Portfolio SAT and SMT Solving of Cardinality Constraints in Sensor Network Optimization

Gergely Kovásznai and Krisztián Gajdár
Eszterházy Károly University, Eger, Hungary

Laura Kovács
TU Wien, Austria and Chalmers University of Technology, Sweden

NOTE:

This preprint has been accepted as a peer-reviewed publication at the SYNASC 2019 conference.

The camera-ready version of this preprint is available at:

https://doi.org/10.1109/SYNASC49474.2019.00021

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Portfolio SAT and SMT Solving of Cardinality Constraints in Sensor Network Optimization

Gergely Kovásznai*, Krisztián Gajdár*, Laura Kovács†
*Eszterházy Károly University, Eger, Hungary
† TU Wien, Austria and Chalmers University of Technology, Sweden

***Abstract*—Wireless Sensor Networks (WSNs) serve as the basis for 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. In previous work, we applied OMT (Optimization Modulo Theories) solvers to maximize a WSN's lifetime, i.e., to generate an optimal sleep/wake-up scheduling for the sensor nodes. We discovered that the bottleneck for the underlying SMT (Satisfiability Modulo Theories) solvers was typically to solve satisfiable instances. In this paper, we encode the WSN verification problem as a set of Boolean cardinality constraints, therefore SAT solvers can also be applied as underlying solvers. We have experimented with different SAT solvers and also with different SAT encodings of Boolean cardinality constraints. Based on our experiments, the SAT-based approach is very powerful on satisfiable instances, but quite poor on unsatisfiable ones. In this paper, we apply both SAT and SMT solvers in a portfolio setting. Based on our experiments, the MiniCARD+Z3 setting can be considered to be the most powerful one, which outperforms OMT solvers by 1-2 orders of magnitude.**

I. Introduction

Considering *Wireless Sensor Network (WSN)* problems, sensor devices are self-powered and, therefore, have limited power supply. It is thus 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 a fraction of that in the active mode. However, coverage (and other security constraints) should be maintained during the entire lifetime of the WSN. It is therefore required to generate *sleep/wake-up scheduling* which does not violate any of those constraints at any time and provides a *maximal lifetime* for the WSN.

In literature, there exist results on modeling WSN problems as *SAT instances*. For instance, [1] proposes a SAT encoding of a WSN's communication model as a directed graph, which is then checked against strong connectivity in [2]. The papers [3], [4] apply a *Satisfiability Modulo Theories (SMT)* formalization to the coverage problem and to other WSN constraints, and then they use SMT solvers to generate an appropriate sleep/wake-up scheduling for a WSN. Yet, these works do not address the maximization problem of the WSN's lifetime, hindering the practical use of these techniques.

In our previous work from [5], we generate *Optimization Modulo Theories (OMT)* benchmarks from WSNs and apply OMT solvers to obtain a sleep/wake-up scheduling for a WSN that maximizes its lifetime. A follow-up paper [6] investigates further aspects of WSNs that make the OMT solving process even more challenging, and the paper reports on experiments with the OMT solvers OptiMathSAT [7], Z3 [8], and Symba [9]. In [10], we introduce our OMT solver called Puli and report on further experiments, showing that Puli significantly outperforms the aforementioned OMT solvers on WSN benchmarks. Puli uses the SMT solver Z3 [11] as underlying solver and calls Z3 iteratively. Within Puli, Z3 shows an interesting behavior: it is fast in solving unsatisfiable instances, but provides poor performance in solving satisfiable ones.

The aim of this paper is to improve our previous results on SMT-based optimization of WSNs. To do so, we apply *SAT solvers parallel to SMT solvers, in a portfolio setting* and hence improve our previous results [10] by a SAT/SMT-based portfolio approach for WSN optimization. Section II introduces basic concepts and definitions on SAT, SMT and OMT solving, as well as the WSN model we apply, including some reliability and security constraints over WSNs. Section III shows how to encode those constraints into SAT and SMT, by using Boolean cardinality constraints as building blocks. In Section IV, we propose a novel approach for WSN optimization by applying SAT and SMT solvers in a portfolio setting. We explain certain search strategies and give details on our implementation. Section V reports on experiments with our portfolio-based WSN optimization approach, with various combinations of different SAT and SMT solvers, as well as different encodings of Boolean cardinality constraints. Based on those experiments, MiniCARD + Z3 provide the best performance together, outperforming OMT solvers in terms of solved instances and also of runtime by 1-2 orders of magnitude. Regarding cardinality encodings, sequential counters outperform the other encodings involved in our experiments, especially in combination with the SAT solver MiniSat.

This research was supported by the grant EFOP-3.6.1-16-2016-00001 "Complex improvement of research capacities and services at Eszterhazy Karoly University".

II. Preliminaries

A *literal* l is a Boolean variable x or its negation $\overline{x}$. A *clause* C is a disjunction of literals. A Boolean formula is in *Conjunctive Normal Form (CNF)*, if it is a conjunction of clauses. We say that a Boolean formula, typically in CNF, is *satisfiable*, if there exists a truth assignment to the Boolean variables of the formula such that the formula evaluates to true. Otherwise, it is said to be *unsatisfiable (UNSAT)*. The *Boolean Satisfiability (SAT)* problem is the problem of determining if a Boolean formula is satisfiable.

Satisfiability Modulo Theories (SMT) is the decision problem of checking satisfiability of a Boolean formula with respect to some background theory. *Optimization Modulo Theories (OMT)* is an extension of SMT which allows finding models that optimize given objective functions. Common theories include the theory of integers, reals, fixed-size bitvectors, etc. The logics that one could use might differ from each other in the linearity or non-linearity of arithmetic and the presence or absence of quantifiers. In this paper, we use the theory of integers combined with linear arithmetic and without quantifiers – denoted as QF_LIA in the SMT-LIB standard [12].

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 shorter the lifetime of the sensor node is, denoted by L_i. The objective for the sensor nodes is to cover a set of $m \geq 1$ points of interest. Let $d_{i,j}$ denote the physical distance between the i^{th} sensor node and the j^{th} point. Let $w_{i,t}$ be a Boolean variable that denotes whether the i^{th} sensor node is awake at the t^{th} time interval. T denotes the lifetime of the WSN. In the rest of this paper, we consider n, m, T arbitrarily fixed and give all definitions relative to them.

The following constraints are defined on a WSN:

(C1) **Lifetime constraint.** For each sensor node, the number of time intervals at which the node is awake must not exceed the node's lifetime.

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

(C2) **Coverage constraint [13], [14], [15].** Every point must be covered by at least K sensor nodes at any time, where $K \geq 1$ is a predefined constant.

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

where $S_j = \{i \mid d_{i,j} \leq r_i\}$.

(C3) **Evasive constraint [16], [4], [5].** In a hostile environment or in critical systems, it is important to protect the sensors from being active for too long. In such applications, each sensor node must not stay active for more than E consecutive time intervals, where $E \geq 1$ is a predefined constant.

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

(C4) **Moving target constraint [4], [5].** To improve resiliency and security, some critical points may require not be covered by the same sensor for more than M consecutive time intervals, where $M \geq 1$ is a predefined constant and $M < E$.

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

where $CR \subseteq [1, m]$ is a predefined set of critical points.

III. SAT and SMT Encoding of WSN Verification

To verify WSNs that are represented as introduced in Section II, we encode the aforementioned constraints (C1)–(C4) both to Boolean logic (SAT) and to SMT. We avoid the use of quantifiers by unrolling all constraints, representing this way (C1)–(C4) as quantifier-free formulas..

In this section we overview related key approaches that our work builds upon to express WSN (correctness) properties as instances of SAT/SMT.

A. Boolean Cardinality Constraints

By a Boolean cardinality constraint we mean a so-called "AtMost" constraint and define it as $\sum_{i=1}^{k} l_i \leq c$, where each l_i is a Boolean literal and $c \in \mathbb{N}$ is a constant. Note that the lifetime constraint (C1), the evasive constraint (C3) and the moving target constraint (C4) each are of this kind.

The coverage constraint (C2) however is a so-called "AtLeast" constraint, where an "AtLeast" constraint is defined as $\sum_{i=1}^{k} l_i \geq c$. It is easy to see that an "AtLeast" constraint can be translated to an "AtMost" constraint $\sum_{i=1}^{k} \overline{l_i} \leq k - c$.

B. SAT Encoding

There are various existing, well-known approaches expressing Boolean cardinality constraints into Boolean logic, for example by using sequential counters [17], cardinality networks [18] or modulo totalizers [19], [20].

Sequential counters [17] encode a Boolean cardinality constraint into the following Boolean formula:

$$\begin{aligned}
& (l_1 \Leftrightarrow s_{1,1}) && \\
\wedge \quad & \overline{s_{1,j}} && \text{for } j \in [2, c] \\
\wedge \quad & (s_{i,1} \Leftrightarrow l_i \vee s_{i-1,1}) && \text{for } i \in [2, n] \\
\wedge \quad & \big(s_{i,j} \Leftrightarrow (l_i \wedge s_{i-1,j-1}) \vee s_{i-1,j}\big) && \text{for } i \in [2, n], j \in [2, c]
\end{aligned}$$

All the Boolean variables $s_{i,j}$ are introduced as fresh variables and the formula above can be converted into its CNF [17].

Cardinality networks [18] yield another, refined approach for encoding Boolean cardinality constraints. For improving reasoning about cardinality constraints encoded, for example, using sequential counters, a cardinality network encoding of a cardinality constraint divides the cardinality constraint into multiple instances of the base operations *half sorting* and *simplified half merging*, which basically work as building blocks, as shown in Figure 1.

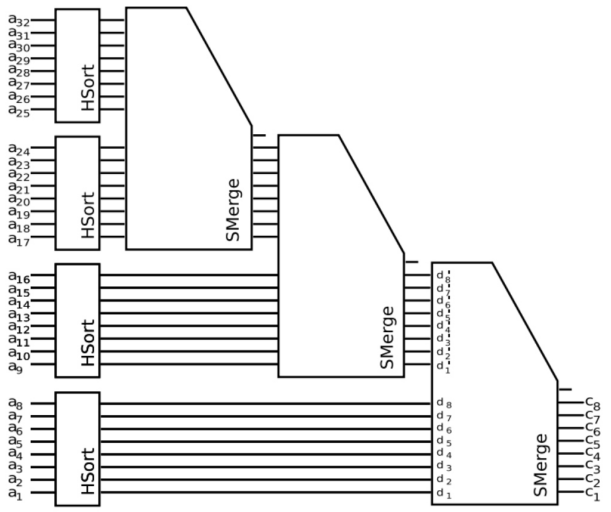

Figure 1. Representation of $\sum_{i=1}^{32} a_i \leq 8$ as a cardinality network [18].

The *modulo totalizer* cardinality encoding [19] and its variant for k-cardinality [20] improve the above described approach based on cardinality network, especially in connection with MaxSAT solving. The modulo totalizer approach of [19] addresses limitations of the half sorting cardinality network approach from [18], by using totalizer encodings from [21] in order to reduce the number of variables during CNF encodings. The modulo totalizer cardinality encoding of [19] decreases the number of clauses used in [21], and hence improves cardinality network encodings during constraint propagation.

C. SMT Encoding

Boolean cardinality constraints can naturally be encoded as SMT properties. In particular, in our work, we use SMT properties expressed in the quantifier-free theory of linear integer arithmetic QF_LIA. However, the Boolean literals l_i from Section III-A need to be converted to express sum properties over them. Such a conversion can easily be obtained using if-then-else expression `(ite` l_i `1 0)` in the SMT-LIB format, where `1` and `0` represent integer constants.

IV. Portfolio Solving for WSN Optimization

We now describe our setting and considerations for WSN optimization. By WSN optimization we mean the search for the maximal lifetime of a WSN. A satisfying model for WSN optimization is a truth assignment to the Boolean variables $w_{i,t}$, which represents an optimal sleep/wake-up scheduling for the sensor nodes.

The maximal lifetime for a WSN can be searched by iteratively calling the underlying SAT/SMT solver with different values for T. In this regard, T denotes the least lifetime we are currently trying to satisfy with the given WSN.

WSN optimization is a *monotonic* optimization problem [10] meaning that, as incrementing the value of T, all the resulting SAT/SMT instances are satisfiable until exceeding the optimum. That is, there exists no UNSAT instances below the optimum.

A. Portfolio SAT and SMT Solving

In previous experiments with OMT solvers [5], [6], [10], we observed that the bottleneck for underlying SMT solvers was typically to solve satisfiable instances. In our current experiments with underlying SAT solvers, we observed the opposite: the SAT-based approach is rather powerful on satisfiable instances and poor on unsatisfiable ones.

Therefore, it seems to be a decent idea to run a SAT solver and an SMT solver parallel, in a portfolio setting. Thus, in each iteration, both the SAT and the SMT encodings of the current WSN verification problem are generated and fed into the underlying SAT and SMT solvers, respectively. Then, the two solvers are executed as parallel processes. Whenever any of those two solvers obtains the result, we terminate the other solver.

B. Search Strategies

In this paper, we experiment with three different strategies for searching the optimal lifetime.

Linear search can be applied by setting T to the lowest possible value 1 and then, after each satisfiable instance, incrementing T by one. When we hit the first UNSAT instance, the optimum $T-1$ can returned.

Binary search uses a lower bound lb and an upper bound ub, and sets the next value of T to $\frac{lb+ub}{2}$. In the WSN context, lb stores the highest lifetime of satisfiable instances so far and, similarly, ub the lowest lifetime of UNSAT instances so far. Since the sensor nodes together cannot be active for longer than the sum of their individual lifetimes (L_i) and at least K sensor nodes must be active in each time interval, we use the initial values

$$lb = 1 \quad \text{and} \quad ub = \frac{\sum_{i=1}^{n} L_i}{K}$$

In [10], we propose a speed-up for *linear search by applying regression analysis*. To use this approach, we need to define a so-called *resource function* f_{RES}, in order to obtain data points $\big(T, f_{\text{RES}}(T)\big)$ for regression analysis. For a WSN, the charge of the batteries of sensor nodes can be considered

to be such a resource, which is continuously decreasing until draining. This is estimated by the following resource function:

$$f_{\text{RES}}(T) \quad = \quad \sum_{i=1}^{n} L_i - \sum_{i=1}^{n} \sum_{t=1}^{T} w_{i,t}$$

The value of the resource function is obtained by summing the lifetime of all the sensor nodes (which is the theoretical maximum of the network's lifetime), and then subtracting the sum of the time intervals that have been used up so far. This is the only search strategy which needs to obtain the satisfying model from the underlying solver in each iteration, which suffers from overhead. However, based on our experiments, this pays off compared to simple linear search.

C. Implementation

Our WSN optimization approach described in the previous sections is implemented in Python. The package PYSAT [22] provides a unified API to several SAT solvers such as MINISAT [23], GLUCOSE [24] and LINGELING [25]. From our point of view, it is even more important that PYSAT supports a lot of encodings for Boolean cardinality constraints, including sequential counters [17], cardinality networks [18] and modulo totalizer [19], [20]. Furthermore, PYSAT offers API to the SAT solver MINICARD [26], which handles Boolean cardinality constraints natively, instead of translating them into CNF.

In a similar manner, the Python package PYSMT [27] provides a unified API to several state-of-the-art SMT solvers over QF_LIA, such as MATHSAT [28], Z3 [11], CVC4 [29] and YICES [30].

To run two solvers parallel to each other, we instantiate `ProcessPool` from the Python module `pathos.multiprocessing` [31], which can run jobs with a non-blocking and unordered map.

V. EXPERIMENTS AND RESULTS

In our experiments, we use the WSN benchmarks from [6], by relying on four different constraint settings and three different density groups. Our experiments were run on 3.6 GHz 8-core CPU with 8 GB memory. The wall clock time limit was set to 1200 seconds and the memory limit to 3 GB. Our implementation, together with our benchmarks and log files, is available at:

https://iot.uni-eszterhazy.hu/en/research/tools

For the sake of comparison with previous results with OMT solvers, we run our PULI solver from [10] with linear search boosted by regression and also with binary search, as well as the OMT solvers OPTIMATHSAT [7], Z3 [8] and SYMBA [9]. Regarding the portfolio-based approach, we run MINISAT (v2.2), GLUCOSE (v4.1), LINGELING (bbc-9230380-160707) and MINICARD (v1.2) as SAT solvers, MATHSAT (v5.5.1) and Z3 (v4.8.4) as SMT solvers.

OMT			***Search***	***Solved/TO***	***Runtime***
PULI			lin-reg	19/1	63.1
			binary	19/1	63.7
OPTIMATHSAT				19/1	173.9
Z3-OPT				10/10	605.8
SYMBA				10/10	654.4
SAT	***Encoding***	***SMT***	***Search***	***Solved/TO***	***Runtime***
MINICARD		MATHSAT	linear	20/0	3.6
			lin-reg	20/0	2.0
			binary	20/0	1.7
		Z3	linear	20/0	4.0
			lin-reg	20/0	2.1
			binary	20/0	1.8
MINISAT	SEQ-COUNTER	MATHSAT	linear	20/0	4.7
			lin-reg	20/0	2.7
			binary	20/0	2.0
		Z3	linear	20/0	5.0
			lin-reg	20/0	3.0
			binary	20/0	2.2
GLUCOSE	SEQ-COUNTER	MATHSAT	linear	20/0	4.7
			lin-reg	20/0	2.8
			binary	20/0	2.2
		Z3	linear	20/0	5.1
			lin-reg	20/0	3.0
			binary	20/0	2.2
LINGELING	SEQ-COUNTER	MATHSAT	linear	20/0	5.8
			lin-reg	20/0	3.2
			binary	20/0	2.8
		Z3	linear	20/0	5.9
			lin-reg	20/0	3.3
			binary	20/0	2.7

Table I
RESULTS FOR WSNS OF 40-50% DENSITY WITH 10 SENSOR NODES, 4 TARGET POINTS, 2-COVERAGE, EVASIVE CONSTRAINT WITH E = 2, AND MOVING TARGET CONSTRAINT WITH M = 1.

Tables I-IV summarize some of the results of our experiments. In the columns "OMT", "SMT" and "SAT" the names of the OMT, SMT and SAT solvers are shown, respectively. The "Encoding" column shows the Boolean cardinality constraint encoding: sequential counters, cardinality networks and k-cardinality modulo totalizer. The column "Search" contains the name of the search strategy, where "lin-reg" stands for linear search boosted with regression. The column "Solved" shows the total number of solved instances, "TO" the number of timeouts, and "Runtime" the average runtime in seconds. None of the solvers exceeded the memory limit on any of the benchmark instances.

In Table I, we can see the results for WSNs of 40-50% density when the WSN constraints defined in Section II are all checked, including the coverage constraint (with $K = 2$),

the evasive constraint (with $E = 2$) and the moving target constraint (with $M = 1$). As it can immediately be seen, our current portfolio-based approach does not timeout at all and outperforms the OMT solvers by 1-2 orders of magnitude. PULI is the fastest OMT solver on the WSN benchmarks in general [10], but it is still slower by 1 order of magnitude than the portfolio approach.

We ran three SAT solvers with all the three cardinality encodings introduced in Section III, but on the WSNs of density 40-50% we did not observe significant difference in performance, therefore the table only shows the results for sequential counters.

Regarding search strategies, regression is able to boost linear search pretty well, since its performance often approaches that of binary search. From now on, we will show only the results for binary search, since binary search outperformed the other two strategies on each of the benchmarks in our experiments.

In Table II, we can see the results for WSNs of 60-70% density, for the same constraint setting. Here, the difference between different cardinality encodings starts to manifest, in terms of the performance of SAT solvers. Using sequential counters pays off far more than using cardinality networks or k-cardinality modulo totalizer, especially when running MINISAT and GLUCOSE. MATHSAT and Z3 provide quite similar runtimes except for when executed in combination with LINGELING.

Table III show results for the most difficult 80-90% density group, but for the easiest constraint setting, i.e., when the evasive constraint and the moving target constraint are not checked. In comparison with Table II, the OMT solvers are even more efficient in solving the current instances, and so is the portfolio-based approach with MINICARD. The other SAT solvers behave completely in the other way around. The solver performance with sequential counters is significantly higher than that of cardinality networks and k-cardinality modulo totalizer. For the latter cardinality encodings, the SAT solvers run out of time quite often, therefore we will drop those encodings in the next table. The combination of LINGELING + Z3 seems an interesting exception, but it is still much slower than the MINICARD-based approaches or the ones that use sequential counters.

Table IV shows the results for the most difficult benchmark: for WSNs of 80-90% density with all the WSN constraints being checked. As it was suggested before, we only show here the results for sequential counters, excepts for the LINGELING+Z3 setting with k-cardinality modulo totalizer encoding, which does not seem competitive anymore.

OMT			***Solved/TO***	***Runtime***
PULI			19/1	75.5
OPTIMATHSAT			16/4	327.2
Z3-OPT			6/14	858.8
SYMBA			7/13	924.5
SAT	***Encoding***	***SMT***	***Solved/TO***	***Runtime***
MINICARD		MATHSAT	20/0	2.8
		Z3	20/0	2.6
MINISAT	SEQ-COUNTER	MATHSAT	20/0	3.5
		Z3	20/0	3.6
	CARD-NETWORK	MATHSAT	20/0	55.1
		Z3	20/0	51.9
	kM-TOTALIZER	MATHSAT	19/1	74.8
		Z3	19/1	70.2
GLUCOSE	SEQ-COUNTER	MATHSAT	20/0	7.6
		Z3	20/0	7.7
	CARD-NETWORK	MATHSAT	20/0	62.2
		Z3	19/1	70.7
	kM-TOTALIZER	MATHSAT	19/1	84.2
		Z3	19/1	104.9
LINGELING	SEQ-COUNTER	MATHSAT	20/0	29.5
		Z3	20/0	8.8
	CARD-NETWORK	MATHSAT	19/1	72.0
		Z3	19/1	71.7
	kM-TOTALIZER	MATHSAT	19/1	79.7
		Z3	20/0	38.5

Table II
RESULTS FOR WSNS OF 60-70% DENSITY WITH 10 SENSOR NODES, 4 TARGET POINTS, 2-COVERAGE, EVASIVE CONSTRAINT WITH E = 2, AND MOVING TARGET CONSTRAINT WITH M = 1, ONLY WITH BINARY SEARCH.

VI. CONCLUSION

In this paper, we propose a novel approach for speeding up the optimization of WSNs, in comparison with previous results with OMT solvers. The current approach models the WSN optimization problem as Boolean cardinality constraints, which are fed into both an underlying SAT solver and an SMT solver. The two solvers are executed in parallel in a portfolio setting. Our experiments with different SAT and SMT solvers show MINICARD + Z3 to be the most efficient combination, which outperforms the OMT solvers by 1-2 orders of magnitude. Our further experiments with different encodings of Boolean cardinality constraints show that it is highly recommended to apply sequential counters as opposed to cardinality networks and k-cardinality modulo totalizer.

As future work, we would like to run further experiments with larger WSNs of 10s or 100s of sensor nodes. This was not yet possible with OMT solvers, but now, with our new portfolio SAT + SMT approach, this objective looks quite realistic to achieve. Another objective is to investigate more realistic WSN models.e For instance, what if the ranges of the sensor nodes are not fixed, but could be adjusted on the fly? In this case, the solvers would try to minimize the scopes every time in order to prolong the WSN's lifetime.

OMT			*Solved/TO*	*Runtime*
PULI			20/0	11.4
OPTIMATHSAT			14/6	794.5
Z3-OPT			20/0	13.3
SYMBA			20/0	155.0
SAT	***Encoding***	***SMT***	***Solved/TO***	***Runtime***
MINICARD		MATHSAT	20/0	1.9
		Z3	20/0	1.6
MINISAT	SEQ-COUNTER	MATHSAT	20/0	3.3
		Z3	20/0	3.2
	CARD-NETWORK	MATHSAT	7/13	852.8
		Z3	16/4	282.3
	kM-TOTALIZER	MATHSAT	2/18	1097.3
		Z3	15/5	322.2
GLUCOSE	SEQ-COUNTER	MATHSAT	20/0	9.4
		Z3	20/0	9.3
	CARD-NETWORK	MATHSAT	6/13	843.1
		Z3	15/5	315.4
	kM-TOTALIZER	MATHSAT	7/13	891.3
		Z3	16/4	260.3
LINGELING	SEQ-COUNTER	MATHSAT	20/0	94.2
		Z3	20/0	16.1
	CARD-NETWORK	MATHSAT	4/16	1065.7
		Z3	17/3	216.6
	kM-TOTALIZER	MATHSAT	11/9	759.3
		Z3	19/1	134.0

Table III
RESULTS FOR WSNS OF 80-90% DENSITY WITH 10 SENSOR NODES, 4 TARGET POINTS, 2-COVERAGE, ONLY WITH BINARY SEARCH.

OMT			*Solved/TO*	*Runtime*
PULI			20/0	42.3
OPTIMATHSAT			20/0	192.8
Z3-OPT			9/11	774.7
SYMBA			4/16	1098.8
SAT	***Encoding***	***SMT***	***Solved/TO***	***Runtime***
MINICARD		MATHSAT	20/0	3.7
		Z3	20/0	3.6
MINISAT	SEQ-COUNTER	MATHSAT	20/0	4.5
		Z3	20/0	4.2
GLUCOSE	SEQ-COUNTER	MATHSAT	20/0	24.9
		Z3	20/0	28.0
LINGELING	SEQ-COUNTER	MATHSAT	20/0	23.3
		Z3	20/0	19.2
	kM-TOTALIZER	Z3	16/4	348.5

Table IV
RESULTS FOR WSNS OF 80-90% DENSITY WITH 10 SENSOR NODES, 4 TARGET POINTS, 2-COVERAGE, EVASIVE CONSTRAINT WITH E = 2, AND MOVING TARGET CONSTRAINT WITH M = 1, ONLY WITH BINARY SEARCH.

In such a case, Boolean cardinality constraints might not be sufficient to encode the entire WSN problem, but only a part of that. We would like to investigate how to apply our portfolio SAT + SMT approach for such models. It might also be worth to experiment with planners as alternatives to SAT solvers, as future work.

ACKNOWLEDGMENTS

This research was also supported by the ERC Starting Grant SYMCAR 639270, the ERC Proof of Concept Grant SYMELS 842066, the Swedish Wallenberg Academy Fellowship TheProSE, and the OMAA grant 101öu8.

REFERENCES

[1] C. Biró, G. Kusper, T. Radványi, S. Király, P. Szigetváry, and P. Takács, "SAT representation of randomly deployed wireless sensor networks," in *Proc. International Conference on Applied Informatics (ICAI)*, 2014, pp. 101–111.

[2] C. Biró and G. Kusper, "Equivalence of strongly connected graphs and black-and-white 2-SAT problems," *Miskolc Mathematical Notes*, vol. 19, no. 2, pp. 755–768, 2018.

[3] W. Kong, M. Li, L. Han, and A. 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)*, 2014, pp. 240–245.

[4] Q. Duan, S. Al-Haj, and E. 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)*, 2012, pp. 316–321.

[5] G. Kovásznai, C. Biró, and B. Erdélyi, "Generating optimal scheduling for wireless sensor networks by using optimization modulo theories solvers," in *Proc. Int. Workshop on Satisfiability Modulo Theories (SMT)*, ser. CEUR, vol. 1889, 2017, pp. 15–27. [Online]. Available: http://ceur-ws.org/Vol-1889/paper2.pdf

[6] G. Kovásznai, B. Erdélyi, and C. Biró, "Investigations of graph properties in terms of wireless sensor network optimization," in *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*. IEEE, 2018, pp. 1–8.

[7] R. Sebastiani and P. Trentin, "OptiMathSAT: A tool for optimization modulo theories," in *Proc. Int. Conf. on Computer-Aided Verification (CAV)*, ser. LNCS, vol. 9206. Springer, 2015, pp. 447–454.

[8] N. Bjørner, A.-D. Phan, and L. Fleckenstein, "μZ - an optimizing SMT solver," in *Proc. Int. Conf. for Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. LNCS, vol. 9206. Springer, 2015, pp. 194–199.

[9] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik, "Symbolic optimization with SMT solvers,"

in *Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. ACM, 2014, pp. 607–618. [Online]. Available: http://doi.acm.org/10.1145/2535838.2535857

[10] G. Kovásznai, C. Biró, and B. Erdélyi, "Puli - A problem-specific OMT solver," in *Proc. 16th International Workshop on Satisfiability Modulo Theories (SMT 2018)*, ser. EasyChair Preprint, no. 371, 2018.

[11] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. TACAS'08/ETAPS'08. Springer-Verlag, 2008, pp. 337–340.

[12] C. Barrett, P. Fontaine, and C. Tinelli, "The SMT-LIB standard: Version 2.6," Department of Computer Science, The University of Iowa, Tech. Rep., 2017, available at `www.SMT-LIB.org`.

[13] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. Int. Workshop on Wireless Sensor Networks and Applications*, ser. WSNA'02. ACM, 2002, pp. 32–41.

[14] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-6615-6

[15] M. Cardei, "Coverage problems in sensor networks," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 899–927. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-7997-1_72

[16] Z. Benenson, F. C. Freiling, and P. M. Cholewinski, "Advanced evasive data storage in sensor networks," in *Proc. Int. Conf. on Mobile Data Management*, ser. MDM'07. IEEE Computer Society, 2007, pp. 146–151. [Online]. Available: http://dx.doi.org/10.1109/MDM.2007.29

[17] C. Sinz, "Towards an optimal CNF encoding of boolean cardinality constraints," in *Proc. Principles and Practice of Constraint Programming (CP)*, ser. Lecture Notes in Computer Science, vol. 3709. Springer, 2005, pp. 827–831.

[18] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, "Cardinality networks: a theoretical and empirical study," *Constraints*, vol. 16, no. 2, pp. 195–221, 2011.

[19] T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita, "Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers," in *25th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2013, pp. 9–17.

[20] A. Morgado, A. Ignatiev, and J. Marques-Silva, "Mscg: Robust core-guided MaxSAT solving," *JSAT*, vol. 9, pp. 129–134, 2014.

[21] O. Bailleux and Y. Boufkhad, "Efficient CNF Encoding of Boolean Cardinality Constraints," in *Proc. 9th International Conference on Principles and Practice of Constraint Programming (CP)*, 2003, pp. 108–122.

[22] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science, vol. 10929. Springer, 2018, pp. 428–437.

[23] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science, vol. 2919. Springer, 2004, pp. 502–518.

[24] G. Audemard and L. Simon, "Lazy clause exchange policy for parallel sat solvers," in *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science, vol. 8561. Springer, 2014, pp. 197–205.

[25] A. Biere, "CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017," in *Proc. of SAT Competition 2017 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Series of Publications B, vol. B-2017-1. University of Helsinki, 2017, pp. 14–15.

[26] M. H. Liffiton and J. C. Maglalang, "More expressive constraints for free," in *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science, vol. 7317. Springer, 2012, pp. 485–486.

[27] M. Gario and A. Micheli, "PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms," in *International Workshop on Satisfiability Modulo Theories (SMT)*, 2015.

[28] A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani, "The MathSAT5 SMT solver," in *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, N. Piterman and S. Smolka, Eds., vol. 7795. Springer, 2013, pp. 93–107.

[29] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli, "CVC4," in *Proc. Int. Conf. on Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, vol. 6806. Springer, 2011, pp. 171–177.

[30] B. Dutertre, "Yices 2.2," in *Proc. Int. Conf. on Computer-Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, A. Biere and R. Bloem, Eds., vol. 8559. Springer, 2014, pp. 737–744.

[31] M. M. McKerns, L. Strand, T. Sullivan, A. Fang, and M. A. Aivazis, "Building a framework for predictive science," *arXiv preprint arXiv:1202.1056*, 2012.