Alcom-FT Technical Report Series
ALCOMFT-TR-02-79

Branch and bound algorithms for the minimum test set problem

K.M.J. De Bontridder* J.K. Lenstra† J.B. Orlin‡
L. Stougie†§

Abstract

In the minimum test set problem a set of items is given together with a collection of subsets of the items, called tests. A smallest set of tests is to be selected such that for every pair of items there is a test in the selection that contains exactly one of the two items. This problem is NP-hard in general. The problem has important applications in biology, pharmacy, and the medical sciences, as well as in coding theory.

We develop a variety of branch and bound algorithms to solve this problem exactly. The variety is in the definition of the branching rules and the lower bounds to prune the search tree. The emerging algorithms are compared both theoretically and empirically.

1 Prelude

The input of the *minimum test set problem* (MTS) consists of a set of *items*, $\{1, ..., n\}$, and a collection $\mathcal{S}$ of *tests*, $T_1, ...T_m \subset \{1, ..., n\}$. A test T_l covers (differentiates) item pair $\{i, j\}$ if either $i \in T_l$ or $j \in T_l$ (but not both), i.e., $|T_l \cap \{i, j\}| = 1$. A family $\mathcal{T} \subset \mathcal{S}$ of tests is a *test set* if each of the $n(n-1)/2$ item pairs is covered by at least one test in $\mathcal{T}$. The problem is to find the test set of minimum cardinality τ^*.

The minimum test set problem arises naturally in the following general setting of identification problems: Given a set of individuals (database entries) and a set of binary attributes that may or may not occur in each individual, the goal

*Institute of Information and Computing Sciences, Universiteit Utrecht, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. e-mail: koendb@cs.uu.nl. Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

†Department of Mathematics, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, The Netherlands. e-mail: {jkl,leen}@win.tue.nl.

‡Department of Operations Research, Massachussets Institute of Technology, Cambridge, Massachusetts, USA. e-mail: orlin@sloan.mit.edu.

§CWI, P.O. Box 94079, 1090GB Amsterdam, The Netherlands. e-mail: stougie@cwi.nl.

is to find the minimal subset of attributes (a test set) such that each individual can be identified uniquely from the information the test set contains. In this way the incidence vector of any individual with the test set is a unique binary signature, which distinguishes the individual from any other individual.

In fact, the motivation for this research dates over 20 years back through a request from the Wageningen University of Agriculture in The Netherlands concerning investigations of potato diseases. Each potato race is vulnerable for a number of diseases. In order to investigate the effect of special treatments on the diseases a selection of potato races had to be made such that any pair of diseases could be distinguished from one another.

There are numerous other important applications in biology, pharmacy, and the medical sciences (see e.g. [Moret & Shapiro 1985] and [De Bontridder et al. 2001]), as well as in coding theory (see [De Bontridder 1997]).

The problem is NP-hard [Garey & Johnson 1979]. It remains NP-hard if the number of items in each test is at most 2 [De Bontridder et al. 2002]. Approximation algorithms for the problem are studied in [Moret & Shapiro 1985] and [De Bontridder et al. 2002].

The main part of [Moret & Shapiro 1985] concerns empirical analysis of branch and bound algorithms for MTS. We use the same basic branch and bound scheme: The nodes of the search tree are defined by partial test sets, which are sets of tests that do not differentiate all item pairs. A child of a node is obtained by adding a test to the partial test set of the node. There are several criteria on which to decide which test to add given the partial test set. Given a partial test set various lower bounds can be defined on the number of tests that need to be added to obtain a test set. The lower bounds are used in combination with the size of a feasible solution found to prune the search tree.

The difference with [Moret & Shapiro 1985] is in the different branching rules and lower bounds employed for pruning the search tree. Moreover, we do not only compare the various resulting branch and bound algorithms empirically, as was done in [Moret & Shapiro 1985], but we also make a theoretical comparison, based on complexity and for the lower bounds on pruning power.

The common ideas behind the algorithms are exposed in Section 2. The various branching rules are presented precisely in Section 3. Among them some have appeared in [Moret & Shapiro 1985] and some are new. Various lower bounds are presented in Section 4, again some of them new and some of them having appeared in [Moret & Shapiro 1985]. In Section 4 we compare the lower bounds theoretically on their pruning power and the time needed to compute.

From among the variety of branch and bound algorithms that can be constructed by combining branching rules and pruning criteria we make a selection in Section 5. We subject this selection in Section 6 to empirical performance analysis. As test problems we take a real-life problem and a set of random instances. The analysis shows a clear picture in favor of one of the algorithms. We compare our results with the conclusions in [Moret & Shapiro 1985]. The results are very encouraging for solving larger size practical problems.

Throughout the paper we use log shortly to denote $\log_2$.

2 Partial test sets and equivalence classes

For solving minimum test set problems we device branch and bound algorithms, in which the internal nodes of the search tree correspond to partial test sets and the leaves to test sets, the root being the empty set. The child of an internal node is defined by adding a test to the partial test set of that node, and discarding some other tests.

A partial test set is a family of tests that does not differentiate each item pair. Any partial test set defines an *equivalence relation* on the set of items.

Definition 2.1 Given a partial test set two items are equivalent if there is no test that differentiates them.

The equivalence relation corresponding to a partial test set $\mathcal{T} \subset \mathcal{S}$ induces a partition of the set of all items into subsets $I_1^{\mathcal{T}}, I_2^{\mathcal{T}}, \cdots, I_t^{\mathcal{T}}$, for some t, such that, for each subset, any two items in it are not differentiated by any test in $\mathcal{T}$. Each item can be in only one set, for if item i_1 is not differentiated from item i_2, and item i_2 is not differentiated from item i_3, then i_1 cannot be differentiated from i_3 either. Regarded in this way, a test set is a set of tests that defines an equivalence relation such that the induced partition consists of exactly n singleton sets.

In the next sections we use the equivalence relations to define for each node branching rules and criteria to prune the tree in that node. The various branching rules and pruning criteria that we define lead to various algorithms. However, they all start with selecting the so-called *essential tests*.

Definition 2.2 A test T is essential if there exists an item pair, for which T is the only test in $\mathcal{S}$ that differentiates the two items.

Therefore we will assume from now on that the root of the search tree includes all essential tests. Notice that the essential tests can be found in $O(n^2m)$ time.

3 Quality criteria and branching rules

In this section we will give various criteria to decide on the quality of tests in nodes of the search tree, i.e., given a partial test set has already been selected. We will use these criteria in designing branching schemes and lower bounds for pruning the search tree. As already mentioned before, we may assume that all essential tests have been included in each node of the search tree.

Let us consider a node represented by the pair $(\mathcal{T}, \mathcal{R})$, with $\mathcal{T}$ the partial test set and $\mathcal{R}$ the set of discarded tests. Let $I_1^{\mathcal{T}}, I_2^{\mathcal{T}}, \cdots, I_t^{\mathcal{T}}$ be the corresponding partition of the items. $\mathcal{T}$ leaves $\sum_{i=1}^{t} |I_i^{\mathcal{T}}|(|I_i^{\mathcal{T}}|+1)/2$ pairs of items undifferentiated. Updating the equivalence classes after adding a test to a partial test set requires $O(n)$ time, in the most straightforward way.

A test $T \notin \mathcal{T}$ decreases the number of undifferentiated item pairs instantaneously by

$$D(T|\mathcal{T}) = \sum_{h=1}^{t} |T \cap I_h^{\mathcal{T}}||I_h^{\mathcal{T}} \backslash T|. \tag{1}$$

This leads to the first quality criterion called the *separation criterion* in [Moret & Shapiro 1985].

Another criterion called the *information criterion* in [Moret & Shapiro 1985], was introduced (in a different form) already in 1961 [Rescigno & Maccacaro 1961], and in the way stated here in [Chang et la. 1970] (see also [Moret 1982]).

Definition 3.1 A partial test set $\mathcal{T}$ has *information value*

$$E(\mathcal{T}) = \log n - \frac{1}{n} \sum_{h=1}^{t} |I_h^{\mathcal{T}}| \log |I_h^{\mathcal{T}}|.$$

Notice that $E(\mathcal{T}) = 0$ if $\mathcal{T} = \emptyset$ and $E(\mathcal{T}) = \log n$ if $\mathcal{T}$ is a test set. Notice also that E is a monotone function: $E(\mathcal{T}') \leq E(\mathcal{T})$ for any $\mathcal{T}' \subset \mathcal{T}$. Addition of a test T to $\mathcal{T}$ will give an increase in information value of

$$\Delta E(T|\mathcal{T}) = E(\mathcal{T} \cup T) - E(\mathcal{T}), \tag{2}$$

which defines the quality criterion.

We add here a new quality criterion for a test given a partial test set, which we call *power of a test*.

Definition 3.2 Given a partial test set $\mathcal{T}$ the power of a test $T \notin \mathcal{T}$, is defined by

$$P(T|\mathcal{T}) = \sum_{h=1}^{t} \min\{|T \cap I_h^{\mathcal{T}}|, |I_h^{\mathcal{T}} \backslash T|\}.$$

A final criterion, also mentioned in [Moret & Shapiro 1985], is the so-called *least-separated pair* criterion and was applied first before already by microbiologists [Rypka et al. 1967], [Rypka et al. 1978], [Payne 1981].

Definition 3.3 Given a partial test set $\mathcal{T}$, item pair $\{i, j\}$ is least-separated if amongst the yet undifferentiated item pairs it is the one for which the minimum number of tests is available that differentiates the pair. Denoting the set of least-separated item pairs, given $\mathcal{T}$ by $I_S(\mathcal{T})$,

$$S(T|\mathcal{T}) = \begin{cases} 1 \text{ if } |T \cup \{i, j\}| = 1; \\ 0 \text{ otherwise.} \end{cases}$$

In case there is not a unique least-separated item pair we choose one of them arbitrarily. [Moret & Shapiro 1985] uses the name "nearly essential tests" to clarify the notion of this criterion: at least one of these tests must be contained in a test set. We follow [Moret & Shapiro 1985] by using this criterion only in combination with any of the previous three for breaking the many ties that application of this criterion will meet.

Each of the quality criteria can be incorporated in the following general branching scheme. For $C \in \{D, \Delta E, P, S\}$,

- Given a node $(\mathcal{T}, \mathcal{R})$, order the tests $T \notin \mathcal{T} \cup \mathcal{R}$, according to non-increasing $C(T|\mathcal{T})$ value: $C(T_1|\mathcal{T})) \geq C(T_2|\mathcal{T}) \geq \cdots \geq C(T_l|\mathcal{T})$, with l the number of tests not in $\mathcal{T} \cup \mathcal{R}$. Create the children $(\mathcal{T} \cup T_1, \mathcal{R})$, $(\mathcal{T} \cup T_2, \mathcal{R} \cup T_1)$, $(\mathcal{T} \cup T_3, \mathcal{R} \cup T_1 \cup T_2)$, $\cdots$, $(\mathcal{T} \cup T_l, \mathcal{R} \cup_{h=1}^{l-1} T_h)$, and explore the children in this order.

Notice that each of the first three branching schemes requires $O(mn \log n)$ computation time in each node of the search tree. The least-separated pair criterion takes less time but will always be combined with one of the other three.

4 Lower Bounds for pruning

We will use the quality criteria from the previous section here to define pruning criteria for the nodes of the search tree. Again consider node $(\mathcal{T}, \mathcal{R})$ with corresponding partition of the items $I_1^{\mathcal{T}}, I_2^{\mathcal{T}}, \cdots, I_t^{\mathcal{T}}$. We define four lower bounds on the number of tests that have to be added to $\mathcal{T}$ to obtain a test set. For each of the lower bounds we also give the complexity of computing them.

4.1 Lower bound by ideal tests

The first one, which we denote $L_1(\mathcal{T})$, is straightforward, and, as notation suggests, independent of the discarded tests. It assumes the existence of *ideal tests* or *even splitting tests*, regardless of the availability of such tests in the instance. Given any partition of the items in equivalence classes, such a test contains half of the items in each of the equivalence classes.

Lemma 4.1 *The number of ideal tests that have to be added to partial test set $\mathcal{T}$ in order to obtain a test set is bounded from below by*

$$L_1(\mathcal{T}) = \lceil \log(\max_{h=1,\cdots,t} I_h^{\mathcal{T}}) \rceil.$$

PROOF. Given a set of q items, defining $\frac{1}{2}q(q-1)$ item pairs, any ideal test differentiates $\lfloor \frac{1}{4}q^2 \rfloor$ of the item pairs. From this observation the lemma follows easily. $\square$

Computing this lower bound has time complexity $O(n)$ in each node of the search tree. This can be improved by using a more sophisticated data structure for storing the sizes of the equivalence classes.

4.2 Lower bounds by power

The lower bound $L_1(\mathcal{T})$ does not use any specific information about the tests available in the problem instance to supplement $\mathcal{T}$ to a test set. Indeed, ideal tests may not be available at all in the node. The following lower bounds do use specific problem instance and node information. They are based on the power $P(T|\mathcal{T})$ of a test T. Clearly, $1 \leq P(T|\mathcal{T}) \leq n/2$, for any $\mathcal{T}$ and any $T \notin \mathcal{T}$. Moreover, notice that $P(T|\mathcal{T}) \geq P(T|\mathcal{T}')$ for any $\mathcal{T} \subset \mathcal{T}'$ and any $T \notin \mathcal{T}'$. Given node $(\mathcal{T}, \mathcal{R})$, we order the l tests not in $\mathcal{T} \cup \mathcal{R}$ according to non-increasing power, $P(T_1|\mathcal{T}) \geq P(T_2|\mathcal{T}) \geq \cdots \geq P(T_l|\mathcal{T})$.

Let $F(n,m)$ denote the minimum total power that m tests need to cover n items. Then, given a partial test cover $\mathcal{T}$, with resulting equivalence classes $I_1^{\mathcal{T}}, I_2^{\mathcal{T}}, \cdots, I_t^{\mathcal{T}}$, a number k of tests has to be selected with total power at least $\sum_{h=1}^{t} F(|I_h^{\mathcal{T}}|, k)$. Using the power ordered tests not in $\mathcal{T} \cup \mathcal{R}$, and assuming that they their cumulative effect is the sum of their individual effects on differentiating item pairs, we obtain our second lower bound on the number of tests to be added to $\mathcal{T}$.

$$L_2(\mathcal{T}, \mathcal{R}) = \min\{k | \sum_{j=1}^{k} P(T_j|\mathcal{T}) \geq \sum_{h=1}^{t} F(|I_h^{\mathcal{T}}|, k)\}.$$

Through a smart dynamic programming recursion the list of values for $F(h,k)$, $h = 1, \ldots, n$, $k = 1, \ldots, m$, can be computed in $O(nm)$ time. We omit the description and its correctness proof in this extended abstract. Moreover, notice that the F-values are instance independent. Thus, such a list needs to be computed once and for all.

Having the list of F-values and the powers of tests in an ordered list, given a partial test set, computing $L_2(\mathcal{T}, \mathcal{R})$ requires $O(n)$ time.

In fact, we can sharpen this lower bound by considering subsets of the set of equivalence classes. Let $H \subset \{1, \cdots, t\}$ and $I_H^{\mathcal{T}} = \cup_{h \in H} I_h^{\mathcal{T}}$. Define the power of T relative to $I_H^{\mathcal{T}}$ as $P_H(T|\mathcal{T}) = \sum_{h \in H} \min\{|T \cap I_h^{\mathcal{T}}|, |I_h^{\mathcal{T}} \setminus T|\}$. For each subset $I_H^{\mathcal{T}}$ a lower bound like $L_2(\mathcal{T}, \mathcal{R})$ must hold, i.e., given an ordering of the tests not in $\mathcal{T} \cup \mathcal{R}$ according to non-increasing power relative to $I_H^{\mathcal{T}}$ $(P_H(T_1|\mathcal{T}) \geq \cdots \geq P_H(T_l|\mathcal{T}))$,

$$L_2^H(\mathcal{T}, \mathcal{R}) = \min\{k | \sum_{j=1}^{k} P_H(T_j|\mathcal{T}) \geq \sum_{h \in H} F(|I_h^{\mathcal{T}}|, k)\}$$

is also a lower bound on the number of tests to be added to $\mathcal{T}$. Taking the maximum over all possible subsets H yields our third lower bound:

$$L_3(\mathcal{T}, \mathcal{R}) = \max_{H \subset \{1, \cdots, t\}} L_2^H(\mathcal{T}, \mathcal{R}).$$

We do not know the complexity of computing $L_3(\mathcal{T}, \mathcal{R})$ given the list of F-values.

The bounds described so far have been presented in increasing quality.

Lemma 4.2 $L_1(\mathcal{T}) \leq L_2(\mathcal{T},\mathcal{R}) \leq L_3(\mathcal{T},\mathcal{R})$.

PROOF. The inequalities are direct from the definitions. $\square$

4.3 Lower bounds by information value

The last lower bound was considered in [Moret & Shapiro 1985]) and is based on the information value of a partial test set. Given a node $(\mathcal{T},\mathcal{R})$, order the l tests not in $\mathcal{T}\cup\mathcal{R}$ as $\Delta E(T_1|\mathcal{T}) \geq \Delta E(T_2|\mathcal{T}) \geq \cdots \geq \Delta E(T_l|\mathcal{T})$.

$$L_4(\mathcal{T},\mathcal{R}) = \min\{k | E(\mathcal{T}) + \sum_{j=1}^{k} \Delta E(T_j|\mathcal{T}) \geq \log m\}$$

is a lower bound on the number of tests needed to raise the information value from $E(\mathcal{T})$ to $\log n$. That L_4 is indeed a lower bound follows directly from the observation that $\Delta E(T|\mathcal{T}) \geq \Delta E(T|\mathcal{T}')$ for any $\mathcal{T} \subset \mathcal{T}'$ and any $T \notin \mathcal{T}'$. The latter observation is mentioned but not proved in [Moret & Shapiro 1985] and indeed is quite intuitive: two or more tests may contain so-called cross-information by differentiating some equal item pairs. A formal proof is given in [De Bontridder 1997]. Given the ordering of the tests on ΔE-value the computation of this lower bound requires $O(n)$ time.

Lemma 4.3 *$L_4(\mathcal{T},\mathcal{R})$ neither dominates $L_3(\mathcal{T},\mathcal{R})$ nor is dominated by $L_1(\mathcal{T})$.*

PROOF. The proof is by examples on which the statement holds. In the first example contains 8 items and 4 tests. $\mathcal{T} = \emptyset$

	$I_1^{\mathcal{T}}$
T_1	11110000
T_2	00110011
T_3	10001000
T_4	00000000

In this example $L_3(\emptyset,\emptyset) = 4$ and $L_4(\emptyset,\emptyset) = \infty$.

The second example contains 24 items and 4 tests. $\mathcal{T}$ defines an iduces partition of the items into 5 subsets.

	$I_1^{\mathcal{T}}$	$I_2^{\mathcal{T}}$	$I_3^{\mathcal{T}}$	$I_4^{\mathcal{T}}$	$I_5^{\mathcal{T}}$
T_1	10	10	10	01	1111111100000000
T_2	01	01	10	10	0011001100110011
T_3	10	01	10	10	1111000011110000
T_4	01	10	10	10	0101010101010101

In this example $L_1(\mathcal{T}) = 4$ and $LB_4(\mathcal{T},\emptyset) = 3$ $\square$

4.4 Pruning criteria

All lower bounds can be used in the following two criteria to prune nodes: Given node $(\mathcal{T}, \mathcal{R})$, lower bound L, and U, the size of the best solution found so far,

- Prune the node if
  (PC1) $L \leq |\mathcal{S}| - |\mathcal{T}| - |\mathcal{R}|$, or
  (PC2) $|\mathcal{T}| + L \geq U$.

The second criterion prunes because descendants of the node cannot yield a better solution then the current best one. The first criterion prunes because the node leaves an infeasible instance.

Another pruning criterion to cut off infeasible nodes is

- Prune the node if
  (PC3) A pair of items exists, undifferentiated by $\mathcal{T}$, for which no test exists in $\mathcal{S} \backslash (\mathcal{T} \cup \mathcal{R})$ that differentiates them.

The last criterion that we use prunes again a node for being inferior to the best found solution. The advantage of this criterion is that no ordering of tests is required. It relies on the following lemma in which we use the notation $P_I(T|\mathcal{T}) = \min\{|I \cap T|, |I \backslash T|\}$

Lemma 4.4 *To cover a set I of $2^n - i$ items, $1 \leq i \leq 2^{n-1} - 1$, with n tests, $T_1, \ldots, T_n$, each of the tests needs to have $P_I(T_j|\emptyset) \geq 2^{n-1} - i$, $j = 1, \ldots, n$*

PROOF. Suppose the theorem is not true, then among the n tests there is at least one, test T say, with $P_I(T|\emptyset) < 2^{n-1} - i$. After selection of this test I splits into I_1 and I_2, with $\max\{|I_1|, |I_2|\} > \max\{i, 2^n - i - 2^{n-1} + 1\} > 2^{n-1}$. Clearly, the remaining $n - 1$ tests are not able to cover a set of more than 2^{n-1} items. □

- Prune the node if, for $L \in \{L_1, L_2, L_3, L_4\}$,
  (PC4) Given $L(\mathcal{T}) = \lceil \log |I_k^{\mathcal{T}}| \rceil$, $L(\mathcal{T}) = U - |\mathcal{T}| - 1$ and there do not exist $L(\mathcal{T})$ tests T with $P_{I_k^{\mathcal{T}}}(T|\mathcal{T}) \geq |I_k^{\mathcal{T}}| - 2^{L(\mathcal{T})-1}$

5 A variety of branch-and-bound algorithms

In the previous sections we have presented branching rules and bounding rules as basic ingredients for branch-and-bound algorithms. We have not specified yet a searching strategy. We have chosen here for the depth first search strategy in order to keep storage of intermediate solutions as limited as possible. Since, each node can have quite a lot of children, by the branching rules from Section 3, the number of intermediate solutions in a breadth first search would become already very large on shallow trees. Depth first search has also the well-known advantage of finding feasible solutions rather quickly, which can then be used in defining upper bounds for pruning the search tree. The depth first search strategy together with the branching and pruning rules give rise to quite a variety of branch-and-bound algorithms. From among all of them we made a selection for testing empirically.

In all of the algorithms we implemented some preprocessing. As mentioned already we add all *essential tests* directly to the root of the search tree. However, also halfway on paths from the root to the leaves nodes may occur with a partial test set and a set of discarded tests such that only one test remains that covers a yet undifferentiated item pair. In that case we add this *locally essential test* to the node's partial test set, before we apply any other branching or bounding to that node. This preprocessing automatically checks pruning criterion PC3.

The selected branch-and-bound algorithms selected are listed below. They are defined by a triple $\{C, L, x\}$. C defines on which of the quality criteria the branching rule is based: $C \in \{D, P, \Delta E, S(*)\}$ (see Section 3), where $*$ is any of $\{D, P, \Delta E\}$ and $S(*)$ indicates the least-separation criterion combined with $*$ for breaking ties. L defines the type of lower bound used in pruning criteria PC1 and PC2 (see Section 4.4): $L \in \{L_1, L_2, L_3, L_4\}$ (see Sections 4.1, 4.2, and 4.3). Finally, $x \in \{Y, N\}$ indicates wether Y or not N pruning criterion PC4 is applied (see Section 4.4). In PC4, if applied, the same lower bound is used as the one for L in $\{C, L, Y\}$. We have selected the following algorithms: (P, L_2, N), (D, L_2, N), $(S(P), L_2, N)$, $(S(D), L_2, N)$, (P, L_3, N), (D, L_4, N), $(S(P), L_4, N)$, $(\Delta E, L_4, N)$, $(S(\Delta E), L_4, N)$, (P, L_2, Y), $(S(P), L_2, Y)$, and $(S(D), L_2, Y)$. In the figures depicting the test results one should read L_k for Lk, $k = 1, 2, 3, 4$ and ΔE for E.

6 Empirical performance analysis

We have investigated the empirical performance of the algorithms defined above by applying them to one real world example from the potato industry, and 140 artificial examples. The potato instance consists of 28 tests and 63 items. The artificial instances are randomly generated and can be characterized by the triple (m,n,p). The parameters m, n, and p denote the number of tests, the number of items, and the probability that an item belongs to a test, respectively. Probability $p \in \{0.1, 0.25, 0.5\}$ is equal for each item and each test. We have chosen $n \in \{25, 50, 100\}$ and $m \in \{25, 50, 100\}$, and generated instances with (50,n,p) for all three values of n and p, and (m,50,p), for all three values of m and p. For each triple we generated 10 feasible instances. Only for the triple $(25, 50, 0.1)$ we were not able to find such an instance. The experiments were performed on a Sun Ultra 10 Creator3D workstation and all computation times are measured in cpu seconds. Below the performance of the various algorithms is compared and conclusions on the relative performance are drawn.

We started with comparing nine algorithms that do not use pruning criterion PC4 on the instances with 50 tests and 50 items. For the three probabilities the number of branches and the average computation time are depicted and Figure 1, respectively. The results clearly show that on these instances, using

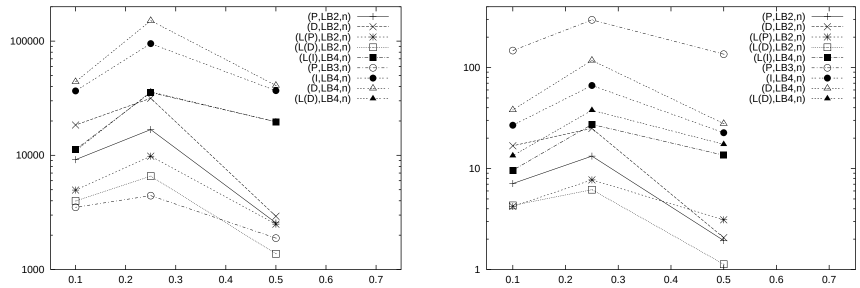

Figure 1: Average number of branches and average computation time of nine algorithms (logarithmic scale).

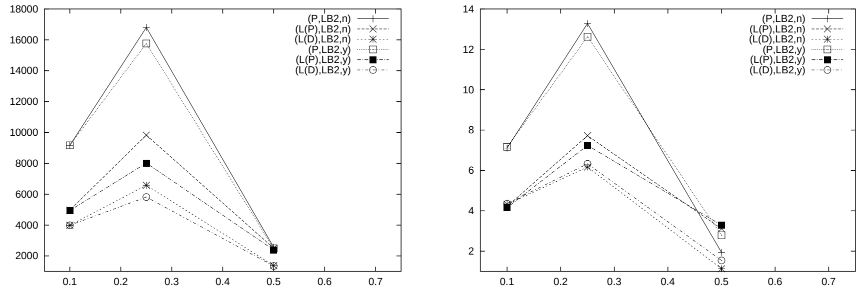

Figure 2: Average number of branches and average computation time of six algorithms.

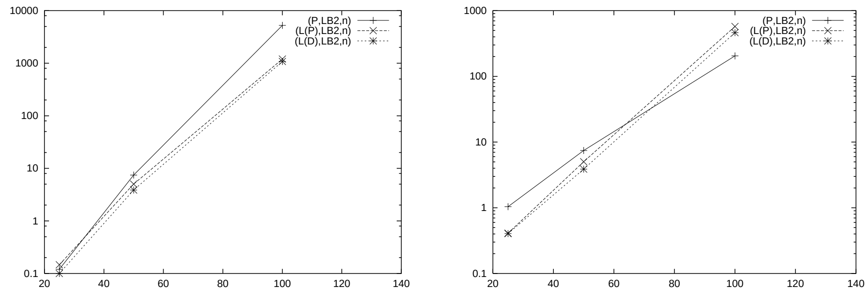

Figure 3: On the right the influence of the number of items and on the left the influence of the number of tests (logarithmic scale).

	(P, L_2, N)	$(S(P), L_2, N)$	$(S(D), L_2, N)$
Number of branches	404	391	334
Computation time	0.2	0.2	0.2

Table 1: Results on the potato instance.

lower bound L_2 is the best choice. The stronger lower bound L_3 is not useful as a consequence of its expensive computation. The algorithms (P, L_2, N), $(S(P), L_2, N)$, and $(S(D), L_2, N)$ come out as most promising. For these three algorithms we studied the effect of applying pruning criterion PC4 on the same instances. The number of branches and the average computation time are depicted in Figure 2, respectively. As we see, the effect of pruning criterion PC4 is small: the saving on the number of branches is often annihilated by the extra computation time. We did not include PC4 in further experiments.

For (P, L_2, N), $(S(P), L_2, N)$, and $(S(D), L_2, N)$ we also studied the influence of the number of items and the number of tests on the average computation time. In Figure 3 we show the influence of the number of items by depicting the average computation time for the instances with 50 tests and the influence of the number of tests by depicting the average computational time for the instances with 50 items. We see that the results of the power as quality criterion improve when the ratio between the number of tests and the number of items increases.

Finally, we tested the three algorithms on the potato instance that initiated the research 20 years ago. As we see in Table 6 the potato instance is not a challenge anymore.

Based on our test results, our first conclusion is that our new power-based bounding algorithms clearly outperform the information-based bounding algorithm which was introduced by [Moret & Shapiro 1985]. We endorse the conclusion of Moret & Shapiro that the most efficient algorithms are based on the least-seperated pair criterion, however combined with the power based bounding, and conclude that $(S(D), L_2, N)$ performs best in our experiments. Further study into the typical structure of problem instances may show that other choices of algorithms may be more appropriate in specific cases.

7 Postlude

We presented branch-and-bound algorithms tailored to minimum test set problems. Some of them are old. The new ones that we propose are based on the *power of tests*. Lower bounds are constructed and analyzed on their pruning power. Empirical testing of the various algorithms indicate that the lower bound based on our new notion of power of tests yields the best results. The test results show encouraging behavior of the better algorithms on reasonably sized problems. However, they also reveal exponentially increasing running times in the number of tests and the number of items. Especially, the number of items has a quite dramatic effect on the running times. Test results, which we did not present showed e.g. that $(50, 100, 0.25)$-instances could not be solved by algorithms using the information criterion within a few hours. More testing is required to find limits to problem sizes that are still solvable by the better of the algorithms.

A practically and theoretically interesting feature we have not considered in the algorithms is the use clever data structures for storing and retrieving information computed. We expect a significant speed-up of the algorithms by implementing smart data structures.

References

[1] K.M.J. De Bontridder, *Methods for solving the Test Cover Problem*, Master's Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, February 1997.

[2] H.Y. Chang, E. Manning, G. Metze, *Fault diagnosis of digital systems*, Wiley, New York, 1970.

[3] M.R. Garey, and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[4] K.M.J. De Bontridder, B.V. Halldórsson, M.M. Halldórsson, C.A.J. Hurkens, J.K. Lenstra, R. Ravi, L. Stougie, *Approximation algorithms for the minimum test set problem*, Forthcoming, 2002.

[5] B.M.E. Moret, Decision trees and diagrams, *Comp. Surveys 14*, 1982, 593–623.

[6] B.M.E. Moret, H.D. Shapiro, On minimizing a set of tests. *SIAM Journal on Scientific and Statistical Computing 6*, 1985, 983–1003.

[7] R.W. Payne, Selection criteria for the construction of efficient diagnostic keys. *Journal of Statistical Planning and Information 5*, 1981, 27–36.

[8] A. Rescigno, G.A. Maccacaro, The information content of biological classification, In C. Cherry (ed.), *Information Theory: Fourth London Symposium*, Butterworths, London, 1961, 437–445.

[9] E.W. Rypka, W.E. Clapper, I.G. Brown, R. Babb, A model for the identification of bacteria. *Journal of General Microbiology 46*, 1967, 407–424.

[10] E.W. Rypka, L. Volkman, E. Kinter, Construction and use of an optimized identification scheme. *Laboratory Magazine 9*, 1978, 32–41.