Sequential diagnosis of *k*-out-of-*n* systems with imperfect tests

Wei W, Talla Nobibon F, Leus R.

KBI_1315

Sequential diagnosis of k-out-of-n systems with imperfect tests

Wenchao Wei*, Fabrice Talla Nobibon†, Roel Leus*‡

Abstract. A k-out-of-n system configuration requires that, for the overall system to be functional, at least k out of the total of n components must be working. This paper addresses the search for a least-cost diagnosis strategy for k-out-of-n systems when the individual component tests are imperfect, which means that a test can identify a component as working when in reality it is down, and vice versa. The inspection procedure stops when the output accuracy exceeds a given threshold. We underline the importance of the possibility of fixing positive and negative predictive values, we examine different classes of diagnosis policies, we discuss global optimality of each of the classes, and we present a polynomial-time algorithm to find a globally optimal policy.

Keywords: system diagnosis, k-out-of-n systems, imperfect tests, sequencing and scheduling.

1 Introduction

System health monitoring for complex systems, such as a space shuttle, aircraft or integrated circuits, is crucial for reducing the likelihood of accidents due to sudden failures, and for improving system availability. It is also imperative that systems be tested before being put into operation, in order to ascertain their functionality. At manufacturing sites, for instance, products are typically inspected at the final stage of production before being shipped to the customer. Electronic equipment (smart mobile phones, laptops, etc.) in particular, which contains components from many different suppliers, has various tests executed throughout the different stages of manufacturing. The same inspection procedure may be repeated thousands of times, and so it is important to minimize the total expected costs in the long run. Additionally, besides this cost directly attributable to the test-set hardware and manpower, field return costs can be reduced by improving output quality through appropriate testing. In this article we search for an inspection *strategy*, which is a dynamic decision rule that decides in which order to test the components, and respects specific stopping criteria. More specifically, we develop algorithms for finding optimal strategies that minimize the expected testing expenses. Throughout the text, we use the terms 'strategy' and 'policy' interchangeably.

*ORSTAT, Faculty of Economics and Business, KU Leuven, Belgium.

†Federal Express (FedEx), Brussels, Belgium.

‡Corresponding author. E-mail: Roel.Leus@kuleuven.be. Postal address: ORSTAT, Faculty of Economics and Business, KU Leuven, Naamsestraat 69, B-3000 Leuven, Belgium. Tel. +32 16 32 69 67. Fax +32 16 32 66 24.

The k-out-of-n configuration is a special case of a complex system that requires that, for the overall system to be functional, at least k out of the total n components must be working. This configuration has a wide range of applications in both industrial and engineering systems [37], such as a multi-display system in a cockpit, the multi-radiator system in a heating system, and the multi-pump system in a hydraulic control system. Consider, for example, an airplane with four engines. Furthermore, suppose that the design of the aircraft is such that at least two engines are required to function for the aircraft to remain airborne. This means that the engines are related in a k-out-of-n configuration, with $k = 2$ and $n = 4$. This is in literature sometimes also referred to as a "2-out-of-4:G" system, where G means the system works or is "good"; a k-out-of-n:G system is equivalent to an $(n-k+1)$-out-of-n:F system, which fails ("F") if at least $(n-k+1)$ components fail. The airplane is tolerant to failures in up to two engines. More generally, the k-out-of-n system configuration represents systems with built-in redundancy. In the case of an automobile with four tires, for example, usually one additional spare tire is on-board the vehicle. Thus, the vehicle can be driven as long as at least four tires are in good condition, leading to a 4-out-of-5 system. A so-called *series* system of statistically independent components is an n-out-of-n system and a *parallel* system is a 1-out-of-n system.

In this paper, we study the case where individual component tests are *imperfect*, which means that a test can observe a component as working when in reality it is down, and vice versa; this can have severe implications. In several safety-critical systems, automatic recovery and reconfiguration mechanisms play a crucial role because an undiscovered fault may lead to a system failure. If a faulty unit is not reconfigured out of the system, it can produce incorrect results, which in turn can contaminate the non-faulty units [4]. In a computer's operating system, for example, system check-up may be wrongly alarmed by a fault that does not exist, and the system might operate improperly, possibly leading to extra memory usage. The consequences of test errors depend on the disposition of the system after testing. For example, if the outcome indicates a working system then the system will be put into operation, otherwise the system may be split into small parts, the faulty ones among which need to be repaired or replaced by functional parts. Obviously, different costs will be incurred in different ensuing situations, but these are neglected in this article: we only focus on the expected cost to assess the state of the system with a specific accuracy.

In this article, we will say that a *positive* test outcome is associated with the discovery of a failure. Imperfect testing can involve two types of test errors (see Figure 1). The

Figure 1: Comparing test outcomes with actual component states

		actual state	
		down	up
test outcome	positive	*true positive*	*false positive* (type-I error)
	negative	*false negative* (type-II error)	*true negative*

first type of error is *false positive* (type-I error), which means the tester concludes the component fails (is 'down') when really it is not; for example, a product can fail a quality test before being shipped, while in reality it is in good condition (component is 'up'). The second type of test error is *false negative* (type-II error), which means the outcome is negative but in reality the component fails, for example, a system check-up failing to detect the fault it was designed to find, in a computer system that really has the fault. *Positive predictive value* is the proportion of positive results that are truly positive; *negative predictive value* is the proportion of negative results that are truly negative. The following expressions show how these two values are related with type-I and type-II error:

$$\begin{aligned}
&\text{positive predictive value} = Pr\{\text{ component down} \mid \text{outcome positive }\},\\
&\text{positive negative value} = Pr\{\text{ component up} \mid \text{outcome negative }\},\\
&\text{type-I error rate} = Pr\{\text{ outcome positive} \mid \text{component up }\},\\
&\text{type-II error rate} = Pr\{\text{ outcome negative} \mid \text{component down }\},
\end{aligned}$$

where $Pr\{A|B\}$ is the probability of event A conditional on event B. Note that positive and negative predictive values are conditioned on the test results and can be used to estimate the probability of a fault in a single component. For definitions of similar terms in diagnostic tests, we refer to [2, 3].

As an illustration, suppose the Wi-Fi module of a smartphone is being tested. Historical data shows that $\chi = 81.25\%$ of the test outcomes is negative. Both the positive and negative predictive value are 0.9. Therefore, if the outcome is positive, we know that the module is not functional with probability 90% and is working with probability 10%; if the outcome is negative, the module is actually down with probability 10% and up with probability 90%. We can infer the a priori probability that the module works as $0.1(1-\chi) + 0.9\chi = 0.75$. The type-I error rate can then be computed as $0.1875 \times 0.1/0.75 = 0.025$ and the type-II error rate is $0.8125 \times 0.1/0.25 = 0.325$. In this example as well as throughout the article, we first fix the positive and negative predictive value and thereby also the complement α, which represents the probability that the outcome is wrong (here $\alpha = 10\%$). The type-I and type-II error rates then follow implicitly from this choice for α combined with either historical test data or with the a priori probability that the module is functional. This contrasts with the existing references to date (e.g., [14, 25]), where the type-I and type-II error rates are typically assumed to be known, which then implies a value for α. We argue that, in a number of applications, the approach described above and used in this paper is more appropriate: when a tester examines a component returning a positive result, he/she then wishes to evaluate what is the probability of the need to reject or repair this component. Type-I and type-II error rates cannot be directly used to answer this question, because they are conditional on whether the component is actually functional or not. Predictive values are therefore very useful measures of diagnostic accuracy in routine inspection practice.

Two types of testing strategies are defined by Butterworth [10]: *sequential* and *non-sequential*. Sequential then means that the testing order of the components is selected before the diagnosis begins, while non-sequential means the testing order is dynamic, in the sense that it can vary dependent on the results of the component tests. The terminology, however, varies between references (compare with Wald [39], for example), and in the general setting of *sequential system diagnosis*, the adjective 'sequential' simply refers to the fact that component tests are conducted one after the other, and never simultaneously (in a scheduling context [26], one could speak of single-machine scheduling). In order to avoid misunderstanding, we therefore resort to the use of the name *elementary* policy to refer to what Butterworth [10] defines as a sequential policy.

There are many sequential testing problems in the reliability literature, where the reliability of a system is the probability that the system functions. To evaluate the reliability, the assumption is that all components are examined; therefore the order in which the components are arranged for inspection does not influence system reliability or inspection costs. When the diagnostic procedure is actively monitored, however, sometimes not all components will need to be examined, and in this situation the sequencing of the tests does matter. We are aware of only one paper that studies the sequencing of imperfect component tests, namely Nachlas et al. [25]. They focus only on series systems, and it turns out that there always exists an optimal elementary policy that is globally optimal. To the best of our knowledge, however, the more general sequencing problem of imperfect tests for k-out-of-n systems has not yet been dealt with in the literature. It is the goal of this paper to fill exactly this gap. We assume that the diagnosis procedure halts once the tests results (output) satisfy a certain condition, namely that output accuracy exceed a threshold value, and the objective is to design an optimal testing strategy such that the total expected costs are minimized.

In the policy classes studied in this article, decisions are made dynamically, meaning that they are conditional on the observations (the outcomes) of the previous tests. As mentioned above, this dynamic character of our policy classes constitutes a very natural motivation for conditioning on the test outcomes (fixing positive and negative predictive value) rather than on the actual (hence, unknown) state of the components (as would be done by selecting type-I and type-II error rates). One can also imagine that under certain circumstances, testing the same component more than once would improve the quality of the output. In this paper, we do not consider such retesting: each component is tested at most once. This will allow us to focus only on the sequencing aspect.

The contributions of this article are fourfold: (1) we describe a general setting for k-out-of-n testing with imperfect tests; (2) we underline the importance of the possibility of fixing positive and negative predictive value first, rather than type-I and type-II error rates; (3) we examine different classes of diagnosis policies and discuss global optimality of each of the classes; and (4) we present a polynomial-time algorithm to find a globally optimal policy. In the process, we also define and analyze other problem variants of k-out-of-n testing. The remainder of this text is structured as follows: the next section (Section 2) provides a review of the relevant literature. Some definitions and a formal problem statement are given in Section 3. A number of observations and results regarding system guarantee are obtained in Section 4, and our main results are established in Section 5. We conclude the article in Section 6.

2 Literature review

An extensive literature review of different types of sequential testing problems can be found in Ünlüyurt [37]. Butterworth [10] shows that the special cases of a parallel ($k = 1$) and a series ($k = n$) system without precedence constraints are polynomially solvable. A polynomial-time algorithm for arbitrary k was presented first in [31], and independently in [8]. Efficient implementations of this algorithm were proposed in [11] (off-line algorithm requiring $O(n^2)$ space and $O(n^2)$ time) and in [32] (on-line algorithm requiring $O(n)$ space and $O(n\log(n))$ time). With general precedence constraints, the testing problem for series systems is NP-hard [12, 22]. Computational results for sequencing tests of k-out-of-n systems with general precedence constraints can be found in [40]. One specific variant of classic k-out-of-n testing is the so-called *conservative* k-out-of-n testing, which is defined in the same way, but testing now continues until either k successful tests are observed or until all n tests have been performed [20].

Most research efforts in the system-testing literature have been directed at finding testing strategies for systems with $k = 1$ and $k = n$ (parallel and series systems) and with special precedence constraints (e.g. a series-parallel and a tree precedence graph). Relatively less attention has been given to imperfect testing, where due to the defect of the test design or unexpected errors, the outcome of a test may not always indicate real conditions of the component but only with a specific probability, otherwise giving the adverse information or even no information. The optimization of sequential search processes with imperfect tests has already been modeled and applied to various domains; we provide a survey below.

Imperfect testing has been introduced for so-called 'search problems.' In the *discrete search problem with a stationary target* [1], an item is assumed to be hidden in one of a set of boxes. Associated with each box i is a prior probability p_i ($\sum p_i = 1$) that the item is hidden in that box and the overlook probability (type-II error) a_i that the item will not be found in a particular search of that box even though the item is actually there. Value a_i remains the same for every search of box i, and the time (cost) consumed in examining box i is c_i. The search procedure does not stop before the item is found. Bellman [7] was the first to describe an optimal policy to minimize the expected search cost by arranging the components in descending order of the ratio $p_i(1-a_i)/c_i$. Using this result, Glus [17] develops an optimal procedure for detecting the breakdown in a complex multi-component system. The stop criterion is the same: the fault can be concealed due to test errors, but is ultimately discovered by repeating tests until it is properly isolated. Wagner and Davis [38] extend the module concept proposed by Gluss [17] for discrete sequential search and refer to it as a 'group activity.' Variations of stationary-object search are addressed in [34, 35].

The discrete search problem can also be seen as a representation of system testing. Besides the cost of testing, which depends on the set of components that are actually tested, Nachlas et al. [25] also consider the consequences of a test error dependent upon the disposition of the system after repair. If a false positive test result occurs, a functioning component is replaced and the failed component is left in place. If the system is then returned to service, the system fails immediately. If a false negative test result occurs, the overall test could indicate that no item fails; if the system is then returned to service, it fails immediately or it might be scrapped. Nachlas et al. add these two events with corresponding cost coefficient into the objective function and perform efficient enumeration of permutations of test sequences to find an optimal one that minimizes the expected total costs for small systems (less than 10 components) with series structure. Note that the false negative results in a *no fault found* situation, whose corresponding cost is independent of the test sequence. Therefore, this cost is ignored in the design of the heuristic method.

Raghvan et al. [28] study the *single-fault detection problem* with unreliable tests, where the fault is inherited from a finite failure source, and there is a finite set of available tests, each of which checks a subset of failure sources. The problem is to design a test policy with minimum expected total diagnostic cost to isolate the failure source with a specified confidence (typically within $[0.95, 0.99]$). This problem is treated as a partially observed Markov decision problem (POMDP), and is solved by a continuous-state dynamic programming recursion. Different fault detection problems are considered in [6, 30].

In the testing department of a typical manufacturing company, for example a semiconductor manufacturer, the objective of the testing process is to achieve a high outgoing-product quality while minimizing the costs of testing, scrapping conforming products, and the opportunity cost of passing non-conforming items. Since the tests are imperfect, applying the same test multiple times is useful for obtaining higher outgoing quality and economic savings. Raouf et al. [29] develop a model where accepted components are repetitively retested and determine the optimal number of repeat inspections for multi-characteristic components to minimize the total expected cost per accepted component due to type-I error, type-II error and cost of inspection. Greenberg and Stokes [18] use the data acquired from retesting rejected items to estimate the probability of testing errors. Note that this stream of literature allows type-I and type-II error rates of tests vary after each test cycle. Ding et al. [14] examine the question whether it is better to repetitively test rejected components or to repetitively test accepted components. Choosing between these two policies depends on the tradeoff between scrapping costs and outgoing quality; see also [13, 27] for variations of this problem. Sequencing issues and imperfect test have both been considered in the inspection of multi-characteristic components; we refer to [29, 33, 36] for examples. The results in the foregoing references, however, do not apply for the case where retesting is not allowed or economically infeasible. Solutions have been published for the latter case also, albeit mainly in a different field; this is the subject of the next paragraph.

The main concern of the reliability literature is the evaluation or the approximation of the reliability of a given system; see for instance Wu and Chen [41] for weighted k-out-of-n systems, Ding et al. [15], who develop approximation procedures for the reliability of multi-state weighted k-out-of-n systems, or Eryilmaz [16], who analyzes the case when components have random weights. The reliability literature has also addressed test sequencing issues, but only for very specific testing assumptions; see [9], for instance. With respect to sequencing with imperfect test information, we are only aware of Nachlas et al. [25], who study series systems. For a k-out-of-n configuration with imperfect information, most of the literature has focused on the design of the system such that it strikes a balance between reliability and cost. In general, the reliability of a k-out-of-n system for fixed k increases with the number n of components. As the required reliability of the system increases, the cost also goes up due to the increase in the number of redundant (idle) components in the system. Marseguerra et al. [24] develop an approach to incorporate uncertainty (component failure probabilities are not known with certainty) into reliability calculations by using Monte-Carlo simulation and genetic algorithms. Amari et al. [4] also the design of systems with built-in redundancy, including k-out-of-n subsystems subjected to imperfect fault coverage (type-II error, see [5] for the definition of fault coverage).

3 Definitions and problem statement

3.1 Definitions

We consider a system consisting of n components, $N = \{1, 2, \ldots, n\}$. In order to discover the state of the system, we can test each component sequentially on a single test machine. Each component is in one of two *states*: either working (up) or not working (down). The system functions (succeeds) if at least k ($k \leq n$) out of the n components are working and malfunctions (fails) if at least $(n-k+1)$ components are not working. Each component i is tested at most once, a setting that has been called [25] 'single-pass' testing. We refer to this single test of component i as 'test i.' The *outcome* of test i is written as $x_i \in \mathbb{B} = \{0, 1\}$ and is also binary: it is either positive or negative; positive (corresponding with $x_i = 0$) means the test detects a fault, negative ($x_i = 1$) means the opposite. All outcomes can be gathered in an n-dimensional binary vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{B}^n$.

We study the situation where the measurements (tests) are imperfect: the outcome can be positive while the component is actually working, and vice versa. We assume that each test has a probability α of producing the wrong outcome; we will refer to α as the *predictive error rate.* Such a common error probability α for all component tests may for instance be inherent in the design of a single unreliable test machine. Specifically, for component i, if the test result is positive then we know the actual state of i is down with probability $(1-\alpha)$ and up with probability α, and mutatis mutandis for a negative outcome. We assume α to be less than 0.5, which seems to be a property of any reasonable testing procedure and which has been mentioned for real-world examples [19]. We define χ_i as the probability that the outcome of test i is negative (x_i equals 1), and p_i is the a priori probability that component i works; it holds that $p_i = \chi_i(1-2\alpha)+\alpha$ ($i = 1, 2, \ldots, n$). For ease of notation, we also define $\lambda_i = 1-\chi_i$ as the probability that test outcome i is positive. Let X_i represent outcome i before testing, which is a Bernoulli random variable with parameter χ_i, and denote by $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ the associated vector of random variables. The realization of each X_i is known only at the end of test i. We assume all variables to be mutually independent.

A *schedule* $\mathbf{s} = (s_1, s_2, \ldots, s_{|\mathbf{s}|})$ is an ordered subset of N; s_t is the index of the test in position t in the schedule $\mathbf{s}$. Let Σ be the set of all schedules, and value c_i represents the cost of test i. We define cost function $f(\mathbf{s})$ for schedule $\mathbf{s}$ as $f(\mathbf{s}) = \sum_{t=1}^{|\mathbf{s}|} c_{s_t}$.

3.2 Problem statement

A solution to the sequencing problem under study is a testing policy, which uses diagnosis experience from test outcomes gathered over time. At each stage, the policy prescribes the next step to be taken: either select a new test to conduct, or stop; the stop/continue decision when not all components have been tested yet is based on the accuracy regarding the system's state. After r tests have been applied according to schedule $\mathbf{s}$, and based on the outcomes $\mathbf{x}$, we conclude that the system functions with a probability $\theta_1(r, \mathbf{x}, \mathbf{s})$ and malfunctions with a probability $\theta_0(r, \mathbf{x}, \mathbf{s})$; we will use $\theta_1(r)$ and $\theta_0(r)$ for short. We study the optimization problem of designing a test policy Π with minimum expected total diagnostic costs. Given a specified threshold value T on system guarantee (e.g., $T = 95\%$), the inspection procedure stops when either $\theta_0(r)$ or $\theta_1(r)$ exceeds T; if this stop criterion is never fulfilled then we perform all the n tests (which maximizes the accuracy of the result). Below, we will use the terms 'guarantee' and 'accuracy' interchangeably; in the context of isolating a single failure source, Raghavan et al. [28] refer to this guarantee as a 'level of confidence.' The value of T should be determined from historical data or by the requirements of the system.

Table 1 contains the parameters of an example 3-out-of-5 system, and we consider a predictive error rate $\alpha = 10\%$. For a given outcome $(1, 1, 1, 1, 1)$ and schedule $(1, 2, 3, 4, 5)$, we have $\theta_1(4) = 94.77\%$ and $\theta_1(5) = 99.14\%$. In case we work with guarantee $T = 95\%$, we will not stop after the first four component tests because the obtained accuracy is not

Table 1: Costs and probabilities of an example 3-out-of-5 instance

i	1	2	3	4	5
c_i	1	1	1	1	1
p_i	$0.5+\epsilon$	0.5	0.5	0.5	$0.5-\epsilon$

yet high enough, and so we also test the fifth component; after this fifth test, $\theta_1(5) > T$ and so we conclude that the system is working. Had we worked with $T = 90\%$, however, then $\theta_1(4) > T$ and the fifth test would not have been necessary to achieve the (lower) desired accuracy threshold.

Following the literature on stochastic scheduling [21], a policy Π can also be seen as a function $\Pi : \mathbb{B}^n \to \Sigma$ that maps outcomes $\mathbf{x}$ to schedules. Note that several outcomes may be mapped to the same schedule. Our objective is to find a policy Π^* within a specific class that minimizes $\mathbb{E}[f(\Pi(\mathbf{X}))]$, with $\mathbb{E}[\cdot]$ the expectation operator with respect to $\mathbf{X}$. Specifically,

$$\mathbb{E}[f(\Pi(\mathbf{X}))] = \sum_{\mathbf{x}\in\mathbb{B}^n} \left(\prod_{i:x_i=1} \chi_i\right)\left(\prod_{i:x_i=0} \lambda_i\right) f(\Pi(\mathbf{x})). \tag{1}$$

We call a policy *globally optimal* if it achieves the minimum expected cost over all possible policies. Tests are conducted one at a time, so a sequence of (a subset of) tests indeed defines a schedule; this is a dominant decision for our objective function (but the same is not true for other objectives such as makespan minimization).

The flexibility inherent in a testing policy allows to schedule different tests dependent on the outcomes of the previously conducted tests: the second component to be tested, for instance, might depend on the result (success or failure) of the first test. To capture this dynamic nature, a policy Π can also be represented by a binary decision tree (BDT). In such a BDT, each non-leaf node is labeled with the index of a component to be tested

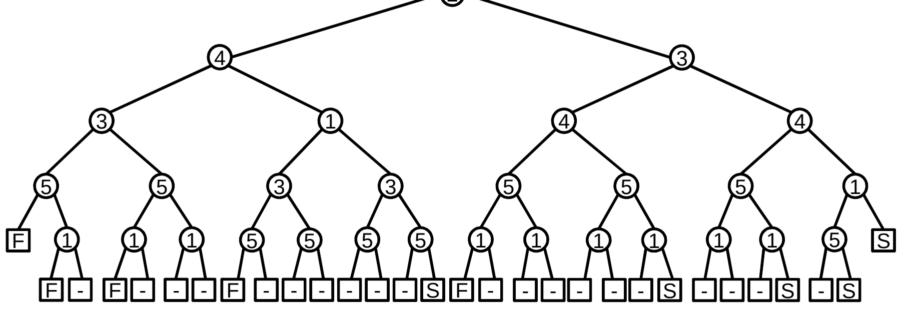

Figure 2: A globally optimal policy of the example instance

and has two child nodes. If the test outcome is positive then the left branch is entered, otherwise the right subtree is taken. Each leaf node either represents a conclusion of a specific system state (successful (S) or failed (F)) with the required accuracy, or indicates that all components have been tested but the state of the system is not identified (labeled by $-$). For a given outcome, the policy generates the schedule stepwise from the root node to a leaf node. Figure 2 depicts a BDT representing a testing policy for the example instance of Table 1 with $T = 90\%$; its expected cost is $4.875 - 0.25\epsilon$.

We call a policy *dominant* if for any two vertices u_1 and u_2 with the same tested component set in the BDT representation of the policy (meaning that the same components were tested in the diagnosis process to reach u_1 and u_2) and with the same number of negative test outcomes in the preceding tests, it holds that the subtrees rooted by u_1 and u_2 are identical (except that the components tested at the root may be different).

Lemma 1. *There always exists a dominant policy that is globally optimal.*

Proof: Consider the BDT of a globally optimal policy Π^* and suppose that there are two vertices u_1 and u_2 in the tree that have the same tested component set (as in the definition of dominant policies) but with different subtrees rooted from u_1 and u_2, so that Π^* is not dominant. By replacing u_1 together with its subtree by u_2 together with its subtree we obtain a different policy Π_1, and similarly we can also replace u_2 and subtree by u_1 and subtree to obtain Π_2. It can be seen that the objective function value of either Π_1 or Π_2 will be at most as high as that of Π^*. By retaining the best policy of either Π_1 or Π_2 and applying similar changes until a dominant policy is obtained, we arrive at a dominant globally optimal policy. The accuracy threshold will also be reached in exactly the same leaf nodes because the accuracy depends only on the number of conducted tests and the number of positive outcomes (see the discussion in Section 4.2). □

Inspired by priority policies for standard scheduling problems [26], we also study *elementary policies*, which are characterized by a total order of the component set N. Such an order can be represented by a list (or permutation) $L = (l_1, l_2, \ldots, l_n)$ of the elements of N. The policy tests the components one by one in the order of the list (so first l_1, then l_2, etc.) until the system's guarantee reaches the threshold value. For a given outcome $\mathbf{x}$ of a k-out-of-n system with imperfect tests, the elementary policy Π characterized by a list L generates a unique schedule $\Pi(\mathbf{x}; L)$ by iterating through the list from left to right. The schedule stops when either θ_1 or θ_0 reaches threshold value T or all the components have been tested. The list $(1, 2, 3, 4, 5)$, for example, characterizes an elementary policy for the instance of Table 1, with corresponding expected costs 4.875

for $T = 90\%$. Clearly, one direct advantage of elementary policies is their compact representation: we can also draw up a BDT for an elementary policy, but this is not necessary: the list provides all the information that is needed.

4 System guarantee

4.1 Computing the θ-values

We define $\kappa^{(\mathbf{x},\mathbf{s})}(w, r)$ as the probability that there are w components actually working among the first r components tested (consequently, $r - w$ components are actually down) according to schedule $\mathbf{s}$ and outcome $\mathbf{x}$. We use $\kappa(w, r)$ for short in the following. We initialize $\kappa(0, 0) = 1$, and $\kappa(w, r) = 0$ when $w = -1$ or $w > r \geq 0$, and we propose a recursive formula to generate each $\kappa(w, r)$ for $w = 0, 1, \ldots, n$; $r = 1, 2, \ldots, n$ with $w \leq r$:

$$\begin{aligned}\kappa(w, r) = \kappa(w, r-1)(x_{s_r}\alpha + (1 - x_{s_r})(1-\alpha)) \\ + \kappa(w-1, r-1)(x_{s_r}(1-\alpha) + (1 - x_{s_r})\alpha).\end{aligned} \tag{2}$$

All other $\kappa(w, r) = 0$. Table 2 gives an illustration of $\kappa(w, r)$ for a given outcome $\mathbf{x} = (1, 0, 1, 0)$ and schedule $\mathbf{s} = (1, 2, 3, 4)$. The time complexity of calculating $\kappa(n, r)$ is $O(nr)$; therefore, all the $\kappa(w, r)$ together can be computed in time $O(n^4)$. The values of $\theta_1(r)$ and $\theta_0(r)$ can now be calculated as follows:

$$\theta_1(r) = \begin{cases} 0 & \text{if } r < k \\ \sum\limits_{w=k}^{r} \kappa(w, r) & \text{if } r \geq k \end{cases} \tag{3}$$

$$\theta_0(r) = \begin{cases} 0 & \text{if } r < n-k+1 \\ \sum\limits_{w=0}^{r-n+k-1} \kappa(w, r) & \text{if } r \geq n-k+1 \end{cases} \tag{4}$$

Proposition 1. *For $0 \leq r \leq n$, we have $\theta_1(r) + \theta_0(r) \leq 1$; equality holds when $r = n$.*

Proof: Using Equation (3) and (4), we have

$$\begin{aligned}\theta_1(r) + \theta_0(r) &\leq \sum_{g=k}^{r} \kappa(g, r) + \sum_{g=0}^{(k-1)+(r-n)} \kappa(g, r) \\ &\leq \sum_{g=0}^{n} \kappa(g, n) = 1.\end{aligned}$$

Table 2: Values of $\kappa(g, r)$ for outcome $\mathbf{x} = (1, 0, 1, 0)$ and schedule $\mathbf{s} = (1, 2, 3, 4)$

$r =$ $x_{s_r} =$	1 1	2 0	3 1	4 0
$w = 0$	α	$\alpha(1-\alpha)$	$\alpha^2(1-\alpha)$	$\alpha^2(1-\alpha)^2$
$w = 1$	$1-\alpha$	$\alpha^2 + (1-\alpha)^2$	$\alpha^3 + 2\alpha(1-\alpha)^2$	$2\alpha^3(1-\alpha) + 2\alpha(1-\alpha)^3$
$w = 2$	0	$\alpha(1-\alpha)$	$2\alpha^2(1-\alpha) + (1-\alpha)^3$	$\alpha^4 + 2\alpha^2(1-\alpha)^2$
$w = 3$	0	0	$\alpha(1-\alpha)^2$	$2\alpha^3(1-\alpha) + 2\alpha(1-\alpha)^3$
$w = 4$	0	0	0	$\alpha^2(1-\alpha)^2$
$\sum_{w=0}^{4} \kappa(w, r) =$	1	1	1	1

It is easy to verify that when $r = n$ (all tests are applied), $\theta_1(r) + \theta_0(r) = 1$ holds. □

A direct consequence is:

Corollary 1. *If $\theta_1(n) > 1 - T$ and $\theta_0(n) > 1 - T$ then the threshold value T will never be reached.*

We now describe some further properties of $\theta_1(r)$ and $\theta_0(r)$.

Proposition 2. *$\theta_1(r)$ and $\theta_0(r)$ are increasing functions of r.*

Proof: We wish to establish that $\theta_1(r+1) \geq \theta_1(r)$ for $0 \leq r \leq n-1$. If $0 \leq r < k$ then $\theta_1(r) = 0$, so $\theta_1(r+1) \geq \theta_1(r)$ is always true. If $k \leq r \leq n-1$, then using Equation (3) we have

$$\theta_1(r+1) = \sum_{w=k}^{r+1} \kappa(w, r+1).$$

Substituting $\kappa(w, r+1)$ by $\kappa(w, r)$ and $\kappa(w-1, r)$ according to Equation (2), we obtain

$$\begin{aligned}\theta_1(r+1) = &\sum_{w=k}^{r+1} \kappa(w, r)(x_{s_{r+1}}\alpha + (1 - x_{s_{r+1}})(1-\alpha)) \\ &+ \sum_{w=k}^{r+1} \kappa(w-1, r)(x_{s_{r+1}}(1-\alpha) + (1 - x_{s_{r+1}})\alpha).\end{aligned}$$

Taking $\kappa(r+1, r)$ and $\kappa(k-1, r)$ out of the first and second summation, respectively, we have

$$\begin{aligned}\theta_1(r+1) = \kappa(r+1,r)(x_{s_{r+1}}\alpha + (1-x_{s_{r+1}})(1-\alpha)) + \sum_{w=k}^{r} \kappa(w,r)(x_{s_{r+1}}\alpha + (1-x_{s_{r+1}})(1-\alpha)) \\ + \kappa(k-1,r)(x_{s_{r+1}}(1-\alpha) + (1-x_{s_{r+1}})\alpha) + \sum_{w=k}^{r} \kappa(w,r)(x_{s_{r+1}}(1-\alpha) + (1-x_{s_{r+1}})\alpha).\end{aligned}$$

With $\kappa(r+1,r) = 0$, this reduces to

$$\begin{aligned}\theta_1(r+1) &= \kappa(k-1,r)(x_{s_{r+1}}(1-\alpha) + (1-x_{s_{r+1}})\alpha) + \sum_{w=k}^{r} \kappa(w,r) \\ &= \kappa(k-1,r)(x_{s_{r+1}}(1-\alpha) + (1-x_{s_{r+1}})\alpha) + \theta_1(r) \\ &\geq \theta_1(r).\end{aligned}$$

We conclude that $\theta_1(r)$ increases with r. An analogous reasoning can be developed for $\theta_0(r)$. □

Informally, the foregoing proposition states that both positive and negative outcomes contribute to both $\theta_1(r)$ and $\theta_0(r)$.

4.2 Comparing the θ-values with the threshold T

For the example instance of given in Table 1 with $T = 95\%$, for another outcome $(1,1,1,1,0)$ and schedule $(1,2,3,4,5)$, we have $\theta_1(4) = 94.77\% < T$ and $\theta_1(5) = 95.2\% > T$. Contrary to what we suggested in Section 3.2, it is wise to stop immediately after observing four negative results in this case because the threshold value will certainly be reached anyway if the remaining last test is applied. Following similar arguments, it can be seen that it is also optimal to stop after observing four positive results. In this section, we will formalize this insight.

Furthermore, performing more tests than when $\alpha = 0$ (perfect testing) is potentially helpful for improving the confidence level; it may be, for instance, that the testing will halt when only $k+1$ negative outcomes and zero positive ones are observed. It is also possible, however, that the diagnosis procedure stops after observing $k-1$ negative and three positive results, since both positive and negative outcomes will increase $\theta_1(r)$ (Proposition 2). The decisions to be made by the tester are therefore: (1) when to halt the testing procedure, and (2) how to sequence the tests such that the total expected costs are minimized. In the remainder of this section, we will answer the first question; we develop a solution to the second question in the next two sections.

In Section 4.1 we defined the accuracy as dependent on one specific pair of outcome and schedule. We observe, however, that as long as the number r of conducted tests is the same and the number of positive outcomes is the same (even if the set of components examined differs), the corresponding guarantee will be identical (see Table 2). We therefore redefine the guarantees $\theta_1(r, t)$ and $\theta_0(r, t)$ as dependent on the number r of tested components and the number t of observed negative outcomes (note the change in arguments). We provide more information on the calculation of $\theta_1(r, t)$ and $\theta_0(r, t)$ below.

Consider the instance in Section 3.2 with $\alpha = 40\%$ and $T = 45\%$. For outcome $(1, 1, 1, 0, 0)$ we have $\theta_1(5, 3) = 53.86\% > T$ and $\theta_0(5, 3) = 45.12\% > T$, whereas both $\theta_1(4, 3) < T$ and $\theta_0(4, 3) < T$. Clearly, this is a situation that renders the classification of the system highly confusing because the thresholds for concluding system success and system failure are reached at the same time. From Proposition 2, we infer that by choosing $T > 50\%$ we will never conclude that the system works and fails at the same time. With the following two lemmas, we provide an alternative testing diagnosis based on simply counting the number of successful and unsuccessful outcomes such that more information is included in the stopping criterion and the foregoing problem cannot occur even with $T \leq 50\%$.

It can be seen that (for $\alpha < 0.5$) if $t_1 < t_2$ then $\theta_1(r, t_1) < \theta_1(r, t_2)$. Define a $(0 \leq a \leq n)$ as the smallest integer such that

$$T \leq \theta_1(n, a). \tag{5}$$

Lemma 2. *If the outcome of the current test is the a-th negative one then it is a dominant decision to stop the test procedure immediately.*

Proof: Assume that after the first r $(r \leq n)$ tests, a negative outcomes are observed and the last one is negative, then the probability the system functions is $\theta_1(r, a)$. If $\theta_1(r, a) \geq T$ then the threshold is attained and the procedure can be interrupted. Conversely, if $\theta_1(r, a) < T$ then the procedure can also be halted because we know that $\theta_1(n, a) \geq T$, which means if we test the remaining components, no matter their outcome, the threshold value T will be reached anyway. □

This lemma also implies that when there is no integer a that fulfills the condition in Equation (5), then it is known before the start of the system diagnosis that we will never reach the conclusion that the system functions with the required accuracy threshold T.

Next we define the integer $b, 0 \leq b \leq n$, as the lowest integer such that

$$T \leq \theta_0(n, n - b). \tag{6}$$

We have the following lemma, which is a counterpart of Lemma 2.

Lemma 3. *If the outcome of the current test is the b-th positive one then it is a dominant decision to stop the test procedure immediately.*

It also follows that when b does not exist then the conclusion of system failure can never be drawn with the required accuracy. If neither a nor b can be found then the testing sequence becomes irrelevant because the diagnosis will conduct all the n tests under all possible outcomes.

Pathological instances can be created, however, where $a < k$ and/or $b < n - k + 1$. Consider, for instance, $k = 1$, $T = 20\%$ and $\alpha = 40\%$; then $a = 0$. From Equation (5), we know that $T \leq \theta_1(n, 0)$ if and only if $a = 0$. From this observation, we derive the condition based on which $a = 0$ holds: $\binom{n}{k}(1-\alpha)^{n-k}\alpha^k + \ldots + \binom{n}{n}\alpha^n \geq T$. In this case, the system can always be identified as working, even without testing any components. Similar argument holds for the case $b = 0$; the corresponding condition is $T \leq \theta_0(n, n)$, which is equivalent with $\binom{n}{n-k+1}(1-\alpha)^{k-1}\alpha^{n-k+1} + \ldots + \binom{n}{n}\alpha^n \geq T$. In the remainder of this article, we will always assume that T is large enough such that both a and $b > 0$.

For most "natural" parameter values, the number of successful and unsuccessful observations required to halt the diagnosis will be at least as high as in the perfect testing situation (with $\alpha = 0$). For the example instance of Section 3.2 with $T = 90\%$, it can be seen that $a = b = 4$ (with $k = 3$, $n = 5$). In the result below, we formally distinguish the parameter settings that are, in some sense, undesirable.

Lemma 4. *If $a+b \leq n$ then there exist outcomes $\mathbf{x}^*$ and policies Π_1 and Π_2 such that Π_1 under $\mathbf{x}^*$ will first observe a working components while less than b failing components are found (and thus conclude that the system functions with the pre-specified accuracy T), whereas Π_2 under $\mathbf{x}^*$ will first observe b failing components while less than a working components are found (and thus halt the diagnosis with the conclusion of system failure). This phenomenon will never occur when $a + b > n$.*

Proof: Given a and b with $a + b \leq n$, construct outcome $\mathbf{x}^*$ as follows: $x_1 = x_2 = \ldots = x_a = 1$ and $x_{a+1} = x_{a+2} = \ldots = x_n = 0$, and consider policy Π_1 that tests the components in increasing index. Clearly, Π_1 will observe a successes under $\mathbf{x}^*$ while observing 0 failures, so if $b > 0$ then Π_1 indeed concludes to system success. Similarly, if Π_2 conducts tests in decreasing index, it will reach the conclusion of system failure as long as $a > 0$. When $a + b > n$, on the other hand, we cannot construct the vector $\mathbf{x}^*$ anymore. □

In conclusion, when $a + b > n$ then the identification of the system state only depends on the test outcomes but not on the sequence in which the tests are executed. This is no longer true, however, when $a+b \leq n$; the latter situation therefore requires modifications in the problem statement (e.g., different stopping criteria or different statement of system identification), and we will not study this situation further in this paper.

4.3 Searching for a and b

We described a recursive method for evaluating $\theta_1(r, \mathbf{x}, \mathbf{s})$ and $\theta_0(r, \mathbf{x}, \mathbf{s})$ in Section 4.1. By combining different outcomes with similar guarantee, however, we can also establish $\theta_1(r,t)$ and $\theta_0(r,t)$ under closed form. Define $\dot{\kappa}(g;t,r)$ as the probability that there are g working components in reality under the observation of t negative results out of r tested components. Clearly, $\dot{\kappa}(g;t,r) = 0$ for $g > r$ or $t > r$. For $g, t, r = 0, 1, 2, \ldots, n$ and $g \leq r, t \leq r$, we have:

Proposition 3.

$$\dot{\kappa}(g;t,r) = \begin{cases} \sum\limits_{l=0}^{\min\{r-g,t\}} (1-\alpha)^{r-m-2l} \alpha^{m+2l} \binom{r-t}{m+l} \binom{t}{l} & \text{if } g \geq t \\ \sum\limits_{l=0}^{\min\{g,r-t\}} (1-\alpha)^{r-m-2l} \alpha^{m+2l} \binom{r-t}{l} \binom{t}{m+l} & \text{otherwise.} \end{cases} \tag{7}$$

Proof: We define N_1 as the set of t components whose outcomes are negative and N_0 containing the $(r-t)$ components with positive outcomes. We focus only on the case $g \geq t$ (the case $g < t$ can be interpreted similarly). Let h represent the number of incorrect test results; this value is at least $m = g - t$ (if $g < t$, then $m = t - g$). We consider the different possible values for h, which correspond with different terms in the summation in Equation (7) via the relation $h = m + 2l$, with l the summation index:

- $h = m$, then all the m incorrect results are from N_0. There are $\binom{r-t}{m}$ ways of choosing m components in N_0 (m false positive observations);
- $h = m + 2$, then $m + 1$ incorrect results are from N_0 and one is from N_1. There are $\binom{r-t}{m+1}$ ways of choosing $m + 1$ components in N_0 ($m + 1$ false positive) and $\binom{t}{1}$ ways of choosing one component in N_1 (one false negative);
- $\ldots$;
- $h = m + 2 \times \min\{r - t - m, t\}$ is the maximum number of incorrect results that can be observed (all components in either N_0 or N_1 have incorrect outcomes). If $r - t - m = r - g < t$, then all tests from N_0 and $r - g$ tests from N_1 are incorrect, and there are $\binom{t}{r-g}$ ways of choosing $r - g$ components in N_1; if $r - g \geq t$, then

$m+t$ tests from N_0 and all tests from N_1 are incorrect, and there are $\binom{r-t}{m+h}$ ways of choosing $m+h$ components in N_0. □

Using the $\dot{\kappa}$-values, for $0 \leq t \leq n$, we have

$$\theta_1(n,t) = \sum_{i=k}^{n} \dot{\kappa}(i;t,n)$$

and

$$\theta_0(n,t) = \sum_{i=0}^{k-1} \dot{\kappa}(i;t,n).$$

If $\theta_1(n,n) \leq T$, then a exists and can be found by comparing T with different θ-values. The search for b can be proceed analogously.

The complexity of calculating $\dot{\kappa}$-values based on Equation (7) is mainly determined by computing the binomial coefficients. Specifically, $\binom{y}{x}$ can be efficiently calculated recursively, with space complexity $O(x)$ and time complexity $O(xy)$ (see Knuth [23]). Overall, the time complexity of finding a and b is O(n4)."

5 An optimal algorithm

In this section, we show that the testing problem with imperfect tests is polynomially solvable, and that a globally optimal solution for some special cases can be represented by a permutation of all the components. We investigate two different settings, namely the case where either a or b does not exist and the case where they both exist. In Section 4.2 we pointed out already that the situation where neither a nor b exists, makes all sequencing decisions irrelevant.

5.1 Either a or b does not exist

If one of the parameters a and b does not exist (but the other one exists) then the k-out-of-n testing problem with imperfect tests as defined in Section 3.2 reduces to *conservative k-out-of-n testing*, which is defined similarly as traditional k-out-of-n testing (with perfect tests), except that we perform tests either until we have observed k tests with negative outcome, or we have performed all n tests [20]. In particular, the diagnosis is not interrupted after observing $n-k+1$ tests with positive outcome.

Proposition 4. *For conservative k-out-of-n testing with perfect tests, the elementary policy represented by the permutation of all n components arranged in nondecreasing* *order of c_i/χ_i is optimal.*

Proof: We define $\pi(U)$ as a permutation of the components in $U \subseteq N$ arranged in nondecreasing order of ratio c_i/χ_i. We first observe that when $k = 1$, the classic and the conservative 1-out-of-n testing problems are actually the same. Consequently (see, for instance, [10]), $\pi(N)$ defines an optimal elementary policy for the conservative testing problem with $k = 1$.

We now use induction on the number n to prove this proposition. When $n = 1$, $\pi(N)$ is indeed optimal because there is only one sequence. When $n = 2$, $\pi(N)$ is optimal for 1-out-of-2 testing and both possible sequences for 2-out-of-2 conservative testing have the same expected costs, namely $c_1 + c_2$, therefore $\pi(N)$ is optimal regardless of the value of k.

Assume now that $\pi(U)$ is optimal for l-out-of-$|U|$ conservative testing with $|U| \leq n-1$ and for all l between 1 and $|U|$; we will show that $\pi(N)$ (with $n = |N|$) is optimal for arbitrary k ($k \geq 2$). Let $\pi_i(U)$ be the sequence with component i in the first position, appended with sequence $\pi(U \setminus \{i\})$ (which we know is optimal for the smaller instance with component set $U \setminus \{i\}$). We will prove that $\pi(N)$ is optimal by showing that

$$\mathbb{E}[f(\Pi_k^E(\pi(N); \mathbf{X}))] \leq \mathbb{E}[f(\Pi_k^E(\pi_i(N); (\mathbf{X}))]$$

for $i = 1, \ldots, n$, and with $\Pi_l^E(L; \mathbf{x})$ the elementary policy for a system with $k = l$ and defined by permutation L (applied to outcome $\mathbf{x}$). For short, below we write $\mathbb{E}[f(\Pi_l^E(L; \mathbf{X}))]$ as $\mathbb{E}_l[L]$.

For $i = 1$, clearly $\pi(N) = \pi_i(N)$, so $\mathbb{E}_k[\pi(N)] = \mathbb{E}_k[\pi_i(N)]$. For $i \neq 1$, note that $\pi_i(N) = (i, 1, 2, \ldots, i-1, i+1, \ldots, n)$, and $\mathbb{E}_k[(i, 1, 2, \ldots, i-1, i+1, \ldots, n)] = \mathbb{E}_k[(1, i, 2, \ldots, i-1, i+1, \ldots, n)]$, because the first $k \geq 2$ components in a sequence will be tested in any case, so interchanging the position of component 1 and i in the test sequence does not alter the total expected costs. Also, by the induction hypothesis, $\mathbb{E}_l[(i, 2, \ldots, i-1, i+1 \ldots, n)] \geq \mathbb{E}_l[(2, 3, \ldots, n)]$ for $l = k$ and $l = k-1$ (where both sequences contain $n-1$ components). Therefore, we have:

$$\begin{aligned}
\mathbb{E}_k[\pi_i(N)] &= \mathbb{E}_k[(1, i, 2, \ldots, i-1, i+1, \ldots, n)] \\
&= c_1 + \chi_1 \mathbb{E}_{k-1}[(i, 2, \ldots, i-1, i+1 \ldots, n)] + \lambda_1 \mathbb{E}_k[(i, 2, \ldots, i-1, i+1 \ldots, n)] \\
&\geq c_1 + \chi_1 \mathbb{E}_{k-1}[(2, \ldots, n)] + \lambda_1 \mathbb{E}_k[(2, \ldots, n)] \\
&= \mathbb{E}_k[\pi(N)].
\end{aligned}$$

Overall, we have $\mathbb{E}_k[\pi(N)] \leq \mathbb{E}_k[\pi_i(N)]$. □

Proposition 5. *For conservative k-out-of-n testing with perfect tests, an optimal elementary policy is also globally optimal.*

Proof: The proof follows the same lines as the proof of Proposition 4. For the base case, when $n = 1$ and $n = 2$, the result is obvious, because any policy is elementary.

Assume now that an elementary policy is also globally optimal for l-out-of-m conservative testing with $m \leq n-1$ and for any value of l. Then for a conservative k-out-of-n system, among policies that start with testing component i, elementary policy $\Pi_k^E(\pi_i(N), \cdot)$ has minimum expected costs $c_i + \chi_i \mathbb{E}_{k-1}[(1,\ldots,i-1,i+1,\ldots,n)] + \lambda_i \mathbb{E}_k[(1,\ldots,i-1,i+1,\ldots,n)]$, where both policies to which the expectation pertains, relate to a system with $n-1$ components and correspond with an identical sequence (from Proposition 4). Therefore, $\pi(N)$ defines an elementary policy that is globally optimal, regardless of the value of k. □

Corollary 2. *When exactly one of the parameters a or b does not exist then the imperfect k-out-of-n testing problem is polynomially solvable.*

5.2 Both a and b exist

Chang et al. [11] propose a polynomial algorithm for the classic k-out-of-n (perfect) testing problem. We use their algorithm in Section 5.2.2, and for the sake of completeness, we first summarize their algorithm below in Section 5.2.1.

5.2.1 The algorithm of Chang et al.

Consider the k-out-of-n testing problem with perfect tests. Relabel the components such that:

$$\frac{c_1}{p_1} \leq \frac{c_2}{p_2} \leq \cdots \leq \frac{c_n}{p_n}$$

and let σ be a permutation such that:

$$\frac{c_{\sigma(1)}}{1-p_{\sigma(1)}} \leq \frac{c_{\sigma(2)}}{1-p_{\sigma(2)}} \leq \cdots \leq \frac{c_{\sigma(n)}}{1-p_{\sigma(n)}}.$$

For any $U \subseteq N$, define $V_i(U)$ to be the subset of U containing the i components with smallest index, so $V_i(U) = \{j \in U | 1 \leq j \leq i\}$. Also, let $F_i(U)$ be the subset of components in U that occupy the first i positions in σ: $F_i(U) = \{\sigma(j) \in U | 1 \leq j \leq i\}$.

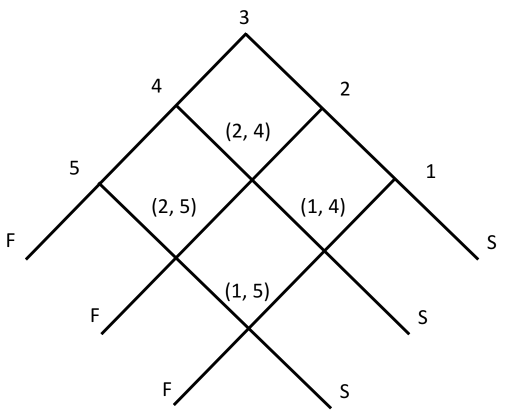

Figure 3: The block-walking representation of a globally optimal policy for the 3-out-of-5 example instance of Section 3.2 with $\alpha = 0$

Finally, we define $SS(U)$ as the component in U with the smallest index, so $SS(U)$ is the single element in $V_1(U)$.

The following procedure from reference [11] produces a globally optimal policy for this problem. Start with testing component $i = SS(V_k(N) \cap F_{n-k+1}(N))$. If component i is working, then we choose component $SS(V_{k-1}(N \setminus \{i\}) \cap V_{n-k+1}(N \setminus \{i\}))$ to test; otherwise we select component $SS(V_k(N \setminus \{i\}) \cap F_{n-k}(N \setminus \{i\}))$ as the next component to be examined. Following this method recursively, we continue the test procedure until either k working components or $n - k + 1$ failing components are found.

The BDT representation of an optimal policy generated by the above procedure can be simplified to a so-called *block-walking* representation. See Figure 3 for an illustration, which represents a globally optimal policy for the example instance with perfect tests. Testing proceeds from top to bottom along the grid. The line crossings on the outside of the grid are labeled by the index of the component to be tested, the nodes at the bottom represent the identification of the state of the system, and the inner crossings are marked by a pair of components, the first one of which will be tested if the previously tested component is working and the second one will be tested if the previously tested component is not working. Similarly as for the BDT representation, if a component is found to be working then the left downward line segment is taken, otherwise the right segment is selected.

5.2.2 Generalized testing

We define a new testing problem (with perfect tests) called *generalized testing problem*, defined similarly as the classic k-out-of-n testing problem in that an instance is defined by a component set N with parameters c_i and p_i, but instead of parameter k a generalized testing instance takes two parameters k_1 and k_0 and the system diagnosis continues until either k_1 working components or k_0 failing components are found. Clearly, the classic k-out-of-n testing problem is a special case of the generalized problem where $k_1 = k$ and $k_0 = n - k + 1$. Similarly, the conservative k-out-of-n testing problem is a subproblem of generalized testing with $k_1 = k$ and $k_0 = n$. We now turn back to k-out-of-n testing with imperfect tests. If both a and b can be found, then the problem cannot directly be reduced to conservative testing but we end up with a generalized (perfect) testing problem with $k_1 = a$ and $k_0 = b$.

The generalized testing problem with $k_0+k_1 = n+1$ is equivalent with classic (perfect) k-out-of-n testing, and so is polynomially solvable by the algorithm of Section 5.2.1. The following theorem contains a generalization of this observation. We will not consider $k_0+k_1 \leq n$ in this text, since this situation exhibits similar ambiguities as those mentioned for the case $a + b \leq n$ in Section 4.2.

Theorem 1. *The generalized testing problem is polynomially solvable when $k_0 + k_1 \geq n + 1$.*

Proof: For an arbitrary instance of generalized testing with $k_0+k_1 \geq n+1$, we construct an instance of classic k-out-of-n' testing, as follows. The component set includes all components of the generalized instance but we add $(k_0+k_1-n-1) \geq 0$ identical dummy components to the system, resulting in a system with $n' = k_0 + k_1 - 1$ components in total, to which we can apply the algorithm of Chang et al. We wish to make sure that the costs of the dummy components are very high such that the dummy components have the highest ratios of both c/p and c/q; each cost can for instance be chosen as $\max\{\frac{c_n}{p_n}; \frac{c_{\sigma(n)}}{1-p_{\sigma(n)}}\}$ and the probability of success as 0.5. In this way, the dummy components will not be tested until all the ordinary components (inherited from the generalized testing instance) have been inspected, and they will always be sequenced last in the diagnosis procedure. Therefore, the probability that dummy components will be tested is independent of the sequence of the ordinary components, such that the contribution of testing all dummy components to the objective function can be calculated independently and remains the same in any optimal policy. Removing the dummy components from the optimal policy then leads to an optimal policy for the original generalized testing instance. $\square$

We define the class of policies generated according to the procedure described in the foregoing proof as *interrupted block-walking* (IBW) policies (they are block-walking policies that are indeed 'interrupted' by removing the dummy components used in the reduction). The following result ensues directly from Theorem 1:

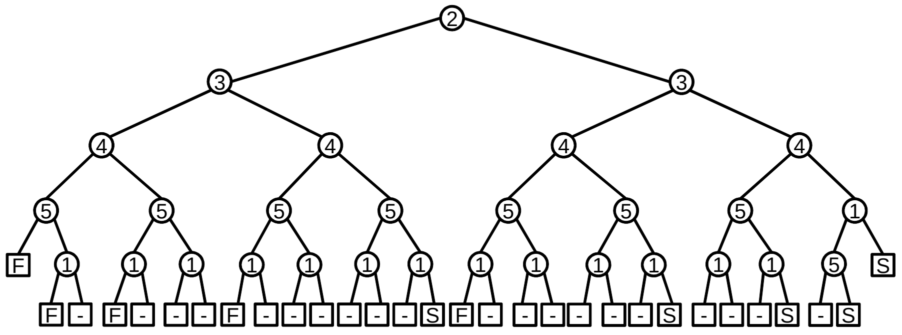

Figure 4: Another globally optimal policy of the example instance in Section 3.2

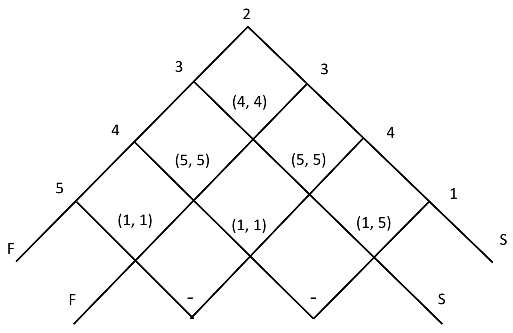

Figure 5: The IBW representation of the globally optimal dominant policy in Figure 4

Corollary 3. *When $a + b \geq n + 1$ then there always exists a globally optimal IBW policy for the imperfect k-out-of-n testing problem; such a policy can be found in polynomial time.*

The class of IBW policies combines the advantages of dominant and of elementary policies, namely global optimality and compact representation (polynomial in n). We note that the class of IBW policies is a subset of the class of dominant policies, but the reverse is not true. For the example instance, the dominant policy depicted in Figure 2 is not an IBW policy. Figure 4 gives another globally optimal dominant policy that does belong to the IBW class, and its compact representation is shown in Figure 5. As before, the bottom end points of the edges in the grid are labeled by a symbol from S, F and $-$, which indicate the system is working, not working or the system's state remains unidentified, respectively (for the specified accuracy threshold).

5.2.3 Counterexample for global optimality of elementary policies

Elementary policies have the benefit of compact representation, but they may 'miss' the global optimum. For the example instance of Section 3.2 with $T = 95\%$ we have $a = b = 4$ by Equations (5) and (6). The globally optimal IBW policy depicted in Figure 5 has expected total costs $4.875 - 0.25\epsilon$, while an optimal elementary characterized by list $(1, 2, 3, 4, 5)$ has expected costs 4.875, which is strictly higher than the global optimum.

6 Summary and outlook on future work

In this article, we have tackled the problem of diagnosing k-out-of-n systems when the tests are imperfect. With fixed positive and negative predictive values and with some restrictions on the values of the parameters, we can transform this problem into a generalized testing problem with perfect tests; we also explain why the combinations of parameter values that are not covered by the transformation are actually undesirable. We describe a polynomial-time algorithm for the generalized testing problem with perfect tests, which implies that the k-out-of-n test sequencing problem is also polynomial (again given specific assumptions on the parameters). We examine different policy classes for the imperfect testing problem, namely the class of dominant policies, of elementary policies and of interrupted block-walking policies. The class of dominant policies always contains a global optimum, while elementary policies are compact in representation (polynomial in the number of components). Interrupted block-walking policies have the merits of both global optimality and of compactness.

For further research, it may be interesting to take retesting into consideration: finished products that do not pass the quality inspection (positive test outcome), for instance, may be retested to reduce the scrapping cost. Accepted components may also be retested to improve outgoing quality. More generally, performing a component test more than once will allow the tester to obtain more information about the system under study. This option was not explored in this article because it requires additional modeling assumptions. A second option for further work to pursue is a different interpretation of imperfect testing, in which the outcome of a particular test might also be unknown or indeterminate instead of simply wrong. This setting has already received some attention in recent literature, see for instance Balakrishnan and Semmelbauer [6].

Appendix A Notation: overview

N the set of components with $N = \{1, 2, \dots, n\}$

c_i costs of testing component i

p_i probability that component i is working; $q_i = 1 - p_i$

χ_i probability that outcome of test i is positive; $\lambda_i = 1 - \chi_i$

α probability that an outcome is wrong

$\mathbf{X}$ $= (X_i, \ldots, X_n)$ test outcome vector, with X_i a Bernoulli random variable

$\mathbf{x}$ $= (x_1, x_2, \ldots, x_n), x_i \in \{0, 1\}$, one realization of vector $\mathbf{X}$

$\mathbf{s}$ $= (s_1, s_2, \ldots, s_{|\mathbf{s}|}), s_t \in N, 1 \leq t \leq |\mathbf{s}|$, schedule indicating the test sequence for a specific outcome

$f(\mathbf{s})$ cost function for schedule $\mathbf{s}$; $f(\mathbf{s}) = \sum_{t=1}^{|\mathbf{s}|} c_{s_t}$

$\theta_1(\cdot)$ probability that the system functions, conditional on the progress of the diagnosis

$\theta_0(\cdot)$ probability that the system fails, conditional on the progress of the diagnosis

T threshold value on system accuracy

References

[1] R. Ahlswede and I. Wegener. *Search Problems.* Wiley, New York, 1987.

[2] D. Altman and J. Bland. Statistics notes: Diagnostic tests 1: sensitivity and specificity. *British Medical Journal*, 308:1552, 1994.

[3] D. Altman and J. Bland. Statistics notes: Diagnostic tests 2: predictive values. *British Medical Journal*, 309:102, 1994.

[4] S. V. Amari, H. Pham, and G. Dill. Optimal design of k-out-of-n:G subsystems subjected to imperfect fault-coverage. *IEEE Transactions on Reliability*, 53(4):567–575, 2004.

[5] T. Arnold. The concept of coverage and its effect on the reliability model of a repairable system. *IEEE Transactions on Computers*, 22:251–254, 1973.

[6] A. Balakrishnan and T. Semmelbauer. Circuit diagnosis support system for electronics asssembly operations. *Decision Support Systems*, 25(4):251–269, 1999.

[7] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

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

[9] D. A. Butler and G. J. Lieberman. Inspection policies for fault location. *Operations Research*, 32(3):566–574, 1984.

[10] R. Butterworth. Some reliability fault-testing models. *Operations Research*, 20(2):335–343, 1972.

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

[12] B. De Reyck and R. Leus. R&D-project scheduling when activities may fail. *IIE Transactions*, 40(4):367–384, 2008.

[13] J. Ding and L. Gong. The effect of testing equipment shift on optimal decisions in a repetitive testing process. *European Journal of Operational Research*, 186(1):330–350, 2008.

[14] J. Ding, B. Greenberg, and H. Matsuo. Repetitive testing strategies when the testing process is imperfect. *Management Science*, 44(10):1367–1378, 1998.

[15] Y. Ding, M. J. Zuo, A. Lisnianski, and W. Li. A framework for reliability approximation of multi-state weighted k-out-of-n systems. *IEEE Transactions on Reliability*, 59(2):297–308, 2010.

[16] S. Eryilmaz. On reliability analysis of a k-out-of-n system with components having random weights. *Reliability Engineering and System Safety*, 109:41–44, 2013.

[17] B. Gluss. An optimum policy for detecting a fault in a complex system. *Operations Research*, 7(4):468–477, 1959.

[18] B. Greenberg and S. Stokes. Repetitive testing in the presence of inspection errors. *Technometrics*, 37:102–111, 1995.

[19] R. K. Gunnarsson and J. Lanke. The predictive value of microbiologic diagnostic tests if asymptomatic carriers are present. *Statistics in Medicine*, 21:1773–1785, 2002.

[20] L. Hellerstein, O. Özkan, and L. Sellie. *Max-Throughput for (Conservative) k-of-n Testing*, volume 7074 of *Lecture Notes in Computer Science*, chapter 72, pages 703–713. Springer Berlin Heidelberg, 2011.

[21] G. Igelmund and F. Radermacher. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, 13:1–28, 1983.

[22] F. P. Kelly. A remark on search and sequencing problems. *Mathematics of Operations Research*, 7(1):154–157, 1982.

[23] D. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part* 1. Addison-Wesley Professional, 2011.

[24] M. Marseguerra, E. Zio, L. Podofillini, and D. Coit. Optimal design of reliable network systems in presence of uncertainty. *IEEE Transactions on Reliability*, 54(2):243–253, 2005.

[25] J. Nachlas, S. Loney, and B. Binney. Diagnostic-strategy selection for series systems. *IEEE Transactions on Reliability*, 39(3):273–280, 1990.

[26] M. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer, 2012.

[27] R. Quinino, E. Colin, and L. Ho. Diagnostic errors and repetitive sequential classifications in on-line process control by attributes. *European Journal of Operational Research*, 201(1):231–238, 2010.

[28] V. Raghavan, M. Shakeri, and K. Pattipati. Test sequencing algorithms with unreliable tests. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, 29(4):347–357, 1999.

[29] A. Raouf, J. Jain, and P. Sathe. A cost-minimization model for multicharacteristic component inspection. *IIE Transactions*, 15:187–194, 1983.

[30] S. Ruan, F. Y. Y. Zhou, K. R. Pattipati, P. Willett, and A. Patterson-Hine. Dynamic multiple-fault diagnosis with imperfect tests. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 39(6):1224–1236, 2009.

[31] S. Salloum. *Optimal testing algorithms for symmetric coherent systems.* PhD thesis, University of Southern California, 1979.

[32] S. Salloum and M. Breuer. Fast optimal diagnosis procedures for k-out-of-n: G systems. *IEEE Transactions on Reliability*, 46(2):283–290, 1997.

[33] J. W. Schmidt and G. K. Bennett. Economic multiattribute acceptance sample. *AIIE Transactions*, 4(3):194–199, 1972.

[34] N. Song and D. Teneketzis. Discrete search with multiple sensors. *Mathematical Methods of Operations Research*, 60(1):1–13, 2004.

[35] L. Stone, J. Stanshine, and C. Persinger. Optimal search in the presence of poisson-distributed false targets. *SIAM Journal on Applied Mathematics*, 23(1):6–27, 1972.

[36] K. Tang and J. Tang. Design of screeing procedures: a review. *Journal of Quality Technology*, 26(3):209–226, 1994.

[37] T. Ünlüyurt. Sequential testing of complex systems: A review. *Discrete Applied Mathematics*, 142:189–205, 2004.

[38] B. Wagner and D. Davis. Discrete sequential search with group activities. *Decision Sciences*, 32(4):557–573, 2001.

[39] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[40] W. Wei, K. Coolen, and R. Leus. Sequential testing policies for complex systems under precedence constraints. *Expert Systems with Applications*, 40:611–620, 2013.

[41] J. Wu and R. Chen. An algorithm for computing the reliability of weighted-k-out-of-n systems. *IEEE Transactions on Reliability*, 43(2):3276–328, 1994.