Generating Optimal Scheduling for Wireless Sensor Networks by Using Optimization Modulo Theories Solvers[*]

Gergely Kovásznai, Csaba Biró, and Balázs Erdélyi

Eszterházy Károly University, Eger, Hungary
IoT Research Institute

Abstract

Wireless Sensor Networks (WSNs) serve as the basis for today's Internet of Things applications. A WSN consists of a number of spatially distributed sensor nodes, which cooperatively monitor physical or environmental conditions. In order to ensure the dependability of WSN functionalities, several reliability and security requirements have to be fulfilled. By applying a Satisfiability Modulo Theories (SMT) formalization for WSNs, a sleep/wake-up scheduling that respects those requirements can be generated by using SMT solvers. It is also important to prolong the lifetime of a WSN as much as possible. The recent success of Optimization Modulo Theories (OMT) approaches makes us able to generate for a WSN a sleep/wake-up scheduling that is optimal in terms of energy efficiency. This makes this kind of WSN optimization an excellent application for OMT solvers. To the best of our knowledge, no scientific work has ever reported any usage of OMT solvers for WSN optimization. In this paper, we introduce an OMT formalization of the aforementioned optimization problem for WSNs, provide a single-hop WSN simulation environment with one of the most common wireless sensor node types, propose several OMT benchmarks extracted from the WSN simulation, and report experiments with three different OMT solvers: Z3, OptiMathSAT, and Symba.

1 Introduction

In today's Internet of Things (IoT) applications, Wireless Sensor Networks (WSNs) provide a rapid and flexible solution for accessing information in many real-world applications. A WSN deploys a large number of small, inexpensive, self-powered devices that can sense their environment and gather local information used to make global decisions. Although research on WSNs was originally motivated by military applications, other applications have also become feasible as the corresponding technology is getting more and more accessible by the public. Such applications [5] are, for instance, (1) *infrastructure security* to provide early detection of any potential threat in critical facilities such as power plants, oil refineries, airports, etc., (2) *environmental and habitat monitoring* to study the response of vegetation and fauna to climatic changes, (3) *traffic monitoring* to control traffic lights and to plan alternatives routes in order to avoid traffic jams and accidents.

In order to ensure the dependability of WSN functionalities, several reliability and security requirements have to be fulfilled. The most researched requirement of that kind is the *coverage problem* which addresses the question of how well the sensors observe the physical environment. Two main types of coverage are distinguished in literature: (1) *area coverage* to cover a given area of interest; (2) *point coverage* to cover a set of target points. Point coverage is sometimes even used to simulate area coverage by using points that approximate an area.

In many cases, especially in military applications or in critical systems, a WSN is deployed in a hostile environment, which makes it necessary for a WSN to fulfill various additional security

[*]Supported by EFOP 3.6.1.

requirements, besides coverage, in order to increase the dependability of WSN functionalities. For instance, WSN deployment may require a sensor node not to monitor the same target point for a long time, otherwise the sensor node is easier to damage, detect or attack. Examples for such additional requirements known from literature are the *partial coverage* [12], the *evasive constraint* [3], and the *moving target constraint* [9].

Since sensor devices are self-powered and, therefore, have limited power supply, it is crucial to apply energy efficient protocols to the sensor nodes by synchronizing their operations. To save energy, a sensor node might eventually enter the sleep mode, in which its power consumption is typically the fraction of that in the active mode. However, coverage (and other security constraints) should be maintained during the entire lifetime of the WSN, therefore we typically want to generate a *sleep/wake-up scheduling* which does not violate any of those constraints at any time and provides a maximal lifetime for the WSN.

Most of the previous works on *lifetime maximization* for WSNs model the problem as an optimization problem and solve it using some heuristic algorithms [16, 6, 5]. They usually scale up to a few hundreds of sensor nodes [16] and a few tens of target points [6], which sometimes comes with the price of losing 100% precise coverage. Most importantly, most of the existing works focus on the coverage problem without giving any special attention to dependability or any of the aforementioned additional security constraints [9]. It seems even infeasible to incorporate those constraints into the existing algorithms.

Satisfiability Modulo Theories (SMT) is a powerful tool to solve constraint satisfaction problems in diverse application areas including software and hardware verification, planning, scheduling, etc. A few previous works apply SMT formalization to the aforementioned WSN constraints and use SMT solvers to generate an appropriate sleep/wake-up scheduling for a WSN. [11] focuses only on the coverage problem and reports experiments with the SMT solver Z3. [9] addresses not only the coverage problem, but also the partial coverage, the evasive constraint, and the moving target constraint. For the experiments in [9], the SMT solver Yices is utilized, and the results show that the approach can scale up to hundreds (or, depending on the sensing range, even up to a few thousands) of sensor nodes. Note that both [11, 9] are only interested in generating a sleep/wake-up scheduling that fulfills the aforementioned constraints, but none of them addresses the maximization problem of the WSN lifetime, which makes them not as practical.

The recent success of *Optimization Modulo Theories (OMT)* approaches makes us able to generate a sleep/wake-up scheduling that provides maximal lifetime for the WSN, while keeping all the flexibility and strength of the SMT-based approaches, namely that different dependability and security constraints can be combined on demand. This makes this kind of WSN optimization an excellent application for OMT solvers. To the best of our knowledge, no scientific work has ever reported any usage of OMT solvers for WSN optimization.

In this paper, we introduce an OMT formalization of the aforementioned lifetime optimization problem for WSNs. We address the coverage as well as the evasive and the moving target constraints. Furthermore, our model allows the sensor nodes to be *heterogeneous*, i.e., they can have different sensing ranges. Note that the SMT-based approaches in [11, 9] deal only with homogeneous sensor nodes. We perform experiments with publicly available OMT solvers such as OptiMathSAT [14], Z3 [4], and Symba [13]. The results of our experiments can hopefully help the SMT community to see what performance today's OMT solvers can provide for WSN optimization, how scalable they are, and how that performance varies when additional constraints, besides coverage, are involved in the optimization process as well. On the top of that, we contribute to the SMT community with new and practical OMT benchmarks, which we make publicly available.

After defining some preliminaries in Sect. 2, we provide a possible way in Sect. 3 to formalize the aforementioned constraints as well as the optimization objective. We also give some details in Sect. 3.1 about how to do the same in the SMT-LIB format. In Sect. 4, we report on experiments with simulated WSNs and existing OMT solvers. We conclude and propose directions for future work in Sect. 5.

2 Preliminaries

Satisfiability Modulo Theories (SMT) is the decision problem of checking satisfiability of logical formulas with respect to some background theory. The most common examples for theories are the integer numbers, the real numbers, the fixed-size bit-vectors, and the arrays. The logics that one could use might differ from each other in the linearity or non-linearity of arithmetic, the presence or absence of quantifiers, or in the presence or absence of uninterpreted functions. In this paper, we are using the quantifier-free logic of linear integer arithmetic with uninterpreted functions (QF_UFLIA). The SMT-LIB format [2], as the common input format for SMT solvers, defines the syntax for QF_UFLIA formulas, where the most important features are as follows:

- No quantifiers $\forall$ and $\exists$ are allowed to be used.
- Every expression must be of type integer or Boolean.
- Only the arithmetic operations addition, subtraction, multiplication, division, and comparison are to use.
- For the sake of linear arithmetic, expressions with multiplication are allowed to be used only in the format $c * t$ where c is a constant.
- It is allowed to use uninterpreted function symbols, i.e., to specify only the signature for such a function symbol.

A QF_UFLIA formula ϕ is satisfiable if there is an assignment of appropriate values to its variables and uninterpreted functions under which ϕ evaluates to true. Most of the SMT solvers support QF_UFLIA, such as CVC [1], MathSAT [7], Z3 [8], and Yices [10].

Many problems of interest, which can be encoded as SMT problems, may require also to find models that are optimal with respect to some objective function [15]. Optimization Modulo Theories (OMT) is an extension of SMT with objective functions to maximize or minimize. An OMT problem, therefore, contains not only a formula to satisfy, but also an expression $\max : obj$ or $\min : obj$, where obj denotes the objective function. In this paper, obj can be an arbitrary QF_UFLIA integer expression. When checking the satisfiability of a QF_UFLIA formula, we are looking for a satisfying assignment under which the value of obj is maximal or minimal, respectively. There exist only a few SMT solvers that provide OMT solving, to the best of our knowledge: OptiMathSAT [14], Z3 [4], and Symba [13]. The syntax for the optimization expression is not part of the SMT-LIB format and, therefore, is specified differently for the different OMT solvers.

3 Optimal Scheduling for WSNs by OMT Solvers

The main objective is to prolong the lifetime of a WSN as much as possible, which basically means to maximize the energy efficiency or, in other words, to minimize the overall energy consumption. Each sensor node has a predefined sensing range which can be heterogeneous in our model. Given the number $n \geq 1$ of the sensor nodes, let r_i denote the range of the i^{th} sensor node. The greater the range is, the less the lifetime of the sensor node is, denoted by L_i.

In this paper, we are focusing on point coverage, where the objective for the sensor nodes is to cover a set of $m \geq 1$ points of interest. The physical location of sensor nodes and points can be neglected in our model, since it is sufficient to know the distance, taken pairwise, between each sensor node and point, denoted by $d_{i,j}$ for the i^{th} sensor node and the j^{th} point.

In order to save energy, and, analogously, to maximize the lifetime of the WSN, the sensor nodes might eventually enter the sleep mode and wake up later on. Let $w_{i,t}$ be a Boolean variable that denotes if the i^{th} sensor node is awake at the t^{th} time interval.

While maximizing the lifetime T of the WSN, it is crucial not to violate certain requirements. Most importantly, for each sensor node, the number of time intervals at which the node is awake must not exceed the node's lifetime. This constraint is called the *lifetime constraint* and can be formalized as follows:

$$\forall i \ (1 \leq i \leq n). \ \sum_{t=1}^{T} w_{i,t} \leq L_i$$

Another crucial requirement is the coverage constraint [5] that requires every point to be covered by at least one sensor node at any time. In some applications, the coverage constraint is generalized by requiring coverage by at least K sensor nodes where $K \geq 1$ is a predefined constant [9, 11]. The *K-coverage constraint* can be formalized as follows:

$$\forall j, t \ (1 \leq j \leq m, \ 1 \leq t \leq T). \ \sum_{i \in S_j} w_{i,t} \geq K$$

where $S_j = \{i \mid d_{i,j} \leq r_i\}$ is the set of sensor nodes which are able to cover the j^{th} point. Fig. 1 illustrates how sleep/wake-up scheduling works with 2-coverage. The blue dots represent the active sensor nodes, the red dots the ones in sleep mode, and the green dots the target points. Blue circles show the sensing ranges of the active sensors. Note that in each time interval, each target point is monitored by (at least) 2 sensor nodes.

In some applications, sensors are deployed in a hostile environment or in critical systems [9], thus it might be important to protect the sensors from being active for too long. In such applications, the so-called *evasive constraint* should be respected as well, which is about prohibiting the sensor nodes to stay active for more than E consecutive time intervals where $E \geq 1$:

$$\forall i, t \ (1 \leq i \leq n, \ 1 \leq t \leq T - E). \ \sum_{t'=t}^{t+E} w_{i,t'} \leq E$$

Fig. 1 illustrates the case for $E = 2$. Note that sensor node 6 (in the middle) switches into sleep mode after being active for 2 consecutive time intervals.

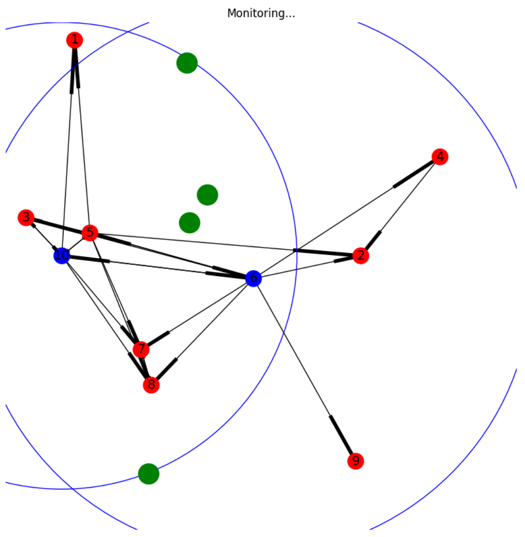
To improve resiliency and security, some critical points may require not to be covered by the same sensor for more than M consecutive time intervals where $M \geq 1$ [9]. This constraint is called the *moving target constraint* and can be formalized as follows:

$$\forall j \in CR, \ \forall i \in S_j, \ \forall t \ (1 \leq t \leq T - M). \ \sum_{t'=t}^{t+M} w_{i,t'} \leq M$$

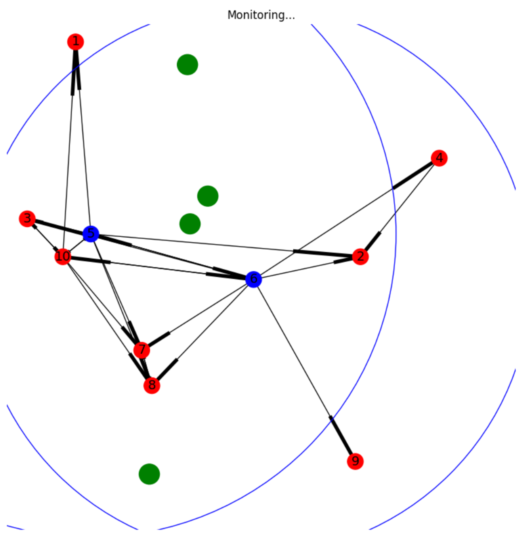
where $CR \subseteq \{j \mid 1 \leq j \leq m\}$ is the set of critical points.

As expected, the *objective function* to maximize is the lifetime of the WSN:

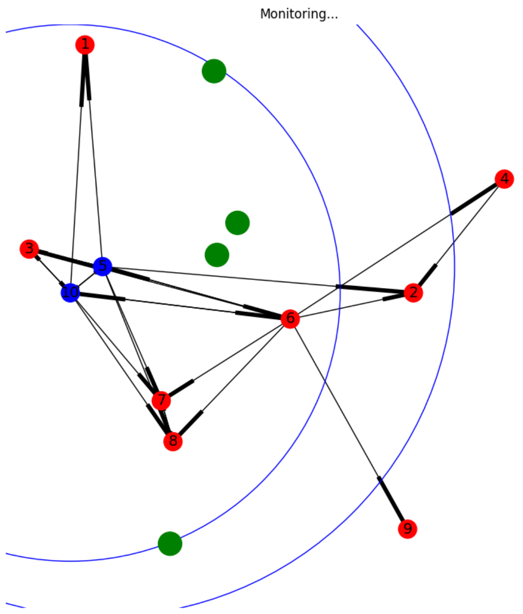
$$\max : T$$



(a) $t = 0$



(b) $t = 1$



(c) $t = 2$

Figure 1: Sleep/wake-up scheduling of sensor nodes for 2-coverage and evasive constraint with $E = 2$. The active nodes (blue dots) are monitoring the target points (green dots).

3.1 SMT-LIB Formalization of the Constraints

When formalizing the lifetime, the coverage, the evasive, and the moving target constraints in the SMT-LIB format [2], one has to reasonably decide which theory and logic to apply, taking the available solvers into consideration as well as the feasibility and the expected runtime of the solving process.

For representing the lifetime of sensor nodes and the time intervals, the best choice is obviously to apply the theory of integer numbers. Similarly, it is sufficient to represent the sensor ranges and the distances between sensors nodes and points as integer numbers. Note that all the aforementioned constraints of interest apply linear arithmetic. Avoiding the use of non-linear arithmetic makes the solving process more feasible, so does the avoiding of quantifiers. All the variables whose value depends on T are represented as uninterpreted functions [1]. Consequently, we have decided to apply the logic QF_UFLIA.

Since the maximal lifetime of the WSN is bounded, one can unroll the constraints in order to avoid the use of quantifiers. Due to the coverage constraint, the sum of the lifetimes of sensor nodes can be used as a time horizon for the WSN: $\widehat{T} = \sum_{i=1}^{n} L_i$

The K-*coverage* constraint can be formalized in SMT-LIB as follows, for all target points j and time intervals t:

```
(>=
  (+  (boolToInt (covers0jAt t)) (boolToInt (covers1jAt t))
  ... (boolToInt (coversnjAt t)) )
  K )
```

where the function (conversijAt t) tells if the sensor node i covers the target point j at the time interval t.

Let us show how to formalize the *evasive constraint* with parameter E. The following assertion should be added for all sensor nodes i and time intervals $t \leq \widehat{T} - E$:

```
(<=
  (+  (boolToInt (w i t)) (boolToInt (w i t+1))
  ... (boolToInt (w i t+E)) )
  E )
```

The moving target constraint can be formalized in a similar manner.

Different OMT solvers require the *optimization objective* to be formalized in different ways. For Z3, the formalization is extremely simple:

```
(maximize T)
```

We experienced that Z3 sometimes returned invalid models for our benchmarks if the translation to Pseudo-Boolean constraints [4] was switched on. We did not investigate the causes and decided to switch it off, as follows:

```
(set-option :opt.elim_01 false)
```

For OptiMathSAT, a lower bound and an upper bound must be specified, too, as follows:

```
(maximize T :local-lb 0 :local-ub T̂)
```

Symba requires a completely different formalization. The objective should be written as an implication with all the constraints of interest on the left-hand side and the following inequality on the right-hand side:

[1] Uninterpreted functions can be eliminated by introducing Ackermann constraints, which comes with the price of quadratic growth in problem size.

```
(=>  $constraints (<= T TOpt)  )
```

where `TOpt`, similar to `T`, is a variable declared by us.

4 Implementation, Benchmarks and Experiments

In this section, we describe the implementation details used to simulate WSNs and to generate benchmarks with different sets of constraints with different parameter values. Furthermore, we report on experiments with three available OMT solvers: OptiMathSAT, Z3, and Symba. Since the strength of our OMT-based approach is that any constraints can be combined with the WSN optimization problem, it makes not much sense to involve in our experiments classical WSN optimization approaches [16, 6, 5] that can only deal with the coverage constraint.

The simulations and experiments were run on 3.60 GHz 8-core CPU with 8 GB memory. The wall clock time limit was set to 600 seconds and the memory limit to 3 GB. The benchmarks and log files are available at `https://iot.uni-eszterhazy.hu/en/research/tools`.

Our simulator and evaluator was implemented in Python language (660 lines of code). The main goal of the experiments is to examine the impact of the different dependability and security constraints to the WSN lifetime maximization problem, as well as the impact of critical parameters such as the number of sensor nodes, the number of target points, the parameters for the coverage, for the evasion and for the moving target constraints, respectively.

4.1 Simulation

For simulating WSNs, we chose an IEEE 802.15.4 compatible sensor node that is able to communicate wirelessly and has common parameters such as a $3V$ power supply [2]. Such sensor nodes are provided with an RF transceiver with an estimated range of $100m$. To accurately define the different ranges for different performance levels, we chose a commonly used RF transceiver, the CC2420 [3]. Although, depending on the operation system of the sensor node, 32 to 256 different performance levels can be defined, the manufacturer of CC2420 publish data only about 8 performance levels. For sensor node simulation, we need to know the power consumption and the sensing range for each performance level. In Tab. 1, data about those performance levels are given, where the estimated ranges can be calculated as follows.

The output power $P(d)$ is directly proportional with the square of the range d:

$$P(d) = pd^2$$

where p is the power density. We can calculate the value of p from the maximum range $d_{max} = 120m$ and the maximum output power $P_{max} = 1mW$ as follows:

$$p = \frac{P_{max}}{d_{max}^2} = \frac{1}{120^2} \approx 6.94 \cdot 10^{-5} mW/m^2$$

From this and the minimum output power $P_{min} = 3.16\mu W$, one can easily calculate the minimum range $d_{min} = 6.75m$.

The estimated range d for the power consumption I can be calculated as follows:

$$d = \frac{d_{max} - d_{min}}{P_{max} - P_{min}}(I - I_{min}) + d_{min}$$

[2] A good example for such a commonly used sensor node is the MICAz mote, cf. `http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf`.

[3] `http://www.ti.com/lit/ds/symlink/cc2420.pdf`

PA_LEVEL	Output Power (dBm)	Output Power (mW)	Power Consumption (mA)	Estimated range (m)
31	0	1.000	17.4	120
27	-1	0.794	16.5	109
23	-3	0.501	15.2	92
19	-5	0.316	13.9	75
15	-7	0.200	12.5	58
11	-10	0.100	11.2	41
7	-15	0.032	9.9	25
3	-25	0.003	8.5	6.75

Table 1: Estimated ranges for the different performance levels of a sensor node equipped with a CC2420 RF transceiver.

In our simulations, the performance level for each sensor node was set randomly in advance, and then the corresponding sensing range was used throughout the simulation, as Fig. 1 illustrates.

4.2 Benchmarks

In the WSN simulation, we generated a network grid of 600 meters by 600 meters. The physical locations of the sensor nodes and the target points were set randomly, as well as the performance levels of the sensor nodes. We generated two benchmark sets from the simulator:

- Harder benchmarks with 10 sensor nodes, 4 target points, 2-coverage constraint, evasive constraint with $E = 3$, and moving target constraint with $M = 2$.
- Easier benchmarks with 10 sensor nodes, 2 target points, 1-coverage constraint, evasive constraint with $E = 2$, and moving target constraint with $M = 1$.

Within each benchmark set, we generated (1) 20 benchmarks with all the constraints enabled, (2) 20 benchmarks with only the moving target constraint disabled, and (3) 20 benchmarks with only the evasive constraint disabled. Since the SMT-LIB formalization of the corresponding optimization problem slightly differs for the different OMT solvers, we generated three variants of each benchmark instance: one for OptiMathSAT, one for Z3, and one for Symba.

4.3 Evaluation Results

Tab. 2 provides an overview of the evaluation results for the harder benchmarks. The total number of solved SAT/UNSAT instances is shown, as well as the number of timeouts ($\#TO$), the highest maximal lifetime (*Opt*) that was found for the given benchmark, the average runtime, the average memory consumption, and the number of crashes (where relevant). Although *Opt* is not a representative value, we use it to illustrate the magnitude of the WSN optimization problem that the given solver is able to cope with.

The results clearly show that OptiMathSAT outperforms the other two solver on the harder benchmarks, although it crashes on one instance with a segmentation fault. It is very interesting that OptiMathSAT does not seem to be sensitive to what type of constraints are involved in the WSN optimization problem, regarding the number of solved instances. On the other hand, runtime and memory consumption decrease if one of the constraints is disabled, especially in the case of the evasive constraint.

	Solver	*#SAT/UNSAT*	*#TO*	*Opt*	*Time*	*Space*	*#Crash*
All constraints on	OptiMathSAT	11/5	4	73	245.5	440.3	
	Z3	2/5	13	72	393.6	449.8	
	Symba	1/5	14	2	423.1	461.9	
Moving target off	OptiMathSAT	10/5	5	74	215.8	314.8	
	Z3	7/5	8	73	258.2	408.3	
	Symba	4/5	11	60	393.5	439.1	
Evasive off	OptiMathSAT	12/4	3	74	149.0	286.5	1
	Z3	7/5	8	72	265.2	468.5	
	Symba	5/5	10	60	355.6	475.0	

Table 2: Results for QF_UFLIA benchmarks with 10 sensor nodes, 4 target points, 2-coverage, evasive constraint with $E = 3$, and moving target constraint with $M = 2$.

When all the constraint are enabled, Z3 and Symba can solve significantly fewer instances than OptiMathSAT does. When one of the constraints is disabled, the performance of Z3 and Symba gets slightly better, although their runtime and memory consumption do not really change. Let us note that we experienced insignificant runtimes on all the UNSAT instances by all the solvers, this is why they can solve all the UNSAT instances, except for the case of crash.

Tab. 3 shows the evaluation results for the easier benchmarks, which consist of SAT instances only. Here, Z3 is clearly the winner since it can solve all the instances at very moderate runtime. Note that the greatest maximal lifetime found by the solvers is 226, therefore we find Z3's performance on those benchmarks quite remarkable.

	Solver	*#SAT/UNSAT*	*#TO*	*Opt*	*Time*	*Space*
All constraints on	Z3	20/0	0	226	63.1	493.1
	OptiMathSAT	12/0	8	159	277.8	520.7
	Symba	9/0	11	159	484.9	436.2
Moving target off	Z3	20/0	0	226	49.2	333.7
	OptiMathSAT	11/0	9	130	311.3	476.1
	Symba	9/0	11	159	488.8	340.0
Evasive off	Z3	20/0	0	226	50.8	300.0
	Symba	19/0	1	226	324.8	334.1
	OptiMathSAT	12/0	8	159	344.1	488.2

Table 3: Results for QF_UFLIA benchmarks with 10 sensor nodes, 2 target points, 1-coverage, evasive constraint with $E = 2$, and moving target constraint with $M = 1$.

Interestingly, OptiMathSAT does not provide better results than the ones on the harder benchmarks. On the contrarily, average runtime and memory consumption have even increased. Surprisingly, Symba remarkably accelerates with the evasive constraint disabled. Symba not only outperforms OptiMathSAT, but it almost reaches the performance of Z3 in terms of solved instances, although Symba's average runtime is still much greater than the one of Z3.

It is also an interesting question if the theory of integer numbers is the best choice. Therefore, we decided to rerun the experiments for the easier benchmarks over real numbers (QF_UFLRA).

The results in Tab. 4 show that Z3 and OptiMathSAT provide almost the same performance as before, while Symba accelerates drastically and can solve almost all the instances.

	Solver	*#SAT/UNSAT*	*#TO*	*Opt*	*Time*	*Space*
All constraints on	Z3	19/0	1	226	87.7	497.1
	Symba	18/0	2	226	223.4	578.5
	OptiMathSAT	12/0	8	159	277.0	507.4
Moving target off	Z3	20/0	0	226	41.4	318.9
	Symba	20/0	0	226	172.5	393.0
	OptiMathSAT	11/0	9	130	310.6	469.9
Evasive off	Z3	20/0	0	226	36.2	284.4
	Symba	20/0	0	226	128.9	340.5
	OptiMathSAT	11/0	9	159	345.2	467.5

Table 4: Results for QF_UFLRA benchmarks with 10 sensor nodes, 2 target points, 1-coverage, evasive constraint with $E = 2$, and moving target constraint with $M = 1$.

To summarize, OptiMathSAT provides the most stable performance and scales the best for the presence/absence of different constraints, and for different parameter settings. Using Z3 is advantageous on instances with fewer target points and lower parameter values, for which it provides amazing runtimes. Although Symba scales the worst over integer numbers, it provides convincing performance over real numbers.

5 Conclusion

In this paper, we investigated a WSN optimization problem: how to maximize the lifetime of the WSN while not violating certain dependability and security constraints. In the paper, three such constraints are addressed: the K-coverage constraint, the evasive contraint, and the moving target constraint. Most of the existing approaches focus on the maximization problem only in combination with K-coverage, while others focus on using multiple constraints, but not addressing the maximization problem at all. By using OMT formalism, it is possible to combine all the aforementioned constraints as well as to specify an optimization objective. Furthermore, an OMT-based approach can be considered extremely flexible and extendable with other constraints on demand.

The recent success of OMT solvers, such as OptiMathSAT, Z3, and Symba, inspired us to do experiments with all of them and investigate how they scale for the size of the WSN, the presence/absence of different constraints, and for different parameter settings. We made the resulting OMT benchmarks and log files publicly available. Although the results show that today's OMT solvers can only deal with a few tens of sensor nodes and target points, OptiMathSAT seems scalable enough for further experiments with larger WSNs, which is part of future work. Since OptiMathSAT and Z3 support the combination of multiple objectives, we are planning to address the minimization of overall energy consumption while maximizing the WSN lifetime.

We are also planning to investigate the possibility to invent an OMT solving approach which is specialized for lifetime maximization in some sense, in order to achieve significant speedup in optimization tasks for WSN applications.

References

[1] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Proc. Int. Conf. on Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 171–177. Springer, 2011.

[2] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.5. Technical report, Department of Computer Science, The University of Iowa, 2015. Available at `www.smt-lib.org`.

[3] Zinaida Benenson, Felix C. Freiling, and Peter M. Cholewinski. Advanced evasive data storage in sensor networks. In *Proc. Int. Conf. on Mobile Data Management*, MDM'07, pages 146–151. IEEE Computer Society, 2007.

[4] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. μZ - an optimizing SMT solver. In *Proc. Int. Conf. for Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 9206 of *LNCS*, pages 194–199. Springer, 2015.

[5] Mihaela Cardei. Coverage problems in sensor networks. In *Handbook of Combinatorial Optimization*, pages 899–927. Springer, 2013.

[6] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005.

[7] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In Nir Piterman and Scott Smolka, editors, *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 7795 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2013.

[8] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proc. Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, TACAS'08/ETAPS'08, pages 337–340. Springer-Verlag, 2008.

[9] Qi Duan, Saeed Al-Haj, and Ehab Al-Shaer. Provable configuration planning for wireless sensor networks. In *Proc. 8th Int. Conf. on Network and Service Management (CNSM) and Workshop on Systems Virtualization Management (SVM)*, pages 316–321, 2012.

[10] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Proc. Int. Conf. on Computer-Aided Verification (CAV)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, 2014.

[11] Weiqiang Kong, Ming Li, Long Han, and Akira Fukuda. An SMT-based accurate algorithm for the K-coverage problem in sensor network. In *Proc. 8th Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, pages 240–245, 2014.

[12] Y. Li, C. Vu, C. Ai, G. Chen, and Y. Zhao. Transforming complete coverage algorithms to partial coverage algorithms for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):695–703, 2011.

[13] Yi Li, Aws Albarghouthi, Zachary Kincaid, Arie Gurfinkel, and Marsha Chechik. Symbolic optimization with SMT solvers. In *Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 607–618. ACM, 2014.

[14] Roberto Sebastiani and Patrick Trentin. OptiMathSAT: A tool for optimization modulo theories. In *Proc. Int. Conf. on Computer-Aided Verification (CAV)*, volume 9206 of *LNCS*, pages 447–454. Springer, 2015.

[15] Roberto Sebastiani and Patrick Trentin. Pushing the envelope of optimization modulo theories with linear-arithmetic cost functions. In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 9035 of *LNCS*, pages 335–349. Springer, 2015.

[16] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proc. Int. Workshop on Wireless Sensor Networks and Applications*, WSNA'02,

pages 32–41. ACM, 2002.