The Generalized Test Collection Problem

September 2019

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

Abstract

The test collection problem, also known as *the minimum test set problem* or *the minimum test cover problem*, selects a minimal set of binary sensors that can determine the state of a monitored system. We generalize this problem by *i*) allowing sensors to produce arbitrary categorical outputs; *ii*) allowing multiple sensor outputs to represent each state of a system and *iii*) including a cost per sensor. The purpose of the planer is to select a set of sensors at a minimum cost that can determine the state of the system. This problem has applications to various complex systems, including, for example, urban water networks, wherein the locations of pipe failures must be identified by sensors installed at specific junctions. In such systems, it is desirable to select the most inexpensive set of sensors that can provide sufficiently reliable information to support decisions regarding the system's maintenance and operations. To address this problem, we present an integer programming model and an effective exact solution method that uses the model's unique structure to reduce its size so that it can easily be solved. The solution method is implemented and demonstrated to be superior to those described in previous studies.

Keywords: Test collection problem, sensor selection, sensor placement, water networks, integer linear programming, constraint reduction

1. Introduction

In this paper, we introduce the *generalized test collection problem* (GTCP). Inputs of this problem consist of the following: (a) a set of potential sensors that measure various attributes of a system where each sensor is associated with an installation cost and (b) a list of all possible sensors' outputs (henceforth referred to as *readings*) where each reading is associated with a specific state of the monitored system.

We define a *signature* as the 'partial reading' obtained from a subset of sensors, e.g., in a system with four sensors, the signature of the reading (0, 1, 5, 0) with respect to sensors 1 and 3 is (0, 5), etc. When the set of selected sensors is known, the set of possible signatures can be immediately derived from the list of possible readings.

The solution to our problem is a subset of sensors whereby each signature can be mapped to a unique state of the monitored system. Rather, the state of the system can always be determined by the set of selected sensors. This solution can be achieved when the signatures of readings related to different states have different outputs for at least α sensors. In a typical situation $\alpha = 1$, however, for the sake of robustness (e.g., when sensors are prone to failure), it is sometimes useful to require a greater value of α.

The considered problem involves a rich generalization of the *minimum test collection problem* (TCP) also known as *the minimum test set problem* or *minimum test cover problem*. The decision version of this problem is an NP-complete problem (Garey and Johnson, 1979), and it is an APX-hard problem (De Bontridder et al. 2003). Using our terminology, the TCP constitutes a special case of the GTCP wherein all sensors produce binary outputs; each reading is associated with a unique binary state ("positive" or "negative"); $\alpha = 1$; and the costs of all of the sensors are identical. Needless to say, many real-life settings do not follow the above assumptions.

This study of the GTCP is motivated by the challenges associated with designing modern monitoring systems in domains such as manufacturing, smart cities, transportation, medicine, homeland security and agriculture. Such systems often consist of many sensors connected to a central processing unit (CPU) that detects the current state of the monitored system. A major task involved when designing these systems is to select a subset of sensors at a minimal cost from a potentially large (and expensive) set of sensors. The state of the system can rarely be inferred from an output obtained from a single sensor. Instead, it is often deduced from a combination of several outputs from different sensors.

Bertolazzi et al. (2016) studied another variant of the TCP for which the number of sensors was given, and the objective was to maximize the Euclidian distance between readings belonging to different classes. The authors refer to this value as the *minimal threshold quantity*. The paper formulates the problem as an integer linear program (ILP) and devises a greedy randomized adaptive search procedure (GRASP) as an appropriate heuristic for solving it. With their numerical experiment, the authors show that their method is effective when many sensors (or features) are present and when the number of instances is moderate.

Many researchers have studied various applications of the TCP. Some have used rather different terminology and formulations than ours. Such applications include

sensor placements for structures motivated by many devices such as wireless sensor networks for energy consumption; sensor networks for locating targets; and water systems. For example, Slijepcevic and Potkonjak (2001) studied the problem of placing sensor nodes into a monitored area so that full coverage can be achieved with minimal energy consumption. They cast it using a heuristic approach to the *set cover problem* (SCP) known as NP-hard. Sela Perelman et al. (2016) also used the SCP to find a desirable set of locations for sensors used to detect pipe failures occurring in an urban water network. The authors also developed a new *augmented greedy heuristic* for solving a problem equivalent to the TCP to locate pipe failures. In a later paper, Sela and Amin (2018) solve a robust version of the SCP problem where their goal is to detect pipe failures that occur when some sensors fail. We use their datasets that are based on an actual large water network to benchmark our solution method. Note that water networks are commonly examined in the scientific literature not only for fault location identification but also for water quality (Ostfeld et al., 2008) and contaminant detection (Krause et al., 2008).

Some authors haven also taken into account sensor costs. For example, Lin and Chiu (2005) considered the sensor placement problem for locating targets under cost constraints. They formulated this problem as a min-max mathematical optimization model and proposed a simulated annealing-based algorithm for solving it.

We make two novel contributions. First, we define the GTCP as a weighted version of the TCP with the goal of minimizing the cost of sensors rather than minimizing their number. Sensor outputs are assumed to taken general categorical values rather than binary ones, and multiple readings may represent the same system state. Second, and more importantly, we devise a successful exact solution strategy that is capable of solving large and realistic instances previously solved only heuristically. Our exact means of solving the GTCP is based on an ILP formulation of the problem coupled with an extensive preprocessing scheme for the elimination of redundant constraints and decision variables. To the best of our knowledge, this is the first effective and scalable exact solution method developed for the TCP and its variants that is based on mathematical programming.

Karwan et al. (1983) presented a review of methods that identify and remove constraint redundancy from ILPs. Later, Paulraj and Sumathi (2010) compared five means of identifying redundant constraints. However, the constraint elimination procedure presented in this study is unique and based on the particular properties of the GTCP and of our formulation.

The rest of this paper is organized as follows. In Section 2, we present notation and an ILP formulation of the problem. In Section 3, we devise the constraint reduction algorithm (CRA). In Section 4, the effectiveness of the proposed algorithm is demonstrated using twelve water networks studied in the literature. Our results are compared to previous solutions and to the effectiveness of solving the considered ILP model directly using a commercial solver. Concluding remarks and avenues for future research are given in Section 5.

2. Notation and ILP formulation

In this section, we demonstrate the GTCP with a small numerical example and we introduce notation and formulate the problem as an ILP. A feasible solution to the considered problem is given by a subset of selected sensors such that the signatures of readings related to different states will result in different outputs for at least a predefined number of sensors. An optimal solution is a feasible solution that minimizes the cost of sensors in the subset.

Let us demonstrate the problem with the following example. Consider a system that consists of four potential sensors such that sensors 1, 2 and 4 produce a binary output while sensor 3 produces a ternary output. Assume that there are seven different possible readings denoted by identifications (IDs) 1-7 and two possible states denoted as "Positive" and "Negative." The input for this small-scale example is presented in Table 1. The last row of the table presents the cost of installing each sensor.

Table 1: All possible sensors and their costs, readings and states

Sensors		**1**	**2**	**3**	**4**	**System State**
IDs and readings	1	0	0	Blue	1	Positive
	2	0	1	Red	1	Positive
	3	1	1	Red	1	Positive
	4	1	0	Gray	1	Positive
	5	1	0	Red	1	Negative
	6	0	1	Blue	0	Negative
	7	0	0	Gray	0	Negative
Sensor cost		**4**	**3**	**6**	**5**	

Note that the signatures of sensors 1, 2, and 3 can be uniquely mapped to the two system states as demonstrated in Table 2 where the resulting signatures and the IDs of the readings from which they originate are presented. Thus, this is a feasible solution to our problem (for $\alpha = 1$). The total cost of these three sensors is 13, which happens to be the optimal solution for this example. Note that when each unique reading is mapped to a single state, the set of all candidate sensors is a feasible solution.

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

Sensors		**1**	**2**	**3**	**System State**
Reading IDs and signatures	1	0	0	Blue	Positive
	2	0	1	Red	Positive
	3	1	1	Red	Positive
	4	1	0	Gray	Positive
	5	1	0	Red	Negative
	6	0	1	Blue	Negative
	7	0	0	Gray	Negative

As a counterexample, consider installing sensors 2 and 4 only. This configuration is not a feasible solution since the signature (0, 1) is obtained from readings 1 and 4, which are related to the "positive" system state, and from reading 5, which is related to the "negative" state. Thus, a design with sensors 2 and 4 only can result in ambiguity regarding the system state.

Let us now present the proposed integer programming formulation for the GTCP. 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 (this assumption is later relaxed).
- R The set of valid readings $R \subseteq V_1 \times V_2 \times \cdots \times V_n$; for each reading $\mathbf{r} \in R$, we refer to the output of the i^{th} sensor with r_i.
- K The set of possible system states $K = \{1, \dots, k\}$.
- $k_{\mathbf{r}}$ The state of the system represented by reading $\mathbf{r} \in R$.
- α The minimum number of selected sensors among which different states require different outputs. Recall that typically $\alpha = 1$.

For each sensor $i \in N$, we define a binary decision variable x_i that is equal to "1" when the sensor is included in the solution configuration and that is equal to "0" otherwise. Now, the GTCP can 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) ensures that for every two readings that represent different system states, at least α sensors with different outputs are included in the configuration.

3. The constraint reduction algorithm

In this section, we introduce an exact solution method that is based on a constraint reduction algorithm (CRA). We first note that the dimension of the 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 readings [$O(|R|^2)$]. For a typical application, we can expect to find thousands of readings, which imply millions of

constraints. Such models can be very difficult to solve explicitly due to memory and computation power limitations.

To solve large instances of the model using a reasonable amount of resources, our algorithm searches for and detects *redundant constraints* and exploits the structure of the problem to fix the values of certain decision variables. In this section, we show that this scheme produces models that can be solved within a relatively short amount of time with a commercial ILP solver. Moreover, in some cases, the entire set of decision variables can be fixed, and the ILP is solved during this preprocessing step.

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

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

Let us denote the collection of sets that define (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 $\mathbf{x}$ that satisfies (2') for S also satisfies (2') for S'.

Proof: Immediately, when $S \subset S'$ and $x_i \geq 0$, then $\sum_{i \in S} x_i \leq \sum_{i \in S'} x_i$.

From the observations 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, when 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 with the small-scale example illustrated in Table 1 with $\alpha = 1$ as shown in Table 3 below. In the first column of the table, we present the number of constraint instances obtained. In columns 2 and 3, we present a pair of readings taken from Table 1 related to different states (note that "B" denotes "blue," "R" denotes "red," and "G" denotes "gray" as the outputs of sensor 3). For example, columns 2 and 3 of the first constraint represent readings 1 and 5 in Table 1, respectively, which are associated with different states (positive and negative, respectively). In column 4, we show the resulting $S_{\mathbf{rq}}$ for this pair. Thus, for the first constraint, sensors 1 and 3 have different outputs and are therefore included in the set $S_{\mathbf{rq}}$. The respective constraint (2') is shown in the rightmost column.

From constraint instances 7 and 10, one can see that sensors 2 and 3 must be included in any feasible solution. Thus, one can fix $x_2 = x_3 = 1$. Then, except constraint instance 12, all constraints can be eliminated based on the observations of Proposition 1. The reduced ILP without redundant constraints is presented below:

$$\min 4x_1 + 5x_4$$

subject to

$$x_1 + x_4 \geq 1$$
$$x_1, x_4 \in \{0,1\}$$

The optimal solution for this ILP with fixed sensors (2 and 3) is $\mathbf{x} = (1,1,1,0)$ at a cost of 13. For this small-scale example, all eliminated constraints are implied by the singleton constraint $x_2 \geq 1$ and $x_3 \geq 1$. Note, however, that constraints can be eliminated due to any other constraints, e.g., $x_2 + x_3 + x_4 \geq 1$ can be eliminated due to constraint $x_2 + x_3 \geq 1$.

Table 3: An explicit example of the ILP (1)-(2) [for $\alpha = 1$]

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

When $\alpha > 1$, any constraint with α variables on the left-hand side implies that the values of these variables are fixed at a value of one. A constraint with fewer than α variables immediately implies infeasibility. Indeed, for the dataset used in the above example, constraints 7 and 10 imply that the problem is infeasible for $\alpha \geq 2$.

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

The algorithm can be parallelized in a relatively simple manner by dividing steps of the inner loop across several processors. Indeed, when checking the inclusion of a particular set S versus each member of the large collection $\mathbb{C}$, each inclusion test can be performed independently. Since the inclusion tests account for almost all of the computational effort exerted, such an approach can reduce the running time by a factor

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 readings R where each reading is mapped to a state.

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

For *two element 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 $S' \subseteq S$
            Remove all $S' \in \mathbb{C}$ such that $S \subset S'$W
            Add S to $\mathbb{C}$
Return $\mathbb{C}$

Figure 1: Pseudocode of the CRA

4. Numerical experiment

To demonstrate the effectiveness of our solution method and to benchmark it, we solve the sensor placement problem introduced by Sela Perelman et al. (2016). Their model uses a graph wherein edges represent pipes, valves, and pumps while nodes represent junctions, and sources in a system. Binary sensors that detect *pressure waves* caused by pipe bursts can be placed on each node in the network. A fault in each edge is associated with a true signal in several sensors located close to the fault location depending on the structure of the network and its topography. Each potential sensor can be affected by several fault locations and each fault may affect several sensors. The goal is to select a subset of nodes where sensors should be placed to detect each fault in the system and to identify its location. The objective is to minimize the number of sensors (implicitly assuming that installation costs are identical at all locations). Their problem clearly reflects a special case of the proposed GTCP model.

We coded the CRA in Python 3 as a single thread application and ran it with PyPy on an i9-9900K Linux machine with 64 GB RAM. The ILP models (both full and reduced) were solved using an IBM CPLEX 12.9 commercial MILP solver with the same machine. Note that CPLEX was implemented as a multithread application that efficiently employed all sixteen cores of the CPU. Sela Perelman et al. (2016) graciously made their input data available to us. These data are based on 12 different realistic water networks designated as Net1-Net12. These networks were adopted from Jolly et al. (2014) and from the Centre of Water Systems University of EXETER and range from a small instance with roughly 100 nodes to one with more than 12,000. Some simplifying assumptions as described in Sela Perelman et al. (2016) were made to map any possible failure in the network to a reading.

In Table 4, we report the performance of the CRA. In the first three columns we present the network name, the number of sensors and the number of unique readings for each network. In the fourth column we present the number of constraints of the full model. Since each state has one unique reading, the number of constraints is $\binom{|R|}{2}$. The

running time of the CRA is presented in the fifth column and the remaining number of constraints after reduction are presented in the sixth column. The solution time of the reduced model in CPLEX is presented in the last column of the table.

Note that in the original input, many readings related to faults in closely located pipes are identical, meaning that these types of faults are not identifiable with any possible sensor placement subject to the sensing technology assumed by Sela Perelman et al. (2016). Following their study, we consider these readings to belong to one state, i.e., one fault location. Moreover, we include a case involving no faults as an additional system state that can be identified whereas Sela Perelman et al. (2016) addressed this issue using a different *detection* model. In this sense, our model is slightly more constrained and may require the use of slightly larger number of sensors for optimal solution.

Table 4: Reduction approach for each network

Dataset features			Constraint reduction model			
Network	Number of nodes	Unique readings	Number of constraints	Reduction time (*sec*)	Remaining constraints	ILP time (*sec*)
Net1	126	110	5,995	0.03	50	0.01
Net2	269	317	50,086	0.12	242	0.03
Net3	420	428	91,378	0.24	262	0.03
Net4	481	549	150,426	0.72	600	0.21
Net5	543	558	155,403	0.53	357	0.03
Net6	791	784	306,936	0.91	678	0.13
Net7	778	761	289,180	0.9	373	0.05
Net8	811	1,058	559,153	4.35	973	0.51
Net9	959	1,006	505,515	1.44	802	0.07
Net10	1,325	1,437	1,031,766	4.81	983	0.2
Net11	1,891	2,094	2,191,371	6.95	1310	0.18
Net12	12,523	13,100	85,798,450	943.91	10846	7200*
* While the problem instance could not be optimally solved within the 7200-second time limit, a near solution was obtained as described in Table 5.						

It is clear from the results presented in Table 4 that our proposed CRA is capable of significantly reducing the size of the ILP model for the GTCP and that its effectiveness improves as the size of the full model increases, e.g., in “Net12”, roughly 99.99% of the constraints are eliminated. This is achieved over a relatively short period of time even with our simple python approach.

Next, we compared the solution time and the results obtained from our algorithm to those achieved using the full model and the state-of-the-art *Augmented Greedy (AG) Algorithm* method developed by Sela Perelman et al. (2016) as reported in their paper.

For this end, we solve both the reduced and full model using CPLEX and a time limit of two hours. This limit was used because in a preliminary experiment, CPLEX used almost 64GB RAM from our machine and encountered memory errors when longer processing times were allowed.

In Table 5, we present the number of selected sensors and solution times for each of the three models. For the reduced and full models, the time also includes the model generation time. For example, in “Net12” where the allocated running time was not

sufficient for CPLEX to converge to the optimal solution, we report the obtained lower bound and the best identified solution.

It is also important to note that our comparison of the solution times of our method and of the AG algorithm presented by Sela Perelman et al. (2016) is not completely balanced because our testing machine was several times faster than that used in their study. However, the fact that the exact solutions obtained with our method require on average 10% fewer sensors is significant regardless of the solution time required.

Table 5: Comparisons of numbers of selected sensors and solution times required for the optimization model (GTCP) and Sela Perelman et al.'s (2016) method.

Network	Reduced model		Full model		AG (Sela Perelman et al., 2016)	
	Number of sensors	Solution time (*sec*)	Number of sensors	Solution time (*sec*)	Number of sensors	Solution time (*sec*)
Net1	45	0.04	45	1.12	**(1)**	5
Net2	86	0.15	86	1.36	98	35
Net3	115	0.27	115	3.22	134	99
Net4	122	0.93	122	8.12	138	296
Net5	151	0.56	151	7.98	164	231
Net6	229	1.04	229	13.5	254	379
Net7	224	0.95	224	14.1	237 **(2)**	559
Net8	166	4.86	166	50.8	195	1684
Net9	327	1.51	327	14.9	359	664
Net10	356	5.01	356	65.4	408	2369
Net11	635	7.13	635	64.3	717	3032
Net12	[3346, 3350]	8144	**(1)**	**(3)**	**(1)**	108000

(1) In these cases, Sela Perelman et al. (2016) reported on a set of sensors that allow for an only partial identification of identifiable faults. These are not comparable to our results, for which complete identification is applied as a hard constraint.
(2) In this case, Sela Perelman et al. (2016) originally reported on 139 sensors but through personal communication with the first author, we found this to be a typo.
(3) For the "Net12" instance, the full model could not be run with the amount of memory available (64 GB) and CPLEX reported errors.

The merits of our CRA are clear from Table 5. While CPLEX is capable of finding an optimal solution for fairly substantial instances of the GTCP, our CRA saves considerable processing time and memory resources and extends the range of instances for which optimal or near-optimal solutions can be obtained using reasonable computational resources.

5. Conclusions

This paper introduces the GTCP and formulates it as an ILP. While the formulation may present a very large number of constraints, we find many of them to be redundant. We introduce a constraint reduction algorithm that can render large models effectively solvable with a commercial solver.

An alternative solution method for the GTCP could involve generating violating constraints of the ILP and adding them to the model gradually in branch and cut procedure. This approach may be a promising one when the number of constraints is

exponential in the dimension of the master problem and when the separation problem can be solved in polynomial time. However, for our model, while the number of constraints is typically very large, it is still quadratic in the input (number of readings). Hence, only a very efficient, sub-quadratic separation algorithm can lead to an effective row generation algorithm. We suspect that no such separation algorithm exists.

We demonstrated the model and our solution method by applying it to a use case of sensor placement in water distribution networks. We show that it can deliver optimal or near-optimal solutions to large-scale instances in relatively short time. The previous literature on this problem presents heuristic methods that deliver solutions with significantly larger number of sensors and that take more time to solve the problem.

There are several possible interesting extensions to our model: First, since the optimal solution of the presented problem involves a set of sensors that can be used to detect all the identifiable states of the system, the optimization model may result in a set of sensors that is too large (or too expensive). For situations in which it is sufficient to detect the state of the system with a high probability, an interesting direction for future research would involve developing a model that considers the tradeoffs between the cost of errors and the cost of sensors. Such a model can also consider a situation where the sensor output is not reliable. That is when there is no deterministic mapping of the readings to states of the system. Finally, our GTCP is based on the assumption that the outputs produced by sensors are discrete values (categorical). A possible extension could consider cases in which some outputs are continuous values (e.g., temperature and pressure).

Acknowledgments

The first author of this paper was supported by a scholarship from the Shlomo Shmeltzer Institute for Smart Transportation at Tel Aviv University.

References

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.

Centre of Water Systems University of EXETER (2015). http://emps.exeter.ac.uk/engineering/research/cws/downloads/benchmarks/.

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.

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

Jolly, M. D., Lothes, A. D., Sebastian Bryson, L., & Ormsbee, L. (2014). Research database of water distribution system models. *Journal of Water Resources Planning and Management*, *140*(4), 410-416.

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*.

Krause, A., Leskovec, J., Guestrin, C., VanBriesen, J., & Faloutsos, C. (2008). Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, *134*(6), 516-526.

Lin, F. Y., & Chiu, P. L. (2005). A near-optimal sensor placement algorithm to achieve complete coverage-discrimination in sensor networks. *IEEE Communications Letters*, *9*(1), 43-45.

Ostfeld, A., Uber, J. G., Salomons, E., Berry, J. W., Hart, W. E., Phillips, C. A., ... & di Pierro, F. (2008). The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, *134*(6), 556-568.

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

Sela, L., & Amin, S. (2018). Robust sensor placement for pipeline monitoring: Mixed integer and greedy optimization. *Advanced Engineering Informatics*, *36*, 55-63.

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.