DIMACS Technical Report 97-82
December 1997

Diagnosing Double Regular Systems

by

Endre Boros[1,2]
RUTCOR
Rutgers University
New Brunswick, New Jersey

Tonguç Ünlüyurt[3,4]
RUTCOR
Rutgers University
New Brunswick, New Jersey

[1]Permanent Member
[2]This research was partially supported by the ONR (Grants N00014-92-J-1375 and N00014-92-J-4083), by DARPA (Contract Number N66001-97-C-8537).
[3]Affiliated Graduate Student Member
[4]This research was partially supported by the ONR (Grants N00014-92-J-1375 and N00014-92-J-4083), by DARPA (Contract Number N66001-97-C-8537).

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

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

ABSTRACT

We consider the problem of testing sequentially the components of a double regular system, when the testing of each component is costly. Generalizing earlier results about k-out-of-n systems, we provide a polynomial time algorithm for the most cost-effficient sequential testing of double regular systems. The algorithm can be implemented to work efficiently both for explicitly given systems, and for systems given by an oracle.

1 Introduction

The problem of diagnosing a complex system through a series of tests of its components arises in many practical situations. In many of these cases, testing a component is costly (it may cost money, time, or incur pain on the patient, etc.), and therefore it is important to determine the right diagnosis policy (the right sequence in which such tests are carried out one-by-one) which minimizes the expected cost of the diagnosis on the long run.

More precisely, we consider complex systems, consisting of several components, in which the state of the system (e.g. working or failing, healthy or unhealthy, etc.) depends on the states of its components. We shall assume that this dependency is known, either by an explicit functional description, or via an oracle.

The problem of diagnosing such a system is a procedure testing the components one-by-one, as long as the state of the whole system is uniquely determined. There are many variations for such a diagnosis problem, and we shall consider in this paper one frequently arising type, in which it is assumed that the cost of testing a component as well as the probability that it is working correctly, are known in advance. Furthermore, it is assumed that the components work, or fail independently of each other. In this case, to a given testing policy one can associate the expected cost, and the problem we consider is to find that policy which minimizes the corresponding expected cost.

Such problems arise in a wide area of applications, including telecommunication (testing connectivity of networks, paging cellular phone customers, etc. see e.g. [11]), testing machines in manufacturing, or sequencing and checking tasks by their priorities on a computer (see e.g. [16]), at a service station (see e.g. [13]), choosing the right order of categories in a quiz show (see e.g. [18]), finding optimal derivation strategies in knowledge bases (see e.g. [17]), best-value (or satisficing) search algorithms (see e.g. [9, 19, 20, 22, 24, 26]), and medical applications, where incoming patients have to be tested against the possibility of some rare but dangerous disease (see e.g. [25]).

In this paper we shall consider *double regular* systems, in which the dependence of the state on its components states satisfies certain conditions (for definitons see Section 3), and show that an optimal testing policy can determined and carried out in polynomial time for such systems. This result generalizes earlier results for k-out-of-n systems. [2, 10]

2 Definitions and Main Results

A *system* is composed of a set $N = \{u_1, u_2, ...u_n\}$ of n components, each of which can be in one of the two states, *working* or *failing*. Since each of these components can, in principle, be in any of the two states, there are 2^n different possible states of the system. A *state vector* $\mathbf{x} = (x_1, ..., x_n)$ is a binary vector, recording the state of the system, i.e. defined as $x_i = 1$ if component u_i is working., and $x_i = 0$ otherwise, for $i = 1, ..., n$. The *structure function* of the system is the Boolean mapping $f : \mathbb{B}^n \mapsto \mathbb{B}$ characterizing the states in which the system is functioning, i.e.

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if the system is functioning in the state } \mathbf{x} \\ 0 & \text{otherwise.} \end{cases}$$

Such a Boolean mapping $f : \mathbb{B}^n \mapsto \mathbb{B}$ is called *monotone* if $f(\mathbf{x}) \geq f(\mathbf{y})$ whenever $\mathbf{x} \geq \mathbf{y}$.

Let us denote a system by the pair (N, f), where N is the set of components and f is the system function. Given a system (N, f), we shall say that its ith component is relevant (with respect to f) if there exists a state vector $\mathbf{x}$ such that $x_i = 0$ and $f(\mathbf{x} \vee \mathbf{e}_i) > f(\mathbf{x})$, where $\mathbf{e}_i$ denotes the ith unit vector. The system (N, f) is called *coherent* if its system function f is a monotone Boolean function, and all of its components are relevant with respect to f.

Sequential diagnosis is a procedure in which the components of a system are inspected, one by one, in order to find out the functionality of the system at the current (not yet known) state. We assume that, when inspecting a component, we can learn the correct state of that component. Typically, one may not need to learn the states of all components to be able to tell the functionality of the system. An inspection *strategy* $\mathcal{S}$ is a rule, which specifies, on the basis of the states of the already inspected components, which component is to be inspected next or the correct value of the structure function at the state vector the system is in. Such an inspection strategy can be represented as a binary decision tree.

Example 1 *Let us consider a system consisting of 3 components* $N = \{u_1, u_2, u_3\}$ *and having*

$$f(x_1, x_2, x_3) = x_1x_2 \vee x_2x_3 \tag{1}$$

as system function. Then Figure 1 represents a possible inspection strategy, in which component u_1 *is inspected first, and depending on if it is working or not, components* u_2 *or* u_3 *are inspected next, etc. E.g., if the system happens to be in state* $\mathbf{x} = (1, 1, 0)$*, then this strategy learns that the system is functioning well after inspecting components* u_1 *and* u_2*, while if the system's state is* $\mathbf{y} = (0, 1, 0)$*, then this strategy will stop after inspecting components* u_1 *and* u_3*, recognizing that the system does not function.*

The main reason to terminate the inspection algorithm as early as possible, and not to check all components, is that inspection is costly. Let us denote by c_i the cost of inspecting component u_i, $i = 1, ..., n$, and let $\mathbf{c} = (c_1, ..., c_n)$. In principle, when describing an inspection strategy, one can go further, and inspect other components, even after being able to tell the functionality of the system. Let us call a strategy *reasonable* if it indeed stops as soon as it learns the functionality of the system. From this point on we shall consider only reasonable strategies, and we shall say strategy instead of reasonable strategy. For instance, the strategy in Figure 1 is a reasonable inspection strategy.

Our main goal is to find a strategy for a given system, which minimizes the cost of inspection. This however, is not a well defined problem, yet, since for different states a different strategy would be optimal.

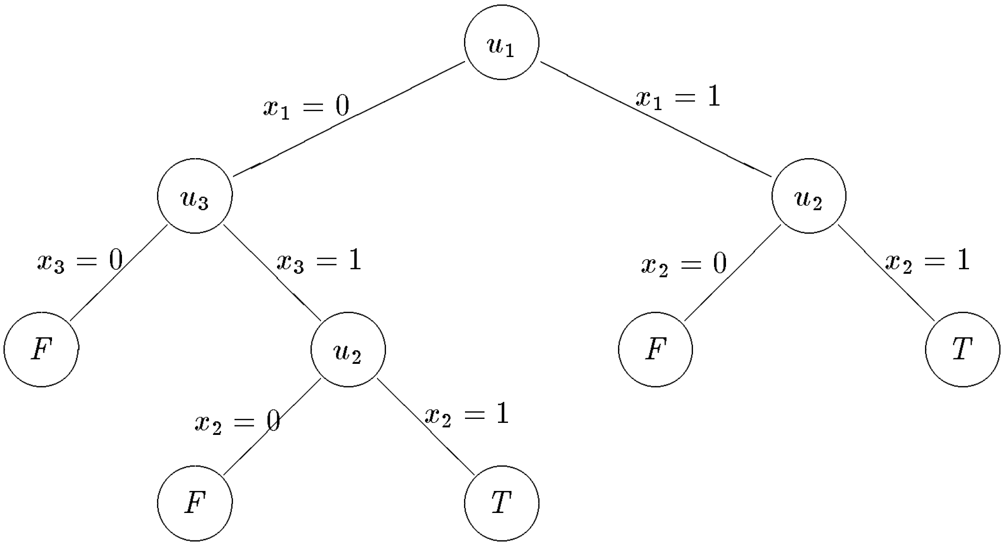

Figure 1: An inspection strategy for the system of Example 1.

Example 2 *Let us return to the system of Example 1. For the state* $\mathbf{x} = (1, 1, 0)$ *the strategy represented in Figure 1 is optimal, since it checks only the first two components (and inspecting less than that would not be enough). However, if the system is in state* $\mathbf{y} = (1, 0, 1)$*, then this strategy inspects all components to learn that the system is not functioning, although, it would be enough to check only component* u_2*.*

Clearly, the cost of inspection depends not only on our strategy, but also on the state the system is at. For this reason, let us assume that an apriori distribution of the states of the system is known. We shall assume in this paper that the component u_i works with probability p_i and fails with probability $q_i = 1 - p_i$, for $i = 1, ..., n$, and that the components work or fail independently of each other. Let $\mathbf{p} = (p_1, ..., p_n)$ denote the probability vector of the system.

Given an instance of the sequential diagnosis problem, i.e. a system (N, f) and the vectors $\mathbf{c}$ and $\mathbf{p}$ of costs and probabilities, respectively, we can reformulate our main goal as to find an inspection strategy $\mathcal{S}$ of the system which has the minimum expected cost.

This problem has been considered in various contexts, and was solved for *series*, *parallel* and *k-out-of-n* systems, see [2, 10]. (A k-out-of-n system is functioning if at least k of its n components work. Series (parallel) systems are special cases corresponding to $k = n$ $(k = 1)$.) We shall generalize these results for a larger class of coherent systems.

Let us associate two permutations σ and π of $\{1, 2, ..., n\}$ to an instance $(f, \mathbf{c}, \mathbf{p})$, satisfying

$$\frac{c_{\sigma_1}}{p_{\sigma_1}} \leq \frac{c_{\sigma_2}}{p_{\sigma_2}} \leq \cdots \leq \frac{c_{\sigma_n}}{p_{\sigma_n}} \text{ and } \frac{c_{\pi_1}}{q_{\pi_1}} \leq \frac{c_{\pi_2}}{q_{\pi_2}} \leq \cdots \leq \frac{c_{\pi_n}}{q_{\pi_n}}, \tag{2}$$

where again $q_i = 1 - p_i$, $i = 1, ..., n$. As it was shown in [8] σ is an optimal order of inspection for a series system, while π is an optimal order for a parallel system. We consider the following algorithm for our problem.

INSPECT($f, \mathbf{c}, \mathbf{p}$):

Input: A coherent system function f, a cost vector $\mathbf{c}$ and a probability vector $\mathbf{p}$.

Output: A subset $I \subseteq \{1, 2, ..., n\}$ containing the indices of the inspected components, together with the values x_i, $i \in I$ of these components at the current state $\mathbf{x}$, and the value $f(\mathbf{x})$.

Step 0: Let $I = \emptyset$, and let us compute the permutations σ and π as in 2.

Step 1: Let k be the smallest index for which $f(\mathbf{y}^k) = 1$, where

$$y^k_{\sigma_j} = \begin{cases} x_{\sigma_j} & \text{if } \sigma_j \in I, \\ 1 & \text{if } \sigma_j \notin I, j \leq k, \\ 0 & \text{if } \sigma_j \notin I, j > k. \end{cases} \tag{3}$$

If there is no such k, then STOP, the system is not functioning correctly, i.e. $f(\mathbf{x}) = 0$.

Step 2: Let l be the smallest index for which $f(\mathbf{z}^k) = 0$, where

$$z^k_{\pi_j} = \begin{cases} x_{\pi_j} & \text{if } \pi_j \in I, \\ 0 & \text{if } \pi_j \notin I, j \leq l, \\ 1 & \text{if } \pi_j \notin I, j > l. \end{cases} \tag{4}$$

If there is no such l, then STOP, the system is working, i.e. $f(\mathbf{x}) = 1$.

Step 3: Choose $i \in (\{\sigma_1, ..., \sigma_k\} \cap \{\pi_1, ..., \pi_l\}) \setminus I$, and learn the value x_i by inspecting component i, and set $I = I \cup \{i\}$. Return to **Step 1**.

We shall show that the above algorithm generates an optimal inspection strategy, i.e. one with the minimum expected cost for coherent systems (N, f), for which f is both σ- and π-regular (for definitions see Section 3). Furthermore, we shall prove that the above algorithm can be carried out by computing the value of f at most $2n^2 + 2n$ times, and by performing $O(n^2 \log n)$ comparisions. Since $k - out - of - n$ systems are regular with respect to any permutation, this result generalizes results of [2, 10].

3 Notations and basic properties

Given a positive Boolean function $f : \mathbb{B}^n \mapsto \mathbb{B}$, we shall say that the variable x_i is *stronger* than x_j $(i \neq j)$ with respect to f, and write $x_i \succeq_f x_j$, if for all binary vectors $\mathbf{x} \in \mathbb{B}^n$ with $x_i = x_j = 0$ we have $f(\mathbf{x} \vee \mathbf{e}_i) \geq f(\mathbf{x} \vee \mathbf{e}_j)$.

It is well-known that this strength relation is a preorder on the set of variables for any positive Boolean function f, (see e.g. [27]). Let us call f *regular* with respect to a permutation σ, if its strength preorder induces σ, i.e. if $x_{\sigma_1} \succeq_f x_{\sigma_2} \succeq_f \cdots \succeq_f x_{\sigma_n}$. For any two points $\mathbf{y}, \mathbf{z} \in B^n$, we say $\mathbf{z}$ is a left-shift of $\mathbf{y}$ if there exists a mapping σ from $\{i : y_i = 1\}$ to $\{i : z_i = 1\}$ such that

1) σ is injective, i.e. $\sigma(i) \neq \sigma(j)$ for $i \neq j$.

2) $\sigma(i) \leq i \; \forall \, i$.

Another characterization of regularity is given by the following: f is regular with $x_1 \succeq_f x_2 \succeq_f \cdots \succeq_f x_n$ iff $\forall$ $\mathbf{y}, \mathbf{z} \in \mathbb{B}^n$ $\mathbf{z}$ is a left shift of $\mathbf{y}$ implies $f(\mathbf{y}) \leq f(\mathbf{z})$.

The *dual* f^d of a Boolean function f is defined by $f^d(\mathbf{x}) = \overline{f}(\overline{\mathbf{x}})$ for all $\mathbf{x} \in \mathbb{B}^n$. It is well-known that a positive Boolean function f and its dual f^d are simultaneously regular (or not) with respect to a given permutation σ, see e.g. [27].

For a given Boolean function f let us denote by $T(f)$ and $F(f)$ the sets of binary vectors at which f takes value 1 and 0, respectively. The vectors in $T(f)$ are called the *true points*, while the elements of $F(f)$ are called the *false points* of f.

Given a subset $S \subseteq \{1, 2, ..., n\}$, let us denote its characteristic vector by $\mathbf{1}^S$, i.e.

$$\mathbf{1}^S_j = \begin{cases} 1 & \text{if } j \in S, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Clearly, for a positive Boolean function f, and for subsets A, B with $\mathbf{1}^A \in T(f)$ and $\mathbf{1}^B \in F(f)$, we have $\mathbf{1}^{A'} \in T(f)$ for all $A' \supseteq A$ and $\mathbf{1}^{B'} \in F(f)$ for all $B' \subseteq B$. Furthermore, f^d is again a positive Boolean function for which

$$T(f^d) = \{\mathbf{1}^S | \mathbf{1}^{\overline{S}} \in F(f)\}, \tag{6}$$

where $\overline{S} = \{1, 2, ..., n\} \setminus S$.

We shall call a subset $I \subseteq \{1, 2, ..., n\}$ an *implicant* of the positive Boolean function f, if $\mathbf{1}^I \in T(f)$. The subset I is called a *prime implicant* if it is an implicant, and no proper subset of it is an implicant.

Given a positive Boolean function f, let us denote by

$$\mathcal{I}(f) = \{I_1, ..., I_s\} \tag{7}$$

the family of its prime implicants, and let

$$\mathcal{J}(f) = \{J_1, ..., J_t\} \tag{8}$$

be the family of the prime implicants of its dual, i.e. $\mathcal{J}(f) = \mathcal{I}(f^d)$. The binary vectors $\mathbf{1}^I$ for $I \in \mathcal{I}(f)$ are called *minimal true points* of f, while vectors of the form $\mathbf{0}^J$ for $J \in \mathcal{J}(f)$ are called *maximal false points* of f, where

$$\mathbf{0}_j^S = \begin{cases} 0 & \text{if } j \in S, \\ 1 & \text{otherwise.} \end{cases} \tag{9}$$

Let us first recall a few well-known properties of positive Boolean functions.

Proposition 1 *A positive Boolean function f and its dual f^d can be represented by the positive disjunctive normal forms,*

$$f(\mathbf{x}) = \bigvee_{I \in \mathcal{I}(f)} \left(\bigwedge_{j \in I} x_j \right), \tag{10}$$

and

$$f^d(\mathbf{x}) = \bigvee_{J \in \mathcal{J}(f)} \left(\bigwedge_{j \in J} x_j \right), \tag{11}$$

respectively. Furthermore, for any $I \in \mathcal{I}(f)$ and $J \in \mathcal{J}(f)$ we have $I \cap J \neq \emptyset$.

Let us remark next that given a positive Boolean function f by its list $\mathcal{I}(f)$ of prime implicants, it may be very difficult to obtain $\mathcal{J}(f)$, the list of its dual prime implicants. First of all the size of $\mathcal{J}(f)$ can be exponentially larger than the size of $\mathcal{I}(f)$. Secondly, there is no known efficient algorithm to obtain $\mathcal{J}(f)$ from $\mathcal{I}(f)$ in time polynomial in these sizes, not even in cases when the size of $\mathcal{J}(f)$ is proportional to the size of $\mathcal{I}(f)$, see e.g. [4, 14, 15, 21].

It is well-known, however that for regular functions the above *dualization* problem can be solved efficiently, see e.g. [3, 12, 23]. In particular, it is shown in [5] that

Proposition 2 *If f is a regular function (with respect to a permutation σ), then*

$$|\mathcal{J}(f)| \leq (n + 1 - \log_2 |\mathcal{I}(f)|)|\mathcal{I}(f)|, \tag{12}$$

furthermore, $\mathcal{J}(f)$ can be generated in $O(n^2|\mathcal{I}(f)|)$ time.

Let us mention finally that if a regular function is given by an oracle, then its prime implicants can be generated efficiently, see e.g. [6, 7].

4 Strategies and binary decision trees

As we noted earlier, inspection strategies can naturally be represented as *binary decision trees*, i.e. as rooted trees, in which every node has two or zero successors. Nodes with zero successors are called the *leaves*, and are labeled by T or F corresponding to the value of the function f the tree represents. Nodes with two successors are labeled by the components of the system. The two arcs leading to the successors of a node labelled by $u_i \in N$ represents the two states this component can be – the right arc corresponding to the working state of u_i, while the left arc corresponding to the failing of component u_i. Let us label the right arc by x_i and the left arc by $\overline{x}_i$. To a node a of $\mathcal{T}$ then let us associate $P(a) \subseteq \{x_1, \overline{x}_1, x_2, \overline{x}_2, ..., x_n, \overline{x}_n\}$ the set of labels along the arcs of the uniqe path connecting the root of $\mathcal{T}$ to a. Let us denote furthermore the set of leaves of $\mathcal{T}$ labeled by T, the so called *true leaves*, by $L^T(\mathcal{T})$, and analogously, let $L^F(\mathcal{T})$ denote the set of leaves of $\mathcal{T}$ labeled by F, i.e. the set of *false leaves.*

Then, we have that

$$f(\mathbf{x}) = \bigvee_{a \in L^T(\mathcal{T})} \bigwedge_{u \in P(a)} u \tag{13}$$

is a DNF representation of the function f, and

$$f^d(\mathbf{x}) = \bigvee_{b \in L^F(\mathcal{T})} \bigwedge_{v \in P(b)} \overline{v} \tag{14}$$

is a DNF representation of f^d.

Example 3 *Let us consider again the system in Example 1, and let us consider the inspection strategy, i.e. the binary decision tree of this system in Figure 2. There are two true leaves and three false leaves, and using the edge labeling we can write*

$$f(x_1, x_2, x_3) = x_1 x_2 \vee \overline{x}_1 x_2 x_3$$

and

$$f^d(x_1, x_2, x_3) = \overline{x}_1 x_2 \vee x_1 \overline{x}_3 x_2 \vee x_1 x_3.$$

Let us call a binary decision tree of f *reasonable*, if it is a reasonable inspection strategy, i.e. if any subtree rooted at a non-leaf node contains both true and false leaves. Let us denote by $\mathbf{BDT}(f)$ the family of all reasonable binary decision trees which represent the Boolean function f.

Let us consider now a coherent system (N, f), a cost vector $\mathbf{c}$ and a probability vector $\mathbf{p}$, and an inspection strategy $\mathcal{S} \in \mathbf{BDT}(f)$ of this system.

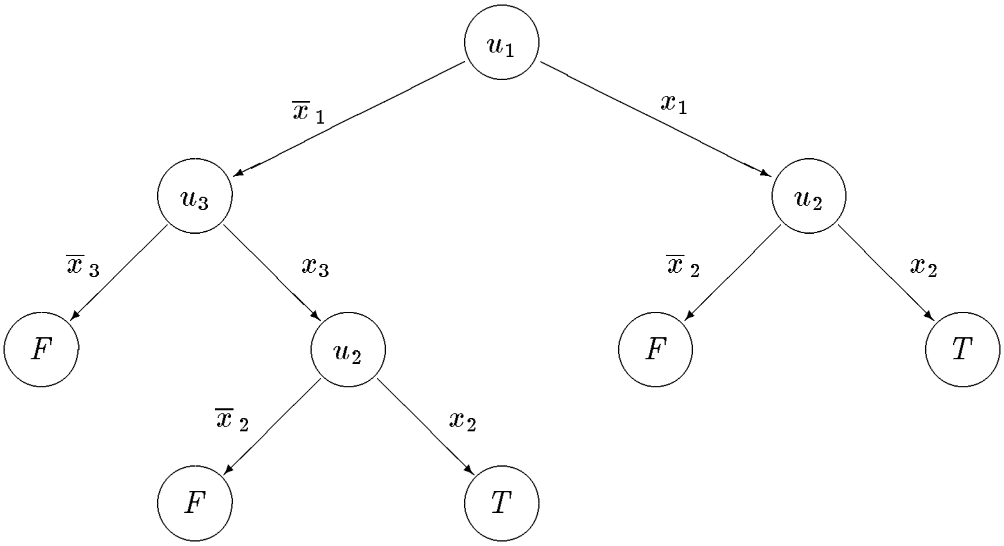

Figure 2: A binary decision tree representing the system of Example 1.

For every state $\mathbf{x} \in \mathbb{B}^n$ the inspection algorithm will terminate in one of the leaf nodes of $\mathcal{S}$. Let us denote by $B(a)$ the set of binary vectors for which the algorithm stops in the leaf node a. Clearly, $B(a) \cap B(b) = \emptyset$ if a and b are different leaves of $\mathcal{S}$, and we have

$$T(f) = \bigcup_{a \in L^T(\mathcal{S})} B(a) \quad \text{and} \quad F(f) = \bigcup_{b \in L^F(\mathcal{S})} B(b). \tag{15}$$

Let us denote by $C(\mathcal{S}, \mathbf{x})$ the cost of inspection by strategy $\mathcal{S}$ if the system is in state $\mathbf{x}$. According to the above remark, this cost is clearly the same for all state vectors which belong to the same leaf. More precisely, if $\mathbf{x} \in B(a)$ for a leaf a, then

$$C(\mathcal{S}, \mathbf{x}) = Cost(\mathcal{S}, a) = \left(\sum_{i: x_i \text{ or } \overline{x}_i \in P(a)} c_i \right). \tag{16}$$

Furthermore, the probability that the system is at one of the states belonging to $B(a)$ is clearly

$$Prob(\mathcal{S}, a) = \left(\prod_{i: x_i \in P(a)} p_i \right) \left(\prod_{i: \overline{x}_i \in P(a)} q_i \right). \tag{17}$$

whereas the probability that the system is in state vector $\mathbf{x}$ is

$$Prob(\mathbf{x}) = \left(\prod_{i: x_i = 1} p_i \right) \left(\prod_{i: x_i = 0} q_i \right). \tag{18}$$

Hence, the expected cost of strategy $\mathcal{S}$ is

$$EC(\mathcal{S}) = \sum_{a \in L^T(\mathcal{S}) \cup L^F(\mathcal{S})} Cost(\mathcal{S}, a) Prob(\mathcal{S}, a). \tag{19}$$

or equivalently

$$EC(\mathcal{S}) = \sum_{\mathbf{x} \in T(f) \cup F(f)} C(\mathcal{S}, \mathbf{x}) Prob(\mathbf{x}). \tag{20}$$

It will be useful to consider another expresion for $EC(\mathcal{S})$. Given a strategy $\mathcal{S}$ let us denote by $P_i(\mathcal{S})$ the probability that component u_i is inspected. Then we have

$$EC(\mathcal{S}) = \sum_{i=1}^{n} c_i P_i(\mathcal{S}). \tag{21}$$

Example 4 *Let us consider again the system of Example 1, and the inspection strategy $\mathcal{S}$ in Figure 2. Let us consider e.g. the rightmost true leaf a of $\mathcal{S}$. For this leaf we have $B(a) = \{(1,1,0),(1,1,1)\}$, $C(\mathcal{S},a) = c_1 + c_2$ and $Prob(\mathcal{S},a) = p_1p_2$. Evaluating these quantities for all the leaves we get by (18)*

$$\begin{aligned} EC(\mathcal{S}) = \; & p_1p_2(c_1+c_2) + p_1q_2(c_1+c_2) + q_1p_3p_2(c_1+c_3+c_2) + \\ & q_1p_3q_2(c_1+c_3+c_2) + q_1q_3(c_1+c_3). \end{aligned}$$

In this strategy the first component is always inspected, hence $P_1(\mathcal{S}) = 1$. Component u_2 is inspected if either u_1 is working, or if u_1 is failing and u_3 is working, hence $P_2(\mathcal{S}) = p_1 + q_1p_3$. Finally, u_3 is inspected only if u_1 is not working, i.e. $P_3(\mathcal{S}) = q_1$. Thus, by (20) we obtain

$$EC(\mathcal{S}) = c_1 + (p_1 + q_1p_3)c_2 + q_1c_3.$$

The equivalence of these two expressions for $EC(\mathcal{S})$ can easily be seen by applying the identities $p_1 + q_1 = p_2 + q_2 = p_3 + q_3 = 1$.

Given a system (N, f), a cost vector $\mathbf{c}$ and a probability vector $\mathbf{p}$, the problem of finding the best sequential inspection strategy can be restated as

$$\min_{\mathcal{S} \in \mathbf{BDT}(f)} EC(\mathcal{S}). \tag{22}$$

We can now formulate our main result.

Theorem 1 *Let us consider a coherent system (N, f), a cost vector $\mathbf{c}$ and a probability vector $\mathbf{p}$, such that f is regular with respect to the permutations σ and π defined by*

$$\frac{c_{\sigma_1}}{p_{\sigma_1}} \leq \cdots \leq \frac{c_{\sigma_n}}{p_{\sigma_n}} \text{ and } \frac{c_{\pi_1}}{q_{\pi_1}} \leq \cdots \leq \frac{c_{\pi_n}}{q_{\pi_n}}.$$

Then the algorithm Inspect *finds an inspection strategy $\mathcal{S} \in \mathbf{BDT}(f)$ minimizing the expected cost $EC(\mathcal{S})$. The inspection according to this strategy can always be carried out by evaluating the value of f most $2n^2 + 2n$ times, and by performing $O(n^2 \log n)$ comparisions.*

5 1(0)-Optimal Strategies

We shall say that $\mathcal{S}$ is *1(0)-Optimal* if $\mathcal{S}$ minimizes

$$EC^{1(0)}(\mathcal{S}) = \sum_{a \in L^T(\mathcal{S})(L^F(\mathcal{S}))} Cost(\mathcal{S}, a) Prob(\mathcal{S}, a). \tag{23}$$

We shall assume that f is a Boolean function, both σ and π-regular. Without any loss of generality we may assume that $\sigma(\pi) = (1, 2, .., n)$.

For a subset I of $\{1, 2, .., n\}$, let us define $l^I = max\{i : i \in I\}$, and for a binary vector $\mathbf{x} \in \mathbb{B}^n$, let $ON(\mathbf{x}) = \{i | x_i = 1\}$ and $OFF(\mathbf{x}) = \{i | x_i = 0\}$.

First, we shall give a proof for a $1 - optimal$ strategy. The proof for $0 - optimal$ strategy will follow by duality. For our proof we need a few lemmas.

Lemma 1 *If $\mathbf{x} \in T(f)$, then there exists a unique $I \in \mathcal{I}(f)$ satisfying $\mathbf{x}_j = \mathbf{1}^I_j$ for $j \leq l^I$.*

Proof. Let I be the lexicographically largest prime implicant such that $I \subseteq ON(\mathbf{x})$. Then I satisfies the condition in the lemma, because otherwise $I \bigcup \{j\} - \{l^I\}$ would be an implicant of f by the regularity of f, contradicting the choice of I as lexicographically largest prime implicant for which $i \subseteq ON(\mathbf{x})$. Finally, the prime implicant I is unique, because the existence of a different prime implicant satisfying the same conditions would violate the primeness of I. □

Example 5 *Let $f = x_1 \bigvee x_2 x_3 \bigvee x_2 x_4$ with $I_1 = \{1\}$, $I_2 = \{2, 3\}$ and $I_3 = \{2, 4\}$, note that f is $\sigma - regular$ with $\sigma = (1, 2, 3, 4)$. If $\mathbf{x} = (0, 1, 1, 1)$, then the associated prime implicant is $I = I_2$, and $l^{I_2} = 3$.*

Thus, for every true point $\mathbf{x}$ we can associate a unique prime implicant I, and its corresponding characteristic vector $\mathbf{1}^I$. We shall say that $\mathbf{x}$ is an extension of I. Let us note that $\mathbf{x}$ and $\mathbf{1}^I$ match exactly at indices $j \leq l^I$. In this way, to each $\mathbf{x} \in T(f)$ we have associated a unique $I \in \mathcal{I}(f)$, and from this point on we shall use the notations $I^{\mathbf{x}}$, $l^{\mathbf{x}}$ and $\mathbf{1}^{\mathbf{x}}$. Let us further define $\hat{l^{\mathbf{x}}}$ to be the index of the last but one 1 digit (the previous 1 before the last) in I^x, and if this does not exist let $\hat{l^{\mathbf{x}}} = 0$.

Let $\gamma(g, h) = \{\mathbf{x} : \mathbf{x} \in T(f), l^{\mathbf{x}} = h, \hat{l^{\mathbf{x}}} = g\}$ for $h \geq g$. We can notice that if $\gamma(g, h) \neq \emptyset$, then $\gamma(g, h') \neq \emptyset$ for $g + 1 \leq h' \leq h$.

Proposition 3 *If $\mathbf{x} \in \gamma(g, h)$ and $\mathbf{y} \in \gamma(g, h')$ then $x_k = y_k$ for $k \leq g$, whenever f is regular with respect to $(1, 2, .., n)$*

Proof. If the above were not true, we could assume without any loss of generality that for some $k \leq g$, $x_k = 1$ and $y_k = 0$. Then by shifting the last 1 in $\mathbf{y}$ to the kth position we would get a left shift of $\mathbf{y}$ which is then a true point by regularity, contradicting the fact that $\mathbf{y} \in \gamma(g, h')$. □

Let $\Gamma(g) = \bigcup_{\forall h} \gamma(g, h)$. Note that if $\gamma(g, h) = \emptyset$ then $\gamma(g, h') = \emptyset$ for $h' \geq h$.

Proposition 4 *The sets* $\Gamma(g)$*,* $g = 0, 1, ..., n-1$ *form a partition of the true points.*

Proof. This proposition follows immediately, since $\Gamma(g)$ consists of those true points $\mathbf{x}$ for which $l^{\mathbf{x}} = g$, which is a uniquely defined index for all true points. $\square$

Example 6 *Suppose that we have* $\mathbf{x} = (1,0,1,1,0,1,0,0,1,1,0,1,0)$*,* $\mathbf{1}^I = (1,0,1,1,0,1,0,0,1,0,0,0,0)$ *and* $max\{h : \gamma(g,h) \neq \emptyset\} = 11$*. Obviously* $g = 6$*, and* $\gamma(6,9)$ *consists of the true points which are extensions of* $\mathbf{1}^I$*, and* $\Gamma(6)$ *consists of the true points which are extensions of the following characteristics vectors of prime implicants:*

$$\begin{array}{r}
\gamma(6,7) \rightarrow (1,0,1,1,0,1,1,0,0,0,0,0,0) \\
\gamma(6,8) \rightarrow (1,0,1,1,0,1,0,1,0,0,0,0,0) \\
\gamma(6,9) \rightarrow (1,0,1,1,0,1,0,0,1,0,0,0,0) \\
\gamma(6,10) \rightarrow (1,0,1,1,0,1,0,0,0,1,0,0,0) \\
\gamma(6,11) \rightarrow (1,0,1,1,0,1,0,0,0,0,1,0,0)
\end{array}$$

Lemma 2 $p_1 + q_1p_2 ..... + q_1q_2...q_{i-2}p_{i-1} + q_1q_2...q_{i-1} = 1$ *for* $i \geq 2$
where $p_i + q_i = 1$*.*

Proof. Suppose we have n independent events such that p_i is the probability that ith event occurs with $p_i + q_i = 1$. The first term in the summation, p_1, is the probability that the first event occurs. The second term, p_1q_2, is the probability that the first term does not and the second term does occur, ..., etc. Finally, the last term, $q_1q_2...q_{i-1}$ is the probability that none of these events occur. Hence this is a partition of all possibilities and the sum of these probabilities must be equal to 1. $\square$

Lemma 3 *Assume* f *is* σ*-regular, where e.g.* $\sigma = (1, 2, .., n)$*. Let* $\mathcal{S}^1$ *be the strategy which inspects the components in their* σ*-order* $u_1, u_2, ..., u_n$ *until the functionality of the system is determined. Then,* $\mathcal{S}^1$ *is a* $1-optimal$ *strategy.*

Proof. We shall prove this lemma by induction on the total number of components. If $n = 1$, then there is only one strategy which is hence optimal. Assume next that the claim is true for less than n components, let $\mathcal{S}^*$ denote a 1-optimal strategy for an n component system, and let $\mathcal{S}^i$ denote the strategy which inspects u_i first and the rest of the strategy is 1-optimal, i.e. the right(left) branches of the root correspond to 1-optimal strategies of the residual smaller component systems.

Note, that by our hypothesis $\mathcal{S}^i$ inspects the components in the order $u_i, u_1, ..., u_{i-1}, u_{i+1}, ..., u_n$, whereas $\mathcal{S}^1$ inspects the components in the order $u_1, u_2, ..., u_n$ until the functionality of the system is determined. Since we have

$$EC^1(\mathcal{S}^*) = min_i EC^1(\mathcal{S}^i), \tag{24}$$

it is sufficient to prove that $EC^1(\mathcal{S}^1) \leq EC^1(\mathcal{S}^i)$ for all i. For this let us write

$$EC^1(\mathcal{S}^1) - EC^1(\mathcal{S}^i) = \sum_{\mathbf{x} \in T(f)} (C(\mathcal{S}^1, x) - C(\mathcal{S}^i, x))Prob(\mathbf{x}). \tag{25}$$

Since the sets $\Gamma(g)$ form a partition of the true points it is sufficient to show that $\sum_{\mathbf{x} \in \Gamma(g)}(C(\mathcal{S}^1, x) - C(\mathcal{S}^i, x))Prob(\mathbf{x}) \leq 0$.

For $\mathbf{x} \in \gamma(g,h) \subseteq \Gamma(g)$ we claim that $C(\mathcal{S}^i, \mathbf{x}) = C(\mathcal{S}^1, \mathbf{x})$, whenever $i \leq h$, since $\mathcal{S}^i$ and $\mathcal{S}^1$ both inspect by the end the same set of components, namely $\{u_1, u_2, .., u_i\}$

Let us hence consider the cases when $h + 1 \leq i$. Note that for $\mathbf{x} \in \gamma(g,h)$, h can be as low as $g+1$. When $h+1 \leq i$ we can have two cases for $\mathbf{x} \in \gamma(g,h)$:

- $C(\mathcal{S}^1, \mathbf{x}) - C(\mathcal{S}^i, \mathbf{x}) = -c_i$ if $x_i = 0$, since in this case $\mathcal{S}^1$ inspects $\{u_1, u_2, .., u_h\}$ and $\mathcal{S}^i$ inspects $\{u_1, u_2, ..u_h, u_i\}$;
- $C(\mathcal{S}^1, \mathbf{x}) - C(\mathcal{S}^i, \mathbf{x}) = \sum_{j=g+1}^{j=h} c_j - c_i$ if $x_i = 1$, since in this case $\mathcal{S}^1$ inspects $\{u_1, u_2, .., u_h\}$ and $\mathcal{S}^i$ inspects $\{u_i, u_1, u_2, .., u_g\}$.

¿From the above discussion it follows that when we are comparing the costs of $\mathcal{S}^1$ and of $\mathcal{S}^i$, it is enough to consider the vectors in $\bigcup_{h=g+1}^{h=i} \gamma(g,h)$. Note that components $1, 2, ..., g$ are inspected by both strategies, and also that all $\mathbf{x} \in \Gamma(g)$ match exactly at indices $j \leq g$, implying thus $Prob(\mathbf{x} \in \gamma(g,h)) = K_0 K_h$, where $K_0 = \prod_{i \leq g, x_i = 0} q_i \prod_{i \leq g, x_i = 1} p_i$ and $K_h = q_{g+1}q_{g+2}..g_{h-1}p_h$. Since K_0 is a common factor in all these probabilities, we can ignore that part in our computations, because the overall diffrence in expected cost will be multiplied by $K_0 \geq 0$.

The following table shows the different cases corresponding to the possible values of h. According to the above, the listed K_h values are proportional to $Prob(\mathbf{x} \in \gamma(g,h))$.

		Components tested by		
h	K_h	$\mathcal{S}^1$	$\mathcal{S}^i$ if $x_i = 1$	$\mathcal{S}^i$ if $x_i = 0$
$g+1$	p_{g+1}	u_{g+1}	u_i	u_{g+1}, u_i
$g+2$	$q_{g+1}p_{g+2}$	$u_{g+1}u_{g+2}$	u_i	u_{g+1}, u_{g+2}, u_i
$g+3$	$q_{g+1}q_{g+2}p_{g+3}$	$u_{g+1}, u_{g+2}, u_{g+3}$	u_i	$u_{g+1}, u_{g+2}, u_{g+3}, u_i$
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..
$i-1$	$q_{g+1}..q_{h^*-2}p_{i-1}$	$u_{g+1}, u_{g+2}..u_{i-1}$	u_i	$u_{g+1}, u_{g+2}...u_{i-1}, u_i$
i	$q_{g+1}q_{g+2}..q_{i-1}p_i$	$u_{g+1}, u_{g+2}..u_i$	u_i	can't occur

From the above table we obtain

$$\sum_{\mathbf{x}\in\Gamma(g}(C(\mathcal{S}^1,\mathbf{x}) - C(\mathcal{S}^i,\mathbf{x}))Prob(\mathbf{x})$$

$$= \sum_{j=g+1}^{i-1} Prob(\text{case } j)[q_i(-c_i) + p_i(-c_i + \sum_{k=g+1}^{j} c_k)] + Prob(\text{case}i)(\sum_{j=g+1}^{i-1} c_j)$$

$$= \sum_{j=g+1}^{i-1} Prob(\text{case } j)(-c_i + p_i \sum_{k=g+1}^{j} c_k) + Prob(\text{case } i)\sum_{j=g+1}^{i-1} c_j$$

$$\begin{aligned} &= c_{g+1}p_i[(\sum_{j=g+1}^{i-1} q_{g+1}q_{g+2}...q_{j-1}p_j) + q_{g+1}q_{g+2}...q_{i-1}] - c_ip_{g+1} \\ &\quad +q_{g+1}[c_{g+2}p_i[(\sum_{j=g+2}^{i-1} q_{g+2}...q_{j-1}p_j) + q_{g+1}q_{g+2}...q_{i-1}] - c_ip_{g+2}] \\ &\quad +q_{g+2}q_{g+3}[c_{g+3}p_i[(\sum_{j=g+3}^{i-1} q_{g+3}...q_{j-1}p_j) + q_{g+1}q_{g+2}...q_{i-1}] - c_ip_{g+3}] \\ &\qquad\vdots \\ &\quad +q_{g+1}....q_{i-2}[c_ip_i(p_{i-1} + q_{i-1}) - c_ip_{i-1}] \end{aligned}$$

Then by Lemma 2 we have

$$\sum_{\mathbf{x}\in\gamma}(C(\mathcal{S}^1,\mathbf{x}) - C(\mathcal{S}^i,\mathbf{x}))Prob(\mathbf{x}) = \sum_{j=g+1}^{j=i-1} q_{g+1}q_{g+2}...q_{j-1}[c_jp_i - p_ic_j].$$

¿From this, since $c_jp_i - p_ic_j \leq 0$ for $j \leq i$, the claim follows. $\square$

Corollary 1 *Assume f is π-regular, $\pi = (1,2,..,n)$. Let $\mathcal{S}^0$ be the strategy which inspects components in the order $u_1, u_2..$ until the functionality of the system is determined, then $\mathcal{S}^0$ is $0-$optimal.*

Proof. Apply the proof of previous theorem to ($f' = f^d$, $\mathbf{p}' = \mathbf{e} - \mathbf{p}$, $\mathbf{c}$) where $\mathbf{e}$ is the vector of all 1s. $\square$

6 Optimal Strategies:

Under our assumptions, we know how to find 1(0)-optimal strategies. Now we try to find an optimal strategy. The following corollary which covers a specific case follows from the previous lemmas.

Corollary 2 *Let us assume that f is a regular function with $x_1 \succeq_f x_2.... \succeq_f x_n$, where $\frac{c_1}{q_1} \leq \frac{c_2}{q_2}... \leq \frac{c_n}{q_n}$ and $\frac{c_1}{p_1} \leq \frac{c_2}{p_2}... \leq \frac{c_n}{p_n}$. Then the strategy $\mathcal{S}^*$, which tests components in the order $u_1, u_2, ...,$ is optimal.*

Proof. We have $EC(\mathcal{S}) = EC^0(\mathcal{S}) + EC^1(\mathcal{S})$, and since by Lemma 3 and corollary 1 both $EC^0(\mathcal{S})$ and $EC^1(\mathcal{S})$ are minimized by $\mathcal{S}^*$, the claim follows. □

Lemma 4 *Let f be a positive regular boolean function with $x_1 \succeq_f x_2 \succeq .... \succeq x_n$. Let I_i, $\mathbf{y}_i$, l_i be the prime implicants, characteristic vectors of prime implicants and indeces of last 1's in the characteristic vectors, respectively, for $i = 1,..,m$, such that $y_1 >_L y_2 >_L ... >_L y_n$. Then $l_1 \leq l_k$ for all k.*

Proof. Let us choose the minimum index j for which the set $\{i : i \leq j\}$ is an implicant. Then this set must be a prime implicant, since f is regular, and we chose the minimum such j. Moreover, it is then the lexicographically largest prime implicant, since if there were any other prime implicant, lexicographically larger than this one, it would contain $\{i : i \leq j\}$ and some more indices, contradicting primeness. Obviously, the index of the last 1 in any other characteristic vector has to be greater than j. □

Let us assume now that $\frac{c_{\sigma_1}}{p_{\sigma_1}} \leq \frac{c_{\sigma_2}}{p_{\sigma_2}} \cdots \leq \frac{c_{\sigma_n}}{p_{\sigma_n}}$ and that $\frac{c_{\pi_1}}{p_{\pi_1}} \leq \frac{c_{\pi_2}}{p_{\pi_2}} \leq \cdots \leq \frac{c_{\pi_n}}{p_{\pi_n}}$. Let us also assume that f is regular with respect to both orders $x_{\sigma_1} \succeq_f x_{\sigma_2} \succeq_f .... \succeq_f x_{\sigma_n}$ and $x_{\pi_1} \succeq_f x_{\pi_2} \succeq_f .... \succeq_f x_{\pi_n}$.

Let $I_1, I_2, ...I_s$ denote the prime implicants of f with corresponding characteristic vectors $y^{I_1,\sigma}, y^{I_2,\sigma}...y^{I_s,\sigma}$ where we have $y^{I_1,\sigma} >_L y^{I_2,\sigma} ... >_L y^{I_s,\sigma}$ when the components are ordered by $\sigma_1, \sigma_2, ...\sigma_n$. Let us also consider J_1, J_2, ..., J_t, the prime implicants of f^d, with characteristic vectors $z^{J_1,\pi}$, $z^{J_2,\pi}$, ..., $z^{J_t,\pi}$, where $z^{J_1,\pi} >_L z^{J_2,\pi} >_L ... >_L z^{J_t,\pi}$ when the components are ordered by $\pi_1, \pi_2 ... \pi_n$. Let $l^{\mathbf{x},\rho}$ denote the index of the last 1 digit of the characteristic vector of the corresponding prime implicant, where ρ denotes the permutation of the components. Let $\mathcal{S}^*$ denote the strategy that inspects next component u_i where $i \in \{\sigma_i | y_i^{I_1,\sigma} = 1\} \bigcap \{\pi_i | z_i^{J_1,\pi} = 1\}$. Let us denote finally by $\mathcal{S}^0$ a 0-optimal strategy, and by $\mathcal{S}^1$ a 1-optimal strategy.

Lemma 5 *If $\mathbf{x} \in T(f)$ then $\mathcal{S}^*$ tests the same set of components as $\mathcal{S}^1$.*

Proof. Strategy $\mathcal{S}^1$ inspects components $\{u_{\sigma_1}, u_{\sigma_2} .... u_{\sigma_{l^{\mathbf{x},\sigma}}}\}$. Since $\mathcal{S}^1$ is 1-optimal, it is enough to show that $\mathcal{S}^*$ inspects a subset of $\{u_{\sigma_1}, u_{\sigma_2} .... u_{\sigma_{l^{\mathbf{x},\sigma}}}\}$ for any $\mathbf{x} \in T(f)$.

Since $l^{I_1,\sigma} \leq l^{\mathbf{x},\sigma}$, the first component inspected by $\mathcal{S}^*$ belongs to $\{u_{\sigma_1}, u_{\sigma_2}, ..., u_{\sigma_{l^{\mathbf{x},\sigma}}}\}$ Let us fix the value of the corresponding variable. The remaining function is still regular with the same order (disregarding the inspected component). Therefore, if we continue to apply the same strategy we will always inspect some u_{σ_i} such that $i \leq l^{\mathbf{x},\sigma}$. Hence the lemma follows. □

Lemma 6 *If $\mathbf{x} \in F(f)$ then $\mathcal{S}^*$ inspects the same set of components as $\mathcal{S}^0$.*

Proof. Same proof as above, just use f^d and its prime implicants. □

Theorem 2 *$\mathcal{S}^*$ is optimal.*

Proof. For an arbitrary strategy $\mathcal{S}$ the expected cost is $EC(\mathcal{S}) = EC^0(\mathcal{S}) + EC^1(\mathcal{S})$. By Lemmas 5 and 6 we have $C_0(\mathcal{S}^*, \mathbf{x}) = C_0(\mathcal{S}^0, \mathbf{x})$ and $C_1(\mathcal{S}^*, \mathbf{x}) = C_1(\mathcal{S}^1, \mathbf{x})$, thus $EC^0(\mathcal{S}^*) = EC^0(\mathcal{S}^0)$ and $EC^1(\mathcal{S}^*) = EC^1(\mathcal{S}^1)$. Here the right hand sides are minimal, hence $\mathcal{S}^*$ is optimal. □

If the double regular functions f and f^d are given by their DNF forms, then one can implement the algorithm by searching through the prime implicants both of f and of f^d to find the lexicographically largest ones, then selecting a component to be inspected in their intersection, and after testing the chosen component, by going through the prime implicants again to fix the value of the corresponding variable, and simplyfi f and f^d in this way. All these can be done in linear time in the number of prime implicants. Let us note that if the inspected component has the value 0, some prime implicants will disappear, and we can find the lexicographically largest prime implicants in the next step by using Lemma 4. If the inspected component has the value 1, then the lexicographically largest prime implicant is typically the same as before, (of course, the inspected component has to be excluded).

Lemma 4 also suggests a way to implement the procedure when the functions f and f^d are given by oracles (actually, an oracle for f is enough since we can calculate the value of f^d by using the oracle for f.) What we need to do in this case is to ask the value of f and f^d for at most n points at each step to find the lexicographically largest prime implicant. (These points should be some of the vectors $(1, 0, ..., 0)$, $(1, 1, 0..., 0)$, ..., $(1, 1, ..., 1)$ one-by-one in this order.) So, at each step of selecting the next component for inspection, we use the oracle at most $2n$ times. After this we have to take the intersection of the two lexicographically largest prime implicants we found, for which in the worst case we need $O(nlogn)$ comparisons. Since we inspect at most n components, we have at most n steps. Hence, to select the components for inspection in the worst case we need $2n^2$ oracle calls and $O(n^2logn)$ comparisons, in total.

In each step, after selecting the component, we inspect it and plug in its value to the function. After fixing the values of those components we have to ask the value of the function at two more points to figure out the correct value of the function.

We can confirm that the value of the function is 0, by making all the unfixed components equal to 1 and asking the oracle if the value of the functions is still 0. In case the function is 1, this can be done by making all the unfixed components equal to 0, and asking the oracle if f has the value 1. These additional questions add up to at most $2n$ oracle calls during the whole procedure, for this task. Therefore, in total we need $2n^2 + 2n$ oracle calls and $O(n^2logn)$ comparisons. (We assume that f satisfies the conditions that the procedure is optimal and f and f^d are given by oracles.)

Let us note that the strategy described as S^* earlier is the same strategy given by Inspect, since in steps 1 and 2, what happens is either we find the value of the function or otherwise we find the lexicographically largest prime (dual prime) implicants. Finally in step 3, if the value of the function is not determined yet, we choose the next component to be inspected from the intersection of these two lexicographically largest implicants of the primal and the dual. This is exactly how S^* was described to work. With the complexity analysis given above and by Theorem 2 we have proved Theorem 1.

7 Conclusion

In this report we propose a polynomial-time, minimum-expected cost algorithm for finding the correct value of a double regular boolean fuction when it is costly to observe its arguments. The algorithm does not require the input function to be double regular hence can be used as a heuristic algorithm for the general case. (It remains to be analyzed how large the error of such a procedure can be.)

Other problems of interest of this type, which has been studied and solved under restrictive assumptions, includes the case where the components are identical, i.e. the cost of inspecting any of the components is c and the probability that a component is working is p for all components, [1]. Another interesting case is when there are precedence constraints on the tests [16, 26]. Also the case when f is the structure function of a series-parallel network is of special interest [2].

References

[1] L. Arseneau. Optimal testing strategies for s,t-series parallel systems. Master's thesis, Combinatorics and Optimization, University of Waterloo, 1996.

[2] Y. Ben-Dov. Optimal testing procedures for special structures of coherent systems. *Management Science*, 27(12):1410–1420, 1981.

[3] P. Bertolazzi and A. Sassano. An $O(mn)$ time algorithm for regular set-covering problems. *Theoretical Computer Science*, 54:237–247, 1987.

[4] J. C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive boolean functions. *Information and Computation*, to appear, 1995.

[5] E. Boros. Dualization of aligned boolean functions. RUTCOR Research Report RRR 9, Rutgers University, RUTCOR, February 1994.

[6] E. Boros, P. L. Hammer, T. Ibaraki, and K. Kawakami. Identifying 2-monotonic positive boolean functions in polynomial time. In W. L. Hsu and R. C. T. Lee, editors, *ISA'91 Algorithms*, number 557 in Springer Lecture Notes in Computer Science, pages 104–115, Berlin-Heidelberg, 1991. Springer-Verlag.

[7] E. Boros, P. L. Hammer, T. Ibaraki, and K. Kawakami. Polynomial time recognition of 2-monotonic positive boolean functions given by an oracle. *SIAM Journal on Computing*, 26:93–109, 1997.

[8] R. Butterworth. Some reliability fault testing models. *Operations Research*, 20:335–343, 1972.

[9] C. L. Chang and J. R. Slage. An admissible and optimal algorithm for searching and-or graphs. *Artificial Intelligence*, 2:117–128, 1971.

[10] Ming-Feng Chang, Weiping Shi, and W. Kent Fuchs. Optimal diagnosis procedures for k-out-of-n structures. *IEEE Transactions on Computers*, 39(4):559–564, 1990.

[11] L. A. Cox, Jr., Yuping Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research*, 21:1–21, 1989.

[12] Y. Crama. Dualization of regular boolean functions. *Discrete Applied Mathematics*, 16:79–85, 1987.

[13] S. O. Duffua and A. Raouf. An optimal sequence in multicharacteristics inspection. *Journal of Optimization Theory and Applications*, 67(1):79–87, 1990.

[14] T. Eiter and B. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. Technical Report CD-TR 16, Technische Universität Wien, Christial Doppler Labor für Expertensysteme, 1991.

[15] M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. Technical Report LCSR-TR 225, Rutgers University, Department of Computer Science, 1994.

[16] M. R. Garey. Optimal task sequencing with precedence constraints. *Discrete Mathematics*, 4:37–56, 1973.

[17] R. Greiner. Finding optimal derivation strategies in redundant knowledge bases. *Artificial Intelligence*, 50:95–115, 1990.

[18] J. B. Kadane. Quiz show problems. *J. Math. Anal. Appl.*, 27:609–623, 1969.

[19] R. Kowalski. Search strategies for theorem proving. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, number 5, pages 181–201, Edinburgh, 1969. Edinburgh University Press.

[20] R. Kowalski. And-or graphs, theorem proving graphs and bi-directional search. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, number 7, pages 167–194, Edinburgh, 1972. Edinburgh University Press.

[21] K. Makino and T. Ibaraki. The maximum latency and identification of positive boolean functions. In D. Z. Du and X. S. Zhang, editors, *ISAAC'94 Algorithms and Computation*, number 834 in Lecture Notes in Computer Science, pages 324–332, Berlin–Heidelberg, 1994. Springer Verlag.

[22] N. J. Nilson. *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.

[23] U. N. Peled and B. Simeone. An $O(nm)$-time algorithm for computing the dual of a regular boolean function. *Discrete Applied Mathematics*, 49:309–323, 1994.

[24] I. Pohl. Bi-dircetional search. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, number 6, pages 127–140, Edinburgh, 1971. Edinburgh University Press.

[25] Yuping Qiu, L. A. Cox, Jr., and Lawrence Davis. Guess-and-verify heuristics for reducing uncertainities in expert classification systems. In *Proceedings of the Eighth Conference on Uncertainity in Artificial Intelligence*, 1992.

[26] H. A. Simon and J. B. Kadane. Optimal problem-solving search: All-ornone solutions. *Artificial Intelligence*, 6:235–247, 1975.

[27] R. O. Winder. *Threshold Logic*. Ph.d. dissertation, Princeton University, Department of Mathematics, 1962.