The Stochastic Test Collection Problem: Modeling and Solution Approaches

Yifat Douek-Pinkovich, Irad Ben-Gal, Tal Raviv
Department of Industrial Engineering, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel
E-mail: yifatdouek@gmail.com, bengal@tauex.tau.ac.il, talraviv@tauex.tau.ac.il

November 2019

Abstract

The well-studied Test Collection Problem (TCP) selects a minimal set of binary tests needed to correctly classify the state of a system. This model has applications in various domains, such as the design of monitoring systems in engineering, communication and healthcare. In this paper, we define the *Stochastic Test Collection Problem* (STCP) that generalizes the TCP. While the original TCP assumes that each combination of the outputs of the tests can be mapped into a class, in the STCP, these combinations are mapped to probability distributions over the classes. Moreover, each test and each type of classification error is associated with some cost. The objective is to select a subset of tests that minimize the weighted sum of the tests' costs and the expected cost of the classification errors. The STCP further generalizes the TCP by allowing general categorical results of the tests rather than binary ones. We apply three metaheuristic methods for the STCP, namely, Tabu Search (TS), Cross-Entropy (CE), and Binary Gravitational Search Algorithm (BGSA). These methods were tested on realistic and publicly available datasets and shown to be successful by comparing their solutions to the optimal ones obtained by an exhaustive search. The latter procedure requires tremendous computational effort and is therefore applicable only to relatively small instances of the problem. The solutions of larger instances, as obtained using the three heuristics methods, were also compared with each other.

Keywords: Combinatorial optimization, The test collection problem, Cross-entropy, Tabu search, Binary gravitational search algorithm

1 Introduction

Consider the following settings. One is given a set of feasible tests with categorical outputs and all possible combinations of these tests' results for a population of tested subjects (e.g., patients). Each such combination of the tests' results is called *reading* and is associated with a discrete probability distribution over a given finite (and typically small) set of *classes* (e.g., diagnoses). Each reading is also associated with a-priori probability (i.e., relative frequency in the population). Each of the tests is associated with a *testing cost* and each possible type of *classification error* is associated with an *error cost*. A classification error type A-B is the error of classifying a subject as B, while it actually belongs to A. We refer to a subset of selected tests as a *configuration.* A possible combination of the results of a given configuration is called a *signature*. Clearly, given a small configuration, different readings may result in an identical signature and hence are indistinguishable. The question of adding additional tests in this case is economical – considering both the testing costs and the error costs as signatures into a class with the goal of minimizing the sum of the expected classification errors and testing costs. The tradeoff between the two components of the objective function is straightforward, since often executing more tests (or more accurate and expensive ones) is likely to reduce the likelihood of classification errors and thus their excepted cost, but increase the total testing cost (e.g., adding more blood tests to improve the patient's diagnosis). The challenge of (and the interest in) the problem stems from the fact that the classification can rarely be inferred from the signature of a single test. Instead, it is concluded from a combination of the results obtained from different tests – the signature – and even then there is a chance of error. A formal definition of the problem requires additional notation and is introduced in Section 2.

The STCP generalizes the well-studied Test Collection Problem (TCP), as discussed in Garey and Johnson (1979), Halldórsson et al. (2001) and De Bontridder et al. (2003), to name only few examples. Most importantly, instead of a one-to-one relation between readings and classes, it allows a probabilistic many-to-many relation and thus enables various readings-to-class mappings, as often happens in reality. In such a case, often a deterministic diagnosis is impossible, hence the need to introduce the classification error cost into the objective function. Moreover, the original version of the TCP assumes that the cost of all the tests is identical, and this merely seeks to minimize the number of tests instead of the most cost-effective ones. Clearly, the TCP is a special case of the STCP, where each reading is associated deterministically with one class, the costs of all the tests are set to one, and the classification error costs are set to prohibitively large numbers (e.g., larger than the number of tests). Note that the TCP is known to be NP-Hard (Garey and Johnson, 1979) and APX-Hard (Halldórsson et al. 2001, De Bontridder et al. 2003). Therefore, in this study we are not trying to generate an efficient, exact algorithm for the STCP, but we focus rather on effective heuristic methods to solve it.

While the TCP, and its stochastic generalization presented here, can be well motivated by a medical testing problem (e.g., selecting the most cost-effective panel of routine medical tests), equivalent problems may occur when monitoring any complex system; for example, water networks, manufacturing plants, greenhouse facilities, vehicles and, naturally, the human body. Modern monitoring systems often comprise many sensors connected to a central processing unit that detects and classifies the

current state of the system. It is often the case that the information obtained from the sensors is noisy and cannot be mapped deterministically to a single class. A major task in the design phase of these systems is to select a subset of sensors with a minimal cost out of potentially very large (and expensive) sets.

To be comparable with the error costs, the testing cost, in our context, is the cost of using the sensor for a single reading. For example, the cost associated with the installation of the sensors should be amortized in terms of a single usage, the equivalent of performing one test.

The STCP extends a previous problem introduced by Bertolazzi et al. (2016) and studied by Douek-Pinkovich et al. (2019) that generalizes the TCP by introducing costs for the sensors (tests). Many authors have studied various applications of the test collection problem, some of them using terminology and formulations that differ somewhat from ours. Such applications include sensor placement for structure, e.g., in Kammer (1991), Sela et al. (2016); robotics, e.g., in Hovland and McCarragher (1997); energy consumption strategies, e.g., in Slijepcevic and Potkonjak (2001); medical diagnosis, e.g., in Wendt and Potkonjak (2011); process monitoring, e.g., in Bacher and Ben-Gal (2017), among others.

In this paper, we apply three metaheuristic methods to solve the STCP, namely, Tabu Search (TS), Cross-Entropy (CE), and Binary Gravitational Search Algorithm (BGSA). We also solve a few small instances of the problem by enumerating all the possible testing configurations and evaluate their solutions in order to obtain an optimal benchmark solution to evaluate the proposed heuristic algorithms.

Tabu Search (TS) is a well-studied and frequently used metaheuristic proposed by Glover (1989) as a general extension of classical local search techniques to overcome local optimum convergences. TS uses ‘intelligence’ to direct the iterative search in a prospective and promising direction. The effectiveness of TS depends on how adaptive memory is used to direct the exploration process. The power of the methodology is illustrated, for example, by Cangalovic et al. (1996), who applied TS to a combinatorial assignment problem. Drezner et al. (2001) and Pacheco et al. (2009) used TS to select a subset of descriptive variables that yields the greatest percentage of hits in a regression model. This method was successfully applied in corporate bankruptcy predictions, credit scoring and other forecasting applications.

The Cross-Entropy (CE) method comprises of a suite of techniques and algorithms for rare-event simulation, importance sampling and combinatorial optimization problems (COP). The method was first introduced by Rubinstein (1997) for the efficient estimation of rare event probabilities in stochastic networks. It was later recognized that one can also apply it to (heuristically) solve hard combinatorial optimization problems, such as the traveling salesman and max-cut (see Rubinstein and Kroese, 2004, and De Boer et al., 2005). Moreover, the CE method was successfully applied to stochastic optimization problems. For example, Alon et al. (2005) successfully implemented the CE method in the buffer allocations problem.

The Gravitational Search Algorithm (GSA) is a heuristic solution method introduced by Rashedi et al. (2009) and inspired by the law of gravity and mass interactions. In this algorithm, the search agents represent a collection of masses, and their interactions are based on the Newtonian laws of gravity and motion. The method was extended to solve combinatorial optimization problems with binary variables by

Rashedi et al. (2010). The extended version is referred to as the Binary Gravitational Search Algorithm (BGSA). Papa et al. (2011) combined the BGSA with a classifier method to provide a fast and accurate framework for feature selection. The BGSA was also applied to nonlinear optimization problems, such as the optimal power flow problem, as seen in Duman et al. (2012).

The Feature Selection (FS) problem is related to the STCP in the sense that in both problems, the goal is to select a set of characteristics of an object that enables its proper classification. However, the two methods differ in their objective and implementation. The STCP considers the tradeoff between the cost of the information obtained for the classification as well as the expected cost of misclassification. On the other hand, in feature selection, the goal is to select a subset of characteristics that minimizes the chances of misclassification due to overfitting or underfitting. Saeys et al. (2007) provided a basic taxonomy of feature selection techniques.

In the naïve Bayes classification literature, Chai et al. (2004) studied the tradeoff between the costs of the tests and misclassification. They presented a greedy procedure in which one new test is selected at each step, based on its potential information gain and cost. The authors also extended this method to batch testing strategy, when several tests are performed at each step. Ling et al. (2004) used a similar approach in decision trees, where the goal is to minimize the expected total testing and misclassification costs.

The contributions of this paper are in introducing the stochastic variant of the TCP and in presenting effective solution methods for it that follow some of the above approaches.

The rest of the paper is organized as follows. In Section 2, we present some formal notation and mathematical formulation of the problem. In Section 3, we demonstrate the properties of the problem using a small illustrative example. Sensitivity analysis is carried out to demonstrate some counterintuitive properties of the problem and its optimal solutions. In Section 4, we present the heuristic methods to solve the STCP based on the TS, CE, and BGSA metaheuristics. In Section 5, our solution methods are tested and compared based on realistic data from the UCI Machine Learning Repository (Dua and Graff, 2019). Some concluding remarks are offered in Section 6.

2 Notation and problem definition

The mathematical formulation of the STCP is based the following notation.

N	Set of candidate tests available in a given system; the number of tests is denoted by $n = \|N\|$. $S \in N$ is a subset of selected tests called *configuration*.
c_i	The cost of test i for all $i \in N$.
V_i	The set of outputs/results that can be obtained from test i.
R	The set of valid readings, $R \subseteq V_1 \times V_2 \times \cdots \times V_n$; for each reading $\tilde{\mathbf{r}} \in R$ we refer to the result of the i^{th} test by $\tilde{r}_i$.
K	The set of possible classes $K = \{1, \dots, k\}$.
λ_{kl}	Misclassification error of type (k, l), $k, l \in K$, i.e., the cost of classifying an object as class l while its true class is k.

$p(\tilde{\mathbf{r}})$ The a-priory probability of obtaining the reading $\tilde{\mathbf{r}} \in R$.

$p(k|\tilde{\mathbf{r}})$ The conditional probability of class $k \in K$ given the reading $\tilde{\mathbf{r}} \in R$.

Let us denote the a-priori probability of each class by $p(\mathrm{k})$ and the conditional probability of each reading given a class by $p(\tilde{\mathbf{r}}|\mathrm{k})$. These two probability values are thus related to each other by Bayes' rule,

$$p(\mathrm{k}) = \sum_{\tilde{\mathbf{r}} \in R} p(\mathrm{k}|\tilde{\mathbf{r}}) p(\tilde{\mathbf{r}}) \tag{1}$$

$$p(\tilde{\mathbf{r}}|\mathrm{k}) = \frac{p(\mathrm{k}|\tilde{\mathbf{r}}) p(\tilde{\mathbf{r}})}{p(\mathrm{k})} \tag{2}$$

For each configuration, $S \subset N$, define $R(S)$ as the set of all its signatures, i.e., partial readings that can be obtained from the results of the selected tests. When the configuration is known, we denote the signature of $\tilde{\mathbf{r}}$ by **r.** The signature $\mathbf{r} \in R(S)$ is a vector of dimension $|S|$. For convenience, the elements, r_i, of the signature vectors, are indexed by the original indices of the tests in N. For example, if $N = \{1,\ldots,5\}$ and $\mathbf{r} \in R(\{1,2,5\})$, then $\mathbf{r} = (r_1, r_2, r_5)$. In this example, r_5 is the third element of **r**. We denote the set of all the possible readings from which signature **r** can be obtained when the configuration is S, by $Q(S, \mathbf{r})$. That is, $Q(S, \mathbf{r}) = \{\tilde{\mathbf{r}} \in R: \tilde{r}_i = r_i \ \ \forall i \in S\}$.

Next, for each configuration $S \subset N$ and $\mathbf{r} \in R(S)$, it is possible to calculate the following three probability components: the probability of a signature given class k, $p_S(\mathbf{r}|k)$; the prior probability $p_S(\mathbf{r})$ of signature $\mathbf{r} \in R(S)$; and the a-posteriori probability $p_S(k|\mathbf{r})$ that the class is $k \in K$ given that the signature is $\mathbf{r} \in R(S)$. These probabilities can be calculated using Equations (3) -(5). For configuration S the conditional probability of class k when observing a signature **r**, $p_S(k|\mathbf{r})$, is calculated in Equation (5) using Bayes' rule.

$$p_S(\mathbf{r}|k) = \sum_{\tilde{\mathbf{r}} \in Q(S,\mathbf{r})} p(\tilde{\mathbf{r}}|k) \tag{3}$$

$$p_S(\mathbf{r}) = \sum_{\tilde{r} \in Q(S,\mathbf{r})} p(\tilde{\mathbf{r}}) = \sum_{k \in K} p(k) \cdot p_S(\mathbf{r}|k) \tag{4}$$

$$p_S(k|\mathbf{r}) = \frac{p_S(\mathbf{r}|k) \cdot p(k)}{p_S(\mathbf{r})} \tag{5}$$

Consider a given configuration $S \subset N$. The expected classification error cost for signature **r** of S if it is mapped to class l is:

$$E_S(\mathbf{r}|l) = \sum_{k \in K} \lambda_{kl} \cdot p_S(k|\mathbf{r}) \tag{6}$$

let $l_S^*: R(S) \longrightarrow K$ be a function that maps each possible signature of S to a class that minimizes the expected classification error cost.

$$l_S^*(\mathbf{r}) = \underset{l \in K}{\operatorname{argmin}} \{E_S(\mathbf{r}|l)\} \tag{7}$$

The minimum expected classification error cost for signature **r** is:

$$E_S^*(\mathbf{r}) = \min_{l \in K} \{E_S(\mathbf{r}|l)\} \tag{8}$$

Note that (7) coincides with the "minimum Bayes risk decision rule," as found in Duda et al. (2012).

The STCP can now be formulated mathematically. Namely, given an instance of the problem $[N, R, K, \mathrm{p}(\tilde{\mathbf{r}}), \mathrm{p}(k|\tilde{\mathbf{r}}), \boldsymbol{\lambda}, \mathbf{c}]$, select a configuration S, such that the total expected classification error and tests costs is minimized,

$$\min_{S \subseteq N} \left\{ \sum_{\mathbf{r} \in R(S)} p_S(\mathbf{r}) E_S^*(\mathbf{r}) + \sum_{i \in S} c_i \right\} \tag{9}$$

Note that given the set of sensors, S, the set of signatures is uniquely defined by $R(S)$ and the optimal mapping of each signature to a class is give by (7).

3 Motivating example

Let us demonstrate the problem by the following small example. Consider a medical testing system comprising three potential tests aimed at detecting a viral disease. Each test produces a binary result, i.e., the result of medical test i may be either $V_i = 0$ or, 1. The input of this instance in terms of the notation presented in Section 2 is:

N	$\{1,2,3\}$
c_i	$[2,\ 0.5,\ 0.5]$
V_i	$\{0,1\}$ for $i = 1,2,3$; the result of each test can be either 0 or 1.
R	$V_1 \times V_2 \times V_3$; all possible combinations of the tests results. See also the first group of columns in Table 1.
K	$\{\text{Negative}, \text{Positive}\}$
λ_{kl}	$\begin{bmatrix} 0 & 50 \\ 50 & 0 \end{bmatrix}$; i.e., both false-positive and false-negative costs are equal to 50.
$p(\tilde{\mathbf{r}})$	See the second group of columns of Table 1.
$p(k\|\tilde{\mathbf{r}})$	See the third group of columns of Table 1.

Table 1: Valid readings $\tilde{\boldsymbol{r}} \in R$, the probabilities $p(\tilde{\boldsymbol{r}})$, $p(\tilde{\boldsymbol{r}}|k)$, and $p(k|\tilde{\boldsymbol{r}})$.

Test Readings, R			$p(\tilde{\mathbf{r}})$	$p(\tilde{\mathbf{r}}\|k)$		$p(k\|\tilde{\mathbf{r}})$	
Test 1	Test 2	Test 3		Negative	Positive	Negative	Positive
0	0	0	0.067	0.035	0.105	0.289	0.711
0	0	1	0.119	0.015	0.245	0.070	0.930
0	1	0	0.105	0.105	0.105	0.550	0.450
0	1	1	0.135	0.045	0.245	0.183	0.817
1	0	0	0.097	0.140	0.045	0.792	0.208
1	0	1	0.080	0.060	0.105	0.411	0.589
1	1	0	0.251	0.420	0.045	0.919	0.081
1	1	1	0.146	0.180	0.105	0.677	0.323

In this small example, the solution can be readily calculated by an exhaustive search of all $2^3 = 8$ possible test configurations. The value of each subset S is calculated by enumerating all the signatures in $R(S)$. An example of such calculations for the configuration $S = \{1,2\}$ is described in Table 2. First, the set of all signatures obtained from the subset S is shown in the first group of columns in Table 2. Next, for each signature $\mathbf{r} \in R(S)$ and each diagnosis $k \in K$, the probabilities $p_S(\mathbf{r}|k)$, $p_S(\mathbf{r})$, and

$p_S(k|\mathbf{r})$ are calculated using (3) -(5) and are shown in the second, third, and fourth group of columns, respectively. Now, the expected error cost of diagnosing l while the true diagnosis is k can be seen in the fifth group of columns and is calculated when l is decided, i.e., $E_S(\mathbf{r}|l)$, as given in (6)(6). Equation (7) shows the diagnosis that minimizes the expected classification error cost for signature $\mathbf{r}$ (sixth group of columns). Its expected classification error cost is given by (6)and can be seen in the seventh group of columns. Multiplying the minimum expected classification error cost by the probability of obtaining each signature $\mathbf{r} \in R(S)$ can be seen in the eighth group of columns. The sum of this column is 12.25, which denotes the expected classification error cost for the subset as given by the first addend of (9). The second addend of (9) indicates the cost of the tests, which is 2.5. Thus, the expected total cost of the subset $S = \{1,2\}$ is 14.75.

Table 2: Calculating the expected total cost for $S = \{1,2\}$

$R(S)$		$p_S(\mathbf{r}\|k)$		$p_S(\mathbf{r})$	$p_S(k\|\mathbf{r})$		$E_S(\mathbf{r}\|l)$		$l_S^*(\mathbf{r})$	$E_S^*(\mathbf{r})$	$p_S(\mathbf{r})E_S^*(\mathbf{r})$
Test1	Test2	Negative	Positive		Negative	Positive	l= Negative	l = Positive			
0	0	0.05	0.35	0.19	0.15	0.85	42.57	7.43	Positive	7.43	1.38
0	1	0.15	0.35	0.24	0.34	0.66	32.81	17.19	Positive	17.19	4.13
1	0	0.2	0.15	0.18	0.62	0.38	19.01	30.99	Negative	19.01	3.38
1	1	0.6	0.15	0.40	0.83	0.17	8.49	41.51	Negative	8.49	3.38
										$\sum_{\mathbf{r}\in R(S)} p_S(\mathbf{r})E_S^*(\mathbf{r})$	12.25
										$\sum_{i\in S} c_i$	2.5
										Expected total cost:	14.75

In Table 3, for each possible configuration (given in the first column), we present the expected classification error cost (second column), the tests cost (third column), and the expected total cost, which is the error cost plus the tests cost (in the fourth column). One can observe that the configuration {1,2,3}, i.e., using all the tests, is the one that minimizes the cost function (9) with a value of 14.01.

Table 3: The expected classification error, tests cost, and expected total cost of each configuration

Configuration	Expected classification error cost	Testing cost	Total cost
{1,2,3}	11.01	3	14.01
{1,2}	12.25	2.5	14.75
{1,3}	12.25	2.5	14.75
{2,3}	15.00	1	16.00
{1}	12.25	2	14.25
{2}	18.13	0.5	18.63
{3}	15.00	0.5	15.50
{ }	22.50	0	22.50

Interestingly, the second best configuration is {1}. Adding tests 2 or 3, i.e., using the configurations {1,2} or {1,3}, results in the same classification error cost as {1} but at a higher testing cost. This demonstrates the complex structure of the problem and the fact that a simple greedy or local search heuristic is unlikely to solve it.

Numerical analysis of the optimal decision, as a function of the classification error costs (false positive, $\lambda_{Negative,Positive}$, and false negative, $\lambda_{Positive,Negative}$) is presented in Figure 1. In this figure, the colors denote the optimal configuration. The vertical black line illustrates the changes in the optimal configuration when the false-negative error cost ranges from 0 to 100 and the cost of the false-positive error is fixed at 35. As seen, the optimal decision can be very sensitive to changes in this parameter. The expected classification error when performing all the tests is always smaller then performing other combinations. However, the optimal decision takes into account the tradeoff between the expected classification error and the cost of the tests.

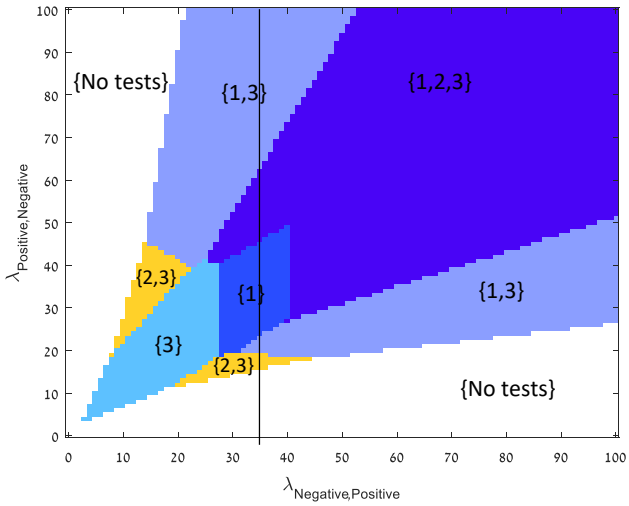

Figure 1: An optimal configuration as a function of the false positive and false negative costs.

Another analysis is performed to test how changes in the prior probabilities $\mathbf{p}(k)$ affect the optimal decision. The results are shown in Figure 2. The parameter $p(negative)$ changes along the horizontal axis; note that $p(positive) = 1 - p(negative)$. The rest of the parameters are fixed to their values, as in the original example. It is clear that when $p(negative) = 1$ or $p(negative) = 0$, i.e., when the

diagnosis is always negative or always positive, the solution is trivial: tests are not required. The number of sensors increases as the entropy of the classes increases.

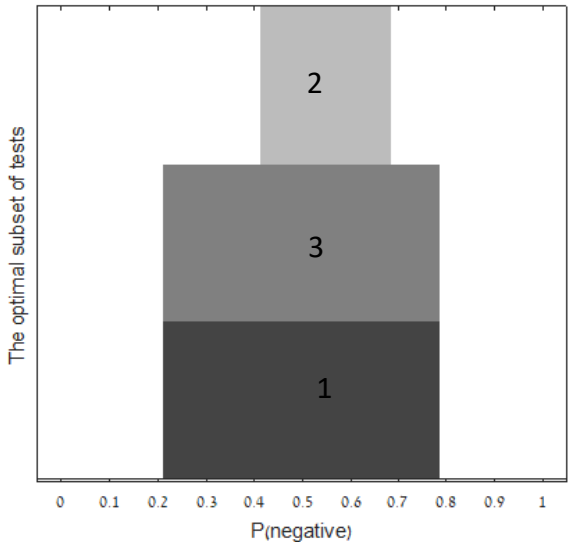

Figure 2: The optimal subset of tests (configuration) as a function of the probability of diagnoses. For example, for P(Negative)=0.3 the optimal configuration is {1,3}.

Note again that this problem could be solved by optimality using an exhaustive search, which is valid when the number of tests is small. However, since the number of configurations grows exponentially with the number of tests, the problem quickly becomes computationally intractable in the number of tests. Therefore, effective heuristic solution methods for solving the problem are required and are presented in the next section.

4 Metaheuristics solution methods to the STCP

This section presents three possible methods to address the STCP based on the known *Tabu Search (TS)*, *Cross-Entropy (CE)*, and *Binary Gravitational Search Algorithm (BGSA)* metaheuristics. We report on all three methods since none of them predominate. Note that the STCP is a long-term design problem, and thus the decision maker may wish to apply all available methods and select the best solution obtained.

Recall that a solution to a problem is defined by the selected configuration, whereas the mapping of each signature to a class is defined by (7). We denote a solution by the characteristic vector $\mathbf{x}$ of this set, i.e., $x_i = 1$ if test i is included and 0 otherwise. The value of a solution, calculated as in (9), is denoted by $g(\mathbf{x})$.

4.1 The TS method

The TS method extends the basic local search techniques to facilitate the exploration of the solution space beyond local optima. Once a local optimum is reached, the method allows one to move to a new solution even if it is inferior. The TS method uses a *tabu list* (TL) to disallow moves that cancel previous moves during several subsequent iterations in order to escape a neighborhood of locally optimal solutions.

The TS algorithm involves three main steps: (a) generate an initial solution and initialize the TL to be empty, (b) explore the current solution's neighborhood defined by a set of candidate moves but excluding moves listed in the TL, and (c) move to the best explored solution and add a new entry to TL to forbid the move that brings the search back to the previous solution. If the TL is longer than a predefined length, the algorithm removes its oldest entries. Steps (b) and (c) are repeated up to a predefined number of iterations or until some other stopping criterion is satisfied.

In our implementation, the initial solution is the empty set ($\mathbf{x} = \mathbf{0}$). Given a current solution $\mathbf{x}$, its value is evaluated with respect to all possible readings as explained and demonstrated in Sections 2 and 3. The neighborhood of $\mathbf{x}$, $N(\mathbf{x})$ is defined by three types of moves: add one test that is not included in the current solution, remove one test from the current solution, and swap a test from the current solution with one that is not included. The set of neighboring solutions induced by each type of the above-mentioned moves are denoted by $A(\mathbf{x})$, $R(\mathbf{x})$, and $S(\mathbf{x})$, respectively, thus $N(\mathbf{x}) = A(\mathbf{x}) \cup R(\mathbf{x}) \cup S(\mathbf{x})$. Note that $|A(\mathbf{x}) + R(\mathbf{x})| = n$ and $|S(\mathbf{x})| \leq \frac{1}{4}n^2$. Each entry in the tabu list consist of one or two tests that should not be added or removed from the solution as long as the entry remains in the list. Add and remove operations add entries with a single test and the swap operation adds an entry with a pair of tests. One is forbidden for removal and the other for appending. If a candidate move involves a test in the tabu list, then its respective solution is excluded from the neighborhood. We denote this reduced neighborhood by $N'(\mathbf{x})$. In our implementation we use a stopping criterion based on the total number of iterations in order to allow a fair comparison with other methods, but other criteria used in the literature may apply. The main algorithm is outlined as a pseudocode in Figure 3.

Initialized
    Set **x,** $\mathbf{x}^*$ and TL as empty
    Set $v^* = g(\mathbf{x})$
Repeat
    Set $v = \infty$
    For each $\mathbf{x}'$ in $N'(\mathbf{x})$
        Set $v' = g(\mathbf{x}')$
        If $v' < v$ then
            $v = v'$
            $\mathbf{y} = \mathbf{x}'$
            Let m be the test(s) by which **x** and **y** differ
    Set $\mathbf{x} = \mathbf{y}$
    If $v < v^*$ then $\mathbf{x}^* = \mathbf{x}$ and $v^* = v$
    Append *m* as an entry to the TL
    If $|TL|$ is greater than the maximal tabu length, remove its first entry
Until a pre-determined number of iterations is performed

Figure 3: Pseudocode of the TS algorithm

4.2 The CE Method

The CE algorithm involves iterative steps whereby each iteration can be broken down into three main phases: (a) generate a random population of solutions using a specified *probabilistic selection rule*; (b) evaluate the value of each of the generated solutions; and (c) update the probabilistic selection rule for the next iteration based on the best solutions (called the *elite set*) and iterate until some stopping criterion is satisfied.

At each iteration we generate w solutions using a multi-Bernoulli distribution with success probabilities $\mathbf{p} = (p_1, \dots, p_n)$, i.e., $\mathbf{x} = (x_1, \dots, x_n)$ such that $x_i \sim Ber(p_i)$. We initialize the probabilities with $p_i = 0.5$, for all i, and update all these probabilities at step (c) of each iteration. In a given iteration of the CE, we use $\mathbf{x}_i^{(j)}$ to denote the i^{th} test in solution j, while $\mathbf{x}^{(j)} \in \{0,1\}^{\text{n}}$ is a binary vector that represents solution j. The probabilities p_i are updated at the end of each iteration based on the best ρw solutions (called the elite set) and subject to exponential smoothing with a weight parameter $\alpha \in [0,1]$. The parameter $\rho \in (0,1)$ defines the relative size of the elite set; recall that w is the number of solutions that we generate at each iteration. w and ρ are selected such that $w\rho$ is an integer. Previous studies used $\rho = 0.1$, i.e., the top ten percent solutions are taken as the elite set. We follow this line. The indices of the solutions in the elite set of iteration t are denoted by $\mathcal{E}_t$. The parameters of the multi-Bernoulli distribution are updated as follows:

$$q_{t,i} = \frac{\sum_{j \in \mathcal{E}_t} \mathbb{I}_{\{\mathbf{x}_i^{(j)}=1\}}}{\rho w}, \quad i = 1, \dots n \tag{10}$$

where $\mathbb{I}_{\{.\}}$ is an *indicator function* defined as follows:

$$\mathbb{I}_{\{condition\}} = \begin{cases} 1, & condition\ holds \\ 0, & otherwise \end{cases}$$

That is, $q_{t,i}$ is the proportion of the solutions that include test i in the elite set of iteration t. The following exponential smoothing formula is then used to update $\mathbf{p}_t$

$$p_{t,i} = \alpha q_{t,i} + (1-\alpha) p_{t-1,i}, \quad i = 1, \dots n \tag{11}$$

We use exponential smoothing to prevent the premature convergence of $\mathrm{p}_{t,i}$ to 0 or 1. It has been empirically shown, e.g., by Alon et al. 2005, that a value of α between $0.7 \leq \alpha \leq 0.9$ often gives the best results. In this study, we use $\alpha = 0.8$.

Several types of stopping criteria have been used in the literature, such as i) Stop when the worst solution in the elite set does not change for a predefined number of consecutive iterations; ii) Stop when all the elements of $\mathbf{p}_t$ are close enough to 0 or 1 and thus no new solutions are likely to be generated; and iii) Stop after a predefined number of iterations or computation time. Clearly, combinations of the above may also apply. In our implementation we use the third stopping criterion to enable a fair comparison with other heuristics given a similar computational effort.

A pseudocode that describes our CE algorithm is presented in Figure 4.

Initialized **p** such that all test probabilities are equal to **0.5**.
Set $v^* = \infty$
Repeat
For j = 1 to w
Generate solution $\mathbf{x}^{(j)}$ such that $x_i^{(j)} \sim Ber(p_i)$
Sort the solutions in **x** in non-decreasing order of $g(\mathbf{x}^{(j)})$
Let $\mathrm{x}^{[1]}, \mathrm{x}^{[2]}$, ... be the solutions in their sorted order
If $g\left(\mathbf{x}^{([1])}\right) < v^*$
$v^* = g\left(\mathbf{x}^{([1])}\right)$ and $x^* = \mathbf{x}^{([1])}$
Update **p** using (11)
Until a pre-determined number of iterations is performed
Return $\mathbf{x}^*, v^*$

Figure 4: Pseudocode of the CE algorithm

4.3 The Binary Gravitational Search Algorithm (BGSA) Method

The Binary Gravitational Search Algorithm (BGSA) is a relatively recent metaheuristic inspired by the Newtonian law of gravitation and motion. Solutions are represented by vectors and considered as objects (also called agents), and their position in a multidimensional space is determined by the elements of these vectors. The mass of each solution is determined by its objective function value. At each iteration of the algorithm, the position and velocity of each object is updated based on its current position, mass and velocity as well as those of the other objects. The mass of each object/solution is then updated based on the values of solutions in the population. The process is repeated until all the objects are merged into one or more heavy objects or another stopping criterion is met.

For the implementation of the BGSA for the STCP, consider an initial set of w solutions each represented by a vector $\mathbf{x}^{(j)} \in \{0,1\}^n$ for $j = 1,2,\dots,w$; we refer to each coordinate of these vectors $x_i^{(j)}$ as the position of j^{th} solution in the i^{th} dimension. These values are updated from iteration to iteration and we use $\mathbf{x}^{(j)}(t)$ to denote the position of the solution at iteration t of the algorithm. Let us further define

$$best(t) = \min_{j\in\{1,\dots,w\}} g\left(\mathbf{x}^{(j)}(t)\right)$$

$$worst(t) = \max_{j\in\{1,\dots,w\}} g\left(\mathbf{x}^{(j)}(t)\right).$$

Based on these values we can calculate a normalized measure of each solution j

$$q_j(t) = \frac{g\left(\mathbf{x}^{(j)}(t)\right) - worst(t)}{best(t) - worst(t)}$$

Next, the mass of each solution j is updated as follows:

$$M_j(t) = \frac{q_j(t)}{\sum_{j'=1}^{w} q_{j'}(t)}. \qquad (12)$$

At a specific time t, the force acting on agent j_1 from agent j_2 is defined as follows:

$$F_i^{(j_1,j_2)}(t) = G_0\left(1-\frac{t}{T}\right)\frac{M_{j_1}(t)\cdot M_{j_2}(t)}{\sum_{i'=1}^{n}\left|x_{i'}^{(j_1)}(t)-x_{i'}^{(j_2)}\right|+\varepsilon}\left(x_i^{(j_2)}(t)-x_i^{(j_1)}\right),$$

where G_0 is a gravitational constant, T is the total number of planned iterations for the algorithm and ε is a small positive constant. Using some preliminary experiments, we set $G_0 = 0.01T$ and $\varepsilon = 2.2\times 10^{-16}$.

Next, we find an elite set E_t comprising the best solutions at iteration t and generate random numbers $p_j(t)\sim U[0,1]$ for each $j \in E_t$. The cardinality of the E_t is set to $\lceil \rho_t w \rceil$ where ρ_t linearly decreases from iteration to iteration according to the following formula:

$$\rho_t = 1-\frac{t}{T}(1-\rho_T),$$

where ρ_T is a parameter of the algorithm, while in our experiment we used $\rho_T = 0.02$. Next, we define the force that acts on solution j in dimension i at iteration t by:

$$F_i^{(j)}(t) = \sum_{j'\in E_t\setminus\{j\}} p_{j'}F_i^{(j,j')}(t).$$

In such a way, at the initial stage all solutions apply pressure on each other, and as the iterations progress, only the few best solutions affect all the others. Now, according to the law of motion, the acceleration of a solution j at iteration t in dimension i is given by:

$$a_i^{(j)}(t) = \frac{F_i^{(j)}(t)}{M_j(t)} \tag{13}$$

The velocity of an agent is considered as a random fraction of its current velocity added to its acceleration:

$$v_i^{(j)}(t) = \pi_j(t)\cdot v_i^{(j)}(t-1) + a_i^{(j)}(t-1), \tag{14}$$

where $v_i^{(j)}(0)$ is initialized to zero and $\pi_j(t)$ is drawn from $U[0,1]$. Moreover, to increase the chance of convergence, the velocity is limited by some parameter v_{max}. That is, $\left|v_i^{(j)}\right| < v_{max}$. We followed (Rashedi et al., 2010) and set $v_{max} = 6$. Based on the velocity in each dimension, i, we flip the position of each agent, j, between 0 and 1 with probability $\left|tanh\left(v_i^{(j)}(t)\right)\right|$ and leave it as $x_i^{(j)}(t)$ otherwise.

$$x_i^{(j)}(t+1) = \begin{cases} 1-x_i^{(j)}(t), & with\ probability\ \left|tanh\left(v_i^{(j)}(t)\right)\right| \\ x_i^{(j)}(t), & otherwise \end{cases} \tag{15}$$

The BGSA algorithm is outlined as a pseudocode in Figure 5.

Initialized
$v^* = \infty$
Draw initial population of w solutions $\mathbf{x}^{(j)}, \quad j = 1,2,\dots,w$
Repeat
$Set = \infty, worst = -\infty$
For j = 1 to w
Evaluate $g(\mathbf{x}^{(j)})$
If $g(\mathbf{x}^{(j)}) < best$
$best = g(\mathbf{x}^{(j)})$
If $g(\mathbf{x}^{(j)}) > worst$
$worst = g(\mathbf{x}^{(j)})$
Calculate M_j // *See (12)*
Calculate $a^{(j)}$ // *See (13)*
Calculate $\mathbf{v}^{(j)}$ // *See (14)*
Update $\mathbf{x}^{(j)}$ // *See (15)*
If $g(\mathbf{x}^{(j)}) < v^*$
$v^* = g(\mathbf{x}^{(j)})$ and $x^* = \mathbf{x}^{(j)}$
Until it reaches the pre-determined number of iterations, T
Return the best solution $\mathbf{x}^{(j)}$

Figure 5: Pseudocode of the BGSA algorithm

4.4 Caching

Recall that calculating the value, $g(\mathbf{x})$, for each solution with a given configuration requires evaluating the signatures of all the readings, which is a computationally demanding task. Indeed, almost all the running time of the three heuristics described above is spent on these value evaluations. In some situations, the same solutions may be required multiple times in the same run of the algorithm. To avoid repeated calculations, we store the value of each calculated solution in a hash table that is indexed by the binary representation of the solution. When the algorithms require the value of $\mathbf{x}$ they first check if it already exists in the table. If it exists, the value is retrieved; otherwise the solution is evaluated and its value is stored in the table. This mechanism significantly reduces the running time of all three heuristics and is especially effective in the last iterations of the two randomized heuristics, namely, the CE and BGSA. In our experiments we managed a different cache for each algorithm in order to properly benchmark them, but in practice the same cache can be used by different algorithms that may be run in parallel or sequentially to obtain the best solution.

5 Experimental results

In this section, we evaluated the effectiveness of the algorithms presented in Section 4. The algorithms were implemented in MATLAB 2018b. The testing environment was an eight-core Intel i7-4790 3.60 GHz CPU, 32 GB of RAM running under Windows 7, 64 bit.

For the evaluation of the algorithms presented above, we used representative datasets from the UCI Machine Learning Repository (Dua and Graff, 2019) with up to 8,124 readings and 68 tests. In some of the datasets, we removed tests and readings to eliminate missing values. Moreover, tests with numerical values were discretized by dividing their values into quintiles or quartiles (depending on the number of readings).

Using this data, we estimated the prior probabilities of the classes and the readings as well as the conditional probability of each class given a reading. Our algorithms were tested with respect to these estimated probabilities.

For each dataset, we used three test cost vectors: one with a fixed (unit) cost per test and two with randomly generated values as described below. Lastly, three classification error cost matrices were created for each combination of dataset and cost vector. All cleaned and processed input data of our experiment are available as an electronic appendix to this paper. In total, the test contained 45 problem instances based on five different UCI datasets.

In Table 4 we show the number of tests, type of test values (continuous or discrete), discretization level (in the case of continuous test values), number of readings and number of classes in the datasets. The information in the table refers to the cleaned data after removing some tests and readings to eliminate the missing values.

Table 4: Characteristics of the five datasets used in the experiments.

Dataset	*Tests*	*Test value type*	*Discretization*	*Readings*	*Classes*
Wine	12	Continuous	Quintiles	6,463	7
Thyroid	21	Continuous	Quintiles	3,103	5
Mushrooms	21	Discrete	-	8,124	2
Cortex nuclear	68	Continuous	Quartiles	1,077	8
Molecular biology	60	Discrete	-	3,190	3

The Wine Dataset from UCI includes two tables, related to red and white wine samples as described in Cortez et al. (2009). We followed Kaggle (Parmar, 2018) and used a merged version of this dataset where the type of the wine was added as a new feature. The class in this dataset is the wine quality score represented by a numerical value in the range 3-9. The classification error matrix is based on the distance between the classified quality and the true quality (a Toeplitz matrix with values in the range 0-6). The (i, j) element of the Toeplitz matrix represents the absolute difference between the two classes. An error cost matrix that is proportional to the Toeplitz matrix reflects the fact that greater errors are costlier. Specifically, we created one matrix that is 20 times the Toeplitz matrix and one that is 30 times that matrix.

In the Thyroid Dataset there are five classes: four related to pathological conditions and one (negative) related to a healthy one. We created two matrices that assign a high

cost to a false negative diagnosis, a low one to a false positive diagnosis, and medium values to the misdiagnosis of a pathological condition.

In the Mushrooms Dataset, each reading should be classified as toxic or nontoxic. The classification matrices were constructed to reflect the fact that a false negative error (classifying a toxic mushroom as an edible one) is much more expensive or dangerous than a false positive error.

In the Cortex Nuclear Dataset, the classes are described by three binary features that define the eight classes. We constructed error cost matrices based on the Hamming distance of this binary description of the class; i.e., the distance can be zero, one, two or three. The matrices were created by multiplying these distances by 100 and by 200.

The Molecular Biology Dataset contains DNA sequences of 60 nucleotides (each nucleotides is a test, in our terminology). Each sequence belongs in one of three classes (exon-intron, intron-exon, or neither). For this dataset, we used three fixed classification error matrices with three different values (low, medium, high).

Our experiment is full factorial. That is, we tested all combinations of the three test cost vectors and three classification error cost matrices – nine runs for each of the five datasets.

For each dataset, we created one *fixed* test cost vector and two random cost vectors that were drawn from $N(1,0.1)$ and from $U(0,2)$. The Thyroid Dataset from UCI included one test cost vector that we used in our experiment after normalizing it to make its mean equal one. In this instance, we used it instead of the cost vector with normally distributed values.

For each dataset, we created three error cost matrices based on particular dataset characteristics. All these error cost matrices have zeros on their diagonal and values off the diagonal, as described in Table 5.

Table 5: Description of the classification error cost matrices.

Dataset	*Error cost matrix 1*	*Error cost matrix 2*	*Error cost matrix 3*
Wine	20 × Toeplitz matrix {0,..,6}	30 × Toplitz matrix {0,..,6}	50 at each element off the diagonal
Thyroid	1500 to false negative 600 to false positive 1200 other errors	3000 to false negative 1200 to false positive 2400 other errors	1500 at each element off the diagonal
Mushrooms*	300 to false positive, 500 to false negative	100 to false positive, 400 to false negative	50 to false positive, 700 to false negative
Cortex Nuclear	100 × *hamming distance*	200 × *hamming distance*	300 at each element off the diagonal
Molecular Biology**	30 at each element off the diagonal	60 at each element off the diagonal	120 at each element off the diagonal

* In the Mushrooms Dataset, the error cost matrix 3 is not fixed.
** In the Molecular Biology Dataset, all error cost matrices are fixed.

We conducted some preliminary experiments to decide upon some of the parameters of the algorithm and discovered that the CE works well with iterations of $20n$ solutions and typically converges before the 50^{th} iteration. We found that the best number of solutions per iteration in BGSA is not affected by the number of tests and that the algorithm works well with 100 solutions per iteration. To make a fair comparison

between these two methods, we set the number of iterations in BGSA to $10n$. This allowed us to keep the total number of evaluated solutions to approximately $1000n$ in both methods. Note, however, that in both cases many of the solutions were sampled more than once and could be retrieved from the cash rather than actually being evaluated. Since the TS is a much faster heuristic, we run it with a limit of 90 iterations but repeat each run three times with tabu list lengths of 0, 2, and 4 while keeping the hash table that stores the cache from iteration to iteration. Note that setting the tabu length to 0 is equivalent to a naïve local search, which typically terminates with a locally optimal solution in less than 90 iterations. The reported solution values for the TS are the best out of the three. We note no single alternative list length predominates the other. The solution times reported for the TS are the sums of the times of the three runs with the different tabu lengths. All the other tuning parameters of the heuristics methods are specified in Section 4.

For each of the $5 \times 3 \times 3$ combinations of datasets, error cost matrices and test cost vectors we applied the three heuristic methods. For the smaller instances (Wine, Thyroid, and Mushrooms) we also computed the exact optimal solution by enumerating all the 2^n possible combinations of the tests. This process was much more competently demanding than any of our methods and is clearly not applicable to cases with larger instances. In Table 6 we present the results for these smaller datasets.

For each run, the solution value is presented first and optimal ones are in boldface. Next, the two components of the solution values are listed – the expected error cost and the tests cost. In addition, the iteration number when the best solution is first found and the solution times in seconds are displayed.

In Table 7, the same results are reported for the larger datasets, namely, Molecular Biology and Cortex Nuclear. Here, we could not compute the exact solution using an exhaustive enumeration and thus the solution values in bold are the best we could find using the three heuristic methods.

It is apparent from Tables 6 and 7 that none of the three solution methods consistently provides a better solution than the others. All instances of the three smaller datasets were solved to optimality using both CE and BGSA and the TS, while missing the optimal solutions occurred in only one case out of the 27. Compared with the brute force approach of enumerating all the solutions, the computational effort required by all three methods is negligible when the number of tests grows. For example, for the Thyroid and Mushrooms instances there are over two million possible configurations, but less than 2,500, 6,000, and 15,000 of them were explored by the TS, CE and BGSA, respectively.

In the larger datasets, the TS provided the best solutions in 15 out of 18 instances and missed the best solution within a small margin of up to 1.5%. In these datasets the CE and BGSA found the best solutions in 8 and 7 cases, respectively.

Table 6: Results summary for Wine, Thyroid, and Mushrooms datasets.

Test cost vector	Error cost matrix	Parameters	Wine			Thyroid			Mushrooms		
			TS	BGSA	CE	TS	BGSA	CE	TS	BGSA	CE
fixed	1	Solution value	**11.602**	**11.602**	**11.602**	**17.058**	**17.058**	**17.058**	**4**	**4**	**4**
		Expected error cost	4.602	4.602	4.602	7.058	7.058	7.058	0	0	0
		Tests cost	7	7	7	10	10	10	4	4	4
		# of iterations until best solution	10/90	2/120	3/50	12/90	106/210	10/50	6/90	66/210	11/50
		Solution time (sec.)	19.2	108.2	64.4	32.3	191.9	103.0	22.7	188.1	56.1
fixed	2	Solution value	**13.765**	**13.765**	**13.765**	**23.762**	**23.762**	**23.762**	**3.591**	**3.591**	**3.591**
		Expected error cost	5.765	5.765	5.765	22.762	22.762	22.762	0.591	0.591	0.591
		Tests cost	8	8	8	1	1	1	3	3	3
		# of iterations until best solution	10/90	4/120	4/50	13/90	124/210	10/50	4/90	132/210	10/50
		Solution time (sec.)	36.7	107.4	62.3	38.5	183.8	96.9	17.3	180.2	54.3
fixed	3	Solution value	**15.388**	**15.388**	**15.388**	**23.568**	**23.568**	**23.568**	**3.295**	**3.295**	**3.295**
		Expected error cost	7.388	7.388	7.388	22.568	22.568	22.568	0.295	0.295	0.295
		Tests cost	8	8	8	1	1	1	3	3	3
		# of iterations until best solution	10/90	11/120	4/50	13/90	115/210	10/50	4/90	98/210	9/50
		Solution time (sec.)	32.7	108.7	61.4	26.4	192.6	89.7	15.2	175.1	58.0
Normal*	1	Solution value	**11.691**	**11.691**	**11.691**	**40.122**	**39.609**	**39.609**	**4.014**	**4.014**	**4.014**
		Expected error cost	4.601	4.601	4.601	40.122	37.609	37.609	0.217	0.217	0.217
		Tests cost	7.090	7.090	7.090	0	2	2	3.797	3.797	3.797
		# of iterations until best solution	8/90	4/120	4/50	1/90	102/210	9/50	5/90	110/210	13/50
		Solution time (sec.)	25.8	105.4	60.3	2.4	60.9	22.4	23.0	178.3	78.0
Normal*	2	Solution value	**13.881**	**13.881**	**13.881**	**60.991**	**60.991**	**60.991**	**3.895**	**3.895**	**3.895**
		Expected error cost	5.765	5.765	5.765	27.071	27.071	27.071	0.098	0.098	0.098
		Tests cost	8.116	8.116	8.116	33.920	33.920	33.920	3.797	3.797	3.797
		# of iterations until best solution	9/90	1/120	4/50	13/90	129/210	13/50	5/90	119/210	13/50
		Solution time (sec.)	28.9	106.7	59.4	20.6	129.4	70.0	11.0	168.6	68.0
Normal*	3	Solution value	**15.504**	**15.504**	**15.504**	**40.122**	**40.122**	**40.122**	**3.846**	**3.846**	**3.846**
		Expected error cost	7.388	7.388	7.388	40.122	40.122	40.122	0.049	0.049	0.049
		Tests cost	8.116	8.116	8.116	0	0	0	3.797	3.797	3.797
		# of iterations until best solution	10/90	8/120	5/50	1/90	72/210	12/50	6/90	113/210	11/50
		Solution time (sec.)	17.9	110.4	62.5	3.0	55.0	21.7	10.9	174.3	68.3
Uniform	1	Solution value	**12.369**	**12.369**	**12.369**	**15.790**	**15.790**	**15.790**	**2.235**	**2.235**	**2.235**
		Expected error cost	5.471	5.471	5.471	8.121	8.121	8.121	0.492	0.492	0.492
		Tests cost	6.898	6.898	6.898	7.669	7.669	7.669	1.743	1.743	1.743
		# of iterations until best solution	8/90	1/120	4/50	12/90	131/210	11/50	6/90	128/210	10/50
		Solution time (sec.)	33.2	94.4	55.9	36.4	177.5	95.0	11.2	198.6	65.1
Uniform	2	Solution value	**15.004**	**15.004**	**15.004**	**22.159**	**22.159**	**22.159**	**2.137**	**2.137**	**2.137**
		Expected error cost	7.288	7.288	7.288	21.743	21.743	21.743	0.394	0.394	0.394
		Tests cost	7.716	7.716	7.716	0.416	0.416	0.416	1.743	1.743	1.743
		# of iterations until best solution	9/90	10/120	5/50	13/90	119/210	12/50	6/90	105/210	9/50
		Solution time (sec.)	22.7	103.7	60.0	33.5	184.5	93.0	7.2	174.6	64.5
Uniform	3	Solution value	**16.974**	**16.974**	**16.974**	**22.183**	**22.183**	**22.183**	**2.038**	**2.038**	**2.038**
		Expected error cost	7.426	7.426	7.426	21.767	21.767	21.767	0.295	0.295	0.295
		Tests cost	9.548	9.548	9.548	0.416	0.416	0.416	1.743	1.743	1.743
		# of iterations until best solution	11/90	8/120	4/50	13/90	128/210	7/50	9/90	143/210	12/50
		Solution time (sec.)	28.5	100.7	60.2	20.8	187.3	95.8	13.4	170.0	60.3

*except in the Thyroid dataset, where the cost vector was taken from UCI

Table 7: Results summary for Cortex Nuclear and Molecular Biology datasets.

Test cost vector	Error cost matrix	Parameters	Cortex nuclear			Molecular biology		
			TS	BGSA	CE	TS	BGSA	CE
fixed	1	Solution value	**7.371**	7.650	8.093	**7.103**	7.969	7.828
		Expected error cost	0.371	0.650	0.093	3.103	0.969	0.828
		Tests cost	7	7	8	4	7	7
		# of iterations until best solution	9/90	642/680	45/50	5/90	569/600	49/50
		Solution time (sec.)	261.6	357.8	315.3	405.6	1561.0	1342.0
fixed	2	Solution value	**7.743**	8.371	8.371	**8.282**	8.395	8.357
		Expected error cost	0.743	0.371	0.371	0.282	0.395	0.357
		Tests cost	7	8	8	8	8	8
		# of iterations until best solution	9/90	674/680	33/50	23/90	521/600	46/50
		Solution time (sec.)	264.7	357.7	320.7	1230.0	1571.0	1417.5
fixed	3	Solution value	**8**	8.836	8.279	**8.564**	8.978	8.865
		Expected error cost	0	0.836	0.279	0.564	0.978	0.865
		Tests cost	8	8	8	8	8	8
		# of iterations until best solution	10/90	650/680	42/50	23/90	565/600	20/50
		Solution time (sec.)	269.3	357.4	322.8	1230.7	1562.6	1456.3
Normal	1	Solution value	6.852	7.353	**6.747**	**6.760**	**6.760**	**6.760**
		Expected error cost	0.464	0.372	0.464	0.903	0.903	0.903
		Tests cost	6.388	6.981	6.283	5.857	5.857	5.857
		# of iterations until best solution	10/90	645/680	42/50	29/90	558/600	41/50
		Solution time (sec.)	169.1	354.2	312.5	528.8	1539.6	1192.8
Normal	2	Solution value	**7.056**	7.293	7.458	**7.189**	**7.189**	**7.189**
		Expected error cost	0.185	0.186	0.371	0.470	0.470	0.470
		Tests cost	6.871	7.107	7.087	6.719	6.719	6.719
		# of iterations until best solution	40/90	659/680	50/50	32/90	581/600	36/50
		Solution time (sec.)	194.4	354.4	322.2	702.7	1564.3	1374.6
Normal	3	Solution value	**7.087**	7.771	7.274	7.675	7.660	**7.652**
		Expected error cost	0	0	0	0.790	0.941	0.753
		Tests cost	7.087	7.771	7.274	6.885	6.719	6.899
		# of iterations until best solution	86/90	645/680	49/50	43/90	570/600	40/50
		Solution time (sec.)	292.3	350.4	320.5	980.4	1541.2	1396.3
Uniform	1	Solution value	**2.224**	**2.224**	**2.224**	**1.758**	**1.758**	**1.758**
		Expected error cost	0.186	0.186	0.186	0.301	0.301	0.301
		Tests cost	2.038	2.038	2.038	1.457	1.457	1.457
		# of iterations until best solution	30/90	615/680	37/50	13/90	526/600	31/50
		Solution time (sec.)	169.3	341.6	277.2	214.1	1467.9	1082.7
Uniform	2	Solution value	2.307	**2.295**	2.307	**1.969**	**1.969**	**1.969**
		Expected error cost	2.094	0	2.094	0.151	0.151	0.151
		Tests cost	0.213	2.295	0.213	1.818	1.818	1.818
		# of iterations until best solution	21/90	609/680	39/50	15/90	534/600	29/50
		Solution time (sec.)	258.8	342.0	292.0	623.0	1465.9	1099.6
Uniform	3	Solution value	**2.295**	2.307	2.307	**2.119**	**2.119**	**2.119**
		Expected error cost	0	2.094	2.094	0.301	0.301	0.301
		Tests cost	2.295	0.213	0.213	1.818	1.818	1.818
		# of iterations until best solution	50/90	620/680	30/50	16/90	533/600	33/50
		Solution time (sec.)	303.4	340.0	290.6	690.0	1468.0	1126.5

We further observed that the best solutions in almost all the runs of the three methods were found in an early iteration (relative to the number of allowed iterations), which implies that with the other tuning parameters used, our stopping criteria were correct, although it may be the case that other criteria could save computation time without scarifying quality. Lastly, it seems that in most of the case the TS outperforms the two other heuristics in terms of computation time. However, we comment that since the STCP is a long-run design problem, a good practice would be to apply all the known heuristics, or to use complete enumeration when the number of tests is small enough.

In Table 8, we present some aggregated statistics that measure the success of our caching mechanism for the three solution methods. In the first row, we present the average number of solutions that are evaluated for each of the nine instances. In the second row, we present the average number of times when the required solution could be obtained from the hash table (cache hits) and thus, there was no need to reevaluate it. The ratio between the number of cache hits and the total number of scanned solutions (actually evaluated and retrieved from the cache) is presented in the third row, entitled "Frac. hits rate". In the fourth row, we present the ratio between the number of evaluated solutions and the number of all the possible ones.

It can be seen that for all the three algorithms, the hash table is beneficial, especially in instances with a small number of tests. When the number of tests grows, the hash table is effective mostly for the TS algorithm that searches in previous good solutions but not so much for the BGSA and CE. We note that for all the three methods, the fraction of evaluated solutions approaches zero as the number of tests grows. Given that all three algorithms spend very most of their computation time in the evaluation of solutions, this implies that their solution time is much shorter than the time needed for complete enumeration.

Table 8: Statistics of the caching mechanism

Solution method	Measure	Wine	Thyroid	Mushrooms	Cortex nuclear	Molecular biology
	# Tests	12	21	21	68	60
TS	# of evaluations	558	1,685	2,329	50,047	29,062
	# of cache hits	4,055	16,664	15,208	68,128	62,248
	Frac. hits rate	0.88	0.91	0.87	0.58	0.68
	Frac evaluated	0.14	0.0008	0.0011	1.7×10^{-16}	2.5×10^{-14}
BGSA	# of evaluations	2,376	13,101	13,989	64,136	56,346
	# of cache hits	9,624	7,899	7,011	3,864	3,654
	Frac. hits rate	0.80	0.38	0.33	0.06	0.06
	Frac evaluated	0.58	0.0062	0.0067	2.2×10^{-16}	4.9×10^{-14}
CE	# of evaluations	1,342	5,858	6,007	63,769	52,298
	# of cache hits	10,658	15,142	14,993	4,231	7,702
	Frac. hits rate	0.89	0.72	0.71	0.06	0.13
	Frac. evaluated	0.33	0.0028	0.0029	2.2×10^{-16}	4.5×10^{-14}

6 Conclusions

This paper introduces a practical problem whereby decisions must be made in the design phase of various systems and processes. The aim is to minimize the sum of the expected error cost associated with classification errors and the testing cost. The STCP is a generalization of the deterministic variant of the minimum TCP, which is already

highly intractable but our numerical experiments show that the three proposed solution methods are effective. Specifically, for a given budget of computational effort, the TS method appears to be superior to the CE and BGSA. However, since none of these methods consistently predominates, when computational resources are available, all the three methods should be applied, as the STCP is typically a long-term decision.

For large instances, the effectiveness of the three methods presented cannot be evaluated in terms of optimality gap, since an optimal solution cannot be calculated. Thus, an interesting direction for future research is to develop an exact method that solves the problem or at least creates good lower bounds for it.

The STCP is related to the well-studied feature selection problem and it may be the case that similar solution methods can be applied to it. However, solutions to the feature selection problem should be evaluated jointly using a classification method to examine their predictive power and control their sensitivity to overfitting, which is a different objective than that of the STCP.

Acknowledgment: The first author of this paper was partially supported by a scholarship from the Shlomo-Shmeltzer Institute. The research was partially supported by the Koret's Digital Living 2030 Grant.

References

Alon, G., Kroese, D. P., Raviv, T., & Rubinstein, R. Y. (2005). Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, *134*(1), 137-151.

Bacher, M., & Ben-Gal, I. (2017). Ensemble-Bayesian SPC: Multi-mode process monitoring for novelty detection. *IISE Transactions*, 49(11), 1014-1030.

Bertolazzi, P., Felici, G., Festa, P., Fiscon, G., & Weitschek, E. (2016). Integer programming models for feature selection: New extensions and a randomized solution algorithm. *European Journal of Operational Research*, *250*(2), 389-399.

Cangalovic, M. M., Kovacevic-Vujcic, V. V., Ivanovic, L., Drazic, M., & Asic, M. D. (1996). Tabu search: A brief survey and some real-life applications. *Yugoslav journal of operations research*, *6*(1), 5-18.

Chai, X., Deng, L., Yang, Q., & Ling, C. X. (2004, November). Test-cost sensitive naive bayes classification. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 51-58). IEEE.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. In *Decision Support Systems*, Elsevier, 47(4):547-553.

De Boer, P. T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, *134*(1), 19-67.

De Bontridder, K. M., Halldórsson, B. V., Halldórsson, M. M., Hurkens, C. A., Lenstra, J. K., Ravi, R., & Stougie, L. (2003). Approximation algorithms for the test cover problem. *Mathematical Programming*, *98*(1-3), 477-491.

Douek-Pinkovich, Y., Ben-Gal, I., and Raviv, T. (2019). The generalized test collection problem. Working paper.

Drezner, Z., Marcoulides, G. A., & Hoven Stohs, M. (2001). Financial applications of a tabu search variable selection model. *Journal of Applied Mathematics and Decision Sciences*, *5*(4), 215-234.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Duman, S., Güvenç, U., Sönmez, Y., & Yörükeren, N. (2012). Optimal power flow using gravitational search algorithm. *Energy Conversion and Management*, *59*, 86-95.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing*, *1*(3), 190-206.

Halldórsson, B. V., Halldórsson, M. M., & Ravi, R. (2001, August). On the approximability of the minimum test collection problem. In *European Symposium on Algorithms* (pp. 158-169). Springer, Berlin, Heidelberg.

Hovland, G. E., & McCarragher, B. J. (1997, April). Dynamic sensor selection for robotic systems. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on* (Vol. 1, pp. 272-277). IEEE.

Kammer, D. C. (1991). Sensor placement for on-orbit modal identification and correlation of large space structures. *Journal of Guidance, Control, and Dynamics*, *14*(2), 251-259.

Ling, C. X., Yang, Q., Wang, J., & Zhang, S. (2004, July). Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning* (p. 69). ACM.

Pacheco, J., Casado, S., & Núñez, L. (2009). A variable selection method based on Tabu search for logistic regression models. *European Journal of Operational Research*, *199*(2), 506-511.

Papa, J. P., Pagnin, A., Schellini, S. A., Spadotto, A., Guido, R. C., Ponti, M., ... & Falcão, A. X. (2011, May). Feature selection through gravitational search algorithm.

In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on* (pp. 2052-2055). IEEE.

Parmar R. (2018). Wine quality. https://www.kaggle.com/rajyellow46/wine-quality#winequalityN.csv.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information sciences*, *179*(13), 2232-2248.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2010). BGSA: binary gravitational search algorithm. *Natural Computing*, *9*(3), 727-745.

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operational Research*, *99*(1), 89-112.

Rubinstein, R.Y. and D.P. Kroese. (2004). The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning. Springer-Verlag, New York

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, *23*(19), 2507-2517.

Sela Perelman, L. S., Abbas, W., Koutsoukos, X., & Amin, S. (2016). Sensor placement for fault location identification in water networks: A minimum test cover approach. *Automatica*, *72*, 166-176.

Slijepcevic, S., & Potkonjak, M. (2001). Power efficient organization of wireless sensor networks. In *Communications, 2001. ICC 2001. IEEE International Conference on* (Vol. 2, pp. 472-476). IEEE.

Wendt, J. B., & Potkonjak, M. (2011, October). Medical diagnostic-based sensor selection. In *Sensors, 2011 IEEE* (pp. 1507-1510). IEEE.