An SMT-based Accurate Algorithm for the K-Coverage Problem in Sensor Network

Weiqiang Kong, Ming Li, Long Han, and Akira Fukuda
Graduate School of IS&EE, Kyushu University, Japan.
weiqiang@qito.kyushu-u.ac.jp, {ziqiangliming, l-han}@f.ait.kyushu-u.ac.jp,
fukuda@ait.kyushu-u.ac.jp

Abstract—**In the context of wireless sensor network (WSN), the K-Coverage problem denotes that each point in a certain network area is covered by at least K sensors at the same time so as to guarantee the quality of services provided by the WSN. In this paper, we first propose a bottom-up modeling method for the K-coverage problem. Based on this method, we investigate a set of iteratively-applicable simplification techniques for simplifying the problem. Furthermore, we propose a satisfiability modulo theory (SMT) based algorithm for computing an accurate solution to the K-coverage problem. Experimental results have shown that our proposed simplification techniques and algorithm provide sufficiently satisfiable performance with respect to both computing speed and problem size.**

Keywords-K-coverage; wireless sensor network; satisfiability modulo theory; accurate algorithm;

I. INTRODUCTION

Wireless sensor network (WSN) is an infrastructure comprised of sensing (measuring), computing, and communication elements that gives an administrator the ability to instrument, observe, and react to events and phenomena in a specified environment [1].

WSN has been developing rapidly in recent years and been applied to many fields including military, business, and agriculture, etc. Coverage problems are one of the most active research topics related to WSN. It is generally necessary to deploy multiple sensors to cover an entire WSN area so as to provide services within the area. Each sensor used in WSN has a limited sensing radius range. A point in a WSN area is said to be covered if it is within the radius range of a sensor. If a point is covered by only one sensor, then it is said to be 1-covered. Consequentially, a WSN area is said to be K-covered if every point in the area is covered by at least K sensors at the same time. Such a restriction is called the K-coverage problem. Due to costs and/or interference among multiple sensors, it is often not practically feasible to deploy an arbitrarily large number of sensors to simply fulfill the K-coverage restriction. How to deploy sensors in a reasonable way so as to decrease the number of sensors while fulfilling the K-coverage restriction, is the research topic of this paper.

The K-coverage problem is a typical combinatorial optimization problem. As the size increase of the target WSN area, the scale of solution space grows exponentially. In a large-scale K-coverage problem, it is generally difficult to compute optimal solutions. Therefore, existing algorithms usually circumvent this problem by sacrificing coverage rate or solution accuracy to improve algorithms' performance [2], [3], [4]. In the case of sacrificing coverage rate, a compromised coverage rate is used to replace the strict 100% rate; in the case of sacrificing solution accuracy, it is allowed to deploy redundant sensors.

In this paper, we investigate high performance algorithms for the K-coverage problem, which should not sacrifice coverage rate (i.e., fulfill strictly 100% rate) and should find the minimum number of sensors. Our contributions made in this paper are as follows: (1) we proposed a bottom-up modeling method for the K-coverage problem; (2) we proposed a set of iteratively-applicable simplification techniques for simplifying the K-coverage problem; (3) furthermore, we proposed an SMT-based efficient algorithm for computing accurate solutions to the K-coverage problem. As shown by our preliminary experiments, the simplification techniques as well as the algorithms provide sufficiently satisfiable performance.

The paper is organized as follows. Section II introduces the modeling method of K-coverage problem; Section III investigates a set of methods for simplifying the K-coverage problem; Section IV describes our proposed SMT-based accurate algorithms and experiment evaluation of the algorithms; Section V concludes the paper.

II. MODELING OF THE K-COVERAGE PROBLEM

Candidate-Points. In a practical environment, positions in which sensors are allowed to be deployed are usually limited. For example, in an indoor environment, they are often deployed on walls or pillars. Therefore, we abstract the positions in which sensors can be deployed and call them as "candidate points for sensor deployment" (called "candidate-points" for simplicity).

For each candidate-point, we can only deploy one sensor (or not deploy). The number and location of candidate-points depend on the physical environment of the target area. As one of the most important input-data of K-coverage algorithms, the number of candidate-points determines directly the complexity and solution space of the problem. For each candidate-point, we can make a choice to deploy a sensor there or not. Therefore, the number of choices is equal to 2^N, where N denotes the number of candidate-points.

Observation-Points. In addition to "candidate-points", to represent the target area under consideration, we define another type of points, which are called as "observation-points". An observation-point is said to be covered by a sensor if and only if the point is within the radius range of the sensor. With this definition, we say that an observation point is K-covered if it is covered by K sensors. If all the observation-points in the target area are covered by sensors, we say that the area is covered by sensors. The distribution of observation-points can be even, or the distribution can be customized according to actual requirements.

The density of observation-points reflects the accuracy of the model, and generally, the higher the density, the more accurate the coverage by the sensors of the target area is. However, as the increase of the number of observation-points, the complexity of the problem also increases. The K-coverage problem can be defined again based on the above two definitions. That is, to find the minimum number of sensors, the deployment of which can satisfy that all observation-points in the target area is K-covered by sensors.

A. Basic Algorithms for Computing K-Coverage

We can easily come up with a simple exhaustive algorithm to compute this problem. The pseudo-code of the algorithm is shown in Algorithm 1. (Actually, this algorithm is also mentioned in [5] and is called as the Original Combination Algorithm). We assume there are a set O of observation-points and a set N of candidate-points; We use C to denote the set of all possible combination of sensor deployment; We use function $k\text{-}covered(c)$ to evaluate if a sensor deployment $c \in C$ satisfies K-coverage, which returns `true` when c satisfies and `false` otherwise; We use function $num(c)$ to denote the number of sensors in c; We use min to hold the minimum number of sensors that satisfies K-coverage; We use $outputDeploy$ to hold the sensor deployment that satisfies K-coverage and has the minimum number of sensors.

Algorithm 1. A Simple Exhaustive Algorithm for K-Coverage

```
1. Input: The sets $O$, $N$, and $C$.
2. Output: The sensor deployment $outputDeploy$
3.
4. for each $c \in C$
5.     if $k\text{-}covered(c) == \texttt{true}$ then
6.         if $num(c) < min$ then
7.             $min = num(c)$;
8.             $outputDeploy = c$;
9.         end if
10.    end if
11. end for
12. return $outputDeploy$
```

However, we can notice that there are a lot of performance deficiencies in Algorithm 1: let us assume that there are N candidate-points, the time complexity of the algorithm is $O(2^N)$ since the number of all possible combination of sensor-deployment is 2^N, and in the worst case, we need to check every combination of them. Even if there are some ways of improving the performance of this algorithm, e.g., by checking combinations in an ascending order of the number of sensors, it is still difficult to change the time complexity essentially. The time complexity is still O(2^N).

Therefore such kind of simple exhaustive algorithms, which are based on the combinations of sensor deployment, is not practically feasible. As the increase of the number of candidate-points, the complexity of such algorithms will grow exponentially.

In addition, there is one more reason for the low performance of Algorithm 1. In the step of computing whether a sensor deployment c satisfies K-coverage (namely the function $k\text{-}covered(c)$), there are a lot of unnecessary (and thus avoidable) repeated calculation. We explain this computation repetition in more detail below.

We use sets to denote the combination of sensor deployment, where the elements of each set are (sensor) candidate-points. As shown in Figures 1 and 2, let us assume that there are two combinations of sensor deployment $C1$ and $C2$, where $C1 = \{s1, s2, s3, s4\}$ and $C2 = \{s1, s2, s3, s5\}$; also assume that there are two observation-points $o1$ and $o2$ in the target area, and the objective coverage considered is 2-coverage. The difference of $C1$ and $C2$ only lies on $s4$ and $s5$, and $s4$ and $s5$ are in positions far away from the observation-point $o1$. In Algorithm 1, we need to check whether $C1$ and $C2$ can make observation-points $o1$ and $o2$ satisfy 2-coverage. Although $C1$ and $C2$ can both satisfy the 2-coverage for $o1$ and $o2$. But it should be noted that, compared to the sensors observable by $o2$, the sensors that can be observed by $o1$ are the same in the two deployments $C1$ and $C2$. Consequentially, if deployment $C1$ could satisfy the 2-coverage for $o1$, then $C2$ could as well. Therefore, it is actually not necessary to check the coverage for $o1$ twice. We can imagine that the performance loss here due to unnecessary repeated computation is huge.

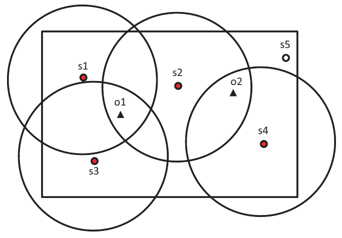

Figure 1. A Sample Combination of Sensor Deployment $C1$

In order to reduce this kind of performance loss analyzed above, we could, instead of directly computing the *combination* of desired sensor deployment, compute in advance the sets of (sensor deployment) candidate-points for each observation-point, which make the observation point satisfy K-coverage. We can see that the sets of candidate-points for

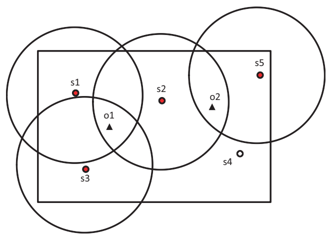

Figure 2. A Sample Combination of Sensor Deployment $C2$

$o1$ is $S11 = \{s1, s2\}$, $S12 = \{s1, s3\}$, $S13 = \{s2, s3\}$; for $o2$, there are two such sets $S21 = \{s2, s4\}$, $S22 = \{s2, s5\}$. Because our goal is to find a minimum-sized number of sensors, we can easily make an assertion that the final result (for our goal of a combination of candidate-points for 2-coverage) must contain and only need to contain one set of $S11$, $S12$, or $S13$, and must contain and only need to contain one subset of $S21$ or $S22$. That is, the final result for the best 2-coverage combination must be one of the follows: $S11 \cup S21 = \{s1, s2, s4\}$, $S12 \cup S21 = \{s1, s2, s3, s4\}$, $S13 \cup S21 = \{s2, s3, s4\}$, $S11 \cup S22 = \{s1, s2, s5\}$, $S12 \cup S22 = \{s1, s2, s3, s5\}$, $S13 \cup S22 = \{s2, s3, s5\}$. Among of them, the combinations with minimum number of sensors are $\{s1, s2, s4\}$, $\{s2, s3, s4\}$, $\{s1, s2, s5\}$, $\{s2, s3, s5\}$. The four sets of sensors are all optimal solutions. As can be seen, we have used here the bottom-up method (by focusing on observation-points) to replace the top-down method (by focusing on candidate-points), and this could avoid the repeated-checking problem mentioned above.

Another advantage of such bottom-up method is that it makes simplification of the K-coverage problem easier.

III. Simplification

We have proposed a set of simplification techniques that can be used as premises of the SMT-based algorithm (to be introduced in Section IV) for simplifying the K-coverage computation problem. These techniques can be applied iteratively for the simplification purpose. In this section, due to space limitation, we only mention the basic ideas of two such techniques. For each observation-point, we can find sets of candidate-points that satisfy the K-coverage restriction. We called such sets as “satisfiable sets” of the observation-point. One observation-point may usually have more than one “satisfiable set”. We denote a single “satisfiable set” as S, and the set of all “satisfiable (sub)sets” as SS.

A. Eliminating Certain Observation-Points

Consider two observation-points $o1$ and $o2$, the sets of satisfiable set of $o1$ is $SS1 = \{S11 = \{s1, s2\}, S12 = \{s1, s3\}\}$ and the sets of satisfiable set of $o2$ is $SS2 = \{S21 = \{s1\}, S22 = \{s2\}, S23 = \{s3\}\}$. Note here that, under the restriction of K-coverage, the element number of satisfiable sets of $o1$ is 2, but the element number of $o2$ is 1. Although seemingly strange, this is possible as an intermediate result of simplification, which will be shown in the next subsection.

For the above example, we can see that $S11$ is the parent set of $S21$ and $S22$ (namely $S21, S22 \subseteq S11$); $S12$ is the parent set of $S23$. We can assert that if $o1$ is satisfied for K-coverage, then $o2$ must be satisfied as well. We can see that $o2$ can be eliminated since its satisfiability is contained by that of $o1$ and the existence of $o2$ does not have any effect on the final result. We call $o2$ as a *similar-point* of $o1$, and similar-points have transitivity but does not have commutativity.

B. Must-Be-Chosen Candidate-Points

Let us assume an observation-point $o1$, which has a *unique* satisfiable set $S1$. It can be asserted that every element $s \in S1$ must be contained in the final optimal result $outputDeploy$ (See Algorithm 1), because, otherwise, $o1$ will never be satisfied for K-coverage. We call the candidate-points contained in this unique satisfiable set as “must-be-chosen candidate-points”. Generally, when the distribution of observation-points is sparse, such kind of observation-points often exist. Eliminating such observation-points can reduce the number of candidate-points, and consequentially, reduce the complexity of the problem directly.

Consider an example: there are three observation-points $o1$, $o2$, and $o3$; the set of satisfiable sets of $o1$ is $SS1 = \{S11 = \{s1, s2\}, S12 = \{s1, s3\}\}$, the set of satisfiable sets of $o2$ is $SS2 = \{S21 = \{s1, s2\}, S22 = \{s3, s4\}\}$, and the set of satisfiable sets of $o3$ is $SS3 = \{S31 = \{s4, s5\}\}$. We can see that there is only one satisfiable set $S31$ for $o3$, so all the elements in $S31$ should be in the final result. We thus mark $s4$ and $s5$ as Must-Be-Chosen candidate points by assigning $s4 = 1$ and $s5 = 1$ (we use 1 to denote that a point is chosen and 0 otherwise); remove $o3$ from the model, and further simplify $SS1$ and $SS2$. We use $SS1'$ and $SS2'$ to denote the simplified results of $SS1$ and $SS2$, respectively. $SS1' = \{S11 = \{s1, s2\}, S12 = \{s1, s3\}\}$ and $SS2' = \{S21 = \{s1, s2\}, S22 = \{s3\}\}$.

We can see that the element numbers of satisfiable sets of $SS2'$ are inconsistent (two for $S21$ and one for $S22$). As mentioned before, such situation is possible during simplification. Furthermore, observation-point $o2$ has now become a similar-point of $o1$. As introduced in Section III.A, $o2$ can be removed. Such kind of iterative simplification can greatly decrease the complexity of the problem. It is necessary to emphasize that multiple simplification may not need to be carried out in a fixed order. In the final computation algorithm for K-coverage, we will form these simplification techniques in a set and try repeatedly these techniques until no further simplification could be done.

IV. An SMT-based Accurate Algorithm

In general, SAT/SMT solving [6] is a technique to find variable-assignments to all the Boolean variables contained in a logical formula so as to make the formula `true`, or to determine that there is not such variable-assignments.

In a SAT problem, the formula is a Boolean expression written using only logical operators of $\wedge$, $\vee$, $\neg$, together with Boolean variables and parentheses. Therefore, the problem here is to, through assigning `true` or `false` to each of the Boolean variables, try to find an assignment that makes the entire formula have the value `true`. If all possible variable-assignments have been tried and no such assignment exists, then the problem here is said to be unsatisfiable.

SMT solving is an extension of SAT solving technique. In SMT problems, the formulas can contain variables of other types such as Integers and Reals etc., rather than merely Boolean variables as in SAT problems. An example of SMT problem is given as follows. The formula here is $(x+1) < 2 \wedge (y \vee z)$, where x is an Integer, y and z are Boolean variables. The formula is satisfiable since there exists at least one variable assignment, for example, $x = 0$, $y =$ `true`, $z =$ `false`. Another formula $(x+1) = 2 \wedge (x \neq 1)$ is obviously unsatisfiable. There are a lot of well-known SMT solvers such as Z3 [7], Yices [8], and CVC3 [9]. In this paper, we use the Z3 solver since it generally performs best.

A. A SMT-based Algorithm with Z3

To utilize the Z3 SMT solver to compute the K-coverage problem, we need in the first place to encode a K-coverage problem into the input language of Z3 (namely a set of Z3-recognizable formulas). Note that we do not directly encode the original K-coverage problem, but instead, we only encode the intermediate problem after simplification, into a set of formulas written in Z3 input language.

The general idea of encoding the problem (after simplification) is as follows: (Step 1) Declare an Integer variable for each of the candidate-points and define their values as either 0 or 1, where 0 denotes `false` and 1 denotes `true` ; (Step 2) Define a set of logical formulas using these variables, which restricts the conditions that are necessary to fulfill the K-coverage restriction; (Step 3) Define a logical formula, which restricts the condition that the sum (of the values) of all the candidate-points should be smaller or equal to the minimum value (of the sum of the candidate-points) computed so far.

To make it easier to understand, we use a concrete example to explain the idea of encoding. In the example: we consider a 2-coverage problem; there are two observation-points; after applying the simplification technique introduced in section III, the sets of satisfiable sets of these two observation-points are $SS1 = \{S11 = \{s1, s2\}, S12 = \{s3, s4\}\}$ and $SS2 = \{S21 = \{s1, s4\}, S22 = \{s2, s3\}\}$. We encode the problem into the following Z3 language (essentially, we use the SMT-LIB 2.0 language [10], a standard language, formulas written in which can be accepted by most state-of-the-art SMT Solvers including Z3):

```
 1. (declare-const s1 Int)
 2. (declare-const s2 Int)
 3. (declare-const s3 Int)
 4. (declare-const s4 Int)

 5. (assert (or (= s1 0) (= s1 1)))
 6. (assert (or (= s2 0) (= s2 1)))
 7. (assert (or (= s3 0) (= s3 1)))
 8. (assert (or (= s4 0) (= s4 1)))

 9. (assert (or (and (= s1 1) (= s2 1))
                (and (= s3 1) (= s4 1))))
10. (assert (or (and (= s1 1) (= s4 1))
                (and (= s2 1) (= s3 1))))
11. (assert (< (+ s1 s2 s3 s4 ) min))

12. (check-sat)
13. (get-model)
```

The first 4 lines are to declare four Integer variables with the names $s1$, $s2$, $s3$, and $s4$ by using the Z3 keyword `declare-const`. Lines from 5 to 9 are to define the possible values of these Integer variables. These lines correspond to (Step 1) mentioned above. Note that `assert` is a Z3 keyword to define a logical formula. For example, `(assert (or (= s1 0) (= s1 1)))` is the same as the logical formula $(s1 = 0) \vee (s1 = 1)$. Multiple formulas defined using the keyword `assert` are logically and-conjuncted. For example, the logical formula defined through lines 5 and 6 is same as $((s1 = 0) \vee (s1 = 1)) \wedge ((s2 = 0) \vee (s2 = 1))$.

Lines from 9 to 10 define the conditions that restricts the candidate-points to satisfy the 2-coverage of the problem. Essentially, the conditions are those defined in $SS1$ and $SS2$. These lines correspond to (Step 2) mentioned above.

Line 11 is to define that the sum of the values of the four candidate-points should be smaller or equal to a minimum value denoted by `min`. This line is the same as the logical formula $(s1 + s2 + s3 + s4) <$ `min`. Note that we did not declare the value of `min` in Z3 language before using it. As will be explained below, this value is not known in advance, so we try different values to find the minimum value.

The last line 12 with `(check-sat)` is simply a command to let the Z3 SMT solver to determine the satisfiability of the formulas defined above. Line 13 demands Z3 to output a variable-assignment if the formulas are satisfiable.

We save the above text into a file, input the file into Z3, and wait for Z3 to return a result. As mentioned above, since initially we do not know the exact value of `min`, so we circumvent this by generating multiple files, which have the same textual contents as above except the value of `min`. We then input these files one by one, in an ascending order of `min`'s values, into Z3. If, for a file, a variable-assignment is found which satisfies the 2-coverage problem, we stop there and get the final optimal solution (namely an optimal sensor deployment that satisfies 2-coverage for all the observation-points); if not, we input the next file with the plus-1 value for `min`, and so on.

For this example, we initially use 1 as the value for `min`, and Z3 returns unsatisfiable for it; we then input the file with `min=2` and input the file, again Z3 returns unsatisfiable. At last, if we set $min = 4$, Z3 returns that these formulas are satisfiable with an assignment $s1 = 1$, $s2 = 1$, $s3 = 0$, and $s4 = 1$. This shows an optimal result with the minimum sensor values of 3.

V. Experiments and Evaluation

We have conducted a series of experiments to evaluate the effectiveness and efficiency of the simplification techniques and the SMT-based algorithm. Our experiment environment is as follows: Windows PC running on a Intel-Core i7-2620M (@2.70GHz, 2.70GHz) CPU with 8.0GB Ram.

In our experiments, we consider 3-coverage problem of four target (rectangle) areas of different sizes, and of 50×50, 90×90, 140×140, and 190×190 units; we consider two kinds of distribution of candidate-points, one is *even* distribution and another is *random* distribution.

For candidate-points, in the case of even distribution, the candidate-points are located in a target area evenly with 14-unit intervals by considering the target area as a grid; in the case of random distribution, the candidate-points are located randomly but with the premise of satisfaction of 3-coverage (In other words, we only consider random distribution that satisfies 3-coverage, and do not taken other cases into account. This is a reasonable premise since our purpose is to examine the effectiveness of simplification). The number of candidate-points increases as with the size increase of target areas. In our experiments, the correspondence between the number of candidate-points and target areas are shown in Table I.

Table I
Correspondence between the Number of Candidate-Points and the Target Areas

Case No.	Target Area	Candidate-Point Number
Case 1	50×50	36
Case 2	90×90	100
Case 3	140×140	225
Case 4	190×190	400

For observation-points, no matter whether the distribution of candidate-points is even or random, observation-points are located evenly in a target area with 1-unit intervals by considering the target area as a grid. For the above target areas, since an observation-point will be deployed at the border (e.g., 0 position) of the area, the numbers of the observation-points are $51 \times 51 = 2601$, $91 \times 91 = 8281$, $141 \times 141 = 19881$, and $191 \times 191 = 36481$.

A. Evaluation of the Simplification Techniques

Through simplification, we can decrease the numbers of observation-points (or candidate-points), and consequentially, simplify the complexity of the target problem. We analyze the effectiveness of the simplification techniques partially mentioned in section III. Note that we do not experiment and analyze the effectiveness of each simplification technique separately since: the set of simplification techniques that we proposed can be iteratively applied; one simplification technique, which could not be applied at a time, may become applicable later after some other simplification techniques have been applied. Therefore, in the following experiments and analysis, the set of simplification techniques will be evaluated as a whole.

First, we evaluate, when the candidate-points are distributed evenly, how the observation-points can be decreased by applying our proposed simplification techniques. The comparison table, before and after applying simplification, is shown in Table II:

Table II
Decrease of Observation-Points with Even Distribution of Candidate-points

Case No.	Before Simplification	After Simplification
Case 1 (50×50)	2601	0
Case 2 (90×90)	8281	64
Case 3 (140×140)	19881	288
Case 4 (190×190)	36481	924

Note that the cases are characterized by the size of target areas (shown in Table I). We can observe that the number of observation-points is decrease extremely. Note also that in Case 1, after simplification the number of observation-points is 0, which seems to be strange. 0 here simply means that an optimal solution to the 3-coverage problem has already been found after simplification.

Next, we evaluate, when the candidate-points are distributed randomly, how the observation-points can be decreased by applying our proposed simplification techniques. The comparison table, before and after applying simplification, is shown as follows (Table III):

Table III
Decrease of Observation-Point with Random Distribution of Candidate-points

Case No.	Before Simplification	After Simplification
Case 1 (50×50)	2601	10
Case 2 (90×90)	8281	26
Case 3 (140×140)	19881	276
Case 4 (190×190)	36481	448

Similarly, we can observe that our proposed simplification techniques are quite effective for decreasing the number of observation-points, which can consequentially reduce the complexity of the coverage problem.

B. Evaluation of the SMT-based Algorithms

We report our experiment results on evaluation of the performance of (1) Algorithm 1 (tuned by following an ascending order of the number of sensors) vs. SMT-based algorithm both with simplification, and (3) SMT-based algorithm with and without simplification. In the experiments, to enlarge the scope of experiment, we consider 9 cases, in which the sizes of the target areas are from $20 \times 20, \ldots, 190 \times 190$.

1) Algorithm 1 without and with Simplification: The experiment data for the nine cases are listed in Table IV. Note that we removed the cases of 20×20, 30×30, and 40×40 since computation time for them is simply 0. We set 9000 seconds as time out.

Table IV
ALGORITHM 1 AND SMT-BASED ALGORITHM BOTH WITH SIMPLIFICATION (TIME IN SECONDS)

Case No.	Algorithm 1	SMT-based Algorithm
Case 4 (50×50)	0.182	0
Case 5 (90×90)	1.045	0
Case 6 (100×100)	14.353	0.96
Case 7 (110×110)	1095.346	1.58
Case 8 (140×140)	9000	14.82
Case 8 (190×190)	9000	186.77

From the results, we can find that, by using the SMT-based algorithm (particularly Z3 in our experiments), the coverage problem for large target areas such as those more than 140×140 can be computed as well.

2) SMT-based Algorithm with and without Simplification: The next question is that: how about if we directly use SMT solving techniques for the problem without simplification? Many optimization techniques have already been implemented in SMT solvers (such as Z3), and it may be possible that those optimization techniques are sufficient for processing the coverage problems, and thus, our proposed simplification techniques may not be actually necessary. To check this possibility, we conducted experiments by using the SMT-based algorithm without and with our proposed simplification techniques. The results are shown in Table V. Note again that we removed the cases of 20×20, 30×30, 40×40, and 50×50 since computation time for them is simply 0.

Table V
SMT-BASED ALGORITHM WITH AND WITHOUT SIMPLIFICATION (TIME IN SECONDS)

Case No.	With Simplification	Without Simplification
Case 5 (90×90)	0	0
Case 6 (100×100)	0.96	117.45
Case 7 (110×110)	1.58	169.83
Case 8 (140×140)	14.82	736.11
Case 8 (190×190)	186.77	5452

From the above comparison, we can observe that our proposed simplification techniques do provide help to decrease the complexity of the coverage problems, which are necessary even if we use the efficient SMT-based algorithms.

VI. CONCLUSIONS AND FUTURE WORK

We have described our proposed modeling, simplification, and SMT-based computation algorithm for the K-coverage problem in the context of wireless sensor network. Experimental results have shown the efficiency of both the simplification and the SMT-based algorithm. There are much to be done as future work. In addition to conducting more experiments on real problems to further investigate the usability of our proposed approaches, our concrete work undergoing at this moment is to further improve the generality of our modeling and computing methods/algorithms, e.g., to consider user-specified restriction condition in our model.

ACKNOWLEDGMENT

This research is partially supported by “Project for Fostering Value-Creation Advanced ICT Frontier Human Resources by Fused Industry-University Cooperation” supported by MEXT, Japan.

REFERENCES

[1] K. Sohraby, D. Minoli, and T. Znati. “Wireless Sensor Networks: Technology, Protocols, and Applications”. Wiley-Interscience, 2007.

[2] Z. Zhou, S. Das, and H. Gupta. “Connected K-Coverage Problem in Sensor Networks”. Proceedings of the 13rd International Conference on Computer Communications and Networks (ICCCN2004), IEEE press, October 2004, pp.373–378.

[3] M. Hefeeda and M. Bagheri. “Randomized K-coverage Algorithms for Dense Sensor Networks”. Proceedings of IEEE INFO-COM 2007 Minisymposium, IEEE press, May 2007, pp.2376–2380.

[4] H. M. Ammari. “On the Connected K-Coverage Problem in Heterogeneous Sensor Nets: The curse of randomness and heterogeneity”. Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS2009), June 2009, pp.265–272, IEEE press.

[5] X.-Y. Li, P.-J. Wan, and O. Frieder. “Coverage in Wireless Ad-hoc Sensor Networks”. IEEE Transactions on Computers, Vol. 52, No. 6, June 2003, pp.753–763.

[6] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. “Handbook of Satisfiability”. IOS Press, 2009, Vol. 185, Chapters 25, 26, pp. 781–885.

[7] L. de Moura and N. Bjorner. “Z3: An Efficient SMT Solver”. Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS2008), March 2008, pp. 337–340, LNCS 4963.

[8] B. Dutertre and L. de Moura. “A Fast Linear-Arithmetic Solver for DPLL(T)”. Proceedings of the 18th International Conference on Computer Aided Verification (CAV2006), August 2006, pp. 81–94, LNCS 4144.

[9] C. Barrett and C. Tinelli. “CVC3”. Proceedings of the 19th International Conference on Computer Aided Verification (CAV2007), LNCS 4590, July 2007, pp. 298–302.

[10] C. Barrett, A. Stump, and C. Tinelli. “The SMT-LIB Standard: Version 2.0”. Latest official release of Version 2.0 of the SMT-LIB standard. Online: http://www.smtlib.org/ [retrieved: June 2014]