A Combinatorial Approach for Optimal Sensor Selection

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@eng.tau.ac.il

Abstract

In complex systems, such as automobiles, aircraft, and even smart cities, it is possible to install a large number of sensors that can be used to monitor their states and make decisions regarding their operation and maintenance. However, because the sensors bear their costs and increase the complexity of the monitoring system, it is desirable to select the most inexpensive set of sensors that is sufficient to provide accurate and reliable information. In this paper, we introduce the *sensor selection problem* and formulate it as an integer linear program. The model is an extension of the *minimum test collection problem* with respect to several constraints, such as the sensors' cost which is not necessarily equal. We present an effective exact solution method that uses the special structure of the integer-programming model to reduce its dimension so it can easily be solved. This solution method was implemented and demonstrated to be superior in comparison to a state-of-the-art integer programming solver. Using publicly available datasets from the UCI Machine learning repository, we demonstrate how the Optimization-based Sensor Selection model can be used as an effective feature selection method as well.

Keywords: Sensor selection, feature selection, integer linear programming, dimension reduction

1. Introduction

Nowadays, monitoring systems are a crucial part of various systems and processes in many domains, such as manufacturing, medicine, transportation, and agriculture. Sensors are used to monitor the operation of complex systems (e.g., plant floors, the human body, aircrafts and call data centers to name just few examples). Modern monitoring systems often consist of many sensors connected to a central processing unit that detects the current state of the whole system. A major task in the design phase of these systems is to select a subset of sensors with a minimal cost out of a potentially large (and expensive) set. This task is challenging to accomplish because the state of complex systems can rarely be inferred from an output obtained from a single sensor. Instead, the state is concluded from a combination of several outputs from different sensors. In this paper, we introduce the *sensor selection problem* (SSP). The inputs for our model consist of the following: (a) a set of potential sensors that measure the systemand generats outputs, with each sensor having an associated installation cost; (b) a list of all possible *instances* of outputs from the sensors, where each instance is associated with a specific state of the monitored system.

We define the notion of *signature* as the 'partial instance' obtained from a subset of installed sensors, e.g., the signature of the instance (0, 1, 5, 0) with respect to sensors 1 and 3 is (0, 5). When the set of installed sensors is known, the set of possible signatures can be immediately derived from the list of possible instances.

The solution to our problem is a subset of sensors to be installed such that each signature can be associated with a unique state. That is, the state of the system can be determined by the set of installed sensors. This is achieved if the signatures of instances that are related to different states have different outputs in at least a predefined number of sensors (also known in the literature as the ‘testability condition’). This number is referred as the *minimal threshold quantity.* In many applications, this threshold is set to one. The objective is to find such a set with minimal cost.

The considered problem is a generalization of the *minimum test collection problem* (TCP), which is known to be NP-Hard (Garey and Johnson,1979) and APX-Hard (De Bontridder et al. 2003). The special case of the TCP, using our terminology, is the case when all the sensors produce binary outputs; each instance is associated with a unique state, the minimal threshold quantity is one, and the installation cost of all the sensors are identical (one unit without loss of generality).

Closely related problems were studied in the artificial intelligence, data mining, and statistical literature, i.e., in feature selection and variable selection problems, see for example Jović et al. (2015). Feature selection method based on a mathematical formulation can be found in Tseng and Huang (2007). They formulated the problem and devised a heuristic method based on the rough set theory. The method was applied on a dataset from the domain of customer relationship management (CRM). Sun et al. (2012) introduced a feature selection method that rely on concepts from cooperative game theory. It uses a measure based on Shapley value. However, the feasible set of these formulations is defined heuristically to represent a set of features that is believed

to enable prediction in practice. In this paper, we follow and extend the TCP by following an axiomatic approach concerning the definition of the feasible set.

The most similar optimization problem that was studied is the one presented in Bertolazzi et al. (2016). They considered a problem where a set of a given cardinality of features is selected so as to maximize the minimal threshold quantity. They formulated the problem as an integer linear program (ILP) and devise a GRASP heuristic to solve it. In their numerical experiment, they showed that their method is effective for the case when the number of sensors (features) is large while the number of instances is moderate.

We consider a similar problem and our unique contribution is in two aspects. First, we cast the SSP as a weighted version of the generalized TCP where the goal is to minimize the cost of the sensors rather than minimize their number. Second, and more important, we devise an exact solution strategy that is specialized in solving cases where the number of instances is large while the number of sensors is moderate.

Our exact solution method for the SSP is based on an ILP formulation of the problem coupled with an extensive pre-processing scheme for the elimination of redundant constraints and decision variables. After this step, the integer program can be easily solved even for datasets with a large number of instances, such as tens of thousands. We refer to this algorithm as IPSS.

Karwan et al. (1983) presented a review of methods for identifying and removing redundancy in ILPs. Later, Paulraj and Sumathi (2010) compared between five methods for identifying the redundant constraints. However, the constraints elimination procedure presented in this study is unique and based on the particular properties of the SSP and our formulation.

In Section 2, we present some formal notation and an ILP formulation of the problem. In Section 3, we devise an algorithm that eliminates many of the constraints and some of the decision variables. Then, the effectiveness of the proposed algorithm is demonstrated using publicly-available datasets. In Section 4, we present the applicability of the SSP and our solution method as a feature selection algorithm by comparing it with the results of feature selection algorithms from the literature on the datasets presented in Section 3. In section 5, we suggest an extension of our model that can handle datasets with continues numerical data and test this extension with a real-life dataset. Some concluding remarks are drawn in Section 6.

2. Notation and ILP Formulation

We consider a design problem where a set of sensors is selected in order to monitor a system and identify its states. The selected set of sensors has to identify all the system states while minimizing the sensors' costs. A feasible solution to the considered problem is given by a subset of sensors to be used such that the signatures of instances that are related to different states will obtain different outputs in at least a predefined number of sensors. This number is referred as the minimal threshold quantity. A feasible solution is a subset of sensors, such that each of its unique signatures can be

associated with a single state. An optimal solution is a feasible solution that minimizes the cost of the sensors in the subset.

Let us demonstrate the problem by the following example. Consider a system consist of four potential sensors, each sensor, in this case, produces a binary output. Assume that there are seven different possible instances (denoted by IDs 1-7) and two possible states: "Positive" and "Negative". The input of this small example is presented in *Table 1*. The last row of the table, ` the cost of installing each of the potential sensors.

Table 1: All possible sensors, their cost, instances and their states

Sensors:		**1**	**2**	**3**	**4**	**State**
IDs and Instances	1	0	0	1	1	Positive
	2	0	1	0	1	Positive
	3	1	1	0	1	Positive
	4	1	0	1	1	Positive
	5	1	0	0	1	Negative
	6	0	1	1	0	Negative
	7	0	0	1	0	Negative
Sensor Cost		**4**	**3**	**6**	**5**	

Note that the signatures of sensors 2, 3, and 4 can be uniquely mapped to the two states, as demonstrated in *Table 2* where the resulted signatures and the IDs of the instances from which they are originated are presented. The total cost of these three sensors is 14, which turns out to be the optimal solution for this example. Clearly, for any legitimate input (i.e., when each unique instance is mapped to a single state), the set of all the candidate sensors is a feasible solution.

Table 2: The optimal solution for the example presented in *Table* 1

Sensors:		**2**	**3**	**4**	**State**
Instance IDs and signatures	1,4	0	1	1	Positive
	2,3	1	0	1	Positive
	5	0	0	1	Negative
	6	1	1	0	Negative
	7	0	1	0	Negative

For a counterexample, consider installing sensors 2 and 4 only. This configuration is not a feasible solution because the signature (0, 1) is obtained from Instances 1 and 4 (which are related to "Positive") and from Instance 5 (which is related to "Negative").

In order to evaluate the quality of a solution (a set of sensors) that is not feasible with respect to the SSP we define the *reliability measure.* This measure allows a better comparison of our solution with ones produced by other methods in the literature, as

seen Section 4. The reliability measure is defined for any set of sensors, including those that represent infeasible solutions, as the hit ratio of the best possible mapping of signatures to states. The measure can be calculated with respect to any particular dataset and set of sensors by associating each signature with its most likely state, i.e., the one that minimizes the number of instances with respect to the classification errors. The SSP with a minimal threshold quantity of one is defined as the selection that minimizes the cost of the sensors that are required to obtain a 100% reliability with respect to a given dataset. On the other hand, feasible solutions can be classified by their minimal threshold quantity.

In *Table* 3 we demonstrate the calculation of the reliability measure for a configuration that consists of sensors 2 and 4 with respect to the dataset in the example of *Table 1*. In Column 6 we count the number of instances associated with the signature, and in Column 7 we count the number of correctly mapped instances. The ratio between their totals (6/7 in this example) is the value of the reliability measure.

Table 3: The obtained signatures from sensors {2,4} and their related states

Sensors:		**2**	**4**	**Positive**	**Negative**	**Instances**	**Correct**
Instance IDs and signatures	1,4,5	0	1	2	1	3	2
	2,3	1	1	2	0	2	2
	6	1	0	0	1	1	1
	7	0	0	0	1	1	1
Total						7	6

Let us now present our integer programming formulation of the SSP. For this purpose, we use the following notation:

- N Set of candidate sensors available in a given system; The number of sensors is denoted by $n = |N|$.
- c_i The cost of installing Sensor i for all $i \in N$.
- V_i The set of all possible outputs that can be obtained from Sensor i. We assume that this is a discrete set (later this assumption is relaxed).
- R The set of valid instances, $R \subseteq V_1 \times V_2 \times \cdots \times V_n$; for each instance $\mathbf{r} \in R$, we refer to the output of the i^{th} sensor by r_i.
- K The set of possible system's states $K = \{1, \dots, k\}$.
- $k_{\mathbf{r}}$ The state of the system given instance $\mathbf{r} \in R$.
- α The minimal threshold quantity

For each sensor $i \in N$, we define a binary decision variable x_i that is equal to "1" if the sensor is included in the configuration and is "0" otherwise. The SSP can now be formulated as follows,

$$\min \sum_{i \in N} c_i x_i \tag{1}$$

Subject to

$$\sum_{i: r_i \neq q_i} x_i \geq \alpha \qquad \forall \mathbf{r}, \mathbf{q} \in R: k_{\mathbf{r}} \neq k_{\mathbf{q}} \tag{2}$$

$$x_i \in \{0,1\} \quad \forall i \in N$$

The objective function (1) minimizes the total cost of sensors in the configuration. The set of constraints (2) ensure that, for every two instances that are related to different states, at least α sensors with a different output are included in the configuration.

3. Solution method

In this section, we present two solution methods for the SSP. In Section 3.1, we introduce an exact solution method and in Section 3.2, we present a simple greedy heuristic that delivers a feasible solution in very short time. In Section 4, we compare the preferences of these methods.

3.1 Integer-programming based sensor selection (IPSS)

We first note that the column dimension of ILP (1)-(2), i.e., the number of decision variables is equal to the number of candidate sensors, and the number of constraints is quadratic in the number of instances [$O(|R|^2)$]. In a typical application, we expect thousands of instances, which imply millions of constraints, whereas the number of candidate sensors is typically much smaller.

To solve large instances of the model in a reasonable time, our algorithm searches and detects *redundant constraints* and exploits the structure of the problem to fix the values of some decision variables. In this section, we show that this scheme leads to models that can be solved almost instantly by an ILP solver. Moreover, in some cases, the entire set of decision variables is fixed, and the ILP is solved to optimality at this preprocessing step.

The basic idea of our algorithm is as follows: recall that, for each pair of instances $\mathbf{r}, \mathbf{q}$ that are associated with different states, there is a constraint in the model. Let us define the set of sensors $S_{\mathbf{rq}} = \{i : r_i \neq q_i\}$.Using this notation, constraints (2) can be rewritten as (2').

$$\sum_{i \in S_{\mathbf{rq}}} x_i \geq \alpha \qquad \forall \mathbf{r}, \mathbf{q} \in R: \; k_{\mathbf{r}} \neq k_{\mathbf{q}} \tag{2'}$$

Let us denote the collection of sets that defines (2') by $\mathbb{C} = \{S_{\mathbf{rq}}: k_{\mathbf{r}} \neq k_{\mathbf{q}}\}$ and note the following:

Proposition 1: Let $S, S' \in \mathbb{C}$ such that $S \subset S'$. Any solution **x** that satisfies (2') for S also satisfies it for S'.

Proof: Immediately from the fact that if $S \subset S'$ and $x_i \geq 0$, then $\sum_{i \in S} x_i \leq \sum_{i \in S'} x_i$.

Using the observation of Proposition 1, many constraints can be detected as redundant and eliminated. Moreover, the existence of $S \in \mathbb{C}$ such that, $|S| = \alpha$, implies that x_i=1 for all $i \in S$. Similarly, if a variable does not appear in any of the remaining constraints its value must be zero in an optimal solution; thus, the (positive) cost of the associated sensor is not paid.

We demonstrate the above idea using the small example of *Table 1* with $\alpha = 1$, as shown in *Table 4* below. In the first column of the table, we present a serial number of the obtained constraint instance. In Columns 2 and 3, we present a pair of instances from *Table 1* that are related to different states. For example, column 2 and 3 of the first constraint represent respectively instance 1 and instance 5 in Table 1 that are associated to different states (Positive and Negative, respectively). In Column 4, we show the resulted $S_{\mathbf{rq}}$ for this pair. Thus, for the first constraint both sensors 1 and 3 have different outputs and therefore included in the set $S_{\mathbf{rq}}$. The respective constraint (2') is shown in the rightmost column.

From Constraint Instances 3, 7, and 10, one can see that sensors 2, 3, and 4 must be included in any feasible solution. Thus, one can fix $x_2 = x_3 = x_4 = 1$ and all the other constraints can be eliminated based on the observation of Proposition 1, implying that $\mathbf{x} = (0,1,1,1)$ is an optimal solution in this case. In other cases, we could end this process with some remaining constraints and non-fixed variable and find the optimal solution using an ILP solver applied for the reduced model.

Table 4: An explicit example of ILP (1)-(2) [for $\alpha = 1$]

Constraint #	**r**	**q**	$S_{\mathbf{rq}}$	Constraint
1	(0,0,1,1)	(1,0,0,1)	{1,3}	$x_1 + x_3 \geq 1$
2	(0,0,1,1)	(0,1,1,0)	{2,4}	$x_2 + x_4 \geq 1$
3	(0,0,1,1)	(0,0,1,0)	{4}	$x_4 \geq 1$
4	(0,1,0,1)	(1,0,0,1)	{1,2}	$x_1 + x_2 \geq 1$
5	(0,1,0,1)	(0,1,1,0)	{3,4}	$x_3 + x_4 \geq 1$
6	(0,1,0,1)	(0,0,1,0)	{2,3,4}	$x_2 + x_3 + x_4 \geq 1$
7	(1,1,0,1)	(1,0,0,1)	{2}	$x_2 \geq 1$
8	(1,1,0,1)	(0,1,1,0)	{1,3,4}	$x_1 + x_3 + x_4 \geq 1$
9	(1,1,0,1)	(0,0,1,0)	{1,2,3,4}	$x_1 + x_2 + x_3 + x_4 \geq 1$
10	(1,0,1,1)	(1,0,0,1)	{3}	$x_3 \geq 1$
11	(1,0,1,1)	(0,1,1,0)	{1,2,4}	$x_1 + x_2 + x_4 \geq 1$
12	(1,0,1,1)	(0,0,1,0)	{1,4}	$x_1 + x_4 \geq 1$

In the case of $\alpha > 1$, any constraint with α variables in the left-hand side implies fixing the values of these variable to one. A constraint with less than α variables immediately implies infeasibility. Indeed, for the dataset in the above example, Constraints 3, 7, and 10 implies that the problem is infeasible for any $\alpha \geq 2$.

The algorithm presented as pseudocode in *Figure 1* detects all the redundant constraint instances and fixes the required decision variables by scanning all the pairs of instances that are related to different states. For each such pair of instances, the set of sensors with different outputs (S) is constructed. The set is added to the collection $\mathbb{C}$ only if it *does not contain* a previously added set. In addition, if the set is contained in previously added sets, then these sets are removed from $\mathbb{C}$. From a computational effort point of view, the dominating parts of the algorithm are the inclusion tests, both $S' \subseteq S$ and $S \subset S'$. It is possible to spare many of these tests by partitioning $\mathbb{C}$ into subsets of equal cardinality and then test the inclusion of a set only versus sets with cardinality that is not smaller.

The algorithm can be parallelized in a relatively simple manner by dividing the work done in the inner loop among several processors. Indeed, when checking the inclusion of a particular set S versus each of the members of the large collection $\mathbb{C}$, each inclusion test can be done independently. Since the inclusion tests are responsible for almost all of the computational effort, such an approach can reduce the running time by a factor that is close to the number of processors. However, the implementation of the algorithm used for the numerical experiment reported below is based on the simpler serial approach.

Input: A set of instances R, where each instance is mapped to a state.

Let $\mathbb{C} = \emptyset$ *// start with an empty collection of constraints*

For all *two elements subsets* $\{r, q\} \subset R$
    If $k_{\mathbf{r}} \neq k_{\mathbf{q}}$ then
        $S = \{i \in N: r_i \neq q_i\}$
        If there is no $S' \in \mathbb{C}$ s.t $S' \subseteq S$
            Delete all $S' \in \mathbb{C}$ such that $S \subset S'$
            Add S to $\mathbb{C}$
Return $\mathbb{C}$

Figure 1: Pseudocode of the constraint dimension reduction algorithm

The dimension reduction algorithm was implemented in Python 2.7 as a single thread application, and the ILP model was solved by IBM Cplex 12.6.3 that can employ all the cores of the CPU. Our testing environment is an eight-core Intel i7-4790 3.60 GHz CPU, 32 GB of RAM running under Windows 7, 64 bit.

The effectiveness of the dimension reduction algorithm was tested on nine datasets from the UCI Machine Learning Repository (Lichman 2013). A summary of the datasets is presented in *Table 5*. Each of the datasets was solved both by using the dimension reduction technique followed by solving the remaining model using Cplex and by solving the original (unreduced) model directly with Cplex. The results presented here are for $\alpha = 1$. In *Table 5*, the model running time of CPLEX (in seconds)

of the IPSS for each dataset, with and without our reduction method, is given. In addition, the number of constraints in the full model and the number of remaining constraints after the reduction is shown in the table. The merits of the proposed method is clear. In particular, for large datasets (e.g., letter recognition and Connect-4), Cplex could not solve the problem due to an out-of-memory error, while the reduced models were solved very quickly. In some of the smaller instances all the constraint were eliminated and all the decision variables were fixed by the IPSS algorithm. The running time of the reduction process grows sharply with the dimensions of the problem but it does not use an excessive amount of memory and can be applied for large instances. Recall that the serial python implementation is benchmarked against a state-of-the-art solver that exploits all the cores of the CPU.

Table 5: Reduction approach with respect to state-of-the-art ILP solver

	Datasets summary			Full model		Reduction algorithm		
Dataset	Features	Instances	States	Obtained constraints	Solution Time (*Sec*)	Remaining Constraints	Reduction time	Solution Time (*Sec*)
Monk 1	6	432	2	46,656	0.74	0	0.19	0
Monk 2	6	432	2	41,180	0.48	0	0.17	0
Monk 3	6	432	2	46,512	0.45	0	0.144	0
Zoo	16	101	7	3,873	0.125	12	0.065	0.037
Tic-tac-toe	9	958	2	207,832	2.72	36	1.136	0.034
Chess	36	3,196	2	2,548,563	118.6	2	21.28	0.035
Mushrooms	22	8,124	2	16,478,528	415.67	39	149.21	0.041
Letter recognition	16	20,000	26	192,300,979	Out of memory	62	1,177	0.505
Connect-4	42	67,557	3	1,133,893,847	Out of memory	697	15,789	0.172

3.2 Greedy sensor selection heuristic

The *greedy sensor selection* (GSS) algorithm starts with a feasible solution that consists of all the sensors. At each iteration, one sensor is considered for removal by checking the feasibility of a solution consisting of the remaining sensors without it. The sensors are scanned in decreasing order of the ratio between their costs and the reliability measure of a configuration that consists of the tested sensor only. We refer to this ratio as the cost ratio. This procedure can yield a feasible solution for the problem in a short time.

We demonstrate this simple idea using the dataset presented in *Table 1* assuming $a = 1$. The cost ratio of sensors 1, 2, 3, and 4 are 7, 5.25, 10.5, and 5.83, respectively. Thus, the first sensor to be considered for removal is 3. However, because the remaining set of sensors $\{1,2,4\}$ is not a feasible solution, Sensor 3 is not removed. The next sensor to be considered is 1. The set of the remaining sensors is $\{2, 3, 4\}$, which is a feasible solution. Thus, Sensor 1 is removed. Next, sensors 4 and then 2 are considered; however, both cannot be removed. Thus, the obtained solution is the set of sensors $\{2,3,4\}$ which is, in this case, happens to be an optimal solution. Note that the GSS can be used to solve the problem with $\alpha > 1$. Sensors are removed from the configuration as long as Constraint (2) is satisfied.

The running time of the GSS is negligible and it guaranteed to return a feasible solution. However, this solution can be suboptimal as we demonstrate the next section.

4. Application in feature selection and numerical experiment

In this section we compare the proposed sensor selection problem to the known feature selection problem. The comparison is based on a rough analogy by which each sensor output can be considered as a feature value and each system state can be consider as a system class value. Note however that despite the considered analogy the two problems are inherently different, since the sensor selection problem is a deterministic optimization problem that guarantees a 100% identification reliability over a given set, while the feature selection problem is a probabilistic procedure that aims towards a high reliability (often lower than 100%) over a test set that is not given in the training stage. Despite these significant differences, we believe that the following comparison study is both interesting and has an educational value.

Feature selection methods are well studied in data mining, machine learning, and statistical pattern recognition literature. Feature selection is often used to remove irrelevant, redundant, and noisy data. Effective feature selection results in speeding up data analysis algorithms and improving their predictive accuracy (Balamurugan and Rajaram, 2009).

Feature selection methods are categorized into three basic approaches: filter, wrapper and embedded. In the filter approach, various subsets of features are explored to find a small subset that allows accurate classification, without directly applying any classification algorithm to test it. Wrapper methods, however, are search algorithms that apply classification algorithm iteratively to find a small subset of features that allows good classification performances. Embedded methods can be addressed as a special class of wrapper based methods, in which the feature selection technique is suited to only a particular classifier. The proposed Optimization-based Sensor Selection (OSS) model can be viewed as an embedded feature selection method because the result is a set of sensors (i.e., the equivalent of features), with the signature of each sensor mapped to a state (i.e., an equivalent of class). However, if the set of instances does not represent all the possible values, which is the typical situation in learning scenarios, one can view the OSS as a filter method. Indeed, for new, yet unseen, signatures, it is possible to apply many known classification methods (e.g., KNN, SVM, and Decisions trees). Note that, in the considered setting, the classification accuracy for the training set is always 100% (i.e., deterministic classification), unlike a typical classification over a training set. Evaluating the classification accuracy using a test-set is not considered and is beyond the scope of this study.

In this section, we examine the applicability of our sensor selection model to the feature selection domain. To this end we apply the proposed model to the benchmark datasets given in *Table 5,* aiming to identify a subset of features that allow a distinction between the classes of the observation deterministically over all the data. We compare the selected features to other sets of features selected by state-of-the-art feature

selection methods taken from the literature. Finally, we calculate the reliability of these selected sets of features.

Before we proceed to the comparison study, let us note that there is an essential difference between the proposed sensor selection problem and the feature selection problem. The former is a well-defined optimization method where the objective function represents the actual problem at hand. In contrast, in feature selection methods the goal is to select a set of features that are likely, in some heuristic sense, to allow successful classification of future instances. Moreover, in the considered problem the more generic goal is to minimize the total cost of the sensors rather than their number which can be represented by a specific case in which the sensor's cost is fixed and identical. Accordingly, for the sake of a fair comparison, we set the cost of all the features to one. In Section 5, however, we present a real-world case study in which a different cost is associated with each sensor/feature.

Table 6 presents the number of sensors selected by the proposed method vis-à-vis state-of-the-art methods from the literature for various datasets. The first column presents the name of the dataset, the second column presents the total number of features. The third column presents the references and summarizes the result from the literature, including the number of selected features and their reliability. The fourth and fifth columns present the number of features selected by the exact IPSS algorithm and by the GSS heuristic, respectively. Note again that both methods yield a set that guarantees a 100% reliability level, unlike a typical classification problem.

For the Mushrooms dataset, four features (namely, odor, spore-print-color, population, and habitat) out of the 22 are found by the OSS model as mandatory features. This solution implies that the toxicity of any of the 8124 mushrooms can be accurately (at a reliability level of 100%) determined based solely on these four features and it cannot be accomplished with any smaller subset of features. Just for comparison purpose, many studies used the Mushrooms dataset as a benchmark. For example, Kohavi and Frasca (1994) found five sensors using a search of useful features subset based on rough set theory. Liu and Setiono (1996) used a probabilistic approach that yields also four features that although different could guarantee a reliability level of 100% that is necessary in such a case, where a mistake can cause a severe health damage.

For the zoo dataset, five sensors out of 16 were found by the OSS model, while for comparison purpose, the method of Balamurugan and Rajaram (2009) selected 13 Features. Wang et al. (2007) used a technique based on the rough sets and particle swarm optimization. Their methods resulted in five features. In a personal correspondence, the authors informed us that these features are 3, 4, 6, 8, and 13. The reliability was calculated based on this information.

Another example of the effectiveness of the proposed algorithm can be seen in the letter recognition dataset. Our proposed algorithm selected 11 sensors out of 16. In other words, it indicates that one can use 11 features to fully determine the actual letter represented by each of the 20,000 instances (samples) in the set. The implication of this finding for those who wish to identify handwritten letters is that it is sufficient to store less than 2^{11} signatures and compare them to the signature of each new input rather

than all the instances in the dataset. Devi (2015) proposed a feature selection algorithm based on simulated annealing that can dichotomically distinguish between each letter versus all the others. The reported features for each letter ranges from 9 to 15, where at least 15 features are necessary to distinguish between all classes. Oh et al. (2004) claimed that they could achieve "good" classification performances for this dataset with 10 or 13 features using a hybrid genetic algorithm. Both papers did not provide the identity of their selected feature and therefore we could not evaluate their reliability.

Following this line of examples, we also tested the OSS on the three Monk's problem datasets. Each problem is given by a logical description of a class, and the learning task is to derive a simple class description (see Thrun et al., 1991). The OSS model selected three, six, and three features for Monk's problems 1, 2, and 3, respectively. Kohavi and Frasca (1994) reported on the same of features for the first and the second problem but suggest two features for the third problem. Once again, the OSS solution is different in the sense that it ensures perfect classification for the training set.

Despite the relatively good results reported above, the proposed OSS cannot always eliminate a large number of features that enable detecting the class deterministically. Because other feature selection methods do not aim at guaranteeing 100% accuracy on the training set, in some cases, they result in smaller number of features. For example, for the connect-4 dataset, the proposed OSS method reduced the number of features from 42 features to 34, whereas Balamurugan and Rajaram (2009) reduced it to 13; unfortunately, since the authors did not provide the selected features, the exact reliability level could not be calculated, yet obviously is lower than 100%. A comparable situation occurs on the Chess dataset. The OSS model proved that at least 29 features are required for complete classification, whereas Hall (1999) found three features using his correlation-based feature selection search strategy with reliability of 55.39%, and Kohavi and Frasca (1994) found ten features with reliability of 97.78%. Another example is the tic-tac-toe dataset, where the OSS reduced only one feature, whereas Kohavi and Frasca (1994) reduced two features, yet again with a cost of some probability of having classification errors.

Table 6: Comparison of the number of selected features and reliability according to IPSS, GSS, and feature selection methods from the literature

Dataset	No. of features	Literature: number of selected features (reliability*)	IPSS	GSS
Monk 1	6	Kohavi and Frasca (1994): 3 (100)	3 (100)	3 (100)
Monk 2	6	Kohavi and Frasca (1994):6 (100)	6 (100)	6 (100)
Monk 3	6	Kohavi and Frasca (1994): 2 (97.23)	3 (100)	3 (100)
Zoo	16	Balamurugan & Rajaram (2009): 13 (N/A) Wang et al. (2006): 5 (100)	5 (100)	5 (100)
Tic-tac-toe	9	Kohavi and Frasca (1994): 7 (99.58)	8 (100)	8 (100)
Chess	36	Hall (1999): 3 (55.39) Kohavi and Frasca (1994): 10 (97.78)	29 (100)	29 (100)
Mushrooms	22	Liu and Setiono (1996): 4 (100) Kohavi and Frasca (1994): 5 (100)	4 (100)	5 (100)
Letter recognition	16	Devi (2015): 15 (N/A) Oh et al. (2004): 10, 13 (N/A)	11 (100)	12 (100)
Connect-4	42	Balamurugan & Rajaram (2009): 13 (N/A)	34 (100)	41 (100)

*full classification is NOT guaranteed!

Note that the GSS heuristic provides an optimal solution for the smaller datasets but, as expected, fails in some of the larger ones, thus motivating the use of the computationally heavy IPSS algorithm.

In conclusion, one can see that the proposed OSS model can be practically used as a feature selection method based on a conservative approach. That is, the method always selects an exactly minimal number of features that are required for a complete classification of the instances in the training set, whereas other methods either result in a larger number of features or in a set that cannot guarantee perfect classification over many of the UCI training datasets.

5. Extension to continuous outputs

Until now, we based our sensor selection model on the assumption that the outputs produced by the sensors are discrete values (categorical). In this section, we extend the model to capture cases, where some of the outputs are continuous values (e.g., such as temperature and pressure). Recall that, for the discrete case, we assumed that the training set contains all the possible instances. This assumption is not feasible in the context of continuous outputs. Note that, if the accuracy of the output values is high, it is very likely that a single sensor will be enough to distinguish among all the categories in the training set, but has very little power with respect to instances that are not included in the dataset. To overcome this difficulty, we aim at considering two continuous output values as different, only if the difference between them is significant enough. For example, if a sensor collects a body temperature, a difference of one degree

may be considered significant; however, a difference of 0.01 degrees is probably not indicative for changing the diagnosis of medical conditions.

One alternative approach is to discretize or categorize such numerical values; note however, that the categorization process itself may result in a loss of relevant information. To overcome this problem, we incorporate the categorization process into the optimization model. All the numerical values in our dataset are normalized and expressed in terms of standard deviation scores around their means. Both the mean and the standard deviation are estimated from the dataset. Two outputs of a sensor (or values of a feature) are considered different if the difference between them, in terms of standard deviations, exceed some predefined *threshold* parameter, denoted by T. Accordingly, to reflect this modification, we revise constraint (2) as follows:

$$\sum_{i\in N:|r_i'-q_i'|>T} x_i + \sum_{i\in C:r_i\neq q_i} x_i \geq \alpha \qquad \forall \mathbf{r},\mathbf{q} \in R: k_{\mathbf{r}} \neq k_{\mathbf{q}} \qquad (2")$$

Where N and C are the sets of continuous and categorical sensors (or features), respectively. r_i' and q_i' denotes the normalized values of the output of sensor i of instances $\mathbf{r}$ and $\mathbf{q}$, respectively. The set of constraints (2") ensures that, for every two instances that are related to different states, the outputs of at least α sensors are significantly different. The values of T and α determine the sensitivity of the model. Higher values typically result in costlier solutions but may allow better classification. The value of T and α parameters can be fine-tuned by solving the problem for various combinations of these parameters. The accuracy obtained with the selected features of each combination can then be evaluated. Using this process, we create an efficiency frontier of the total cost versus the model accuracy.

To examine the applicability of the sensor selection model on continuous sensors, we tested the Pima Indians Diabetes (PID) dataset that is also acquired from the UCI machine learning repository. Diabetes is a disease in which the body is unable to properly use and store glucose. Poorly managed diabetes can lead to a host of long-term complications, including heart attacks, strokes, blindness, and kidney failure. The PID dataset consists of eight continuous features, i.e., diagnostic measurements, as presented in *Table 7*. The task is to predict, based on these measurements, whether a patient has diabetes or not. All the patients in the dataset are females, at least 21 years old of Pima Indian heritage. The data has 768 instances (patients); however, many of them contain missing values. Like other studies that use this dataset, e.g., Karegowda et al. (2010), we used only the 392 instances with no missing values. In this group 130 women were positively diagnosed with diabetes.

We solved the OSS model for this dataset assuming unit sensor (feature) cost with $\alpha = 1{,}2$ and with $T = 0, 0.01, 0.02, \ldots, 0.32$. Feasible solutions could be found for $T \leq 0.31$ where $\alpha = 1$ and $T \leq 0.19$, where $\alpha = 2$. Each of the obtained configurations was evaluated in a 10-fold cross validation process using the *k*-nearest neighbors (*k*-NN) classification method with $k = 1, \ldots, 30$ and three different metrics namely: Euclidian, Cityblock and Mahalanobis. The best k and the best metrics, in terms of mean classification error (MCE) were used for each configuration.

In *Figure 2* we plot all the 52 solutions for the various α and T combinations on the cost versus MCE plan. We obtained three points on the efficiency frontier, all with $\alpha = 1$. One with features $\{6,7\}$ when $T = 0$. In this case the MCE is 0.327. The second is with features $\{2,6,8\}$ when $T = 0.07$ and MCE is 0.212. The last point on the frontier is with sensors $\{2,6,7,8\}$ when $T = 0.17, 0.18, 0.19$. In this case, the MCE is 0.204.

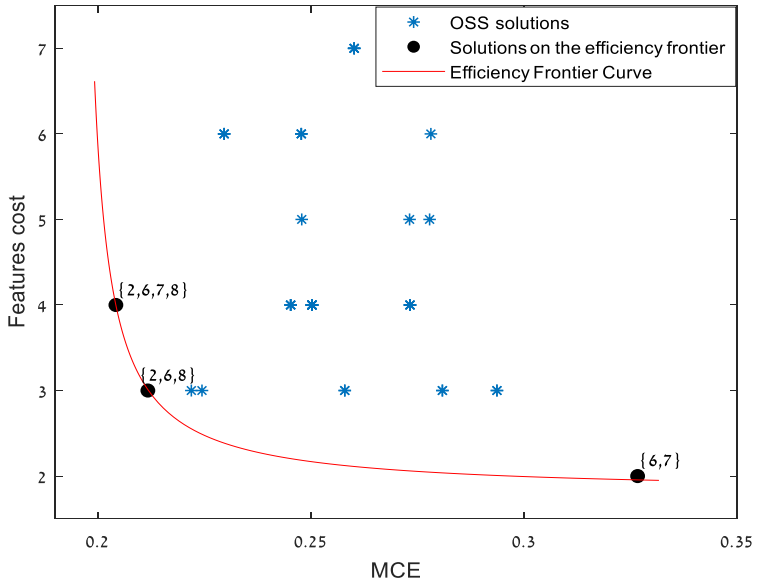

Figure 2: Feature's cost versus MCE for all OSS solutions of the various α and T

To benchmark our method, we compared our results with the results of Karegowda et al. (2010). They used two filter approaches: one based on C4.5 and another based on a genetic algorithm (GA). We used the same training and testing procedure with *k*-NN to test the features selected by these methods. We note that Karegowda et al. (2010) reports on similar classification results.

In *Table 7*, the solutions obtained by our method (with $T = 0.07$ and $T \in [0.17,0.19]$) are compared to the results of C4.5 and GA. Indeed, two of the three points on our efficiency frontier dominates the results obtained by the above-mentioned methods in terms of both MCE and costs (number of features).

Table 7: Description of PID features

Feature index	Feature name	OSS T=.07	OSS T=.19	C4.5	GA
1	number of pregnancies				
2	plasma glucose concentration at 2 hours	X	X	X	X
3	diastolic blood pressure (mm Hg)			X	
4	triceps skinfold thickness (mm)				
5	2-Hour serum insulin (mu U/ml)				X
6	BMI (weight in kg/(height in m)^2)	X	X	X	X
7	diabetes pedigree function		X	X	
8	Age (years)	X	X	X	X
MCE		**0.212**	**0.204**	0.227	0.214

In practice, medical tests are related to different data collection costs. For example, measuring the weight of a person (easy and cheap to perform) versus measuring her plasma glucose level (may be more expensive and painful). Indeed, the Pima dataset specifies the cost of each of the eight tests. The second column of *Table 8*, presents the cost of each feature (test). The optimal set of features obtained by the OSS, in this case with $T = 0.02$ is presented in the third column of the table and the result for C4.5 and GA are presented in the two right most columns. The OSS selects tests at a total cost of $3 while the other methods, which are oblivious to the testing cost, selects much more expensive tests ($21.61 and $42.39) that yield only slightly better prediction. Thus, using the proposed OSS approach one can observe and address a tradeoff between the classification performance and the cost, unlike many conventional machine learning methods that do not take it into account.

Table 8: Solution of the weighted version

Feature name	Test Cost ($)	OSS T=.02	C4.5	GA
number of pregnancies	1.00			
plasma glucose concentration at 2 hours	17.61		X	X
diastolic blood pressure (mm Hg)	1.00		X	
triceps skinfold thickness (mm)	1.00			
2-Hour serum insulin (mu U/ml)	22.78			X
BMI (weight in kg/(height in m)^2)	1.00	X	X	X
diabetes pedigree function	1.00	X	X	
Age (years)	1.00	X	X	X
MCE		0.258	0.227	0.214
Testing cost		$3	$21.61	$42.39

6. Conclusions

This paper introduce a problem of optimal sensor selection and propose a method for solving it. The objective is to minimize the total cost of a set of selected sensors while guaranteeing a full identification of various system states. The considered problem is a generalization of the minimum test collection problem (TCP) which is known to be NP-Hard and APX-Hard. An ILP formulation is presented, and a method to reduce its dimension is devised based on the problem characteristics. The reduced ILP model is shown to obtain an exact solution for large instances of the problem, as demonstrated in an extensive numerical study that is reported here.

The SSP can also be considered as a generalization, or modification of the well-known feature selection problem. It extends the feature selection problem in the sense that a specific cost can be assigned to each feature (sensor). Moreover, it provides both a lower bound on the number of required features that can guarantee a zero-classification error, as well as a method to evaluate the tradeoff between the model accuracy vs. the feature costs. The proposed OSS method was applied to various datasets and compared to other feature selection methods. It is shown to be superior in many use cases and address the above phenomena.

Since the optimal solution of the presented problem is a set of sensors (features) that can be used to identify fully the state of each instance, the OSS may result in a set of sensors that is too large (or too expensive). For situations in which a small probability of classification errors can be tolerated, an interesting direction for future research is to develop a model that considers more rigorously the tradeoffs between the cost of misclassification and the cost of the sensors. Another direction for future research is related to the use of the proposed method to a learning scheme, in which the algorithm is executed over a training set while the performance of the selected sensors is evaluated over a new test set.

Acknowledgment
This work is partially supported by Shlomo Shmeltzer Institute for Smart Transportation grant.

References

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., & Protasi, M. (2012). Complexity and approximation: Combinatorial optimization problems and their approximability properties. Springer Science & Business Media.

Balamurugan, S. A. A., & Rajaram, R. (2009) Effective and efficient feature selection for large-scale data using Bayes' theorem. *International Journal of Automation and Computing* 6(1), 62-71.

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.

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.

Devi, V. S. (2015, December). Class Specific Feature Selection Using Simulated Annealing. In *International Conference on Mining Intelligence and Knowledge Exploration* (pp. 12-21). Springer, Cham.

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

Hall, M. A. (1999). Correlation-based feature selection for machine learning. PhD Thesis. The University of Waikato, New Zealand.

Jović, A., Brkić, K., & Bogunović, N. (2015, May). A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on* (pp. 1200–1205). IEEE.

Karegowda, A. G., Manjunath A. S., & Jayaram M. A. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management, 2(2),* 271-277.

Karwan, M. H., Lofti, V., Telgen, J., & Zionts, S. (1983). Redundancy in Mathematical Programming: a State-of-the-Art Survey, volume 206 of *Lecture Notes in Economics and Mathematical Systems*.

Kohavi, R., & Frasca, B. (1994, November). Useful feature subsets and rough set reducts. In *Third International Workshop on Rough Sets and Soft Computing* (pp. 310-317).

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Liu, H., & Setiono, R. (1996, July). A Probabilistic approach to feature selection- a filter solution. In *ICML* (Vol. 96, pp. 319-327).

Oh, I. S., Lee, J. S., & Moon, B. R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on pattern analysis and machine intelligence, 26(11),* 1424-1437.

Paulraj, S., & Sumathi, P. (2010). A comparative study of redundant constraints identification methods in linear programming problems. *Mathematical Problems in Engineering, 2010*.

Sun, X., Liu, Y., Li, J., Zhu, J., Liu, X., & Chen, H. (2012). Using cooperative game theory to optimize the feature selection problem. *Neurocomputing, 97*, 86-93.

Thrun, S., Bala, J. Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., & Zhang, J. (1991). The MONK's problem: A performance comparison of different learning algorithms. (Technical Report CMU-CS-91-197). Pittsburgh, PA: Carnegie Mellon.

Tseng, T. L. B., & Huang, C. C. (2007). Rough set-based approach to feature selection in customer relationship management. *Omega*, *35*(4), 365-383.

Wang, X., Yang, J., Teng, X., Xia, W., & Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern recognition letters, 28(4)*, 459-471.