their application into mathematical modeling and computational simulation of biological systems.

Ulisses M. Braga-Neto is an Associate Professor at the Department of Electrical and Computer Engineering and a member of the Center for Bioinformatics and Genomic Systems Engineering at Texas A&M University, College Station, TX. He holds a Ph.D. degree in Electrical and Computer Engineering from The Johns Hopkins University, Baltimore, MD and has held post-doctoral positions at the University of Texas M.D. Anderson Cancer Center, Houston, TX and at the Oswaldo Cruz Foundation, Recife, Brazil. His research interests include Pattern Recognition and Genomic Signal Processing. Dr. Braga-Neto is a Senior Member of the IEEE. He is author of the textbook "Error Estimation for Pattern Recognition" (IEEE-Wiley, 2015) and has received the NSF CAREER Award for his work in this area.

Junior Barrera received a B.Sc. in Electrical Engineering (Universidade de So Paulo-USP, Brazil), a M.Sc. in Applied Computing (Instituto Nacional de Pesquisas Espaciais - INPE, Brazil) and a Ph.D. in Electrical Engineering (Universidade de São Paulo- USP, Brazil). He is currently full professor at the Department of Computer Science of the Institute of Mathematics and Statistics of the Universidade de São Paulo, Brazil. His main research topics are machine learning, mathematical morphology and bioinformatics.

Edward R. Dougherty is a Professor in the Department of Electrical and Computer Engineering at Texas A&M University in College Station, TX, where he holds the Robert M. Kennedy '26 Chair in Electrical Engineering and is Director of the Genomic Signal Processing Laboratory. He is also Scientific Director of the Center for Bioinformatics and Genomic Systems Engineering at Texas A&M University. He holds a Ph.D. in mathematics from Rutgers University and an M.S. in Computer Science from Stevens Institute of Technology, and has been awarded the Doctor Honoris Causa by the Tampere University of Technology in Finland. He is a fellow of both IEEE and SPIE, has received the SPIE President's Award, and served as the editor of the SPIE/IS&T Journal of Electronic Imaging. At Texas A&M University he has received the Association of Former Students Distinguished Achievement Award in Research, been named Fellow of the Texas Engineering Experiment Station, and named Halliburton Professor of the Dwight Look College of Engineering. Prof. Dougherty is author of 17 books, editor of 5 others, and author of more than 300 journal papers.