Fast Branch & Bound Algorithms for Optimal Feature Selection

Petr Somol, Pavel Pudil, *Member*, *IEEE*, and Josef Kittler, *Member*, *IEEE*

Abstract—A novel search principle for optimal feature subset selection using the Branch & Bound method is introduced. Thanks to a simple mechanism for predicting criterion values, a considerable amount of time can be saved by avoiding many slow criterion evaluations. We propose two implementations of the proposed prediction mechanism that are suitable for use with nonrecursive and recursive criterion forms, respectively. Both algorithms find the optimum usually several times faster than any other known Branch & Bound algorithm. As the algorithm computational efficiency is crucial, due to the exponential nature of the search problem, we also investigate other factors that affect the search performance of all Branch & Bound algorithms. Using a set of synthetic criteria, we show that the speed of the Branch & Bound algorithms strongly depends on the diversity among features, feature stability with respect to different subsets, and criterion function dependence on feature set size. We identify the scenarios where the search is accelerated the most dramatically (finish in linear time), as well as the worst conditions. We verify our conclusions experimentally on three real data sets using traditional probabilistic distance criteria.

Index Terms—Subset search, feature selection, search tree, optimum search, subset selection, dimensionality reduction, artificial intelligence.

1 Introduction

The problem of optimal feature selection (or, more generally, of subset selection) is very difficult, primarily because of its computational complexity. The optimization space of all the subsets of a given cardinality that have to be evaluated to find the optimal set of features among an initial, larger set of measurements is of combinatorial complexity and its exhaustive search is impractical even for problems of relatively small size. Over the years, this challenge has motivated a considerable research effort aiming at speeding up the search process. This has been achieved to a certain extent either by relaxing the rigor of optimality a solution to the feature selection problem has to satisfy [1], [2], [3], [9], [10], [14] or by introducing heuristic measures which can help to identify parts of the search space that can be left unexplored without any danger of missing the optimal solution [3], [5].

Among the search methods that strive to reduce the computational burden without the loss of optimality, the *Branch & Bound* (B & B) algorithm and its ancestors have been receiving the most attention. While the B & B algorithm idea is well-known and considered one of the essential tools in artificial intelligence [13], [15], [16], [18], [21], [22], [25], it was first adopted for the purpose of feature selection by Narendra and Fukunaga [20]. In this context, it has been later studied in more detail and extended [5], [7], [14], [28]. This family of algorithms gains computational cost savings by the application of the set inclusion monotonicity property that the feature selection criterion function employed by these algorithms must satisfy. Accordingly, given a collection of features and the corresponding value of a feature set criterion function satisfying the monotonicity property, the function value will increase for any arbitrary superset of this collection. In other words, by augmenting a feature set by additional measurements, the criterion function will grow monotonically.

The application of the set inclusion monotonicity property allows the user to create short cuts in the search tree representing the feature set optimization process. In turn, this reduces the number of nodes and branches of the search tree that have to be explored to find the optimal solution and, hence, the enhanced computational efficiency. Although, in the worst case, the B & B algorithm is still of exponential complexity, in practice, this simple heuristics can result in substantial performance gains.

Considerable effort has been invested into acceleration of the B & B algorithm over the years. Many modified versions of the algorithm have been defined, gaining speed or other advantages for different specific problems [4], [12], [17], [18], usually by relaxing the generality or optimality of the B & B concept. Parallelized B & B algorithms have been defined and studied [6], [8], [26], [27]. Concurrently, the optimality of the feature selection process was put into a broader context in [11] by introducing the notion of Filter and Wrapper-based search processes. Our aim, in contrast, was to improve the core B & B concept itself, i.e., to define a new way of B & B search tree construction, which would result in substantial computational time savings without any impact on the applicability of the algorithm and without raising any other conditions than those defined in the original B & B algorithm [20].

- *P. Somol is with the Department of Pattern Recognition, Institute of Information Theory and Automation of the Academy of Sciences, Pod vodárenskou vezí 4, Prague 8, 182 08, Czech Republic. E-mail: somol@utia.cas.cz.*
- *P. Pudil is with the Faculty of Management, Prague University of Economics, Jarosovská 1117/II, Jindrichuv Hradec, 377 01, Czech Republic. E-mail: pudil@fm.vse.cz.*
- *J. Kittler is with the Centre for Vision, Speech, and Signal Processing, University of Surrey, Guildford GU2 7XH, UK. E-mail: j.kittler@eim.surrey.ac.uk.*

Manuscript received 21 Jan. 2003; revised 6 Oct. 2003; accepted 16 Dec. 2003. Recommended for acceptance by M. Pietikainen.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 118169.

TABLE 1
Nomenclature and Main Characteristics of Considered Algorithms

Branch & Bound		*In-level node ordering*	*Further computational savings*
Basic	(BBB)	NO	NO
Improved	(IBB)	computed	NO
Partial Prediction	(BBPP)	predicted	NO
Fast	(FBB)	predicted	by prediction

Thus, in this paper, we propose an additional heuristic mechanism to facilitate a further speed up of the B & B algorithm. The idea is motivated by the observation that practically all the computational time during optimization is devoted to evaluating the objective function at different nodes of the search tree, in other words, for different feature sets. If some of these function values could be predicted, we would avoid lengthy function evaluation and achieve additional economies in the feature set search. We propose a method for computing objective function value predictions based on the statistics of the effect of discarding individual measurements gathered from previously evaluated feature sets. These predictions are then exploited in two new algorithms. In order to ensure that the optimization procedure remains optimal, no search tree cut-off is based on a predicted value. Instead, for any hypothesized cut-off, the true function value is computed and the decision made on that basis. We show that this strategy accelerates the search process significantly.

The paper is organized as follows: In the following section, we introduce the necessary mathematical preliminaries and provide a brief overview of the "Basic" B & B algorithm (free of all additional heuristics) and a more detailed description of its "Improved" version [3], [5]. In Section 3, the drawbacks of these search strategies are identified. In Section 4, we introduce the idea of criterion value prediction and present a new algorithm which makes use of this mechanism for search tree expansion node ordering. In Section 5, we then extend the prediction concept to gain computational savings by applying it to multiple levels of the search tree during its traversal. The properties of the proposed schemes are discussed in Section 6. The results of experiments designed to demonstrate the effectiveness of the advocated algorithms are presented in Section 7. The paper is drawn to a conclusion in Section 8. For an overview of the algorithms to be discussed, see Table 1.

2 Preliminaries

Consider a problem of selecting d features from an initial set of D measurements using objective function J as a criterion of effectiveness. The Branch & Bound approach aims to solve this search problem by making use of the monotonicity property of certain feature selection criterion function.

Let $\bar{\chi}_j$ be the set of features obtained by removing j features $y_1, y_2, \cdots, y_j$ from the set Y of all D features, i.e.,

$$\bar{\chi}_j = Y \setminus \{y_1, y_2, \cdots, y_j\}. \tag{1}$$

The *monotonicity condition* assumes that, for feature subsets $\bar{\chi}_1, \bar{\chi}_2, \cdots, \bar{\chi}_j$, where

$$\bar{\chi}_1 \supset \bar{\chi}_2 \supset \cdots \supset \bar{\chi}_j,$$

the criterion function J fulfills

$$J(\bar{\chi}_1) \geq J(\bar{\chi}_2) \geq \cdots \geq J(\bar{\chi}_j). \tag{2}$$

The monotonicity property helps to identify parts of the search space which cannot possibly contain the optimal solution to the feature selection problem.

Before introducing the new algorithms in Sections 4 and 5, let us summarize the B & B principle briefly. The algorithm constructs a search tree where the root represents the set of all D features and leaves represent target subsets of d features. While traversing the tree down to leaves the algorithm successively removes single features from the current set of "candidates" ($\bar{\chi}_k$ in the kth level). The algorithm keeps the information about both the currently best subset $\mathcal{X}$ and the criterion value X^* it yields (we denote this value the *bound*). Anytime the criterion value in some internal node is found to be lower than the current *bound*, due to the condition (2), the whole subtree may be cut off and many computations may be omitted. The course of the B & B algorithm is illustrated in Fig. 1, where an example problem solution is shown for $d = 2$ and $D = 5$. The dashed arrows illustrate the way of tracking the search tree. For details, see [3], [5], [20].

Several improvements of the core B & B scheme are known. For each B & B search tree, a "minimum solution tree" [28] can be obtained by excluding nonbranching internal nodes, i.e., by shortening paths.

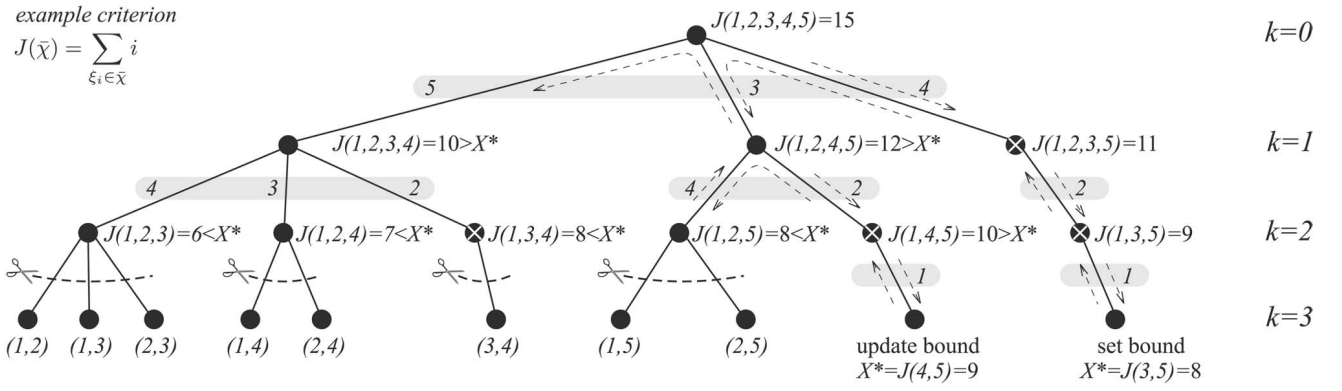


Fig. 1. Example of "branch & bound" problem solution, where $d = 2$ features are to be selected from the set of $D = 5$ features to maximize an illustrative synthetic criterion.

Remark. For the sake of simplicity, we do not reflect this simple acceleration technique in the algorithm descriptions, but we implemented it in all the algorithms selected for experimentation.

We base our investigation on the "Improved" B & B algorithm [5] which we shall refer to as IBB. Let the *criterion value decrease* be the difference between the current criterion value and the value after removal of one feature. Let *bad* features be those features whose removal from the current candidate set causes only a slight *criterion value decrease*. Conversely, let *good* features be those whose removal causes a significant *criterion value decrease*. (At this stage, we do not need to quantify what a slight or significant decrease is.) It is apparent that, given a search tree topology, different feature assignments to its edges may be defined. The IBB algorithm aims to position *bad* features to the right, less dense part of the tree and *good* features to its left, more dense part. For such an ordering we may expect faster *bound* increase because the preferential removal of *bad* features should maintain the candidate criterion value at a high level. Consequently, removing *good* features from the candidate sets in later stages of search improves the chances of the criterion value dropping below the current *bound* and, therefore, allowing more effective subtree cut-offs.

The effect of this *in-level node ordering* heuristics can be illustrated on Fig. 1. The example tree is not an IBB tree because the first level is not ordered (feature sequence 5, 3, 4 as emphasized by gray background). If changed to conform to IBB ordering (5, 4, 3), the algorithm would find the optimum immediately in the rightmost leaf and one more criterion evaluation would be saved ($J(4,5)$). Note that evaluating the criterion function in the crossed nodes is not necessary as they lie on paths. Bypassing path nodes would reduce the tree to the "minimum solution tree." IBB, in combination with the "minimum solution tree," is consequently considered the most effective B & B algorithm.

In order to establish the framework for new algorithm definitions, let us first describe the IBB algorithm formally. The necessary notation has been adopted from [3]:

- k: tree level ($k = 0$ denotes the root),
- $\bar{\chi}_k = \{\xi_j \mid j = 1, 2, \cdots, D-k\}$: current "candidate" feature subset at kth tree level,
- q_k: number of current node descendants (in consecutive tree level),
- $\mathcal{Q}_k = \{Q_{k,1}, Q_{k,2}, \ldots, Q_{k,q_k}\}$: ordered set of features assigned to edges leading to current node descendants (note that "candidate" subsets $\bar{\chi}_{k+1}$ are fully determined by features $Q_{k,i}$ for $i = 1, \cdots q_k$),
- $\mathbf{J}_k = [J_{k,1}, J_{k,2}, \ldots, J_{k,q_k}]^{\mathrm{T}}$: vector of criterion values corresponding to current node descendants in consecutive tree level ($J_{k,i} = J(\bar{\chi}_k \setminus \{Q_{k,i}\})$ for $i = 1, \cdots, q_k$),
- $\Psi = \{\psi_j \mid j = 1, 2, \cdots, r\}$: control set of r features being currently available for search-tree construction, i.e., for building the set $\mathcal{Q}_k$; set Ψ serves for maintaining the search tree topology,
- $\mathcal{X} = \{x_j \mid j = 1, 2, \cdots, d\}$: current best feature subset,
- X^*: current *bound* (crit. value corresponding to $\mathcal{X}$).

Remark. Values q_j, sets $\mathcal{Q}_j$, and vectors $\mathbf{J}_j$ are to be stored for all $j = 0, \cdots, k$ to allow backtracking.

The Improved Branch & Bound Algorithm

Initialization:

$k = 0$, $\bar{\chi}_0 = Y$, $\Psi = Y$, $r = D$,

X^*: lowest possible value (computer dependent).

STEP 1. *Select descendants of the current node to form the consecutive tree level:* First set their number to $q_k = r - (D - d - k - 1)$. Construct an ordered set $\mathcal{Q}_k$ and vector $\mathbf{J}_k$ as follows: Sort all features $\psi_j \in \Psi, j = 1, \cdots, r$ according to their current true criterion value decreases

$$J(\bar{\chi}_k \setminus \{\psi_{j_1}\}) \leq J(\bar{\chi}_k \setminus \{\psi_{j_2}\}) \leq \cdots \leq J(\bar{\chi}_k \setminus \{\psi_{j_r}\})$$

and successively choose the first q_k features among them. Let

$Q_{k,i} = \psi_{j_i}$ for $i = 1, \cdots, q_k$

$J_{k,i} = J(\bar{\chi}_k \setminus \{\psi_{j_i}\})$ for $i = 1, \cdots, q_k$

To avoid future duplicate testing, exclude features ψ_{j_i} from further tree construction, let $\Psi = \Psi \setminus Q_k$ and $r = r - q_k$.

STEP 2. *Test the right-most descendant node (connected by the Q_{k,q_k}-edge)*: If $q_k = 0$, then all descendants have been tested and go to **Step 4** (backtracking). If $J_{k,q_k} < X^*$, then go to **Step 3**. Else let $\bar{\chi}_{k+1} = \bar{\chi}_k \setminus \{Q_{k,q_k}\}$. If $k + 1 = D - d$, then you have reached a leaf and go to **Step 5**. Otherwise go to the consecutive level: Let $k = k + 1$ and go to **Step 1**.

STEP 3. *Descendant node connected by the Q_{k,q_k}-edge (and its subtree) may be cut off:* Return feature Q_{k,q_k} to the set of features available for tree construction, i.e., let $\Psi = \Psi \cup \{Q_{k,q_k}\}$ and $r = r + 1$, $\mathcal{Q}_k = \mathcal{Q}_k \setminus \{Q_{k,q_k}\}$ and $q_k = q_k - 1$ and continue with its left neighbor; go to **Step 2**.

STEP 4. *Backtracking:* Let $k = k - 1$. If $k = -1$, then the complete tree has been searched through; stop the algorithm. Otherwise, return feature Q_{k,q_k} to the set of "candidates:" Let $\bar{\chi}_k = \bar{\chi}_{k+1} \cup \{Q_{k,q_k}\}$ and go to **Step 3**.

STEP 5. *Update the bound value:* Let $X^* = J_{k,q_k}$. Store the currently best subset $\mathcal{X} = \bar{\chi}_{k+1}$ and go to **Step 2**.

3 Drawbacks of the Traditional Branch & Bound Algorithms

When compared to the exhaustive search (ES), every B & B algorithm requires additional computations. Not only the target subsets of d features $\bar{\chi}_{D-d}$, but also their supersets $\bar{\chi}_{D-d-j}$, $j = 1, 2, \cdots, D - d$ have to be evaluated.

The B & B principle does not guarantee that enough subtrees will be cut off to keep the total number of criterion computations lower than their number in ES. The worst theoretical case would arise when we defined a criterion function $J(\bar{\chi}_k) = |\bar{\chi}_k| \equiv D - k$; the criterion function would be computed in every tree node.

A possibility of a weaker B & B performance may result from the following simple facts: 1) Nearer to the root, the criterion value computation is usually slower (evaluated feature subsets are larger) and 2) nearer to the root, subtree cut-offs are less frequent (higher criterion values following from larger subsets are compared to the *bound*, which is updated in leaves). The B & B algorithm usually spends most of the time by tedious, less promising evaluation of the tree nodes in levels closer to the root. This effect is to be expected, especially for $d \ll D$.

In the IBB algorithm, a significant number of additional computations is needed for the heuristic ordering of search tree nodes. The advantage following from these computations is not guaranteed because a slightly better heuristic

organization of the search tree under otherwise disadvantageous conditions can be outweighed by the additional computational time (see Fig. 9 where IBB runs slower than the "Basic" B & B).

4 Improving the "Improved" Algorithm

Let us illustrate the main IBB drawback by referring to Fig. 1. When constructing level 1, i.e., when specifying the ordering of the root descendants, IBB evaluates the *criterion value decrease* for every available feature (all five features), although only three features are to be assigned to the first level edges.

Our aim is to find the same (or very similar) ordering of tree nodes as given by means of the IBB algorithm with a smaller number of criterion evaluations. To achieve this goal we utilize a simple prediction mechanism. The innovation proposed here is that the consecutive tree levels be constructed in several phases. First, the *criterion value decrease* is quickly *predicted* for every feature currently available for the tree construction. The features are then sorted in descending order according to the *predicted criterion value decreases*. The true criterion value is then computed only for the required number of features (beginning from the feature with the highest *predicted criterion value decrease*) and the consecutive tree level is formed. In this way, the total number of true criterion computations per level remains comparable to the "Basic" B & B algorithm [5] without any tree node ordering, although the tree is actually constructed by means of an ordering heuristics similar to the one defined in IBB.

The prediction mechanism utilizes the current statistics of *criterion value decreases* caused by removing a particular feature accumulated during the search process. Note that single features are removed in different tree construction stages and that we need to collect the prediction information separately for every feature. First, let $\mathbf{A} = [A_1, A_2, \ldots, A_D]^{\mathrm{T}}$ be a *vector of feature contributions to the criterion value* (let us call it "*contribution vector*") which stores, for each feature, the individual average *criterion value decrease* caused by removing the feature from current "candidate" subsets. Next, let $\mathbf{S} = [S_1, S_2, \ldots, S_D]^{\mathrm{T}}$ be a *counter vector* recording the number of *criterion value decrease* evaluations for every individual feature.

In order to state the new B & B algorithm which utilizes partial prediction (BBPP), we shall use the notation defined for the IBB description (Section 2).

Whenever the algorithm removes some feature y_i from the current "candidate" subset and computes the corresponding real criterion value $J(\bar{\chi}_k \setminus \{y_i\})$ at kth tree level, the *prediction information* is updated as

$$A_{y_i} = \frac{A_{y_i} \cdot S_{y_i} + (J(\bar{\chi}_k) - J(\bar{\chi}_k \setminus \{y_i\})) \cdot 1}{S_{y_i} + 1} \tag{3}$$

and

$$S_{y_i} = S_{y_i} + 1. \tag{4}$$

with A_{y_i} and S_{y_i} initially set to zero for all $i = 1, \cdots, D$.

The Branch & Bound with Partial Prediction Algorithm
Initialization:

$k = 0$, $\bar{\chi}_0 = Y$, $\Psi = Y$, $r = D$,

X^*: lowest possible value (computer dependent).

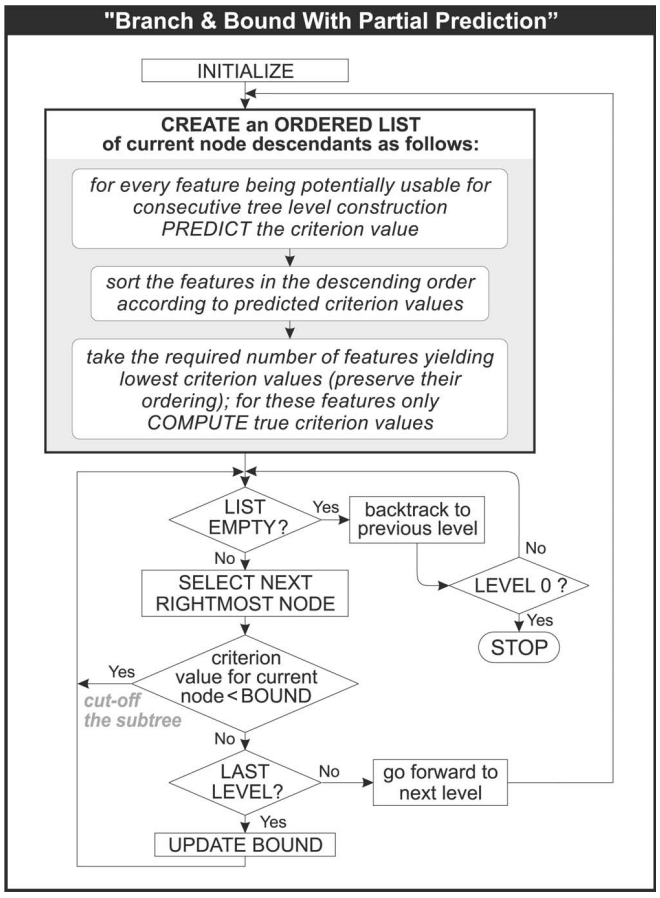


Fig. 2. Simplified diagram of the new BBPP algorithm.

STEP 1: *Select descendants of the current node to form the consecutive tree level*: First set their number to $q_k = r - (D - d - k - 1)$. Construct an ordered set $\mathcal{Q}_k$ and vector $\mathbf{J}_k$ as follows: Sort all features $\psi_j \in \Psi, j = 1, \cdots, r$ in the descending order according to their $A_{\psi_j}, j = 1, \cdots, r$ values, i.e.,

$$A_{\psi_{j_1}} \geq A_{\psi_{j_2}} \geq \cdots \geq A_{\psi_{j_r}}$$

and choose successively first q_k features among them. Let

$Q_{k,i} = \psi_{j_i}$ for $i = 1, \cdots, q_k$

$J_{k,i} = J(\bar{\chi}_k \setminus \{\psi_{j_i}\})$ for $i = 1, \cdots, q_k$

To avoid future duplicate testing, exclude features ψ_{j_i} from further tree construction, let $\Psi = \Psi \setminus Q_k$ and $r = r - q_k$.

STEPS 2, 3, 4, and 5 are the same as in IBB.

Properties of the BBPP algorithm are discussed in Sections 6 and 7. The simplified algorithm diagram is shown in Fig. 2.

5 Fast Branch & Bound

We propose the so-called Fast Branch & Bound (FBB) algorithm which aims to further reduce the number of criterion function computations in internal search tree nodes by means of a stronger prediction mechanism. The algorithm attempts to utilize the knowledge of *criterion value decreases* for future predictions of the criterion values. Both the computed and predicted criterion values are treated equally, i.e., are used for ordering tree nodes, testing subtree cut-off possibilities, etc. However, prediction is allowed under certain

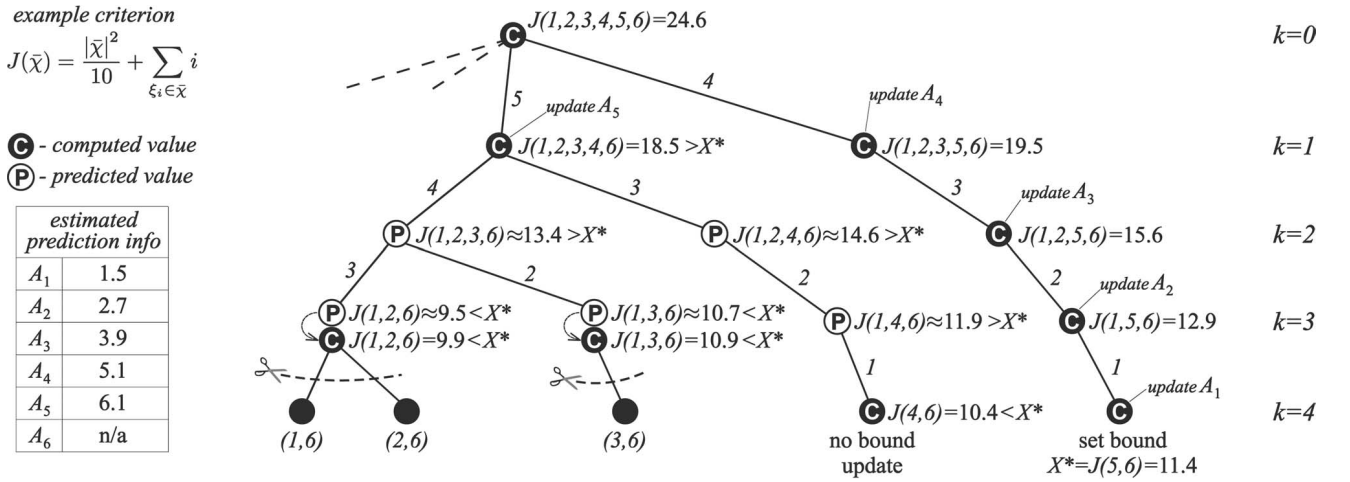


Fig. 3. Illustration of the Fast Branch & Bound prediction mechanism on a synthetic problem where $d = 2$ and $D = 6$.

circumstances only, e.g., not in leaves and not to trigger a node cut-off.

If a predicted criterion value remains significantly higher than the current *bound*, we may expect that even the real value would not be lower and, therefore, the corresponding subtree should not be cut off. In this situation, the algorithm continues to construct the consecutive tree level. But, if the predicted value comes close to the *bound* (and, therefore, there is a chance that the real value is lower than the *bound*) the actual criterion value must be computed. Only if true criterion values prove to be lower than the current *bound*, may subtrees be cut-off. Note that this prediction scheme does not affect the optimality of the obtained results. Inaccurate predictions will result in nothing worse than constructing subtrees, which would have been cut off by the classical B & B algorithm. However, this drawback is usually strongly outweighed by criterion computation savings achieved in other internal nodes, especially near the root, where the criterion computation tends to be slower due to a higher cardinality of the evaluated subsets.

Obviously, the following trade off arises: We want to use prediction as frequently as possible to save time. On the other hand, we have to compute true criterion values to tune (or at least to obtain) the information needed for the proposed prediction mechanism.

The FBB algorithm utilizes the same individual averaged *criterion value decrease* information stored in the *contribution* and *counter vectors* as BBPP. However, the FBB prediction mechanism is controlled more precisely because of its stronger influence on the search tree construction. Before every criterion evaluation, the record in the *counter vector* is checked for the appropriate feature entry. If the prediction mechanism has accumulated enough information (the counter value is higher than a prespecified *minimum*), the required criterion value will be predicted; otherwise, its true value will be computed.

Since it is important to distinguish between predicted and true criterion values, we introduce a vector of value types (we will denote it *type vector*) for the nodes of the current tree level. If a criterion value is predicted, the type value is set to "P." If the value is computed, the type flag is set to "C." The difference between criterion values computed in the current internal node and its parent node is used to update the information in the *contribution vector* only if both values have actually been computed (both for the current node and its parent node, "C" is recorded in the *type vector*). Should we attempt to update the *contribution vector* also by the difference of predicted or mixed type values, we could seriously degrade the prediction mechanism accuracy.

See Fig. 3 for illustration of the described mechanism. The figure shows how the prediction mechanism learns whenever two subsequent criterion values are computed and later uses this information to replace criterion evaluation by a simple subtraction. The predicted values, being only approximations of the true criterion values, do not suffice to cut off subtrees and must be verified by true criterion evaluation whenever tree cutting seems possible (see nodes representing subsets 1, 2, 6 and 1, 3, 6) to preserve the optimality of the results.

The prediction capability is curtailed in the early phases of the algorithm due to the lack of information. Therefore, we may expect frequent *contribution vector* updates. Later, with the number of "reliable" features (whose contribution to the criterion value has been evaluated for more than the required *minimum* number of times) increasing, the criterion values are obtained by means of prediction more frequently and the update process is deployed less frequently.

The aim of the advocated prediction mechanism is not only to reduce the number of criterion evaluations in internal nodes when the algorithm traverses the tree down to the leaves, but also to estimate the point (node) when the criterion value falls below the current *bound* and the corresponding subtree should be cut-off.

Let us refer to the case when the algorithm stops the prediction too soon (and the true value is still higher than the current *bound*) as a *pessimistic prediction error*. On the other hand, we shall refer to the case when the algorithm utilizes prediction too long (misses the real possibility of cutting off current subtree and continues to construct the consecutive tree level) as an *optimistic prediction error*. The prediction mechanism behavior may be modulated by introducing an *optimism* constant in the following way: Let every value from the *contribution vector* be multiplied by the *optimism* constant before being used for prediction. Higher *optimism* constant values will tend to protect the algorithm from missing real possibilities to cut off subtrees, but, on the other hand, the predicted criterion values will

have the tendency to decrease faster and fall below the current *bound* sooner than necessary.

To describe the FBB algorithm, we shall use the notation defined for the IBB (Section 2) and BBPP (Section 4) descriptions together with additional constants and symbols:

- $\delta \geq 1$: *minimum number of evaluations* (1 by default),
- $\gamma \geq 0$: *optimism* (1 by default),
- $\mathbf{T}_k = [T_{k,1}, T_{k,2}, \ldots, T_{k,q_k}]^{\mathrm{T}}, T_{k,i} \in$ {"C," "P"} for $i = 1, \cdots, q_k$: criterion value *type vector* (records the type of $J_{k,i}$ values),
- $\mathbf{V} = [v_1, v_2, \ldots, v_{q_k}]^{\mathrm{T}}$: temporary sort vector,

Remark. Values q_j, sets $\mathcal{Q}_j$, and vectors $\mathbf{J}_j$, $\mathbf{T}_j$ are to be stored for all $j = 0, \cdots, k$ to allow backtracking.

Whenever the algorithm removes some feature y_i from the current "candidate" subset and computes the corresponding true criterion value $J(\bar{\chi}_k \setminus \{y_i\})$ at kth tree level and if the predecessor value $J(\bar{\chi}_k) \equiv J(\bar{\chi}_{k-1} \setminus \{y_j\})$ (after previous removal of some feature y_j) had also been computed (as indicated by $T_{k-1,y_j} =$ "C"), A_{y_i} and S_{y_i} are updated as follows:

$$A_{y_i} = \frac{A_{y_i} \cdot S_{y_i} + J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})}{S_{y_i} + 1} \tag{5}$$

$$S_{y_i} = S_{y_i} + 1. \tag{6}$$

The Fast Branch & Bound Algorithm

In addition to the initialization described for BBPP in Section 4, set the δ and γ values according to guidelines in Sections 6.1 and 7.

STEP 1. *Select descendants of the current node to form the consecutive tree level:* First, set their number $q_k = r - (D - d - k - 1)$. Construct $\mathcal{Q}_k$, $\mathbf{J}_k$, and $\mathbf{T}_k$ as follows: For every feature $\psi_j \in \Psi, j = 1, \cdots, r$ if $k + 1 < D - d$ (nodes are not leaves) and $S_{\psi_j} > \delta$ (prediction allowed), then

$$v_j = J_{k-1,q_{k-1}} - A_{\psi_j}$$

i.e., predict by subtracting the appropriate prediction value based on ψ_j feature from the criterion value obtained in the parent node, otherwise the value must be computed, i.e.,

$$v_j = J(\bar{\chi}_k \setminus \{\psi_j\}).$$

After obtaining all v_j values, sort them in the ascending order, i.e.,

$$v_{j_1} \leq v_{j_2} \leq \cdots \leq v_{j_r}$$

and, for $i = 1, \cdots, q_k$, set
$Q_{k,i} = \psi_{j_i}$
$J_{k,i} = v_{j_i}$ if v_{j_i} records a computed value, or
$J_{k,i} = J_{k-1,q_{k-1}} - \gamma \cdot A_{\psi_{j_i}}$ otherwise,
$T_{k,i} =$ "C" if v_{j_i} records a computed value, or
$T_{k,i} =$ "P" otherwise.
To avoid duplicate testing set $\Psi = \Psi \setminus \mathcal{Q}_k$ and $r = r - q_k$.

STEP 2. *Test the right-most descendant node (connected by the Q_{k,q_k}-edge)*: If $q_k = 0$, then all descendants have been tested and go to **Step 4** (backtracking). If $T_{k,q_k} =$ "P" and $J_{k,q_k} < X^*$, compute the true value $J_{k,q_k} = J(\bar{\chi}_k \setminus \{Q_{k,q_k}\})$ and mark $T_{k,q_k} =$ "C." If $T_{k,q_k} =$ "C" and $J_{k,q_k} < X^*$, then go to **Step 3**. Else let $\bar{\chi}_{k+1} = \bar{\chi}_k \setminus \{Q_{k,q_k}\}$. If $k + 1 = D - d$, then a leaf has been reached and go to **Step 5**. Otherwise go to the next level: Let $k = k + 1$ and go to **Step 1**.

STEPS 3, 4, and 5 are the same as in IBB.

Remark. In Step 1, for $k = 0$ the term $J_{-1,q_{-1}}$ denotes the criterion value on the set of all features, $J(Y)$.

The simplified algorithm diagram is shown in Fig. 4.

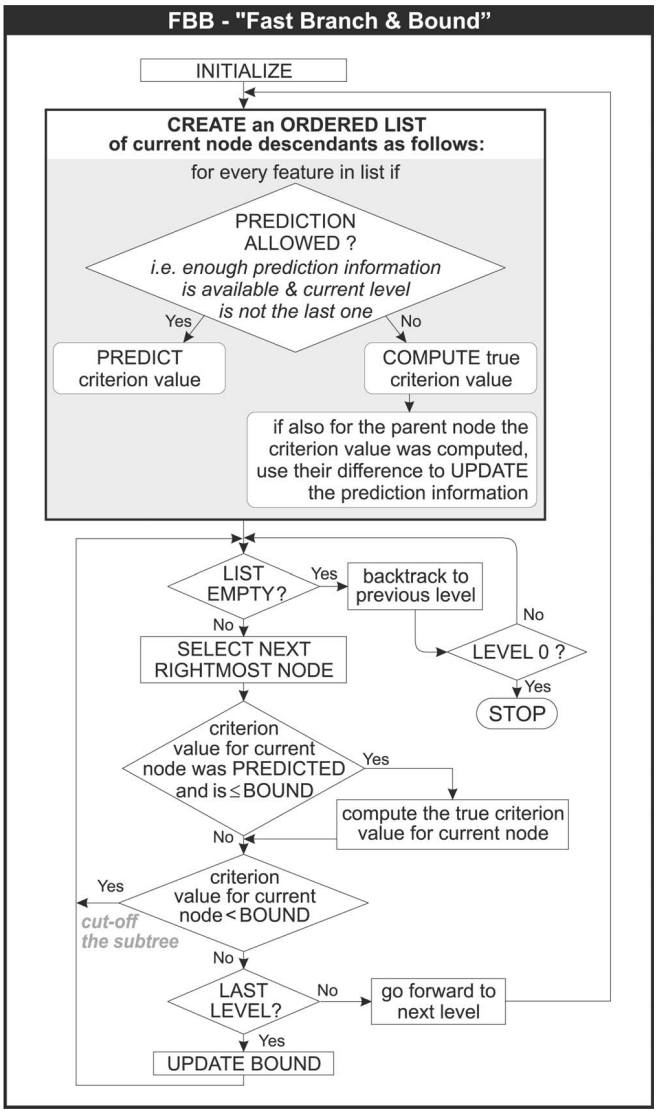


Fig. 4. Simplified diagram of the new FBB algorithm.

6 New Algorithm Properties

The performance of any B & B algorithm depends significantly on properties of the adopted criterion function and the evaluated data set. Apart from these external factors that affect the speed of the search process in all B & B variants (and will be discussed in detail in Section 7), the two new algorithms exhibit several distinct properties.

Both the new BBPP and FBB algorithms utilize a prediction mechanism that is based on heuristic assumptions and as such cannot be expected to work flawlessly under all conditions. In this sense, both the FBB and BBPP may be expected to be most effective and superior to IBB if the prediction mechanism succeeds both in learning and, later, in replacing true criterion values by sufficiently

reliable estimates. This is more likely to function if the individual feature contributions to the criterion value do not fluctuate strongly in relation to different subsets. In Section 7, we show that this is not a limiting factor, at least in the context of feature selection.

Assuming the prediction mechanism does not fail totally, the FBB can be expected to run faster than BBPP as it not only replicates the same functionality but extends it further. On the other hand, the BBPP efficiency may be expected to be less dependent on prediction accuracy. A potential prediction failure would have only indirect influence on the overall algorithm performance; wrong ordering of internal tree nodes would result in less effective subtree cut-offs, but, on the other hand, the basic advantage over the IBB algorithm—reducing the number of criterion function evaluations to order node descendants whenever the next tree level is constructed—would remain preserved.

The prediction mechanism appears to be relatively robust in the sense that, even with strong prediction inaccuracies, the FBB and especially BBPP remain able to overperform other B & B algorithms. Prediction mechanism inaccuracy may lead to a worse tree organization in BBPP or to additional true criterion computations in FBB (especially by prolonging branches as a result of optimistic prediction errors), but that rarely outweighs the number of computations saved anyway.

Unlike classical B & B algorithms, both the new FBB and BBPP always spend some additional time on maintaining the prediction mechanism. However, this additional time is negligible, especially when compared to the time savings gained from the eliminated nonrecursive criterion evaluations in the context of feature selection. When BBPP is used in conjunction with fast recursive criterion forms, the additional time still proves to be short enough to ensure an overall algorithm speedup.

It should be noted that alternative (e.g., context dependent) prediction mechanisms can be considered as well. When designing prediction mechanisms, the practitioner should keep in mind that: 1) The mechanism should learn quickly and be able to start prediction soon; otherwise, it would not serve its purpose and 2) the more sophisticated and computationally complex the mechanism is, the lower the overall performance gain—its complexity must remain considerably lower than that of true criterion evaluation. The mechanism we present in this paper has proven well-suited for the purpose.

6.1 FBB-Specific Properties

Obviously, FBB cannot be used with recursive criterion forms, where calculating $J(\bar{\chi}_k)$ value requires the knowledge of previous value $J(\bar{\chi}_{k-1})$. As FBB eliminates criterion function evaluations in many internal nodes, consecutive criterion values could not be then computed recursively in descendant nodes. The FBB algorithm can be expected to be most advantageous when used with nonrecursive criterion functions of high-computational complexity.

The two constants γ and δ that enable a precise control of FBB operation can be considered optional and set to default values $\gamma = 1$ and $\delta = 1$ for general purposes. For some tasks, different values may prove slightly better, but there is typically no way to identify them without accomplishing the actual B & B search.

Despite that, it should be noted that the *optimism* constant γ can affect the search process more significantly than the *minimum number of evaluations* constant δ. Values $\gamma > 1$ will make the algorithm behavior pessimistic, i.e., reduce the possible occurrence of optimistic prediction errors. The more pessimistic the algorithm behavior, the less prediction takes place (becomes restricted to tree levels closer to the root) and, as a result, the algorithm operates more similarly to the predictionless IBB. Values $0 < \gamma < 1$ will make the algorithm behavior optimistic, i.e., reduce the occurrence of pessimistic prediction errors. This behavior is theoretically more dangerous than the pessimistic one as, in this case, the algorithm may predict too extensively and postpone cutting decisions to deeper tree levels where the number of nodes to be evaluated increases. Ultimately, for $\gamma = 0$, the algorithm will collect the required prediction information and then simply evaluate all leafs, as all internal nodes would have been omitted by means of prediction—in this sense, the FBB would become equal to ES. The γ and δ influence is further discussed in Section 7.1.

7 Experiments

To illustrate the behavior of the algorithms considered, we present a series of experiments. Besides experiments on real data with standard probabilistic criteria, we include various synthetic data-independent tests defined to demonstrate how algorithm performance changes under different model conditions.

It should be emphasized that all the considered algorithms are optimal with respect to the chosen criterion function and yield identical subsets. The resulting classification performance is thus equal for all of them and depends on factors outside the scope of this paper (criterion choice, etc.). The only important algorithm difference we study here is therefore the computational complexity. To show the complexity, we primarily compare computational times. However, as the computational time may be affected by the properties of the operating system and compiler used, we compare the total number of criterion evaluations as well.

7.1 Synthetic Experiments

As stated before, the actual performance of any Branch & Bound algorithm is considerably affected by the properties of the used criterion functions and data sets. To demonstrate such properties, we have defined a set of synthetic illustrative criteria independent of particular data sets. The results have been collected in Fig. 5.

Fig. 5a shows the worst-case scenario when no branch cutting takes place in all B & B variants because all features yield equal contributions to criterion value. The following series of Figs. 5b, 5c, 5d, 5e, and 5f shows how the increasing diversity of features (in the sense of their contributions) leads to more effective branch cutting in all algorithms, while the prediction-based BBPP and FBB seem to utilize this increasing diversity better than IBB. An extreme case is shown in Fig. 5g where each feature yields higher contribution than is the sum of contributions of all features with lower indices. As a result, the *in-level ordering* heuristics succeeds in constructing a tree where the optimum is found in the first branch and all remaining branches are cut off immediately afterward, thus the algorithm finishes in linear time. Fig. 5h shows how the FBB prediction heuristics can be made dysfunctional if the

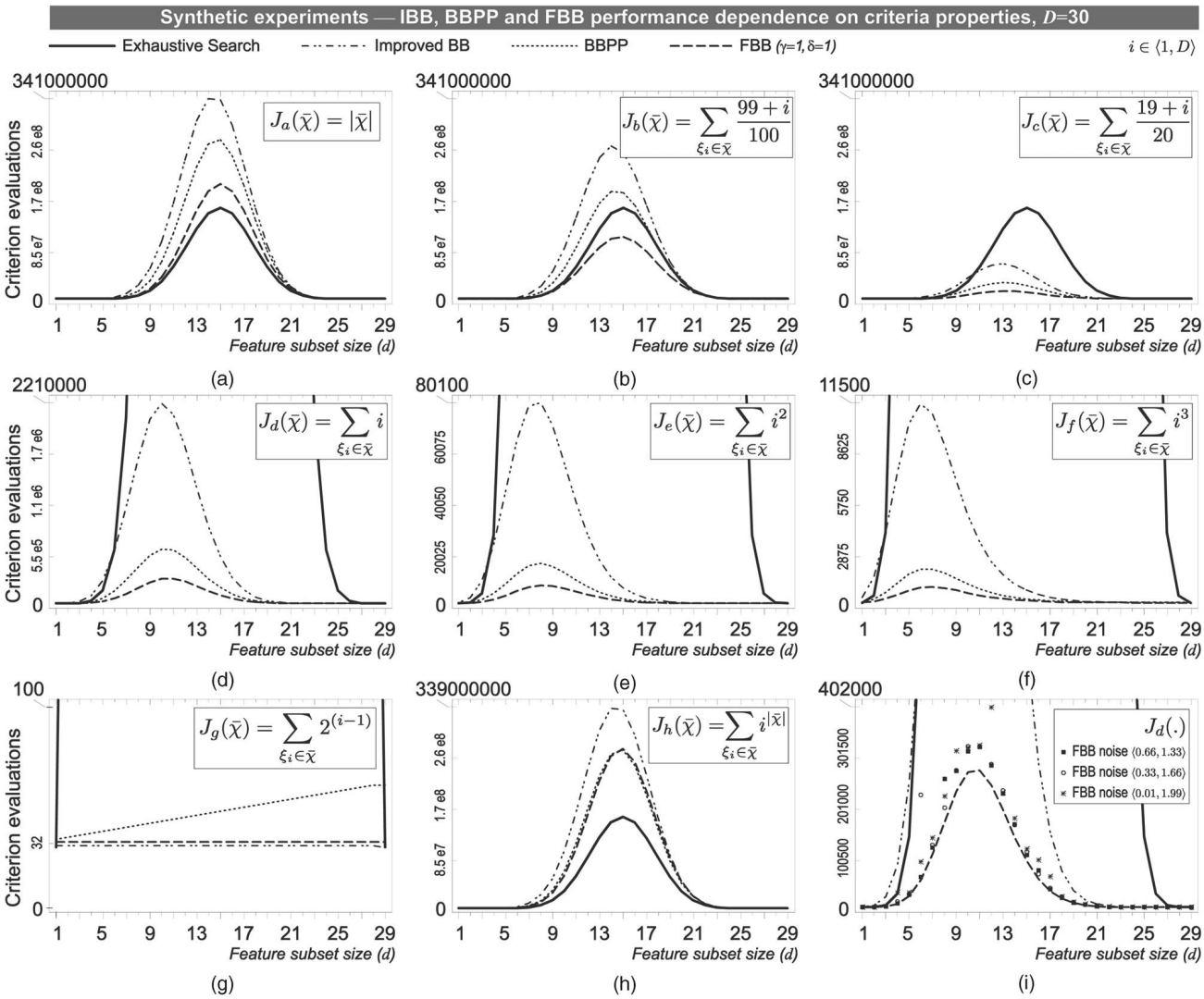


Fig. 5. Synthetic criteria demonstrating the Branch & Bound behavior. (a) Worst-case scenario that completely prohibits branch cutting. (b), (c), (d), (e), and (f) Series of examples demonstrating the advantageous effect of increasing diversity among features. (g) Best-case scenario allowing the most effective branch cutting. (h) FBB prediction learning failure scenario. (i) Influence of noise on FBB learning.

criterion prohibits the prediction mechanism from learning. This is achieved by making the criterion value strongly subset-size dependent. Fig. 5i shows how FBB performance decreases as a result of noisy learning. The noise is added in the prediction learning phase by multiplying the true feature contributions by random coefficients sampled from a uniform distribution on a given interval. Note that even relatively strong noise (interval $\langle 0.01, 1.99 \rangle$) does not necessarily deteriorate the FBB performance to that of the IBB.

To summarize our observations, we can state that the performance of the advanced B & B algorithms depends primarily on the following characteristics of the criterion evaluation process:

1. All features should exhibit stable and measurable differences between each other (in the sense of contribution to criterion value). The more equal the feature contributions are, the worse the performance of any B & B algorithm. The reason is clear—when traversing branches down from root to leaves, the criterion value decreases by the removed feature contributions. In the case of almost equal contributions the criterion values in each level would also remain similar. Cutting-off thus can almost never take place. It can also be observed that all algorithms with *in-level ordering* heuristics (IBB, BBPP, FBB) create trees where features containing less distinguishable amount of information are tested more often among each other (appear in more nodes) than the unique ones.
2. Feature contribution should not vary randomly with respect to different subsets. Unstable or noisy feature behavior reduces the learning performance of the prediction mechanisms (with particular consequences in FBB). Moreover, the *in-level ordering* heuristics would then produce more or less random orderings that would consequently degrade the overall performance of all IBB, BBPP, and FBB (see Fig. 5i).
3. Criterion value should not depend too strongly on subset size (tree level). Too high differences between contributions of the same feature among different tree levels may prevent the FBB prediction mechanism (as defined in this paper) from predicting reasonable values in consecutive tree levels. Consequently, the *in-level ordering* performance

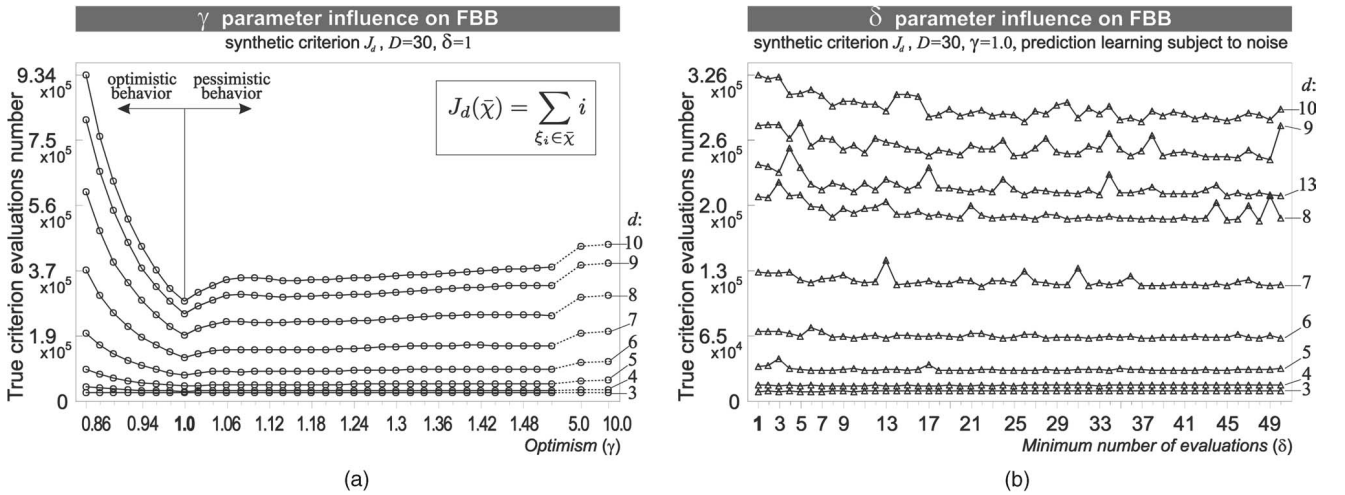


Fig. 6. Influence of parameters γ and δ on the FBB prediction mechanism.

drops in all IBB, BBPP, and FBB similarly to the case of noisy behavior (see Fig. 5h).

An important observation is that, regardless of whether particular synthetic criteria deteriorated or accelerated the search process, the prediction-based algorithms (BBPP, FBB) remained, in practice, always superior to the referential IBB.

Using synthetic tests, we have also investigated the influence of FBB parameters γ and δ. Fig. 6a confirms the supposition that optimistic errors (invoked here by setting $\gamma < 1$) may deteriorate the FBB performance significantly more than pessimistic ones (invoked here by setting $\gamma > 1$). For this reason, values of $\gamma > 1$ may prove advantageous in real problems to prevent optimistic behavior of the prediction mechanism that may be caused by possible noise in data, learning problems, and/or criterion properties. Nevertheless, the graph shows that the γ value of 1 is theoretically the best and should be considered the default value. Fig. 6b shows that setting $\delta > 1$ may result in a slight performance gain in cases when, due to noise or instability of feature contributions, the prediction mechanism needs longer learning before becoming able to predict reasonably well. To demonstrate this effect, we let the algorithm learn feature contributions inaccurately—each multiplied by a random coefficient from a uniform distribution on $\langle 0.5, 1.5 \rangle$. The performance gain follows from the fact that initial tree construction stages (ordering in levels near the root) may prove crucial for later effective branch cut-offs. Initial wrong ordering may delay *bound* convergence to optimum and, consequently, reduce the effectiveness of branch cutting. Experiments show, that δ affects the search process considerably less than γ. Values $\delta > 1$ can be recommended, but may result in approximately $d * \delta$ true criterion evaluations that could have been possibly replaced by means of prediction if δ was lower. The number $d * \delta$ becomes insignificant for problems with very high number of configurations to be tested (typically, when $d \approx D/2$).

7.2 Real Data Experiments

The algorithms have been tested on a number of different data sets. Here, we show and discuss the results computed on the 2–class WDBC mammogram data from the Wisconsin Diagnostic Breast Center (30 features, 357 benign, and 212 malignant samples) and 3–class WAVEFORM data (40 features, 1,692 class 1, and 1,693 class 2 samples, we used the first two classes only to meet criterion limitations), both obtained via the UCI repository [19] http://ftp.ics.uci.edu. We also used 2–class SPEECH data originating at British Telecom (15 features, 682 word "yes," and 736 word "no" samples) obtained from the Centre for Vision, Speech, and Signal Processing, University of Surrey. To emphasize the universality of our concept, we used three different criterion functions: Bhattacharyya, Divergence, and Patrick-Fischer, respectively. We used both the recursive (where possible) and nonrecursive forms of the Bhattacharyya distance. The performance of different methods is illustrated in Figs. 7, 8, and 9, showing the graphs of (a) computational time and (b) number of criterion evaluations. Tables 2, 3, and 4 show the statistics that form the basis of the BBPP and FBB prediction mechanisms (estimated using IBB with no computations omitted by means of prediction). In addition to vectors $\mathbf{A}$ (feature contributions) and $\mathbf{S}$ (numbers of feature contribution evaluations), the tables show the deviation of A_i values and the individual criterion value of each feature.

In case of nonrecursive criterion forms, we implemented all B & B algorithms to construct the "minimum solution tree" [28]. The FBB parameters have been set to default values $\gamma = 1$ and $\delta = 1$ for all tests.

The exponential nature of the optimal search problem is well-illustrated by the fact that, for 15-dimensional data (Fig. 9), the ES remained relatively comparable to B & B algorithms, while, for 30-dimensional data and $d = 15$ (Fig. 7), it required approximately $140\times$ more criterion evaluations than IBB and, in case of 40-dimensional data and $d = 20$ (Fig. 8), the difference reached 1.3×10^8. According to expectations, the effectiveness of the "Basic" B & B [5] also proved to be strongly inferior to the more advanced B & B algorithms, except in the low-dimensional SPEECH case (Fig. 9) where the combination of low dimension and Patrick-Fischer criterion properties degraded the performance of more advanced algorithms.

Similarly to the previous synthetic tests, Figs. 7, 8, and 9 confirm that prediction-based algorithms (BBPP, FBB) practically always operate faster than the referential IBB, while FBB operates mostly faster than or equally fast as BBPP.

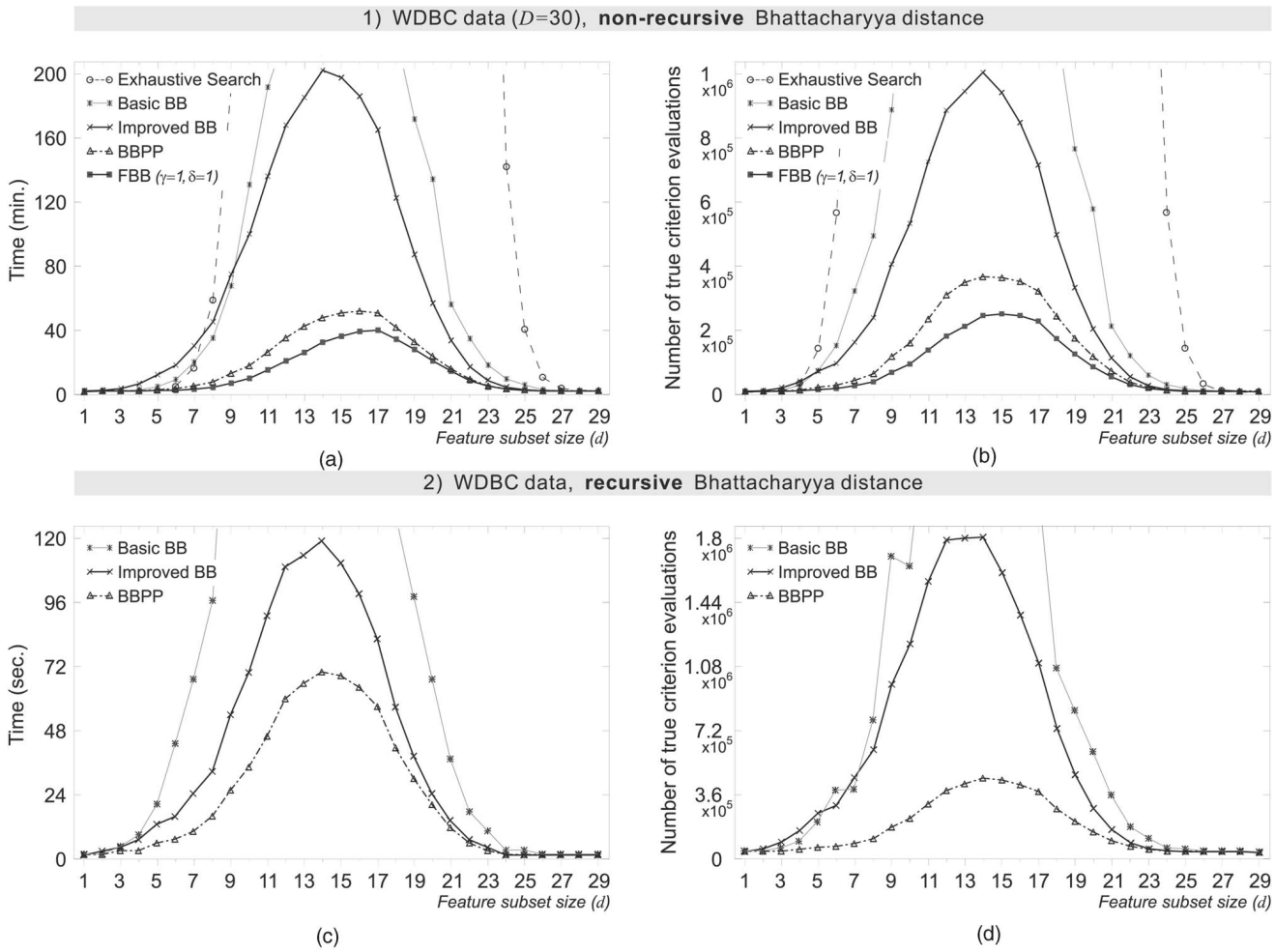


Fig. 7. Optimal subset search methods performance on 2-class WDBC mammogram data. Results computed on a Pentium II-350 MHz computer.

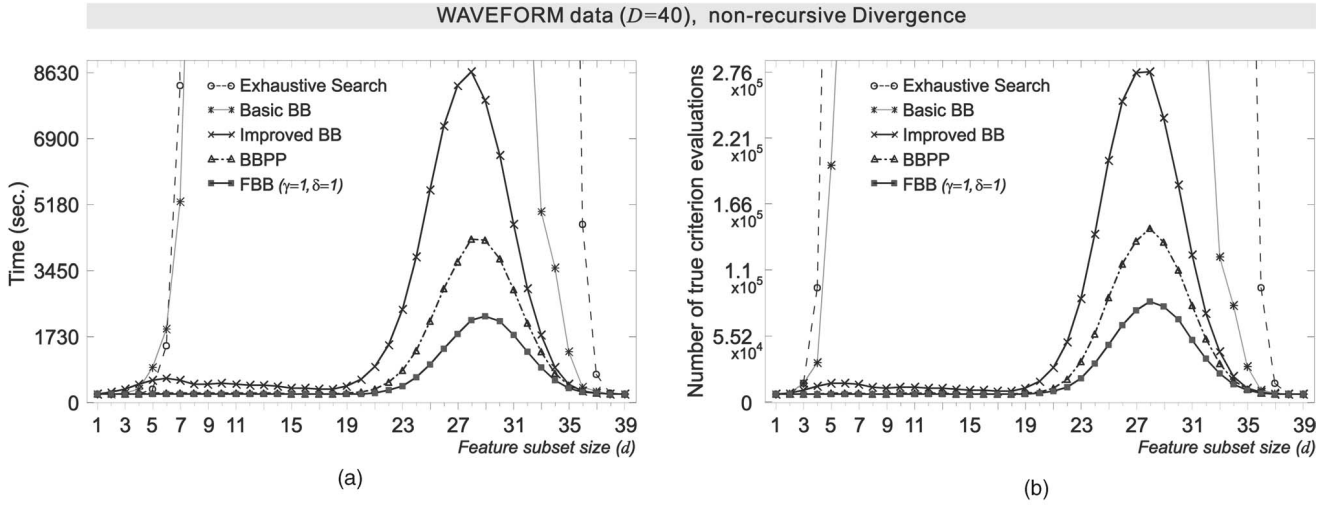


Fig. 8. Optimal subset search methods performance on 2-class WAVEFORM data. Results computed on a Pentium II-350 MHz computer.

Depending on the data set and the criteria used, the FBB has been shown to run approximately 1.5 to 10 times faster than IBB in feature selection tasks with the computationally most expensive settings of d close to $D/2$.

The WDBC example (Fig. 7) shows the behavior of the algorithms considered, with the Bhattacharyya distance as a criterion. Table 2 shows that reliable prediction information (as indicated by low A_i deviation) could be estimated using the Bhattacharyya distance and that, for the WDBC data set, the contributions of different features remain reasonably distinguishable. Both the *in-level ordering* heuristics and prediction mechanisms operated as expected.

The WAVEFORM example (Fig. 9) exhibits a noticeable phenomenon where the IBB, BBPP and FBB algorithms

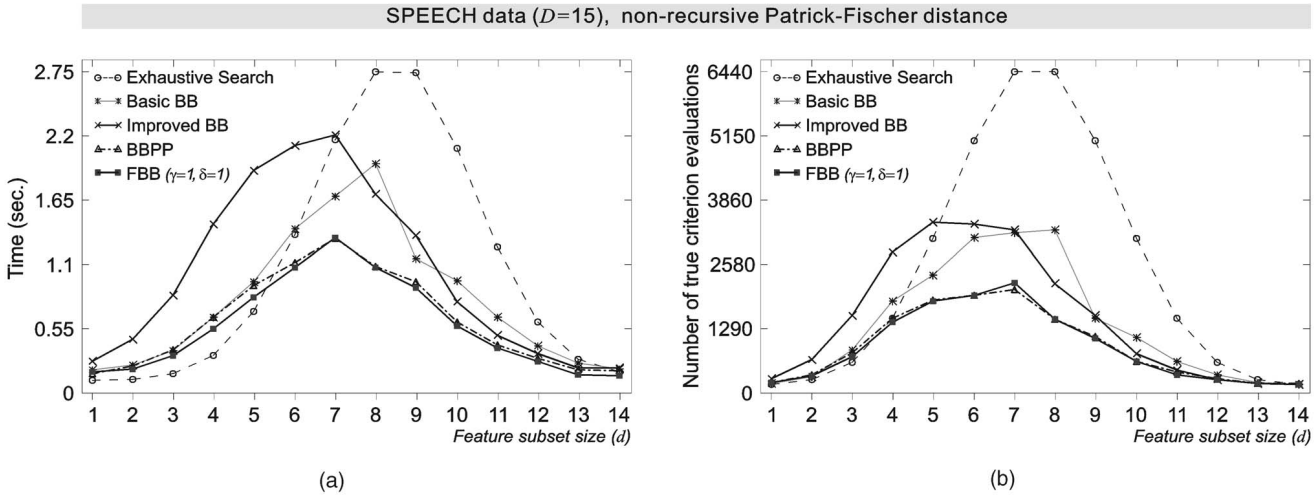


Fig. 9. Optimal subset search methods performance on 2-class SPEECH data. Results computed on a Pentium II-350 MHz computer.

TABLE 2
Feature Contribution Statistics Obtained Using the IBB Algorithm to Select 15 Out of 30 Features of the WDBC Data Set to Maximize Bhattacharyya Distance

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A_i	**0.269**	**0.066**	**0.272**	**0.651**	**0.118**	**0.114**	**0.195**	**0.130**	**0.085**	**0.077**	**0.214**	**0.119**	**0.126**	**0.843**	**0.127**
$A_i dev.$	0.076	0.022	0.098	0.048	0.030	0.020	0.041	0.029	0.028	0.019	0.087	0.021	0.040	0	0.034
S_i	312	95630	525	4	30327	56149	1400	23549	93275	89565	395	51923	12972	1	10083
$J_B(\{i\})$	0.586	0.113	0.629	0.641	0.073	0.307	0.498	0.790	0.064	0.003	0.463	0.010	0.483	0.803	0.003
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A_i	**0.133**	**0.441**	**0.122**	**0.105**	**0.198**	**0.340**	**0.083**	**0.267**	**0.797**	**0.095**	**0.199**	**0.103**	**0.132**	**0.123**	**0.094**
$A_i dev.$	0.031	0.004	0.026	0.015	0.051	0.108	0.026	0.084	0.011	0.017	0.031	0.027	0.019	0.015	0.032
S_i	7746	7	31892	77102	452	31	91836	142	2	88359	1037	9441	32543	51707	38630
$J_B(\{i\})$	0.050	0.081	0.103	0.032	0.036	0.787	0.140	0.816	0.811	0.110	0.327	0.389	0.820	0.175	0.101

$J_B(Y) = 7.53989$. *Best subset* $\mathcal{X} = \{1, 3, 4, 6, 7, 11, 14, 15, 16, 17, 21, 23, 24, 26, 27\}$. $J_B(\mathcal{X}) = 5.741$.

TABLE 3
Feature Contribution Statistics Obtained Using the IBB Algorithm to Select 20 Out of 40 Features of the Waveform Data Set to Maximize Divergence

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A_i	**0.090**	**0.115**	**0.364**	**0.429**	**0.727**	**0.756**	**0.934**	**1.456**	**3.839**	**4.535**	**5.131**	**2.304**	**0.736**	**0.712**
$A_i dev.$	0.005	0.016	0.015	0.031	0.047	0.091	0	0	0	0	0	0	0.051	0.076
S_i	56	17	10	8	3	4	1	1	1	1	1	1	5	6
$J_D(\{i\})$	1.7e-4	0.002	0.005	0.004	0.008	0.088	0.430	1.328	3.130	2.945	3.182	0.759	3.5e-4	0.497
i	15	16	17	18	19	20	21	22	23	24	25	26	27	28
A_i	**3.660**	**3.337**	**4.073**	**1.500**	**0.756**	**0.493**	**0.092**	**0.041**	**0.065**	**0.039**	**0.069**	**0.035**	**0.040**	**0.052**
$A_i dev.$	0	0	0	0	0.007	0.017	0.005	0.005	0.004	0.008	0.005	0.009	0.008	0.006
S_i	1	1	1	1	2	7	46	703	523	699	358	699	703	635
$J_D(\{i\})$	2.551	2.840	3.281	1.800	0.769	0.220	2.5e-6	2.2e-4	0.005	0.002	0.002	0.002	0.002	4.5e-5
i	29	30	31	32	33	34	35	36	37	38	39	40		
A_i	**0.058**	**0.072**	**0.050**	**0.073**	**0.089**	**0.073**	**0.044**	**0.050**	**0.060**	**0.041**	**0.063**	**0.032**		
$A_i dev.$	0.006	0.007	0.007	0.004	0.004	0.006	0.004	0.006	0.006	0.004	0.005	0.002		
S_i	592	280	677	261	72	227	668	681	548	698	560	703		
$J_D(\{i\})$	0.002	0.002	5.4e-4	2.6e-4	0.002	4e-4	0.003	0.005	2.8e-4	0.005	2.5e-4	9.4e-4		

$J_D(Y) = 33.8236$. *Best subset* $\mathcal{X} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21\}$. $J_D(\mathcal{X}) = 32.7742$.

operated exceptionally quickly up to subset size $d = 20$. The most likely explanation of this fact is that about half of the features represent noise only (as confirmed in data documentation). The difference between the estimated contributions of informative and noisy features proven instrumental for the *in-level ordering* heuristics being effective, as shown before in Section 7.1.

The SPEECH example (Fig. 9) is representative of a more difficult task (in the sense of B & B effectiveness) where the criterion (Patrick-Fischer) exhibits behavior that is disadvantageous for all advanced B & B algorithms. Table 4 suggests that Patrick-Fischer distance yields values strongly subset-size dependent. The average feature contributions A_i are significantly different from individual values $J_P(\{i\})$, while the contribution deviation is extremely high. This is the most likely cause of the relatively weak performance of IBB in relation to the "Basic" B & B and for the degradation of FBB to BBPP level. This example also illustrates the fact that ES may operate faster than all B & B algorithms for d close to 0, where, due to the search tree depth and low total number of configurations, B & B may have to evaluate the criterion more times than ES.

TABLE 4
Feature Contribution Statistics Obtained Using the IBB Algorithm to Select Seven out of 15 Features of the Speech Data Set to Maximize Patrick-Fischer Distance

i	1	2	3	4	5	6	7	8
A_i	**161814.60**	**989640.78**	**397470.44**	**370585.33**	**226872.75**	**5296360.67**	**946963.92**	**617125.62**
$A_i dev.$	1342773.565	4262852.321	2620923.968	2649106.231	1833531.444	11513004.74	6273012.913	4237324.195
S_i	439	96	258	266	414	13	57	112
$J_P(\{i\})$	0.367341	0.106052	0.346372	0.317521	0.150978	0.465456	0.385585	0.00149085
i	9	10	11	12	13	14	15	
A_i	**48500000**	**518048.73**	**593087.18**	**18256666.67**	**170793.95**	**303314.61**	**3887584.33**	
$A_i dev.$	0	4166274.724	3390783.969	20820881.08	1451834.918	2240092.05	9341389.259	
S_i	1	120	162	3	439	336	20	
$J_P(\{i\})$	0.955186	0.872747	0.746423	1.68501	1.46025	1.00877	2.17263	

$J_P(Y) = 52,069,500$. *Best subset* $\mathcal{X} = \{7, 8, 9, 10, 11, 12, 15\}$. $J_P(\mathcal{X}) = 8,995.47$.

Note that the graphs of FBB and BBPP exhibit, in all cases, a slight shift to the right when compared to IBB. The prediction-based algorithm acceleration relates to the number of criterion evaluation savings in internal search tree nodes. Therefore, with decreasing d, the tree depth increases and allows a more effective prediction mechanism operation.

Tables 2, 3, and 4 illustrate not only the feature contribution behavior but also the general performance of the *in-level ordering* heuristics present in all IBB, BBPP, and FBB algorithms. Note that features yielding low contribution to the criterion value are evaluated more frequently than features with high contributions. This shows how the algorithm accelerates the improvement of the *bound* and how most contributing features become "accepted" soon and excluded from further testing. Besides that, the tables illustrate the known fact that individual criterion values do not represent well the feature importance. Note how strongly some individual $J(\{i\})$ differ from the respective A_i values.

Remark. The number of computations in the recursive and nonrecursive cases in Fig. 7 differs. The graphs illustrate the difference between the search with and without "minimum solution tree" [28] that could not have been employed with recursive criterion computation.

8 Conclusion

Based on a detailed study of the Branch & Bound principle, we developed two novel optimal subset search algorithms which eliminate a significant number of computations. The proposed procedures are the Branch & Bound with Partial Prediction Algorithm and the Fast Branch & Bound Algorithm. The FBB algorithm is especially suitable for use with computationally expensive criterion functions. The BBPP algorithm is well-suited for use with both recursive and nonrecursive criterion forms. Both algorithms were experimentally shown to be significantly more efficient than any other traditional Branch & Bound algorithm while yielding identical optimal results.

We described the algorithms in the context of feature selection, but we suppose their applicability can be much broader.

References

[1] R. Caruana and D. Freitag, "Greedy Attribute Selection," *Proc. Int'l Conf. Machine Learning,* pp. 28-36, 1994.

[2] N. Chaikla and Y. Qi, "Genetic Algorithms in Feature Selection," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics,* vol. 5, pp. 538-540, 1999.

[3] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* Prentice Hall, 1982.

[4] I. Foroutan and J. Sklansky, "Feature Selection for Automatic Classification of Non-Gaussian Data," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 17, pp. 187-198, 1987.

[5] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. Academic Press, Inc., 1990.

[6] M. Gengler and G. Coray, "A Parallel Best-First Branch and Bound Algorithm and Its Aximatization," *Parallel Algorithms and Applications,* vol. 2, pp. 61-80, 1994.

[7] Y. Hamamoto, S. Uchimura, Y. Matsuura, T. Kanaoka, and S. Tomita, "Evaluation of the Branch and Bound Algorithm for Feature Selection," *Pattern Recognition Letters,* vol. 11, no. 7, pp. 453-456, July 1990.

[8] A. Iamnitchi and I. Foster, "A Problem-Specific Fault-Tolerance Mechanism for Asynchronous, Distributed Systems," *Proc. Int'l Conf. Parallel Processing,* pp. 4-14, 2000.

[9] A.K. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, pp. 153-158, 1997.

[10] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680, 1983.

[11] R. Kohavi and G.H. John, "Wrappers for Feature Subset Selection," *Artificial Intelligence,* vol. 97, nos. 1-2, pp. 273-324, 1997.

[12] D. Koller and M. Sahami, "Toward Optimal Feature Selection," *Proc. 13th Int'l Conf. Machine Learning,* pp. 284-292, 1996.

[13] R.E. Korf, "Artificial Intelligence Search Algorithms," *Handbook of Algorithms and Theory of Computation,* chapter 36, CRC press, 1999.

[14] M. Kudo and J. Sklansky, "Comparison of Algorithms that Select Features for Pattern Classifiers," *Pattern Recognition,* vol. 33, no. I, pp. 25-41, Jan. 2000.

[15] V. Kumar and L.N. Kanal, "A General Branch and Bound Formulation for Understanding and Synthesizing and/or Tree Search Procedures," *Artificial Intelligence,* vol. 21, pp. 179-198, 1983.

[16] E.L. Lawler and D.E. Wood, "Branch and Bound Methods: A Survey," *Operations Research,* vol. 149, pp. 699-719, 1966.

[17] H. Liu, H. Motoda, and M. Dash, "A Monotonic Measure for Optimal Feature Selection," *Proc. European Conf. Machine Learning,* pp. 101-106, 1998.

[18] A. Mitschele-Thiel, "Optimal Compile-Time Scheduling and Network Configuration," *Transputers '94: Advanced Research and Industrial Applications* IOS Press, pp. 153-164, 1994.

[19] P.M. Murphy and D.W. Aha, "UCI Repository of Machine Learning Databases (Machine-Readable Data Repository)," Dept. of Information and Computer Science, Univ. of California, Irvine, 1994.

[20] P.M. Narendra and K. Fukunaga, "A Branch and Bound Algorithm for Feature Subset Selection," *IEEE Trans. Computers,* vol. 26, no. 9, pp. 917-922, Sept. 1977.

[21] N.J. Nilsson, *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.

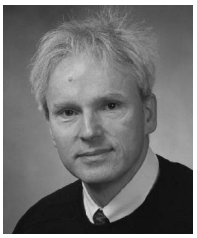
[22] N.J. Nilsson, *Artificial Intelligence: A New Synthesis.* Morgan Kaufmann 1998.

[23] P. Somol, P. Pudil, F.J. Ferri, and J. Kittler, "Fast Branch & Bound Algorithm in Feature Selection," *Proc. Fourth World Multiconf. Systemics, Cybernetics, and Informatics,* vol. 7, Part 1, pp. 646-651, 2000.

[24] P. Somol, P. Pudil, and J. Grim, "Branch & Bound Algorithm with Partial Prediction for Use with Recursive and Non-Recursive Criterion Forms," *Lecture Notes in Computer Science,* vol. 2013, pp. 230-238, 2001.
[25] G.I. Webb, "OPUS: An Efficient Admissible Algorithm for Unordered Search," *J. Artificial Intelligence Research,* vol. 3, pp. 431-465, 1995.
[26] Ch. Xu, S. Tschke, and B. Monien, "Performance Evaluation of Load Distribution Strategies in Parallel Branch and Bound Computations," *Proc. Seventh IEEE Symp. Parallel and Distributed Processing,* pp. 402-405, 1995.
[27] M.K. Yang and C.R. Das, "A Parallel Optimal Branch-and-Bound Algorithm for MIN-Based Multiprocessors," *Proc. IEEE 1999 Int'l Conf. Parallel Processing,* pp. 112-119, 1999.
[28] B. Yu and B. Yuan, "A More Efficient Branch and Bound Algorithm for Feature Selection," *Pattern Recognition,* vol. 26, pp. 883-889, 1993.

Petr Somol received the MSc and PhD degrees from the Faculty of Mathematics and Physics, Charles University, Prague, both in computer science. He is with the Department of Pattern Recognition at the Institute of Information Theory, Academy of Sciences of the Czech Republic, and also with the Medical Informatics Unit, IPH, the University of Cambridge, United Kingdom. His current interests include statistical approach to pattern recognition, combinatorial algorithms, graphics, and modern programming.

Pavel Pudil is the dean of the Faculty of Management, Prague University of Economics and is also with the Department of Pattern Recognition at the Institute of Information Theory, Academy of Sciences of the Czech Republic. He has published more than 100 papers regarding statistical pattern recognition, especially feature selection and mixture modeling. From 1996 to 1999, he was an IAPR Technical Committee 1 Chairman. In 2000, he was elected an IAPR fellow and he is a member of the IEEE and the IEEE Computer Society.



Josef Kittler is a professor of machine intelligence and director of the Centre for Vision, Speech, and Signal Processing at the University of Surrey. He has worked on various theoretical aspects of pattern recognition and image analysis and on many applications, including personal identity authentication, automatic inspection, target detection, detection of microcalcifications in digital mammograms, video coding and retrieval, remote sensing, robot vision, speech recognition, and document processing. He has coauthored the book *Pattern Recognition: A Statistical Approach* (Prentice Hall) and has published more than 400 papers. He is a member of the editorial boards of *Image and Vision Computing*, *Pattern Recognition Letters*, *Pattern Recognition and Artificial Intelligence*, *Pattern Analysis and Applications*, and *Machine Vision and Applications*. He is a member of the IEEE and the IEEE Computer Society.