A NOVEL MINIMIZATION METHOD FOR SENSOR DEPLOYMENT VIA HEURISTIC 2-SAT SOLUTION

*Waleed Ahmed and **Ali Muhammad Ali Rushdi

Abstract- The tasks of guard placement or sensor deployment in an art gallery, a museum or in the corridors of public and security buildings pose the same problem, which requires placing the guards or sensors so as to cover a specified set of nodes with a minimum number of sensors or guards, thereby reducing the overall cost of the system as well assist power consumption. Generally, minimization can be done using optimization techniques such as linear programming, but in case of sensor deployment or guard placement there is a need either to place or not to place the sensor or guard, and hence only Boolean or binary values are used. Therefore, in order to optimize such a problem, we use the special case of linear integer programming known as Boolean integer linear programming (0-1 ILP). Other algorithms like Pseudo-Boolean SAT Solvers can also be used for the minimization purpose. In this paper, we introduce these minimization algorithms for the sensor deployment problem. We also contribute a greed-based heuristic, which utilizes the fact that the pertinent propositional formulas have variables of purely un-complemented literals. This heuristic has much less computational cost compared to those of 0-1 ILP and the Pseudo-Boolean SAT Solvers.

Index Terms- Boolean Satisfiability (SAT), Integer linear programming, Pseudo-Boolean SAT-Solvers, Sensor deployment

I. INTRODUCTION

Sensor networks are installed everywhere to monitor and measure real world phenomena. They play a significant role in our daily day life [1]. Sensors could be deployed in such a way that they can monitor the whole network region assigned to them. The method of deployment of sensors affects the monitoring area and the cost of the system. Low budget usually forces the system designer to install the minimum number of sensors, and to demand that sensors be deployed in such a way to cover a maximum area.

In this paper, we will discuss one of the applications of deploying sensors for monitoring purposes in some random network topology, with the aim that they cover the whole required region with the least number of sensors so as to reduce the installed cost of the system.

The type of coverage can be deterministic or probabilistic (stochastic) [2]. In deterministic coverage, there is some prior knowledge of where the sensors should be positioned. By contrast, in stochastic coverage, the sensors are deployed merely according to some known distribution and there is no prior knowledge of where the sensors should be exactly placed. Here, we will use the deterministic technique i.e., the places of the sensors are well defined from the outset. In order to achieve this purpose, we model our system by using propositional formulas and apply different optimizing technique such as binary integer linear programming, Pseudo-Boolean SAT-Solvers, and a greedy-based heuristic to achieve our objective.

The problem which we present is this paper is correlated with the problem of alarm and guard placement [3, 4]. In the alarm placement problem, alarms are placed at each node of the failure propagation graph, such that the identification of a failed node can be achieved uniquely and efficiently. Whenever a fault occurs it propagates and activates one of several alarms and then by using a diagnostic algorithm the node responsible for causing the fault can be found. In the same fashion, the communication paths are attached with some nodes. Sensors are installed to monitor these communication paths so that if any communication path fails, an alarm is issued and then by using a diagnostic algorithm the responsible communication path can be found. Likewise, in the art gallery problem, the guards are placed in such a way that the whole area can be monitored. Similarly, the sensors are deployed in secure buildings, libraries and museums in the corridors such that they can monitor the whole region.

The rest of this paper is organized as follows. Section 2 discusses Material and Methods, wherein three methods are presented namely, binary integer linear programming, Pseudo-Boolean SAT Solvers and a greedy-based heuristic. Section 3 demonstrates these methods via three scenarios or examples. Section 4 concludes the paper.

II. MATERIAL & METHODS

Today in the age of the modern era, sensors are installed everywhere from daily life systems to complex systems such as wireless networks [5], vehicular networks, biomedical systems and aircraft systems. Therefore, a lot

*Department of Electrical and Computer Engineering, King Abdulaziz University, Saudi Arabia, wahmad0006@stu.kau.edu.sa
**Department of Electrical and Computer Engineering, King Abdulaziz University, Saudi Arabia. arushdi@kau.edu.sa
of research has been carried out on the sensor placement problem in several applications such as in road transportation networks [6, 7], robotic navigation systems [8], effective coverage and surveillance of distributed networks [9, 10], and for diagnosis of faults [11]. The sensors should be deployed in such a way that the energy consumption of the system gets reduced [12, 13]. The problem of selecting the places where the sensors should be deployed is very important. Sensors should be placed somewhere such that the cost of the system should be minimized and the amount of energy or power consumption should be reduced. Reducing the cost of the system and reducing the power consumption of the system are two major constraints of the system. So here, we satisfy these constraints by minimizing the number of sensors to be deployed or installed.

In this paper, we are taking a random topology that is modeled by a graph consisting of paths and nodes. A path represents the region which should be sensed or monitored, and a node represents the place where a sensor should be deployed. Here, each path is attached to precisely two nodes so that each path can be monitored by installing a sensor at either node (or, maybe, both nodes). So, the path can be modeled by using a clause Ci which is a disjunction of the two literals depicting the two nodes attached to the path. Each node is represented by a literal X (i), where, Xi=1 represents the sensor presence while Xi=0 represents its absence.

In order to monitor all the paths, all the corresponding clauses should be satisfiable, which is represented by the propositional formula,

\[ f = C_1 \lor C_2 \ldots \lor C_n. \] (1)

To achieve our goal of reducing the cost of the system and power reduction, we have to minimize the number of sensor nodes. The lower the number of sensors, the lower will be the cost of the system and the lower will be the power consumed by the system.

A. Integer Linear Programming

The integer linear programming (ILP) is a well-known optimizing algorithm based on the simplex method to solve maximum/minimum constrained optimization problem, wherein the variables are just limited to integer values [14]. If the variables are further forced to be of Boolean values only then the algorithm is known as binary ILP or 0-1ILP. The propositional formulas can be transformed to ILP having a certain objective function and subject to some set of constraints. Here, the objective function is the number of sensor nodes and the constraints are that all the paths should be monitored.

The transformation of a propositional formula to a binary integer linear programming (0-1ILP) is achieved via a well-known technique [15], which can be explained by the following example. Let us consider the following clause:

\[ (X_1 \lor \bar{X}_2 \lor X_3), \] (2)

Where X_ (i) represents a Boolean variable and can be converted to an equivalent integer Y_ (i) in real arithmetic as follows:

\[ X_i = 1 - Y_i \]

So the above clause can be written as:

\[ Y_1 + 1 - Y_2 + Y_3 \geq 1, \text{ where } Y_1, Y_2, Y_3 \in \{0, 1\}. \] (3)

In the above equation the inequality symbol (\geq) is used because at least one literal should be asserted to satisfy the clause.

Coming to our problem of sensor deployment, the objective function is to minimize the number of sensors which can be encoded as follows:

\[ \text{Min}X_1 + X_2 + X_3 \ldots + X_n, \] (4)

While the constraint is that all the paths (clauses) should be monitored i.e., at least one literal should be asserted, which can be encoded as follows

\[ f = C_1 \land C_2 \land C_3 \land \ldots \land C_m \] (5)

Where,

\[ C_i = (X_i \lor \bar{X}_j), \text{ where } i \neq j \]

Subject to:

\[ C_1 \geq 1 \Rightarrow Y_b + Y_n \geq 1 \]

\[ C_2 \geq 1 \Rightarrow Y_a + Y_e \geq 1 \]

\[ C_m \geq 1 \Rightarrow Y_f \geq 1 \] (6)

Where \( Y_i \in \{0, 1\} \)

Solution of the binary integer linear programming gives us the minimum number of sensors required to be deployed to monitor the whole region, while the values of the X_i's give us the nodes where the sensors are placed to monitor the whole region.

B. Pseudo-Boolean SAT-Solvers

Modern SAT solvers [16-18] are based on the original DPLL algorithm, which is composed of branching, unit propagation (whose recursive form is known as Boolean constraint propagation) and backtrack searching. The algorithm performs a search process that traverses the space of 2n variable assignments until a satisfying assignment is found. The GRASP algorithm enhances the basic idea of DPLL, adds a conflicting clause to the formula, and introduces a new learning mechanism from conflicting assignments which helps to perform non-chronological backtracking by learning a clause from
conflicting assignments [19, 20]. Gomes [21] further enhances the efficiency of SAT solvers by implementing a restart strategy in search algorithms, which restarts the search space to the root level when a certain amount of conflict has been reached. The limit on the number of conflicts varies in different SAT-Solvers. Two of the most common restarting policies are taken from [22] and that using adaptive strategy [23]. The third main enhancement was the efficient implementation of Boolean Constraint Propagation (BCP) as it is one of the main features of DPLL. The Chaff solver implements two-literal watching, which efficiently reduces the overhead of the BCP [16]. Recent improvements have included methods of effective conflict-based adaptive branching such as Dynamic Largest Individual Sum (DLIS) [24] and Variable State Independent Decaying Sum (VSIDS) [25], which further improve the performance of modern SAT-Solvers.

We stress that SAT-Solvers deal with propositional problems, and their goal is to find the truth assignment that satisfies all given clauses or to report that no satisfying assignment exists otherwise. They cannot solve the minimization/maximization problem having constraints. So, the SAT-solvers need to be extended to solve an optimization problem. Some of the extended SAT-Solvers solvers, called pseudo-Boolean SAT solvers, are CPLEX [26], Z3OPT [27], SAT4J [28], and PBS [29]. The difference between these and the above integer linear programming algorithm is that this algorithm uses the simplex method, while these solvers use SAT-based integer linear programming which works well compared to the simplex-based optimization algorithm.

C. A Greedy-Based Heuristic

The above methods for binary integer linear programming and Psuedo-Boolean Solvers are computationally intractable as they seek exact algorithmic solutions of an NP-hard problem. In order to tackle such a problem, we use a greedy-based heuristic, which achieves a reasonable computational efficiency, at the expense of occasionally failing short of obtaining the exact minimal solution. It performs a local search for the node which has the largest number of paths attached to it and inserts a sensor there. It iteratively performs this until all the clauses are satisfied. The pseudo-code of this heuristic algorithm is given below.

Minimize sensors()
{
Calculate the number of paths (clauses) attached to each node (literal).
Install a sensor at the node attached to the maximum number of paths (clauses). If there is a tie, select one of the tied nodes arbitrarily.

After installing this sensor, delete all paths (clauses) which are attached to the node of that sensor, since these paths are now monitored and need no further coverage.

Go to step 1 and repeat the process until all paths (clauses) are exhausted (i.e., deleted).
}

The above three methods have been applied on 3 scenarios shown as below.

III. EXAMPLES

A. Scenario 1

Fig. 1. A network topology, consisting of nodes and paths, with red nodes representing the nodes where the sensors should be deployed.

Fig. 1. shows a randomly chosen mesh topology, in which there is a total 23 nodes represented as Xi and 34 communication paths or 34 corridors on the floor of a building to be monitored represented by Ci. An obvious upper bound on the number of sensors for the diagnosis of all communication paths or, to monitor all the corridors is 23 (obtained by placing a sensor at every node). This is clearly a bad choice in terms of cost and power. In order to reduce the number of sensors, we model the current situation as a Boolean logic 2-SAT problem, which is represented by the following formula. In the first step, search shows that the X5 has the maximum number of nodes attached to it. So the sensor is installed at node 5, and the paths attached to it are
deleted and the equation \( f \) is updated into equation \( f_1 \). The process continues until all paths are deleted.

\[
f = (X_1VX_7)(X_2VX_7)(X_2VX_8)(X_3VX_8)(X_4VX_14)(X_5VX_{10})(X_5VX_{14})(X_6VX_{10})(X_7VX_{11})(X_7VX_8)(X_7VX_{12})
\]

\[\] (8)

\[
f_1 = (X_1VX_7)(X_2VX_7)(X_2VX_8)(X_3VX_8)(X_5VX_{10})(X_6VX_{10})(X_7VX_{11})(X_7VX_{12})(X_7VX_{13})(X_8VX_{13})(X_8VX_{20})(X_10VX_{15})(X_11VX_{19})(X_12VX_{20})(X_13VX_{17})(X_13VX_{20})(X_16VX_{17})(X_17VX_{18})(X_17VX_{20})(X_17VX_{21})(X_17VX_{22})(X_19VX_{20})(X_20VX_{21})(X_20VX_{23})
\] (7)

\[
f_2 = (X_1VX_7)(X_2VX_7)(X_5VX_{10})(X_6VX_{10})(X_7VX_{11})(X_7VX_{12})(X_10VX_{15})(X_11VX_{19})(X_12VX_{20})(X_13VX_{17})(X_13VX_{20})(X_16VX_{17})(X_17VX_{18})(X_17VX_{20})(X_17VX_{21})(X_17VX_{22})(X_19VX_{20})(X_20VX_{21})(X_20VX_{23})
\] (7)

\[
f_3 = (X_1VX_7)(X_2VX_7)(X_5VX_{10})(X_6VX_{10})(X_7VX_{11})(X_7VX_{12})(X_10VX_{15})(X_11VX_{19})(X_12VX_{20})(X_13VX_{20})(X_19VX_{20})(X_20VX_{21})(X_20VX_{23})
\] (7)

\[
f_4 = (X_1VX_7)(X_2VX_7)(X_5VX_{10})(X_6VX_{10})(X_7VX_{11})(X_7VX_{12})(X_10VX_{15})(X_11VX_{19})
\] (7)

\[
f_5 = (X_5VX_{10})(X_6VX_{10})(X_10VX_{15})(X_11VX_{19})
\] (7)

\[
f_6 = (X_11VX_{19})
\] (7)

Here each path is modeled by clause \( Ci \). The conjunction of paths (clauses) modeled the whole topology which is represented by 2-SAT CNF formulas shown in equ. (7). For the monitoring of all these areas, all the branches should be satisfiable. In order to reduce the number of sensor nodes we use the above heuristic algorithm, which results in that only 7 sensors are needed to cover all these paths. These 7 sensors are encircled in the table above, and further are shown as red nodes in Fig. 1 and as red literals in equ. (8).

\[
f = (X_1VX_7)(X_2VX_7)(X_2VX_8)(X_3VX_8)(X_4VX_14)(X_5VX_{10})(X_5VX_{14})(X_6VX_{10})(X_7VX_{11})(X_7VX_8)(X_7VX_{12})(X_8VX_{12})(X_8VX_{13})(X_8VX_{20})(X_9VX_{14})(X_10VX_{14})(X_10VX_{15})(X_11VX_{19})(X_12VX_{20})(X_13VX_{17})(X_13VX_{20})(X_14VX_{16})(X_14VX_{17})(X_14VX_{18})(X_17VX_{20})(X_17VX_{21})(X_17VX_{22})(X_19VX_{20})(X_20VX_{21})(X_20VX_{23})
\] (8)

The above topology is also solved by binary integer linear programming, and by a Pseudo-Boolean Solver. Both give us the following solution \( X_7 = 1, X_8 = 1, X_{10} = 1, X_{14} = 1, X_{17} = 1, X_{19} = 1 \) and \( X_{20} = 1 \) which is exactly the same solution obtained above.

B. Scenario 2

The randomly chosen mesh topology of scenario 2 is shown in Fig. 2. There are 14 nodes and 23 paths. The topology is modeled by using 2-SAT CNF Boolean equation as shown in equ. (9).

\[
f = (X_1VX_5)(X_2VX_5)(X_3VX_5)(X_3VX_{10})(X_4VX_5)(X_4VX_8)(X_5VX_6)(X_5VX_{10})(X_5VX_{14})(X_7VX_8)(X_7VX_{11})(X_8VX_{12})(X_8VX_{13})(X_9VX_{10})(X_9VX_1)(X_{10VX_{13}})(X_{10VX_{14}})(X_{11VX_{12}})(X_{11VX_{14}})(X_{12VX_{13}})(X_{12VX_{14}})
\] (9)

\[
f_1 = (X_3VX_{10})(X_4VX_8)(X_6VX_{10})(X_7VX_{11})(X_8VX_{11})(X_8VX_{12})(X_9VX_{10})(X_9VX_{12})(X_9VX_{13})(X_{10VX_{13}})(X_{11VX_{12}})(X_{11VX_{14}})(X_{12VX_{13}})(X_{12VX_{14}})(9a)
\]

\[
f_2 = (X_3VX_{10})(X_4VX_8)(X_6VX_{10})(X_7VX_{11})(X_8VX_{11})(X_9VX_{10})(X_{10VX_{13}})(X_{11VX_{12}})(X_{11VX_{14}})(9b)
\]

\[
f_3 = (X_4VX_8)(X_7VX_{12})(X_7VX_{11})(X_8VX_{11})(X_9VX_{10})(X_{10VX_{13}})(X_{11VX_{12}})(9c)
\]

\[
f_4 = (X_7VX_{11})(X_9VX_{13})(X_{11VX_{12}})(9d)
\]

\[
f_5 = (X_9VX_{13})(9e)
\]

\[
\text{Fig. 2. Another network topology with red nodes representing the nodes where the sensors should be deployed.}
\]

Now by applying the method of minimizing technique, the number of sensors required to measure all the paths are calculated to be 6. These 6 nodes are represented by red color in equ. (10) and shown also red in Fig. 2.

\[
f = (X_1VX_5)(X_2VX_5)(X_3VX_5)(X_3VX_{10})(X_4VX_5)(X_4VX_8)(X_5VX_6)(X_5VX_{10})(X_5VX_{14})(X_7VX_8)(X_7VX_{11})(X_8VX_{12})(X_9VX_{10})(X_9VX_{12})(X_{10VX_{13}})(X_{11VX_{12}})(X_{11VX_{14}})(X_{12VX_{13}})(X_{12VX_{14}})
\] (10)
The above topology is also solved by binary integer linear programming, and by a Pseudo-Boolean Solver. Both give us the following solution X5=1, X8=1, X10, X11=1, X12=1 and X13=1, which is similar to the solution obtained above, but with node 13 replacing node 9.

C. Scenario 3

The randomly chosen mesh topology of scenario 3 is shown in Fig. 3. There are 16 nodes and 25 paths. The topology is modeled by using 2-SAT CNF Boolean equation as shown in equ. (11). We apply our optimization heuristic which finds the required sensor nodes as shown by equ. (12) and as identified as red in Fig. 3 a third network technology.

\[ f_2=(X_1X_2)(X_1X_6)(X_2X_3)(X_2X_7)(X_6X_11)(X_7X_11)(X_7X_13)(X_11X_12)(X_12X_13)(X_13X_14) \]  
\[ f_3=(X_1X_2)(X_1X_6)(X_2X_3)(X_6X_11)(X_11X_12)(X_12X_13)(X_13X_14) \]  
\[ f_4=(X_2X_3)(X_6X_11)(X_11X_12)(X_12X_13)(X_13X_14) \]  
\[ f_5=(X_2X_3)(X_12X_13)(X_13X_14) \]  
\[ f_6=(X_2X_3) \]  
\[ f=(X_1X_2)(X_1X_6)(X_2X_3)(X_2X_7)(X_2X_8)(X_3X_8)(X_4X_8)(X_4X_10)(X_5X_10)(X_6X_7)(X_6X_11)(X_7X_8)(X_7X_11)(X_7X_13)(X_8X_9)(X_8X_12)(X_8X_13)(X_8X_14)(X_9X_10)(X_{10}X_{14})(X_{10}X_{15})(X_{10}X_{16})(X_{11}X_{12})(X_{12}X_{13})(X_{13}X_{14}) \]  

Fig.3. A third network topology, again with red nodes representing the nodes where the sensors should be deployed.

The above topology is also solved by binary integer linear programming, and by a Pseudo-Boolean Solver. Both give us the following 6-node solution X2=1, X6=1, X8=1, X10=1, X11=1, X13=1 which is similar to the 7-node solution obtained above, but with the single node 6 replacing the two nodes 1 and 7. In this particular case, our heuristic fell short of obtaining the exact minimal. However, it did a relatively good job at a reasonably short time.

IV. CONCLUSION

Work on this paper is quite related to the celebrated Boolean Satisfiability problem (SAT), which is well known to be NP-complete (computationally intractable) in its generality, and in its special case k-SAT, k≥ 3, and polynomial-time solvable in some special cases including Horn, renamable-Horn, 2-SAT, unit and exclusive-OR. Our current problem involves both the 2-SAT and unate special cases (combined), which are not computationally intractable. However, our problem solves an unate 2-SAT problem repeatedly in pursuit of exact minimization. Therefore, it performs a search that is exponential in the number of nodes in the worst case, and hence it is still computationally intensive. In fact, our current problem of finding a satisfying assignment that minimizes the number of variables to be set to 1 is NP-complete even for the present satisfiable 2-SAT formulas. This problem is receiving attention under the name of main ones 2-Sat, or the more general name of minimum-cost 2-SAT. Therefore, the novel fast heuristic
presented herein is certainly a welcome useful addition to the practical contemporary problem of sensor deployment. As demonstrated by three arbitrarily-generated examples, this heuristic is far from being much inferior in performance to exact minimization algorithms. Detailed computational comparison of the automated version of this heuristic with the exact minimization algorithms is warranted. The comparison should involve the positive factor of speeding up of computation versus the negative one of the increase in the number of sensors deployed.

Another line of interesting and potentially fruitful research is to compare the present heuristic (which obtains a minimal or an almost minimal set of sensors to be deployed) which has the more ambitious goal of providing all solutions to the unate covering problem by obtaining all irredundant sets of such sensors. The present heuristic arises from the greedy concept of selecting the current node having maximum attached paths. The same heuristic, or other competing ones, could be systematically derived from Petrick Method by augmenting it with certain Branch and Bound concepts.

REFERENCES


