On Prediction Mechanisms in Fast Branch & Bound Algorithms

Petr Somol[1,2], Pavel Pudil[2,1], and Jiří Grim[1]

[1] Dept. of Pattern Recognition, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 182 08 Prague 8, e-mail: somol@utia.cas.cz, Home page: www.utia.cas.cz/user_data/PR_dept

[2] Faculty of Management of the Prague University of Economics, 377 01 Jindřichův Hradec, Czech Republic, e-mail: pudil@fm.vse.cz, Home page: www.fm.vse.cz

Abstract. The idea of using the Branch & Bound search for optimal feature selection has been recently refined by introducing additional predicting heuristics that is able to considerably accelerate the search process while keeping the optimality of results unaffected. The heuristics is used most extensively in the so-called Fast Branch & Bound algorithm, where it replaces many slow criterion function computations by means of fast predictions. In this paper we investigate alternative prediction mechanisms. The alternatives are shown potentially useful for simplification and speed-up of the algorithm. We demonstrate the robustness of the prediction mechanism concept on real data experiments.

Keywords: subset search, feature selection, search tree, optimal search, subset selection, dimensionality reduction.

1 Introduction

The problem of optimal feature selection (or more generally of subset selection) is difficult especially because of its time complexity. Any known optimal search algorithm has an exponential nature. The only alternative to the exhaustive search is the *Branch & Bound* (B&B) algorithm [5, 2] and ancestor algorithms based on a similar principle. Two of the recent algorithm versions utilize a concept of prediction mechanism that enables considerable acceleration of the search process without affecting the optimality of results. The Branch & Bound with Partial Prediction (BBPP) [7] uses a prediction mechanism to avoid many criterion evaluations that are unavoidable in older algorithm versions for finding efficient ordering of features inside the search tree. The Fast Branch & Bound (FBB) [6] extends the prediction mechanism to enable bypassing of non-prospective branch sections. The speed-up of BBPP and FBB over older algorithms depends strongly on the efficacy of the underlying prediction mechanisms. In this paper we define and investigate alternative prediction mechanisms in addition to the original one. We show that alternative mechanisms may further improve the FBB speed and also simplify its implementation.

1.1 Preliminaries

All the algorithms addressed in this paper require the criterion function fulfilling the *monotonicity condition*. Consider a problem od selecting d features from an initial set of D measurements using objective function J as a criterion of subset 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.

$$\bar{\chi}_j = \{\xi_i | \xi_i \in Y, 1 \leq i \leq D; \xi_i \neq y_k, \forall k\}$$

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

$$J(\bar{\chi}_1) \geq J(\bar{\chi}_2) \geq \cdots \geq J(\bar{\chi}_j). \tag{1}$$

Each B&B algorithm constructs a search tree where each node represents some set of "candidates". The root represents the set of all D features and leafs represent target subsets of d features. While traversing the tree down to leafs the algorithm successively removes single features from the current "candidate"set ($\bar{\chi}_k$ in the k-th level) and evaluates the criterion value. In leafs the information about both the currently best subset $\mathcal{X}$ and the '*bound*' $X^* = J(\mathcal{X})$ is updated. Anytime the criterion value in some internal node is found to be lower than the current *bound*, due to the condition (1) the whole sub-tree may be cut-off and many computations may be omitted. For details see [1, 2, 5].

Several improvements of this scheme are known. The "Improved" B&B algorithm [2] combined with the "minimum solution tree" [9] concept can be considered the fastest non-predicting algorithm. This algorithm (to be referred as IBB) improves the search speed by optimising the tree topology and bypassing redundant computations in paths leading to leafs. The Fast Branch & Bound includes both of these improvements and incoroporates additional mechanisms to further reduce the impact of some of the principal B&B drawbacks [6] what makes it approximately 2 to 20 times faster than IBB in feature selection tasks, depending on data and criterion properties. As all optimal algorithms yield equal results at a cost of (in principle) exponential computational time, speed becomes the most important property to compare. It should be noted, that releasing the rigor of result optimality in sub-optimal algorithms is the only way to achieve fundamentally higher computational speed of polynomial nature.

2 Fast Branch & Bound

Let the *criterion value decrease* be the difference between the criterion value for the current feature subset and the value after removal of one feature. The FBB

uses *criterion value decreases* estimates for future predictions of the criterion values. Prediction is used only in non-leaf nodes and can not trigger a node cut-off. If a predicted criterion value remains significantly higher than the current *bound*, it may be expected that even the real value would not be lower and therefore the corresponding sub-tree could not be cut-off. In this situation the FBB proceeds to the consecutive tree level. But, if the predicted value drops below the *bound*, the actual criterion value must be computed to evaluate the cut-off chance. Sub-trees may be cut-off only if true criterion values prove to be lower than the current *bound*, what preserves the optimality of the final result. Note that the only impact of possibly inaccurate predictions is prolonging some branches. However, this drawback is usually strongly outweighed by the overall criterion computation savings.

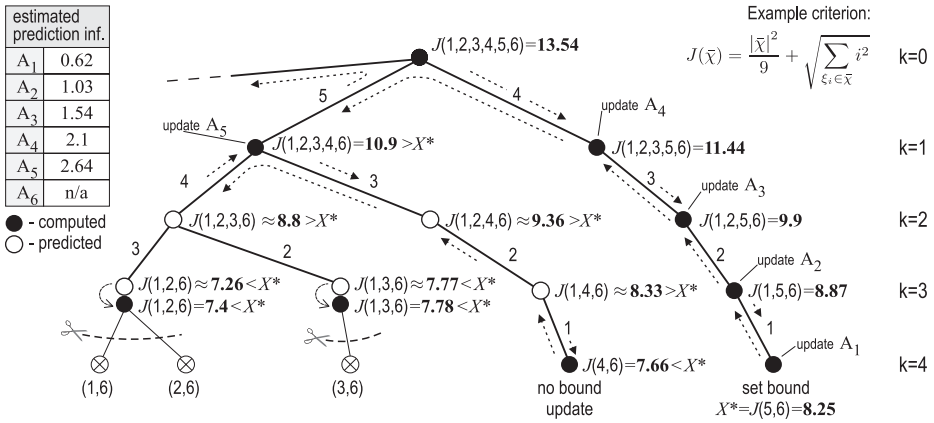

Fig. 1. Illustration of the prediction mechanism in Fast Branch & Bound for a problem of selecting $d = 2$ features out of $D = 6$ to maximise a synthetic criterion function.

See Fig. 1 for an illustration of the search process. The Figure shows initial stages of the search process on a synthetic problem where $d = 2$, $D = 6$. The prediction mechanism learns whenever two subsequent criterion values are computed (here for simplicity $A_i = J(\bar{\chi}) - J(\bar{\chi} \setminus \{i\})$ for $i = 4, 3, 2, 1, 5$) and later uses this information to replace criterion evaluation by a simple subtraction (in white nodes $J(\bar{\chi} \setminus \{i\}) \approx J(\bar{\chi}) - A_i$). The predicted values, being only approximations of the true criterion values, do not suffice to cut-off sub-trees 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.

2.1 Fast Branch & Bound with the default prediction mechanism

The FBB *default prediction mechanism* is based on averaging feature contributions individually for each feature, independently on current tree level k. Averages are kept in 'contribution' vector $\mathbf{A} = [A_1, A_2, \ldots, A_D]^{\mathrm{T}}$, while the number

of averaged values is stored in 'counter' vector $\mathbf{S} = [S_1, S_2, \ldots, S_D]^{\mathrm{T}}$. Both vectors are initially zeroed. Whenever FBB 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 k-th tree level, and if also 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 been computed (as indicated by T_{k-1,y_j} ="C"), the prediction mechanism vectors are updated as follows:

$$A_{y_i} = \frac{A_{y_i} \cdot S_{y_i} + J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})}{S_{y_i} + 1}, \quad S_{y_i} = S_{y_i} + 1 \tag{2}$$

For the formal FBB description we shall use the notion adopted from [1]:

$\bar{\chi}_k = \{\xi_j \mid j = 1, 2, \cdots, D-k\}$ – current "candidate" set at k-th 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 $\mathcal{X}$).

$\delta \geq 1$ – *minimum number of evaluations, by default*= 1,

$\gamma \geq 0$ – *optimism, by default*= 1,

$\mathbf{T}_k = [T_{k,1}, T_{k,2}, \ldots, T_{k,q_k}]^{\mathrm{T}}$, $T_{k,i} \in \{$"C","P"$\}$ for $i = 1, \cdots, q_k$ – criterion value *type vector* (records the type of $J_{k,i}$ values–computed or predicted),

$\mathbf{V} = [v_1, v_2, \ldots, v_{q_k}]^{\mathrm{T}}$ – 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.

The Fast Branch & Bound Algorithm

Initialization: $k = 0$, $\bar{\chi}_0 = Y$, $\Psi = Y$, $r = D$, $\delta = 1$, $\gamma = 1$, $X^* = -\infty$.

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 leafs) 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, **else** (nodes are leafs or prediction not allowed) 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 all $i = 1, \cdots, q_k$:

set $Q_{k,i} = \psi_{j_i}$ and

if v_{j_i} records a computed value, then set $J_{k,i} = v_{j_i}$ and $T_{k,i}$ ="C"
else set $J_{k,i} = J_{k-1,q_{k-1}} - \gamma \cdot A_{\psi_{j_i}}$ and $T_{k,i}$ ="P".
To avoid duplicate testing set $\Psi = \bar{\Psi} \setminus \mathcal{Q}_k$ 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 were tested and go to **Step 4** (backtracking). If T_{k,q_k} ="P" and $J_{k,q_k} < X^*$, then 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**, else go to next level: let $k = k + 1$ and go to **Step 1**.
STEP 3: *Descendant node connected by the Q_{k,q_k}-edge (and its sub-tree) 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 neighbour; go to **Step 2**.
STEP 4: *Backtracking:* Let $k = k - 1$. If $k = -1$, then the complete tree had been searched through; stop the algorithm, else 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 Prediction Mechanisms

Here we define a set of alternative prediction mechanisms. Technically we change only the A_{y_i} estimation, i.e. formula (2). The estimation takes place under the same conditions as described in the preceeding section. The use of the alternative A_{y_i} values inside FBB instead of the default is principally the same, taking place in **STEP 1** only. For a list of defined mechanisms see Table 1.

The simplest *last-value* mechanism directly re-uses only the last computed contribution value. It is based on assumption that feature behaviour does not change too dramatically in local context.

The *level-based averaging* predictor should reduce the impact of *criterion value decrease* estimation errors with criterion functions yielding values strongly dependent on feature set size. As the estimation takes place separately for each tree level, this predictor may become compromised by the delay of prediction start and by the relatively lower number of true values available for learning when compared to the default, global averaging predictor.

The *maximising* and *minimising* predictors are likely to be outperformed by the others as they obviously yield biased predictions. Their purpose is to test the FBB vulnerability to "optimistic" (in case of *minimising*) and "pessimistic" (in case of *maximising*) errors caused by the predictor. The $A_{y_i}^{Max}$ and $A_{y_i}^{Min}$ values are expected here to cause similar effect as setting the optimism parameter γ either > 1 or < 1. In case of too pessimistic behaviour the accelerating effect of prediction mechanism on the FBB algorithm as a whole deteriorates, but the maximum number of search tree nodes remains equal to that of the IBB. In case of too optimistic behaviour the FBB algorithm can unwantedly track the tree

Description	Definition	Comment
averaging	formula (2)	Default.
last-value	$A^{L}_{y_i} = J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})$	Uses only the last value.
maximising	Let $A^{Max}_{y_i} = J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})$ only if $A^{Max}_{y_i} < J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})$	Uses the maximum value obtained so-far.
minimising	Let $A^{Min}_{y_i} = J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})$ only if $A^{Min}_{y_i} > J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})$	Uses the minimum value obtained so-far.
(max+min)/2	$A^{Mid}_{y_i} = (A^{Max}_{y_i} + A^{Min}_{y_i})/2$	"Middle" value.
level-based averaging	$A^{Lev}_{y_i,k} = \frac{A^{Lev}_{y_i,k} \cdot S_{y_i} + J_{k-1,y_j} - J(\bar{\chi}_k \setminus \{y_i\})}{S_{y_i}+1}$	Same as default, but separately for each subset size.
individual	$A^{Ind}_{y_i} = J(\{y_i\})$	Constant predictor.
reverse individual	$A^{Rev}_{y_i} = J(Y) - J(Y \setminus \{y_i\})$	Constant predictor.

Table 1. List of considered prediction mechanisms.

branches deeper than IBB what could result in worse performance loss then in the pessimistic case (for details see [6]). The *(max+min)/2* value is used as an alternative predictor as well.

The *individual* value predictor uses constant individual criterion values for each feature. It is defined to demonstrate the fact that individual feature evaluation often is not sufficient to estimate the value of feature sets. The *reverse individual* predictor analogously uses only the constant individual feature contributions with respect to the full set Y.

4 Experiments

The different predictors in the FBB algorithm were tested on a number of different data sets. Here we show results computed on 2-class mammogram Wisconsin Diagnostic Breast Center (WDBC) data (30 features, 357 benign and 212 malignant samples) and WAVEFORM data (40 features of which 19 represent noise, 1692 class 1 and 1653 class 2 samples) obtained via the UCI repository (ftp.ics.uci.edu)and 2-class SPEECH data originating at British Telecom (15 features, 682 word "yes" and 736 word "no" samples). We used the Bhattacharyya, Divergence and Patrick-Fischer distances. The Patrick-Fischer distance is considered difficult for use in B&B because of its strong dependence on evaluated set size. We used Pentium4-2,6Ghz CPU for all tests. As all the algorithms are optimal, they yield identical subsets identified by the same maximum criterion value. The only important difference between considered algorithms is therefore in computational time or in number of true criterion evaluations. Due to limited space we present only the graphs of computational time. It should be only noted, that practically all experiments proved a straightforward dependence between the number of criterion evaluations and overall time. However, it is difficult to describe this dependence precisely as the number of evaluations does not depend on criterion computational complexity while the overall computational time does.

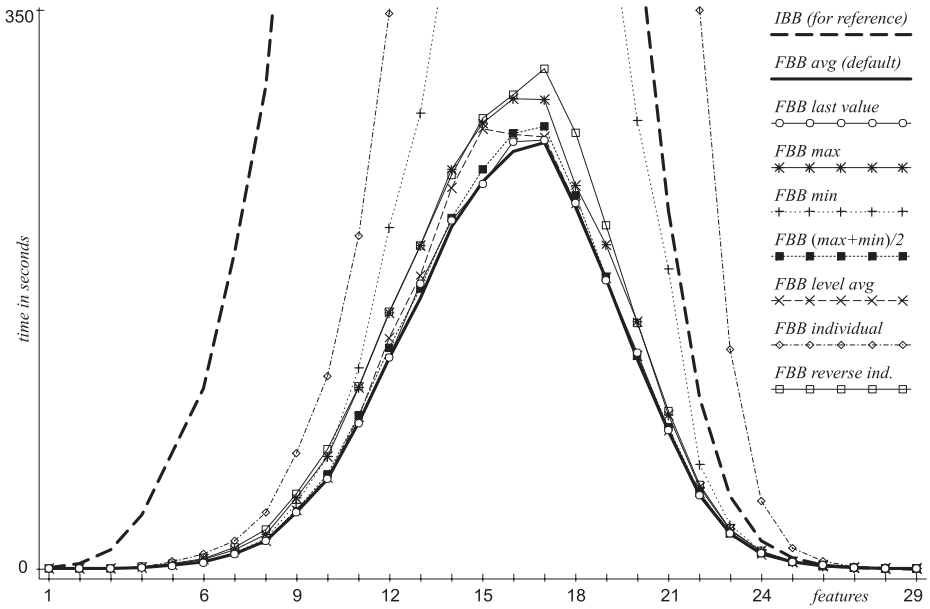

Fig. 2. FBB prediction efficiency – Bhattacharyya distance, WDBC data.

The results in Figures 2,3 and 4 show that the default *averaging* mechanism in FBB remains the best choice for general purpose. It markedly outperforms the referential IBB in all cases. Figures 2 and 3 show unexpectedly good performance of the simplest *last-value* predictor that becomes the fastest one in isolated cases. However, it becomes totally compromised in combination with the Patrick-Fischer criterion (Fig. 4). This illustrates the limits of using local information for generalization depending on criterion properties. The *level-based averaging* predictor performs comparably to the default *averaging*. It performs slightly worse in easier cases (Figs. 2,3) and slightly better in the difficult case (Fig. 4). It can be expected that the overall *level-based averaging* performance would improve with increasing problem sizes where more data becomes available for learning. A better than expected performance is observed for the *(max+min)/2* predictor that proves to be sufficiently good in simpler cases and excellent in the difficult case. It becomes a meaningful alternative to the default *averaging*, showing a good generalization ability.

The performance of the *maximising* predictor in Figs. 2 and 3 is noticeably worse than that of the default *averaging* but better in Fig. 4. This is the result of the invoked "pessimistic" algorithm behaviour, which slows-down the search in easy cases but helps to reduce the negative effect of "optimistic" errors if the learning process is compromised by noise in data or criterion properties. Similar or worse behaviour can be observed for the *reverse individual* predictor. The *minimising* and *individual* value predictors are shown to have a strongly negative impact on FBB performance. In case of the *minimising* predictor it confirms

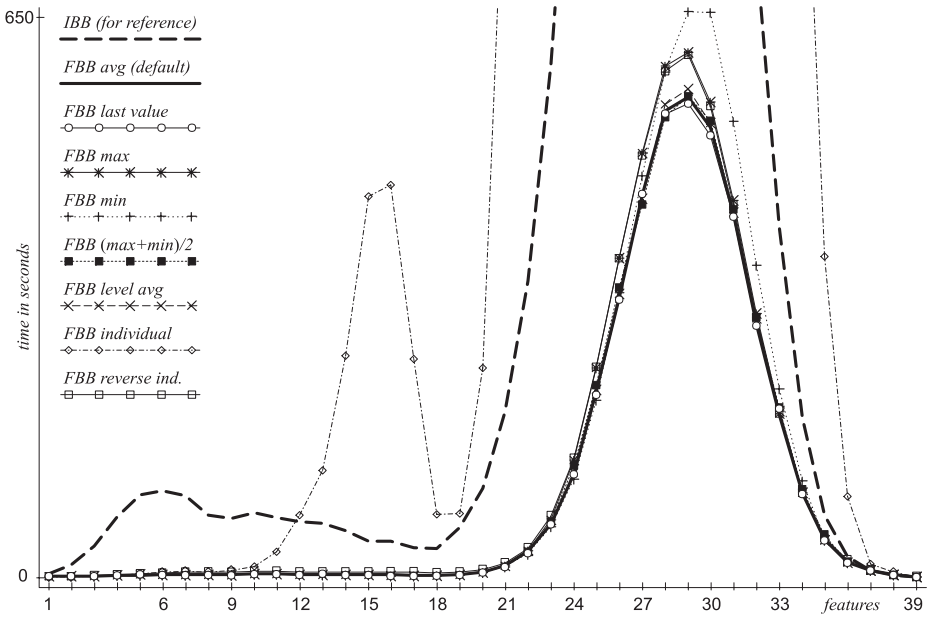

Fig. 3. FBB prediction efficiency – Divergence, WAVEFORM data.

the assumption that "optimistic" algorithm bahaviour (tracking branches deeper than necessary) can strongly deteriorate the FBB performance. The weak *individual* value predictor performance confirms that individual feature importance does not represent well its importance with respect to other features in a set. Remark: A more detailed study of additional aspects affecting the general B&B performance can be found in [8].

5 Conclusion

We have discussed in detail the recent Fast Branch & Bound optimal feature selection algorithm with respect to its core concept of prediction mechanism. Alternative predictors have been defined and investigated. The original *averaging* prediction mechanism has been verified to be the good option for general purpose. However, the simple *last-value* predictor shows to perform equally well for some criteria while being simpler to implement and requiring less computational overhead. The *level-based averaging* predictor is to be recommended especially for high-dimensional tasks and/or criteria where the feature contributions are known to be subset-size dependent. The *(max+min)/2* predictor has proved to be worth consideration as an alternative for general purpose. Regardless the differences between these prediction mechanisms the performance of the respective FBB algorithm versions is generally better than that of the older optimal search algorithms like the Improved Branch & Bound. This demonstrates the robustness of the prediction mechanism concept as such.

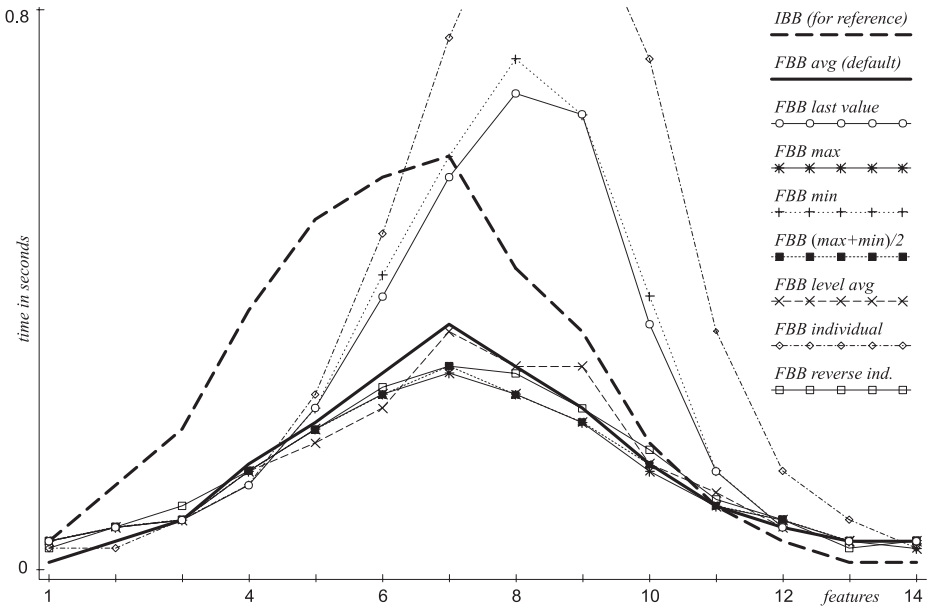

Fig. 4. FBB prediction efficiency – Patrick-Fischer distance, SPEECH data.

References

1. P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.
2. K. Fukunaga. *Introduction to Statistical Pattern Recognition: 2nd edition*. Academic Press, Inc., 1990.
3. Y. Hamamoto, S. Uchimura, Y. Matsuura, T. Kanaoka and S. Tomita. Evaluation of the branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 11(7): 453–456, July 1990.
4. M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(I): 25–41, January 2000.
5. P.M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26: 917–922, September 1977.
6. P. Somol, P. Pudil, F. J.Ferri and J. Kittler. Fast Branch & Bound Algorithm in Feature Selection. *Proc. 4th World Multiconference on Systemics, Cybernetics and Informatics SCI 2000, Orlando, Florida*, Vol VII, Part 1: 646–651, 2000.
7. 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*, Springer Verlag, Vol. 2013, 230–238, 2001.
8. P. Somol, P. Pudil, and J. Kittler. Fast Branch & Bound Algorithms in Feature Selection. *To appear in IEEE Transactions on PAMI*, July 2004.
9. B. Yu and B. Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, 26: 883–889, 1993.