# Heuristic Solutions to the Target Identifiability Problem in Directional Sensor Networks

Wilson M. Tan and Stephen A. Jarvis

Performance Computing and Visualisation Group

Department of Computer Science

University of Warwick, UK

**Abstract**

Existing algorithms for orienting sensors in directional sensor networks have primarily concerned themselves with the problem of maximizing the number of covered targets, assuming that target identification is a non-issue. Such an assumption however, does not hold true in all situations. In this paper, heuristic algorithms for choosing active sensors and orienting them with the goal of balancing coverage and identifiability are presented. The performance of the algorithms are verified via extensive simulations, and shown to confer increased target identifiability compared to algorithms originally designed to simply maximize the number of targets covered.

## 1 Introduction and motivation

Directional sensors are sensors whose sensing capabilities are limited within an angle range [7][8]. In comparison, an omnidirectional sensor's sensing range covers everything around it. In geometric terms, the covered area of an omnidirectional sensor is a circle centered on the sensor, while that of a directional sensor is a sector. In WSNs, nodes that are *directional* can imply that the node has directional capability in sensing and/or communication [7]. In this paper, we solely focus on the sensing capability, and thus will interchangeably use the terms 'nodes' and 'sensors'.

Examples of sensors that are inherently directional in nature include video sensors [11], ultrasonic sensors [3], and infrared sensors [13]. Acoustic sensors (or microphones) can also potentially be directional [9], although most of the existing work in WSN literature so far have utilized omnidirectional microphones [12][6].

Because the sensed region of a directional node is constrained within an angle range, of primary concern is the *total sensing coverage* provided by such a network of directional sensors. A problem that frequently needs to be solved is *'Given a number of directional sensors distributed in space, how should each sensor's sensing region be oriented such that the total sensing coverage of the network is maximized?'*.

Coverage-maximizing algorithms proposed in literature fall into two categories: those that are *target-centric* and those that are *area-centric*.

In target-centric algorithms, it is assumed that there are a finite number of *static* targets distributed in the area, and it is desired that as many as possible of such targets be covered. Sometimes, for issues of energy efficiency, it is also desired that the covering be also done with the least number of sensors possible - such a problem is formalized in [1] as the Maximum Coverage Minimum Sensors (*MCMS*) Problem.

In area-centric algorithms, there are no specific objects or targets of interest; the entire area is of sensing interest, and it is desired that as much of it is covered. In such algorithms, the problem is equivalent to minimizing the *overlap* between the sensed regions of sensors. An example of such an algorithm was presented in [16].

In this work, we solely focus on target-centric algorithms.

Most coverage algorithms focus on maximizing the number of targets covered. This is reasonable when the targets are continually being monitored, and easy target identification is built-in to the system. This is true to a certain extent for visual sensor networks: for instance, a Closed Circuit Television (CCTV) camera (being remotely watched by a person) monitoring a street intersection. However, this assumption does not hold true for all situations. For instance, if we assume that a single acoustic sensor is monitoring two possible sound sources, in general, the sensor will be able to detect that a sound was generated, but not which source generated it - at least not unless the sound generated by each source is distinct, and even then, not without further digital signal processing.

As a slight deviation from all previous studies, we shall work with the following assumptions

1. Targets generate *events*, and these events randomly occur. Events oc-

cur only one at a time, and they are brief enough that they do not overlap in time.

2. Events can be detected by the sensor if the source is within the sensor's sensed region, but the events themselves say nothing about which source generated it.

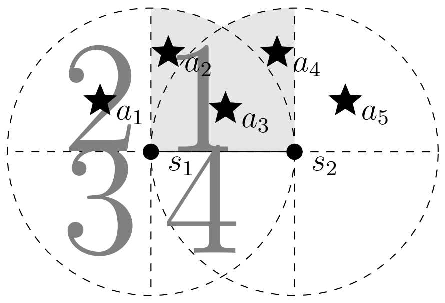


Figure 1: Diagram for the first example.

In this work, instead of just focusing on the number of targets covered, we will also concern ourselves with the identifiability of the targets or event sources.

As a motivating example, we present Figure 1, where the dots $s_1$ and $s_2$ are sensors while the stars $a_1$ - $a_5$ are the targets. The dotted regions around each sensor represent the possible covered or sensed region of each sensor. Each sensor can be oriented to be in 1 of 4 possible orientations. Orientation 1 represents the region covered by the sector from 0° to 90°, 2 covers 90° to 180°, 3 covers 180° to 270°, and 4 covers 270° to 360°. In this example, the boundaries of the sensed regions of each sensor are aligned with the cardinal directions (and with each other). The 4 possible orientations for $s_1$ are labeled in Figure 1. It must be noted however, that such an alignment is not required by the algorithms that will be discussed. In Figure 1 $s_1$ is in orientation 1, while $s_2$ is in orientation 2. With these orientations, $a_2$ is covered by $s_1$, $a_3$ is covered by $s_1$ and $s_2$, and $a_4$ is covered by $a_2$. A total of 3 targets are covered. $a_1$ and $a_5$ are not covered.

It is easy to verify that 3 is the maximum number of targets that can be covered by any network configuration. A *network configuration* is a set of active sensors, each with its corresponding orientation.

However, the configuration $s_1$ - 1 (shorthand for $s_1$ in orientation 1), $s_2$ - 2 is not the only configuration that yields 3 covered targets. $s_1$ - 2, $s_2$ - 2 also yields 3 covered targets, as will $s_1$ - 1, $s_2$ - 1. There are differences however in the identifiabilities conferred by these configurations to the targets that they cover.

Let us begin with $s_1$ - 2, $s_2$ - 2. When $a_1$ generates an event, $s_1$ will be able to detect it, and we know for certain that $a_1$ generated the event since it is the only target covered by $s_1$. When $a_3$ generates an event, $s_2$ will be able to detect it, but we are not sure whether it was $a_3$ or $a_4$ that generated the event. The best that can be done is hazard a guess with 50% probability of being correct. The same analysis holds for $s_1$ - 1, $s_2$ - 1.

Compare this with $s_1$ - 1, $s_2$ - 2. When $a_2$ generates and event, $s_1$ will be able to detect it. At first glance, it seems like we might not be able to distinguish whether it was $a_2$ or $a_3$ which generated the event since both are covered by $s_1$. However, we can know that it is *not* $a_3$, since $s_2$ did not detect anything. Hence, it must be $a_2$ which generated the event. In other words, whether a target generated an event or not can be deduced not just from which sensors detected something, but also from those that did *not* detect anything.

We call the set of sensor states (where *state* indicates whether a sensor detected something or not) which signifies a target generating an event as the target's *syndrome*. A syndrome is a tuple of values, one for each active sensor in the system, each denoting whether a sensor will detect anything upon the target generating an event. For a sensor $x$, let $s_x$ be the element for the sensor in the tuple if the sensor will detect anything, and $s_x$' if it will not. In our latest example, $a_2$ has the syndrome $s_1 s_2'$. Table 1 enumerates the syndrome for each covered target in each of the network configuration that yields 3 covered targets.

| | $s_1$ - 1, $s_2$ - 2 | $s_1$ - 2, $s_2$ - 2 | $s_1$ - 1, $s_2$ - 1 |
|---|---|---|---|
| $a_1$ | - | $s_1 s_2'$ | - |
| $a_2$ | $s_1 s_2'$ | - | $s_1 s_2'$ |
| $a_3$ | $s_1 s_2$ | $s_1' s_2$ | $s_1 s_2'$ |
| $a_4$ | $s_1' s_2$ | $s_1' s_2$ | - |
| $a_5$ | - | - | $s_1' s_2$ |

Table 1: Configurations that yield 3 covered targets, and their resulting syndromes.

In Table 1, we can clearly see why the configuration $s_1$-1, $s_2$-2 affords better target identifiability: in that configuration, each covered target has its own syndrome. In comparison, in the other two configurations, two targets have to share a single syndrome, resulting in ambiguity when identifying their events.

It must be noted that some aspects of Figure 1 do not hold true in other situations.

Firstly, sometimes, assigning a single syndrome for each target is simply impossible. Nevertheless, it is conceivable that even in such situations, it is desirable to minimize the ambiguity between events as much as possible.

Secondly, in other networks (especially those that are *underprovisioned*, meaning there are few sensors relative to targets), it is possible that improved target identifiability will come at the cost of less targets covered. The acceptable trade-off between the number of targets covered and their identifiability will vary from one application to the next.

Another use of the concept of identifiability is in *overprovisioned* networks - that is, networks where there is a surplus in the number of sensors, and even after the maximum possible number of targets has been covered, there are still sensors that are not covering anything. In previous studies, the extra sensors are used in extending the network lifetime: sensors form *cover sets* that take turns covering the targets [2]. Clearly, another possible use for such extra sensors is in increasing the identifiability of targets.

This paper makes three contributions: firstly, the introduction of the concept of *syndromes*; secondly, the definition of the Maximum Target Identifiability-Aware Utility with Minimum Sensors (MTIAUMS) problem; and finally, six heuristic algorithms that determine network configurations that strike a balance between identifiability and the number of targets covered.

This paper is structured as follows. The notations utilized are discussed in Section 2. The problem is formally defined, and its NP-hardness proven in Section 3. Centralized heuristic algorithms are presented in Section 4, while distributed heuristic algorithms are presented in Section 5. The methodology used to test the algorithms, and the results of the simulations, are presented in Section 6. A discussion of related work follows in Section 7, while Section 8 concludes the paper.

## 2 Notations

The following notations are used in this paper

- $M$: number of targets.
- $a_i$: a specific target, $1 \leq i \leq M$.
- $A$: the set of all targets $A = \{a_1, a_2, ... a_M\}$.
- $N$: number of nodes.
- $s_i$: a specific node, $1 \leq i \leq N$.
- $S$: the set of all nodes $S = \{s_1, s_2, ... s_N\}$.
- $r$: the sensing radius; a target is said to be covered by one of the node's orientations if the distance between the target and the node is less than or equal to the sensing radius.
- $W$: number of orientations with which each node can work with.
- $\phi_{i,j}$: set of targets covered by node $i$ when it is working with orientation $j$, $1 \leq i \leq N$, $0 \leq j \leq W$. Note that we allow $j$ to take on the value of 0: this indicates that the node's sensing mechanism is inactive, thus $\phi_{i,0} = \{\}$, $\forall$ $i$.
- $\Phi_i$: set of all targets within range of $s_i$, *regardless* of orientation; $\Phi_i = \{\cup\ \phi_{i,j} \mid 1 \leq j \leq W\}$.

- $Q_i$: set of nodes comprised of one-hop neighbours of $s_i$.
- $\Phi'_i$: set of all targets within range of $s_i$'s one-hop neighbours, sans those also seen by $s_i$; $\Phi'_i = \{\cup \, \Phi_j \setminus \Phi_i \mid s_j \in Q_i\}$.
- $\alpha$: user-defined parameter indicating the desired trade-off between the number of targets covered and their identifiability; higher value indicates more preference for number of targets covered; $0 \leq \alpha \leq 1$.
- $Z$: network configuration, set of active sensors and corresponding orientations; set of (i, j) pairs, where i $\in S$, 1 $\leq$ j $\leq$ W.
- $u_i$: target utility for a given $Z$, $1 \leq i \leq M$.
- $U$: sum of all target utilities for a given $Z$.

## 2.1 Coverage by a specific orientation

Once a target is ascertained to be within a node's sensing radius, the specific orientation of the node covering the target can be determined by

1. Let $D$ be the node, $E$ the target, and $F$ a point immediately to the right of the node. From these points, define the vectors $DE$ and $DF$ (which should be parallel to the x-axis).
2. Compute the angle $\theta$ using

$$\theta = \frac{DE \cdot DF}{|DE||DF|} \tag{1}$$

3. In this study, we assume that the orientation numbers are assigned in a counter-clockwise fashion. The orientation numbers are assigned starting from a reference axis, which in this study is coincidental with the x-axis. Let the offset (in degrees) of the reference axis be referred to as $ref_{offset}$. The specific orientation $n$ which covers a certain target can then be determined using

$$(\frac{360°}{W} + ref_{offset}) \times (n-1) \leq \theta < (\frac{360°}{W} + ref_{offset}) \times n \qquad (2)$$

## 3 Problem definition

To give consideration to the fact that not all setups will be like that in Figure 1 (where the network configuration which maximizes the number of targets covered also maximizes the number of syndromes), we first introduce the concept of *target utility*, $u_i$

$$u_i = \alpha + (1-\alpha)(certainty_i) \qquad (3)$$

$u_i$ will depend on the network configuration Z. $\alpha$ is a parameter (with value between 0 and 1, inclusive) defined by the user, which indicates the desired trade-off between the number of targets covered and identifiability: a higher $\alpha$ indicates that the number of covered targets is more important than the number of syndromes in the system, a lower $\alpha$ value indicates the reverse. $certainty_i$ is the level of certainty with which a target can be identified when it generates an event. Like $u_i$, it is dependent on Z (it is the reason why $u_i$

is dependent on Z). It can be defined as

$$certainty_i = \frac{1}{\text{\# of targets sharing the same syndrome as } a_i} \tag{4}$$

Building on the concept of target utility, we define system utility, $U$:

$$U = \sum_{i=1}^{M} u_i \tag{5}$$

Another definition for $U$ will be

$$\begin{aligned} U = \alpha(\text{number of targets covered}) + \\ (1 - \alpha)(\text{number of syndromes in the system}) \end{aligned} \tag{6}$$

The second term holds because if all the certainties of all covered targets are summed, one will end up with the number of syndromes in the system. Unless explicitly stated, references to 'utility' in this paper must be taken to mean 'system utility'.

Our goal is to provide coverage to targets, maximizing the system utility as defined by Equation 6, while activating as few sensors as possible. We call this problem the Maximum Target Identifiability-Aware Utility with Minimum Sensors (MTIAUMS) problem. A more formal statement of the problem will be: *Given a set of targets A and a set of sensors S (each with W possible sensing orientations), find a network configuration Z (consisting of a set of active sensors, along with their corresponding orientations), such that the resulting system utility U is maximized and the cardinality of Z is* *minimized.*

It must be noted that the computed system utility is affected by the ratio of targets to sensors, and their densities (targets per unit area, sensors per unit area). If there are significantly more targets than sensors (several orders of magnitude higher, for instance), it is actually possible for heuristic algorithms that only aim to maximize the number of covered targets to attain a higher system utility than our algorithms. While the algorithms will still confer better identifiability to covered targets in such a situation, that can possibly be hidden by the fact that there are significantly more possible targets that can be covered (first term, Equation 6) than there are possible syndromes that can be generated (second term, Equation 6). In such cases, it might be helpful to take the metric $\frac{targets}{syndrome}$ into account when evaluating solutions. In this work, we will deal with cases wherein the number of targets is comparable to the number of sensors.

In this study, we propose heuristic solutions to the MTIAUMS problem. A Heuristic solution is useful since the problem is NP-hard (this will be proven in the next subsection).

## 3.1 NP-hardness of the problem

We prove the NP-hardness of the problem through the *'Proof by Restriction'* method [5]. When $\alpha = 1$, the MTIAUMS problem becomes the Maximum Coverage with Minimum Sensors (MCMS) problem defined in [1]. The MCMS problem therefore is a special case of the MTIAUMS problem. It is proven in [1] that the MCMS problem is NP-hard - therefore, the MTIAUMS problem is also NP-hard.

# 4 Centralized algorithms

Centralized algorithms are presented first. Syndrome counting is discussed in Section 4.1. Our first heuristic algorithm, TIA-CGA, is presented in Section 4.2. Two other centralized algorithms are discussed in Section 4.3.

## 4.1 Counting syndromes

The capability to count the number of syndromes associated with a specific network configuration is of primary importance to algorithms that will be presented later. To count the covered targets and to evaluate the level of identifiability afforded by a given network configuration (or set of sensor orientations), we introduce the concept of *coverage matrix*. A coverage matrix is formed from a given $Z$. A coverage matrix has its rows indexed by the members of $A$ and its columns indexed by the members of $S$. Let $z_{i,j}$ be a member of a coverage matrix

$$z_{i,j} = \begin{cases} 1 \text{ if } i \in \phi_{j,k} \text{ s.t. (j, k)} \in Z; \\ 0 \text{ otherwise, inc. (j, k)} \notin Z \ \forall \text{ k, } 1 \leq \text{k} \leq \text{W.} \end{cases} \tag{7}$$

To count the number of syndromes in a given configuration, we use Algorithm 1. One of the inputs to Algorithm 1 is an array called *covered* whose element is 1 if the index corresponding to the target is covered in the coverage matrix (at least one non-zero value in the corresponding row). The array *covered* can be easily generated along with the coverage matrix via

Equation 7.

**Algorithm 1** Syndrome counting algorithm

```
1: Inputs: coverage matrix with elements $z_{i,j}$ (generated using Equation 7);
   an array called covered whose element is 1 if the index corresponding to
   the target is covered in the coverage matrix (at least one non-zero value
   in the corresponding row)
2: Output: syndromes - the number of syndromes in the system when
   solution embodied by the coverage matrix is applied
3: $syndromes \leftarrow 0$
4: processed($i$) $\leftarrow 0\ \forall i \in A$
5: for $1 \leq i \leq M$ do
6:     if (processed($i$) == 0) && (covered($i$) == 1) then
7:         $syndromes \leftarrow syndromes + 1$
8:         for $i + 1 \leq j \leq M$ do
9:             if (processed($j$) == 0) && (covered($j$) == 1) then
10:                same $\leftarrow 1$
11:                for $1 \leq k \leq N$ do
12:                    if $z_{i,k}$ != $z_{j,k}$ then
13:                        same $\leftarrow 0$
14:                    end if
15:                end for
16:                if same == 1 then
17:                    processed($j$) $\leftarrow 1$
18:                end if
19:            end if
20:        end for
21:        processed($i$) $\leftarrow 1$
22:    end if
23: end for
```

Algorithm 1 works by comparing the syndromes of each target in a pair-wise fashion. In the coverage matrix, the syndrome of a target is represented by the values in the row corresponding to the target number. For example, target $a_1$'s syndrome will be the concatenation of the values in row 1: $z_{1,j}, 0 \leq j \leq N$. Each target, unless already 'processed', becomes a reference at some point in the algorithm. The algorithm chooses the reference row sequentially (Algorithm 1, Line 5). The reference row is then compared with rows with higher numbers (Algorithm 1, Line 8). The actual element-by-element comparison of the syndromes happens in Algorithm 1 Lines 11-15. If a row has exactly the same values as the reference row, it is marked as already processed (Algorithm 1, Line 21) and loses the chance to become a reference row itself. It will also not be eligible for comparison with any other future reference rows. The successful selection of a new reference row effectively represents the 'discovery' of a new syndrome and the *syndromes* variable is incremented by 1 (Algorithm 1, Line 7).

The maximum number of target-target comparisons that can possibly be performed in the process of counting syndromes is equivalent to the number of pairwise combination of targets (Algorithm 1, Lines 6 and 9), or $\frac{M!}{2!(M-2)!}$. Within a target-target comparison, it is checked whether or not both targets are covered by each sensor (N, Algorithm 1, Line 11). Therefore, the computational complexity of Algorithm 1 is $\frac{NM!}{2!(M-2)!}$.

## 4.2 Target Identifiability-Aware Centralized Greedy Algorithm

The Target Identifiability-Aware Centralized Greedy Algorithm (TIA-CGA) is a heuristic algorithm that produces network configurations that take into account both the number of targets covered and the identifiability of those targets. The TIA-CGA is primarily derived from the Centralized Greedy Algorithm (CGA) presented in [1]. The CGA is a greedy algorithm which at each stage of the solution computation chooses the sensor-orientation pair

**Algorithm 2** Target Identifiability-Aware Centralized Greedy Algorithm

```
 1: Inputs: $A$; $S$; $\alpha$; $\phi_{i,j}$s
 2: Output: $Z$ - network configuration, a set of (ID of active node, configuration of active node) pairs
 3: $Z \leftarrow \emptyset$
 4: $V \leftarrow A$                ▷ at the end, will contain all uncovered targets
 5: $Y \leftarrow S$                ▷ at the end, will contain all inactive nodes
 6: new_utility ← 0
 7: while 1 do
 8:     old_utility ← new_utility
 9:     for (i, j) s.t. $s_i \in Y$, $0 \leq j \leq W$ do
10:         $TempNodeSet \leftarrow \emptyset$
11:         $TempNodeSet \leftarrow Z \cup \{$(i, j)$\}$
12:         $TempCovMatrix$ ← coverage matrix generated from $TempNodeSet$    ▷ generate using Equation 7
13:         $Q_{i,j}$ ← SyndromeCount($TempCovMatrix$, associated *covered* array)    ▷ compute number of syndromes using Algorithm 1
14:         $U_{i,j} \leftarrow (\alpha) \times (|\phi_{i,j} \cap V|) + (1 - \alpha) \times (Q_{i,j})$
15:     end for
16:     (i, j) ← $\arg\max_{s_i \in Y, 0 \leq j \leq W} U_{i,j}$
17:     new_utility ← maximum $Ui,j$ determined in previous step
18:     if new_utility - old_utility ≤ 0 then
19:         break        ▷ utility does not increase anymore, algorithm ends
20:     else
21:         $Z \leftarrow Z \cup \{$(i, j)$\}$
22:         $V \leftarrow V \setminus \phi_{i,j}$
23:         $Y \leftarrow Y \setminus \{(s_i)\}$
24:     end if
25: end while
```

which adds the highest number of targets to those already covered. The CGA main loop can be seen in Algorithm 3, Lines 7-22 sans Lines 8-11 and 14. The main difference between the TIA-CGA and the CGA is that in an iteration, instead of choosing the sensor-orientation pair which adds the most number of newly covered targets, the TIA-CGA chooses the pair which results in the greatest additional system utility to the system. To do this, for each remaining unchosen sensor-orientation pair, it computes the number of additional targets that will be covered if the pair was chosen next, and the number of syndromes that the system will have (Algorithm 2, Lines 10-13). These two values are weighted by the priority factors $\alpha$ and 1 - $\alpha$, respectively, and then added together (Algorithm 2, Line 14). Algorithm 2 Line 16 chooses the pair which adds the greatest system utility to the system. The algorithm will stop when the best pair found no longer improves the system utility (Algorithm 2, Lines 18-19); otherwise, the pair is added to the solution (Algorithm 2, Line 21), the sensor is removed from the set of sensors viable for selection in the next iteration (Algorithm 2, Line 23), and the loop begins anew.

To derive the computational complexity of Algorithm 2, it must be noted that in each of its iteration, it needs two pieces of information to choose the sensor-orientation pair: the number of additional targets each sensor-orientation pair will cover (if chosen), and the number of syndromes each sensor-orientation pair will add to the system (if chosen).

The number of targets covered can be determined in MNW steps, in keeping with the value derived in [1].

For the number of syndromes, there are NW sensor-orientation pairs to evaluate. In each evaluation, a coverage matrix is generated (Algorithm 2, Line 13, Equation 7). The coverage matrix has MN elements, so it can be assumed that it will take MN steps to evaluate. The coverage matrix is processed by Algorithm 1, with complexity $\frac{NM!}{2(M-2)!}$. Therefore, in each iteration of the main loop of Algorithm 2, the complexity due to syndrome counting will be $NW(MN + \frac{NM!}{2(M-2)!})$.

The evaluation of the system utility and the choosing of the sensor-sensor configuration pair can be done in NW steps. There are a maximum N iterations of the main loop of Algorithm 2. Therefore, the overall complexity of Algorithm 2 is $N(MNW + NW(MN + \frac{NM!}{2(M-2)!}) + NW)$.

## 4.3 2-stage algorithms

The next algorithms that will be introduced are the 2-stage algorithms. One of these algorithms is based on the Centralized Force-based Algorithm (CFA), so a short introduction on CFA is in order.

The CFA is an alternative to CGA, proposed in [10]. Like CGA, CFA aims to produce network configurations that maximize the number of targets covered, and does so greedily. Unlike CGA however, when building the solution, it does not solely rely on the number of targets that will be covered by each sensor-sensor configuration pair. Instead, at each step of the computation, it chooses on the basis of the 'force' exerted by a sensor configuration (or direction) *on* the sensor. This force is defined as the ratio of the targets covered by a specific sensor configuration to the total number of targets covered by the sensor (Equation 8).

$$F_{i,j} = \frac{|\phi_{i,j}|}{|\Phi_i|} \tag{8}$$

The 2-stage algorithms basically apply CGA or CFA first to the problem and then attempt to increase the number of syndromes in the system by greedily using the sensors left unselected. When the first stage is CGA, the algorithm is called 2-stage Target Identifiability-Aware Centralized Greedy

**Algorithm 3** 2-stage Target Identifiability-Aware Greedy Algorithm

```
1: Inputs: $A$; $S$; $\phi_{i,j}$s; CFAStage1 - binary variable, 1 if desired first stage
   is CFA, 0 if CGA
2: Output: $Z$ - network configuration, a set of (ID of active node, configuration of active node) pairs
3: $Z \leftarrow \emptyset$
4: $V \leftarrow A$                    ▷ at the end, will contain all uncovered targets
5: $Y \leftarrow S$                    ▷ at the end, will contain all inactive nodes
6: new_syndrome_count ← 0
7: while 1 do
8:     if CFAStage1 == 1 then                                  ▷ First stage is CFA
9:         Compute $F_{i,j} = \frac{|\phi_{i,j} \cap V|}{|\Phi_i \cap V|}$ $\forall$ $s_i \in Y$, $0 \leq j \leq W$
10:        (i, j) ← $\arg\max_{s_i \in Y, 0 \leq j \leq W}$ $F_{i,j}$
11:    else                                                    ▷ First stage is CGA
12:        Compute $|\phi_{i,j} \cap V|$ $\forall$ $s_i \in Y$, $0 \leq j \leq W$
13:        (i, j) ← $\arg\max_{s_i \in Y, 0 \leq j \leq W}$ $|\phi_{i,j} \cap V|$
14:    end if
15:    if $|\phi_{i,j} \cap V|$ == 0 then
16:        break          ▷ nothing new can be covered anymore, stage 1 ends
17:    else
18:        $Z \leftarrow Z \cup \{$(i, j)$\}$
19:        $V \leftarrow V \setminus \phi_{i,j}$
20:        $Y \leftarrow Y \setminus \{(s_i)\}$
21:    end if
22: end while
```

Algorithm (2S-CGA), and when the first stage is CFA, the algorithm is called 2-stage Target Identifiability-Aware Centralized Force-based Algorithm (2S-CFA).

In Algorithm 3, the first stage can be found in Lines 7-22. As previously mentioned, the first stage can either be CGA or CFA. In the interest of saving space, Algorithm 3 is made to be capable of using both CGA and CFA, with the choice of which algorithm to use now dependent on the binary variable *CFAStage1*, which is assumed to be an algorithm input. *CFAStage1* having the value of *1* denotes that CFA should be used (Algorithm 3, Lines 9-10),

```
23: while 1 do
24:     old_syndrome_count ← new_syndrome_count
25:     for (i, j) s.t. $s_i \in Y$, $0 \leq j \leq W$ do
26:         $TempNodeSet \leftarrow \emptyset$
27:         $TempNodeSet \leftarrow Z \cup \{$(i, j)$\}$
28:         $TempCovMatrix$ ← coverage matrix generated from $TempNodeSet$    ▷ generate using Equation 7
29:         $Q_{i,j}$ ← SyndromeCount($TempCovMatrix$, associated *covered* array)    ▷ compute number of syndromes using Algorithm 1
30:     end for
31:     (i, j) ← $\arg\max_{s_i \in Y, 0 \leq j \leq W} Qi, j$
32:     new_syndrome_count ← maximum $Qi, j$ determined in previous step
33:     if new_syndrome_count - old_syndrome_count $\leq 0$ then
34:         break    ▷ nothing can be covered that will increase the number of syndromes anymore, stage 2 and algorithm ends
35:     else
36:         $Z \leftarrow Z \cup \{$(i, j)$\}$
37:         $Y \leftarrow Y \setminus \{(s_i)\}$
38:     end if
39: end while
```

while *CFAStage1* having the value of *0* denotes that CGA should be used (Algorithm 3, Lines 12-13). We call this algorithm, which represents both 2-stage algorithms, the *2-stage Target Identifiability-Aware Greedy Algorithm.*

The second stage can be found in Algorithm 3 Lines 23-39. In each iteration of the loop, the number of additional syndromes that can possibly be added by each remaining sensor-orientation pair is computed (Algorithm 3, Lines 25-30). If the most that can be added by any sensor-orientation pair is 0 (or negative), the stage and the algorithm will end (Algorithm 3, Lines 33-34); otherwise, the sensor-orientation pair is added to the solution (Algorithm 3, Line 36), the sensor is removed from the set of viable or inactive sensors (Algorithm 3, Line 37), and the loop begins anew.

It must be noted that strictly speaking, the two 2-stage algorithms are not exact alternatives to the TIA-CGA. TIA-CGA always aims to maximize the system utility, thus taking into account both targets covered and syndromes generated at each step. The 2-stage algorithms only take the syndromes into account after all targets that can possibly be covered are covered. In situations where all sensors are utilized for covering targets (no leftovers) - the 2-stage algorithms degenerate into CGA or CFA (depending on which version is being used).

The complexity of the first stage of Algorithm 3 follows that of CGA or CFA: $N_1(MN_1W + N_1W)$. $N_1$ is the number of sensors that will be chosen by the first step. If all sensors are utilized in covering targets (no leftovers), $N_1 = N$, and the complexity of Algorithm 3 becomes the same as that of CGA or CFA.

The complexity of the second stage is $N_2(N_2W(MN_2 + \frac{N_2M!}{2(M-2)!}) + N_2W)$. This is the same as that of Algorithm 2, but without the component for counting covered targets. $N_2$ is the number of unselected sensors left after the first stage. The sum of $N_1$ and $N_2$ cannot exceed N (i.e., $N_1 + N_2 \leq N$). The complexity of Algorithm 3 is then $N_1(MN_1W + N_1W) + N_2(N_2W(MN_2 + \frac{N_2M!}{2(M-2)!}) + N_2W)$, $N_1 + N_2 \leq N$. The majority of the complexity of Algorithm 2 stems from the computation necessary to count syndromes - by sparing some (if not most) sensors from such a step (by splitting $N$ into $N_1$ and $N_2$), the two 2-stage algorithms usually end up with shorter runtimes than Algorithm 2. This will be empirically verified in Section 6.

# 5 Distributed algorithms

Communication costs (in terms of energy and time) will make the transmission of data between the nodes and the base station (which will run the algorithm in a centralized solution) prohibitive as the network grows in size - hence, a distributed solution is at times more practical than a centralized one. Relevant data structures to the distributed algorithms will first be discussed in Section 5.1. Our first distributed algorithm, 2S-TIA-DGA, is introduced in Section 5.2 and its operation discussed in Section 5.3. A comparison with two algorithms for solving MCMS will be given in Section 5.4, and its time complexity is discussed in Section 5.5. 2-stage heuristic algorithms are introduced and discussed in Section 5.6.

## 5.1 Data structures

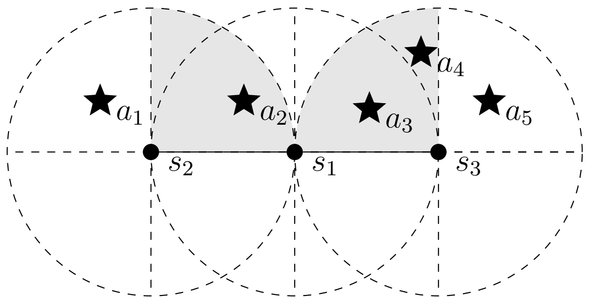


Figure 2: Diagram for the fourth example.

We first introduce two data structures on which the algorithm will rely on. The data structures can be found in each node (as each node will run the algorithm).

The first data structure is the *seen_targets* matrix. The *seen_targets* matrix has its rows indexed by the members of $\Phi_i$, and its columns indexed by the members of $Q_i$. A member of the *seen_targets* matrix (let us call it $z_{j,k}$) is defined by

$$z_{j,k} = \begin{cases} 1 \text{ if } s_k\text{'s current orientation covers } a_j; \\ 0 \text{ otherwise.} \end{cases} \tag{9}$$

Assume that we have the setup in Figure 2, and that we are taking the point of view of $s_1$ (which has not yet decided on its orientation). Also assume that $s_1$ has a lower priority than either $s_2$ or $s_3$ (priorities will be discussed later) and that it has just received a protocol message from each of the nodes, informing it of their chosen orientations. The *seen_targets* matrix of $s_1$ will then be:

| | | $Q_1$ | |
|---|---|---|---|
| | | $s_2$ | $s_3$ |
| $\Phi_1$ | $a_2$ | 1 | 0 |
| | $a_3$ | 0 | 1 |

The second data structure is the *unseen_targets* matrix. Like the *seen_targets* matrix, it has its columns indexed by $Q_i$. The rows on the other hand, are indexed by $\Phi'_i$. The rows can be built dynamically (added as the node receives messages from its neighbors), or be built at an initial information exchange stage. The members of the *unseen_targets* matrix are defined in the same way as the members of the *seen_targets* matrix (Equation 9). Continuing with our example, if we take the point of view of $s_1$, the *unseen_targets* matrix will then be:

|  |  | $Q_1$ |  |
|---|---|---|---|
|  |  | $s_2$ | $s_3$ |
| $\Phi'_1$ | $a_1$ | 0 | 0 |
|  | $a_4$ | 0 | 1 |
|  | $a_5$ | 0 | 0 |

## 5.2 Algorithm overview

The algorithm Target Identifiability-Aware Distributed Greedy Algorithm (TIA-DGA) is shown in Algorithm 4. In TIA-DGA, each node chooses the orientation that has the highest total system utility and then announces its choice to its neighbouring nodes. To avoid double counting, a node will only count a certain target if it has not been covered yet by a node with higher priority. An order of priority between the nodes (or at least within neighbouring nodes) is necessary to ensure that the distributed algorithm will eventually terminate [1]. Priorities in TIA-DGA are assigned by the number of targets that a node can see, or $|\Phi_i|$. The idea behind this is that nodes that can cover a large number of targets (and thus, have higher priorities) will take care of covering targets while nodes that do not have as many will then increase the number of syndromes. Ties are broken using the node ID numbers (which are assumed to be unique throughout the network).

## 5.3 Algorithm operation

Upon being initialized, a node will set its priority to the number of targets that it covers in all orientations (Algorithm 4, Line 3). It will then proceed to execute the *SetAndAnnounceBestOrientation* function (Algorithm 5).

**Algorithm 4** Target Identifiability-Aware Distributed Greedy Algorithm

```
1: Inputs: $\alpha$; $\phi_{i,j}$s; W; $Q_i$; $\Phi_i$; $\Phi'_i$;
2: Output: $j\_chosen$ - chosen orientation for the sensor node
3: Set priority $PR_i = |\Phi_i|$
4: SetAndAnnounceBestOrientation();
5: while a protocol message is received from sensor n $\in Q_i$ s.t. $PR_n \geq PR_i$ do
6:     if $PR_n == PR_i$ then
7:         if $n > i$ then
8:             continue;
9:         else
10:            stop processing message, throw it away
11:        end if
12:    end if
13:    Based on the content of the message and Equation 9, update seen_targets matrix and unseen_targets matrix
14:    SetAndAnnounceBestOrientation();
15: end while
```

For each of the possible orientations, the number of targets that can possibly be acquired can be determined by counting the number of relevant rows in the *seen_targets* matrix with all-zero entries (Algorithm 5, Line 3): because the relevant rows are those whose index belong to $\phi_{i,j}$, each row is representative of a target that can be seen by the sensor; the row having all-zero entries signifies that it has not been covered yet by any other sensor. The algorithm will then determine how many additional syndromes it can create by choosing the orientation under consideration. This can be done by counting the unique row patterns (we define *pattern* here as the concatenation of the column-by-column values) in the *seen_targets* matrix that have at least one equivalent row pattern in the *unseen_targets* matrix (Algorithm 5, Line 4). The idea behind this is that a new syndrome is actually assigned to a target whenever a node chooses to cover it. However, it is possible that no

**Algorithm 5** Algorithm for setting and announcing best orientation (SetAndAnnounceBestOrientation)

```
1: j_old ← j_chosen
2: for j s.t. $1 \leq j \leq W$ do
3:     $P_j$ ← number of rows in *seen_targets* matrix with all 0s, s.t. target corresponding to the row $\in \phi_{i,j}$
4:     $R_j$ ← number of *unique* rows in *seen_targets* matrix that can also be found in *unseen_targets* matrix, s.t. target corresponding to the *seen_targets* matrix row $\in \phi_{i,j}$
5:     if $P_j$ != 0 then
6:         $R_j \leftarrow R_j + 1$
7:     end if
8:     $U_j \leftarrow (\alpha) \times (P_j) + (1 - \alpha) \times (R_j)$
9: end for
10: if $\arg\max_{1 \leq j \leq W} U_j == 0$ then
11:     j_chosen ← 0
12: else
13:     j_chosen ← $\arg\max_{1 \leq j \leq W} U_j$
14: end if
15: if j_chosen != j_old then
16:     Set orientation to j_chosen
17:     Send a protocol message including priority and $\phi_{i,j\_chosen}$
18: end if
```

new syndrome is added to the system as the assigned syndrome can simply be a *redefinition* of the previous syndrome. The *addition* of a new syndrome to the system will only happen if and only if

1. There are other targets that share the target's current syndrome, and
2. Those targets will *not* share the new syndrome that will be assigned.

Going back to the previous example, should $s_1$ choose orientation 1, it will add a new syndrome to the system: $a_3$ will be given a syndrome different from that of $a_4$, which it used to share the same syndrome with. Should

$s_1$ choose orientation 2, the syndrome of $a_2$ will be *redefined*, but no new syndrome will be added to the system, since only $a_2$ uses the old syndrome.

After the number of possibly newly covered targets and possibly newly added syndromes are counted, they are then weighted and added, resulting in the utility (Algorithm 5, Line 8) for the orientation. This is done for each possible orientation.

If there is no utility that can be had from choosing any orientation, the node will turn the sensing mechanism off (Algorithm 5, Line 11). It is assumed however, that even with the sensing mechanism turned off, the node can still send and receive protocol messages. If there is at least one orientation with a non-zero additional utility, the orientation with the highest additional utility is chosen (Algorithm 5, Line 13), and the orientation set to it (Algorithm 5, Line 16). The node will then broadcast a protocol message containing its priority, and the targets that it has chosen to cover with its orientation (Algorithm 5, Line 17). To minimize congestion and save energy, a protocol message is only sent if the node changed its chosen orientation (Algorithm 5, Line 15). Take note that the protocol message will contain $\phi_{i,j}$, meaning, it contains *all* targets covered by the chosen orientation, even those already covered by another node.

Assuming that the node is still in its initialization stage, the matrices *seen_targets* and *unseen_targets* will not contain anything (basically, this is the same as the node assuming that all of its neighbours are turned off).

After the initialization stage, the node will begin receiving messages from its neighbours. After determining that a message should be processed (i.e., sender has higher priority than the node), the node will update the matrices *seen_targets* and *unseen_targets* with the contents of the message (Algorithm 4, Line 13), and the function *SetAndAnnounceBestOrientation* is executed.

## 5.4 Comparison with DGA and DFA

We compare our algorithm with the Distributed Greedy Algorithm (DGA) [1] and the Distributed Force-based Algorithm (DFA) [10], two distributed heuristic algorithms designed to maximize the number of targets covered by a network of directional sensors.

In DGA, the priorities are randomly assigned, and the nodes choose the orientation with the highest number of targets not covered by neighbouring nodes with higher priorities.

In DFA, the priorities are determined by the highest 'force' the node experiences in any orientation, with the force in a given orientation defined by Equation 8. If the force values of two neighbouring nodes are similar, the number of targets covered are then compared; if those are also equal, node ID values are used to ultimately break the tie. Like in DGA, the nodes choose the orientation with the highest number of targets not covered by neighbouring nodes with higher priorities.

## 5.5 Time complexity

Similar to DGA (and DFA), nodes in TIA-DGA have definite priorities, ensuring termination in finite time. Another implication of this is that it also shares the two algorithms' time complexity in the worst case (which happens when sensors reach their final states one-by-one in the order of their priorities), which is O($n^2$) [1].

## 5.6 2-stage distributed algorithms

We also introduce 2-stage algorithms for solving the MTIAUMS problem in a distributed way. Similar to their centralized counterparts, the idea behind the distributed 2-stage algorithms is to cover the targets first using distributed heuristic algorithms designed for solving the MCMS problem (first stage). The number of syndromes is then increased using the remaining sensors (second stage). We introduce two 2-stage algorithms: 2-stage Distributed Greedy Algorithm (2S-DGA) and 2-stage Distributed Force-based Algorithm (2S-DFA). The difference between the two lies in the MCMS-solving algorithm used in the first stage: 2S-DGA uses DGA for its first stage, while 2S-DFA uses DFA for its first stage. An algorithm both representing 2S-DGA and 2S-DFA is presented in Algorithm 6.

The primary difference between DGA and DFA lies with how the priorities of the nodes are determined. DGA randomly assigns priorities, assuming that the random numbers assigned are unique among the nodes (Algorithm 6, Line 5). DFA assigns priorities based on the orientation with the most number of targets covered (Algorithm 6, Lines 7-8). Since the priority values assigned by DFA are not necessarily unique, a mechanism for tie-breaking is needed (Algorithm 6, Lines 13-23). Algorithm 6 decides on the priority to assign using the binary variable *DFAStage1* (Algorithm 6, Line 4), which is assumed to be a user input. A value of 0 for *DFAStage1* denotes that the algorithm being represented is 2S-DGA, while a value of 1 denotes that the algorithm being represented is 2S-DFA. The stage the algorithm is in is denoted by the variable $CurrentState_i$. The first stage of the algorithm can be found in Algorithm 6 Lines 11-26 while the second stage can be found in Algorithm 6 Lines 29-43.

In the first stage, a message received by the node is processed as long as the message is from a node with a higher priority or while the supporting function *TimerFired* returns $False$ (Algorithm 6, Line 11). *TimerFired* is a function which returns $False$ when the timer has not fired yet, and $True$ after the timer has fired. We assume that the timer is a countdown timer which fires after a certain amount of time has elapsed after the timer is started. The timer is started (and the amount of time the timer will wait for specified) using the supporting function *StartTimer*. The timer is simultaneously stopped and reset using the supporting function *StopTimer*. The three supporting functions are tabulated and described in Table 2.

The messages processed in the first stage cause the data structures *seen_targets* and *unseen_targets* to be updated (Algorithm 6, Line 24), and the function *TwoStageSetAndAnnounceBestOrientation* (Algorithm 7) to be called. Similar to its counterpart for TIA-DGA, *TwoStageSetAndAnnounceBestOrientation* chooses the orientation with the highest additional utility. Unlike Algorithm 5 however, Algorithm 7 defines utility differently. While Algorithm 5 defines utility as a weighted sum of the number of covered targets and the number of syndromes (Algorithm 5, Line 8) utility for Algorithm 7 in the first stage is solely determined by the number of covered targets (Algorithm 7, Line 9). If the chosen orientation of the node is changed, the choice

| Function | Input parameters | Return value | Description |
|---|---|---|---|
| StartTimer | time before timer fires | none | starts the timer |
| StopTimer | none | none | stops and resets the timer |
| TimerFired | none | binary | returns $True$ if timer has fired; returns $False$ if otherwise |

Table 2: Supporting functions for Algorithm 6 and Algorithm 7

is broadcast (Algorithm 7, Line 21). At the end of Algorithm 7, the timer is started with the supporting function *StartTimer*, and the timer duration is specified to be equivalent to the value *TIMEOUT*. *TIMEOUT* is an amount of time sufficient for the entire network to be covered by DGA or DFA. The timer is stopped or reset every time a message is received (Algorithm 6, Line 12), assuming of course that the timer has not fired first. The situation of a message from a neighbouring node in Stage 1 arriving after the node's timer has fired or elapsed should *never* happen if the variable *TIMEOUT* was properly set.

The firing of the timer indicates that that the first stage is over and the algorithm now moves to the second stage (Algorithm 6, Line 27). It must be noted that nodes that got an orientation assignment from the first stage will no longer participate in the second stage (Algorithm 6, Line 28).

The operation of the second stage is different from that of the first stage primarily in that the timer no longer plays a role in the admission of packets or messages (Algorithm 6, Line 29) and Algorithm 7 now defines utility based on the number of syndromes generated (Algorithm 7, Line 11).

**Algorithm 6** 2-stage Target Identifiability-Aware Distributed Generic Algorithm

```
 1: Inputs: $\alpha$; $\phi_{i,j}$s; W; $Q_i$; $\Phi_i$; $\Phi'_i$; DFAStage1 - binary variable, 1 if desired
    first stage is is DFA, 0 if DGA
 2: Output: $j\_chosen$ - chosen orientation for the sensor node
 3: Set $CurrentState_i = 1$
 4: if $DFAStage1 == 0$ then
 5:     Set priority $PR_i$ = unique random number
 6: else
 7:     j_highest $\leftarrow \arg\max_{1 \leq j \leq W} |\phi_{i,j}|$
 8:     Set priority $PR_i = |\phi_{i,j\_highest}|$
 9: end if
10: TwoStageSetAndAnnounceBestOrientation();
11: while (TimerFired() $== FALSE$) $AND$ (a protocol message is received
    from sensor n $\in Q_i$ s.t. $PR_n \geq PR_i$) do
12:     TimerStop();
13:     if $PR_n == PR_i$ then
14:         if $\Phi_i > \Phi_n$ then
15:             continue;
16:         else
17:             if $n > i$ then
18:                 continue;
19:             else
20:                 stop processing message, throw it away
21:             end if
22:         end if
23:     end if
24:     Based on the content of the message and Equation 9, update
        $seen\_targets$ matrix and $unseen\_targets$ matrix
25:     TwoStageSetAndAnnounceBestOrientation();
26: end while
27: Set $CurrentState_i = 2$
```

```
28: if j_chosen == 0 then
29:     while a protocol message is received from sensor n $\in Q_i$ s.t. $PR_n \geq PR_i$ do
30:         if $PR_n == PR_i$ then
31:             if $\Phi_i > \Phi_n$ then
32:                 continue;
33:             else
34:                 if $n > i$ then
35:                     continue;
36:                 else
37:                     stop processing message, throw it away
38:                 end if
39:             end if
40:         end if
41:         Based on the content of the message and Equation 9, update seen_targets matrix and unseen_targets matrix
42:         TwoStageSetAndAnnounceBestOrientation();
43:     end while
44: end if
```

# 6 Methodology and simulation results

## 6.1 Methodology

We implement the algorithms and run the experiments in Matlab. Different parameters are varied in different simulations; however, unless otherwise stated, default parameters for a setup are: 50 targets and 50 sensors randomly distributed over a 50 x 50 space, with each sensor having a sensing radius of 10, and 4 possible sensing orientations. Similar to Figure 1, the axes dividing the sensing regions are aligned to a North-East-West-South orientation; however, it should be noted that the results should not be any different if it is otherwise. Also, unless otherwise stated, results presented are the averages of 1000 experiment runs.

**Algorithm 7** Algorithm for setting and announcing best orientation, 2-stage version (TwoStageSetAndAnnounceBestOrientation)

```
1: j_old ← j_chosen
2: for j s.t. $1 \leq j \leq W$ do
3:     $P_j$ ← number of rows in *seen_targets* matrix with all 0s, s.t. target corresponding to the row $\in \phi_{i,j}$
4:     $R_j$ ← number of *unique* rows in *seen_targets* matrix that can also be found in *unseen_targets* matrix, s.t. target corresponding to the *seen_targets* matrix row $\in \phi_{i,j}$
5:     if $P_j$ != 0 then
6:         $R_j \leftarrow R_j + 1$
7:     end if
8:     if $CurrentState_i = 1$ then
9:         $U_j \leftarrow P_j$
10:    else
11:        $U_j \leftarrow R_j$
12:    end if
13: end for
14: if $\arg\max_{1 \leq j \leq W} U_j == 0$ then
15:     j_chosen ← 0
16: else
17:     j_chosen ← $\arg\max_{1 \leq j \leq W} U_j$
18: end if
19: if j_chosen != j_old then
20:     Set orientation to j_chosen
21:     Send a protocol message including priority and $\phi_{i,j\_chosen}$
22: end if
23: if $CurrentState_i = 1$ then
24:     StartTimer(TIMEOUT);
25: end if
```

The effect of $\alpha$ on the heuristic algorithms' performance is evaluated in Section 6.2. The effect of the number of sensors, number of targets, etc., is evaluated in Section 6.3. Section 6.4 will compare the execution times of TIA-CGA and the two centralized 2-stage algorithms. A limited validation of the heuristic algorithms' performance against that optimal solutions is provided in Section 6.5.

## 6.2 Effect of $\alpha$ on heuristic algorithms

We test the effect of the parameter $\alpha$ on the performance of TIA-CGA and TIA-DGA using a setup with 50 targets and 50 sensors. Figure 3 plots the % of targets covered, % of sensors that are active, and the number of syndromes. Five values for $\alpha$ are tested: 0, 0.25, 0.5, 0.75, and 1.0. To provide a point of comparison, we also plot the metrics for 2S-CGA, 2S-CFA, 2S-DGA, and 2S-DFA. As expected, the plots for the four are horizontal lines, since they are not parameterizable by $\alpha$.

The first thing that must be noted from Figure 3 is how similar the results are for $\alpha = 0.25$, $\alpha = 0.5$, and $\alpha = 0.75$. This signifies that while the algorithms allow users to specify the acceptable level of trade-off between coverage and identifiability, in reality and practice, sensor-target setups offer a limited number of possible configurations and solutions. Therefore different values of $\alpha$ (except 0 and 1.0) can result in the same solution. This does not mean however that $\alpha$ does not matter, as the results for $\alpha = 0$ and $\alpha = 1.0$ will show.

When $\alpha = 0$, the algorithm degenerates into a syndrome-maximizing algorithm.

For TIA-CGA, $\alpha = 0$ utilizes the same number of sensors as the three middle $\alpha$ values: 30% of the total (Figure 3b). With $\alpha = 0$, the TIA-CGA covers slightly fewer targets than the solution for all other $\alpha$ values (Figure 3a) - 77% compared to $\alpha = 0.25$'s 79%, for instance. Surprisingly, $\alpha = 0$ actually has a slightly lower average number of syndromes compared to the three middle $\alpha$ values (Figure 3c) - 22.07 compared to $\alpha = 0.25$'s 22.41.

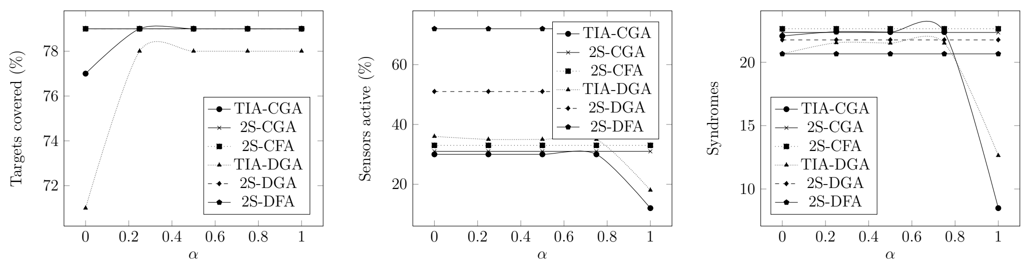
For TIA-DGA, $\alpha = 0$ utilizes a slightly higher number of sensors than the three middle $\alpha$ values: 36% compared to 35% for $\alpha = 0.25$ (Figure 3b). With $\alpha = 0$, TIA-DGA covers less targets than the solution for all other $\alpha$ values (Figure 3a) - 71% compared to $\alpha = 0.25$'s 78%, for instance. Like TIA-CGA, at $\alpha = 0$, TIA-DGA actually has a lower average number of syndromes compared to the three middle $\alpha$ values (Figure 3c).

$\alpha = 1.0$ signifies that only the number of covered target matters, and the algorithm degenerates into a coverage-maximizing algorithm.

TIA-CGA at $\alpha = 1.0$ covers almost the same number of targets as the middle $\alpha$ values (Figure 3a), but does so with significantly less sensors: 12%, compared to 30% of $\alpha = 0.25$ (Figure 3b). With $\alpha = 1.0$, TIA-CGA ends up with a significantly smaller number of syndromes than at other $\alpha$ values (Figure 3c) - 8.49, against $\alpha = 0.25$'s 22.41.

TIA-DGA at $\alpha = 1.0$ also covers almost the same number of targets as the middle $\alpha$ values (Figure 3a), and also does so with significantly less sensors: 18%, compared to 35% of $\alpha = 0.25$ (Figure 3b). With $\alpha = 1.0$, TIA-DGA likewise ends up with a significantly smaller number of syndromes than at other $\alpha$ values (Figure 3c) - 12.64, against $\alpha = 0.25$'s 21.54.

In summary, Figure 3 illustrates that because of the limited number of solutions offered by sensor-target setups, the solutions for values of $\alpha$ between 0 and 1.0 do not differ by much from each other. However, a non-0, non-1.0 $\alpha$ value still offers a middle ground between $\alpha = 0$ and $\alpha = 1.0$; or more accurately, it offers the 'best of both worlds' in terms of targets covered and syndromes generated.



(a) Targets covered (b) Active sensors (c) Syndromes

Figure 3: Effect of $\alpha$ on heuristic algorithms. 50 targets, 50 sensors, 50x50 space, 4 sensing orientations, sensing radius = 10.

## 6.3 Main comparison

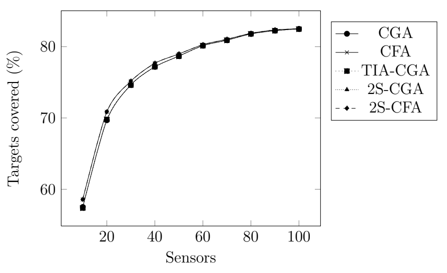
In this subsection, we test the effect of the number of sensors, number of targets, number of possible orientations, and the sensing radius on the metrics number of targets covered, number of sensors that are active, number of syndromes generated, and the system utility garnered. For the distributed algorithms we also include the *total* number of broadcasts made by *all* nodes. We assume that two nodes that can cover at least one target in common can communicate with one another (i.e., are one-hop neighbours in the network topology). We also assume that the nodes utilize a perfect media access control (MAC) protocol, and that there are no retransmissions due to interference (or hidden and exposed terminal problems). Such an assumption is difficult to realise in actual implementations, but the results from the simulations will at least give us an approximate measure of the communication costs associated with the distributed heuristic algorithms.

### 6.3.1 Number of sensors

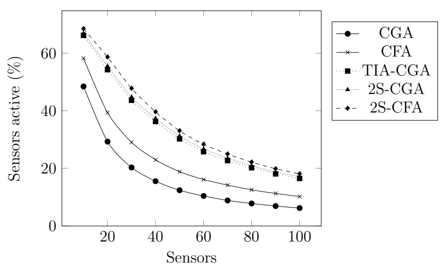
For the first simulation set, we vary the number of sensors from 10-100 and hold the number of targets constant at 50.

The results for the centralized algorithms are shown in Figure 4. As can be seen in Figure 4a, as more sensors are added, the % of covered targets increases, but the increase eventually slows down. All three heuristic algorithms track CGA and CFA very closely. The diminishing increase in the number of covered targets indicate that the system is becoming more and more over-provisioned, with an increasing number of sensors becoming idle since they no longer have any targets to cover or syndromes to contribute. This is consistent with Figure 4b which shows the drop in % of sensors which are active. Among the heuristic algorithms, 2S-CFA consistently utilizes the most sensors, followed by 2S-CGA, and then TIA-CGA. In Figure 4c, we see that consistent with their aims, all three heuristic algorithms consistently have significantly higher number of syndromes than either CGA or CFA. The number of syndromes increases with the number of sensors, but the increase becomes more and more attenuated as the number of sensors increases. The plateauing in the number of syndromes occurs much earlier for CGA and CFA. The same pattern is seen in Figure 4d, which shows the system utility. 2S-CFA has a slight but consistent advantage over the other two heuristic algorithms when it comes to the system utility garnered.

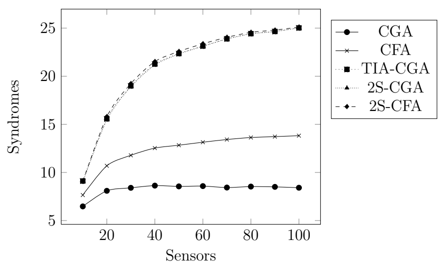
The results for the distributed algorithms are shown in Figure 5. As can be seen in Figure 5a, as more sensors are added, the % of covered targets goes up, but the increase eventually slows down. The % of covered targets



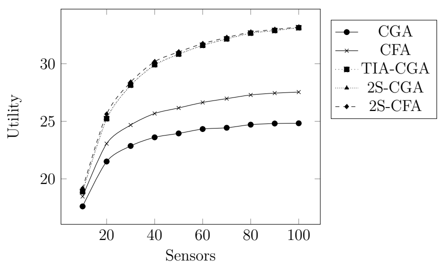
(a) Targets covered



(b) Active sensors



(c) Syndromes



(d) Utility

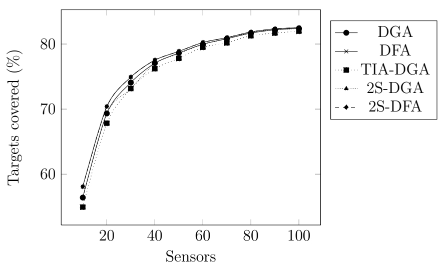
Figure 4: Effect of varying the number of sensors on *centralized* algorithms. $\alpha = 0.5$, 50 targets, 50x50 space, 4 sensing orientations, sensing radius = 10.

are very similar for all algorithms, with TIA-DGA lagging just very slightly behind the others. The plateauing indicates that the system is becoming more and more over-provisioned, with more and more sensors becoming idle since they no longer have any targets to cover or syndromes to contribute. Consistent with this, Figure 5b shows the drop in % of active sensors. The plots are quite similar for DGA and DFA, with the plots of the three heuristic algorithms being higher - this is because TIA-DGA, 2S-DGA, and 2S-DFA utilize the additional sensors for increasing the number of syndromes. Between the three heuristic algorithms, TIA-DGA utilizes a slightly lower number of active sensors than the two 2-stage algorithms. In Figure 5c, we see that consistent with their aims, TIA-DGA, 2S-DGA, and 2S-DFA consistently have a higher number of syndromes than either DGA or DFA. The number of syndromes increases with the number of sensors, but the increase becomes more and more attenuated as the number of sensors increases. The number of syndromes plateaus much earlier for DGA and DFA. Between the three distributed heuristic algorithms, and with respect to the number of syndromes, 2S-DFA performs best, followed very closely by 2S-DGA, and then TIA-DGA. The pattern for the number of syndromes is repeated for the system utility (Figure 5d). As for the number of broadcasts made (Figure 5e), all five algorithms follow a linear (increasing) relationship with the number of sensors. The number of broadcasts for DGA and DFA are highly similar. The slope for the TIA-DGA is slightly higher than that of DGA and DFA. Both 2S-DGA and 2S-DFA have noticeably higher slopes than TIA-DGA, with 2S-DFA having a slightly higher slope than 2S-DGA.

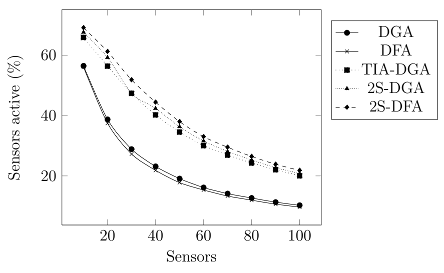
### 6.3.2 Number of targets

For the second simulation set, we vary the number of targets from 10-100 and hold the number of sensors constant at 50.

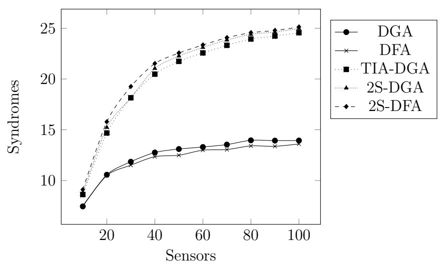
The results for the centralized algorithms are plotted in Figure 6. Figure 6a shows that the % of targets covered remains constant all throughout the values tested (this means that the absolute number of targets covered *increases*). As the number of targets increases, the % of sensors that are active also increases (Figure 6b). We also see in Figure 6b that compared to CGA and CFA, the three heuristic algorithms consistently have more sensors that are active. Among the heuristic algorithms, 2S-CFA consistently utilizes more sensors than 2S-CGA, which in turn, consistently utilizes more than TIA-CGA. Consistent with their aims, Figure 6c shows that the heuris-



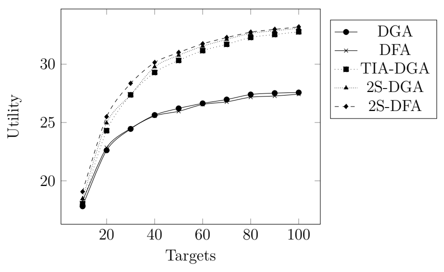
(a) Targets covered



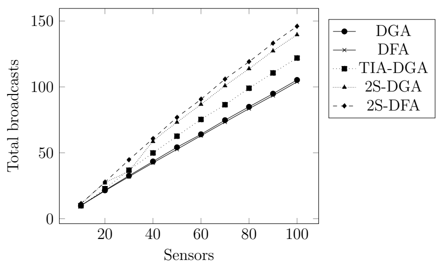
(b) Active sensors



(c) Syndromes



(d) Utility

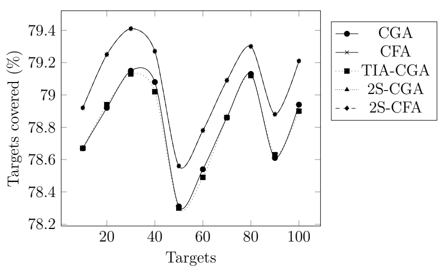


(e) Broadcasts

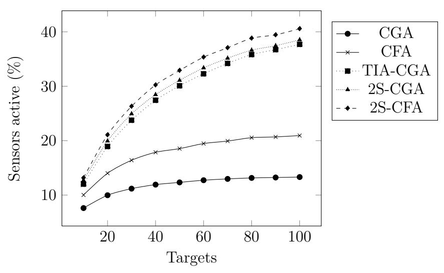
Figure 5: Effect of varying the number of sensors on *distributed* algorithms. $\alpha = 0.5$, 50 targets, 50x50 space, 4 sensing orientations, sensing radius = 10.

tic algorithms consistently have more syndromes than CGA and CFA. The same can also be said about the system utility, shown in Figure 6d. When it comes to the system utility garnered, 2S-CFA once again has a slight but consistent advantage over the two other heuristic algorithms.

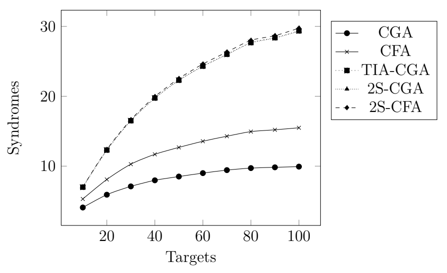
The results for the distributed algorithms are plotted in Figure 7. Figure 7a shows that the % of targets covered remains constant all throughout the values tested, with only a very slight variation across the values tested



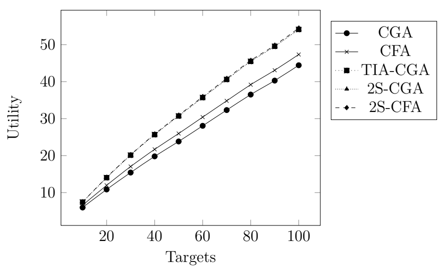
(a) Targets covered



(b) Active sensors



(c) Syndromes



(d) Utility

Figure 6: Effect of varying the number of targets on *centralized* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 4 sensing orientations, sensing radius = 10.

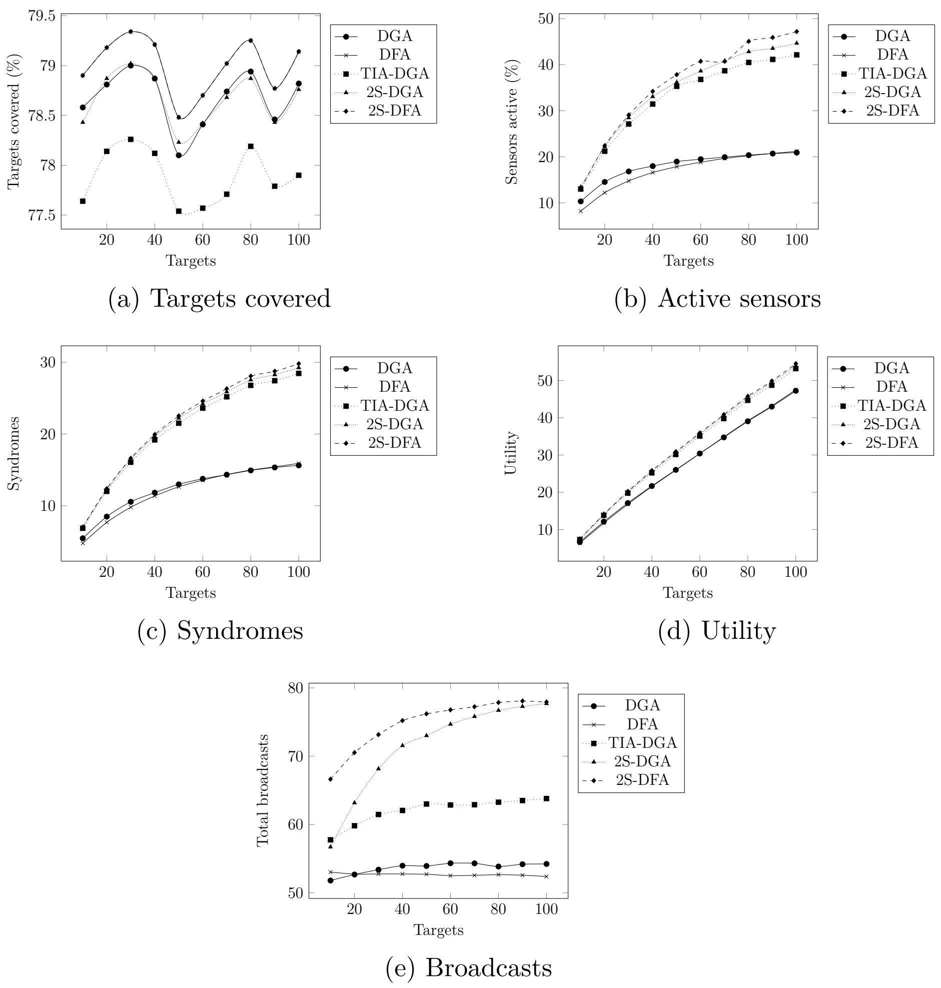
(this means that the absolute number of targets covered *increases*). It can also be seen in the plot that TIA-DGA trails the other four algorithms when it comes to the % of targets covered. The number of active sensors increases for all algorithms, although the increase is much steeper for TIA-DGA, 2S-DGA and 2S-DFA than DGA or DFA (Figure 7b). The increase in the number of active sensors can also be seen to plateau, as already active sensors prove sufficient to cover the targets that are added. Between the three distributed algorithms that are target-identifiability aware, the increase is greatest for 2S-DFA, followed by 2S-DGA, and finally, TIA-DGA. Consistent with the algorithms' aims, Figures 7c and 7d show that TIA-DGA, 2S-DGA and 2S-DFA consistently have more syndromes and higher system utilities than both DGA and DFA, with the difference increasing with the number of targets.

Between the three distributed algorithms for MTIAUMS, the same ordering seen before in Figure 7b is again repeated in Figures 7c and 7d: 2S-DFA closely followed by 2S-DGA, which is then closely followed by TIA-DGA. As for broadcasts, it can be seen in Figure 7e that DGA and DFA utilize significantly less broadcast messages than the other three algorithms, and the number of broadcast messages stay more or less the same even whwn the number of targets is increased. In comparison, the number of broadcasts made by TIA-DGA, 2S-DGA and 2S-DFA increases with the number of targets. 2S-DFA consistently makes more broadcasts than TIA-DGA. 2S-DGA starts with less broadcasts than TIA-DGA, but eventually catches up with 2S-DFA.

### 6.3.3 Number of sensor orientations

For the third simulation set, we vary the number of possible sensor orientations from 2-8 and hold the number of sensors and targets constant at 50.

The results for the centralized algorithms are plotted in Figure 8. It must be noted that an increase in the number of possible sensor orientations implies a decrease in the size of the sensed region. In Figure 8a it can be seen that when it comes to the number of targets covered, the increase in the number of possible sensor orientations only slightly affects CFA and 2S-CFA. It has the effect of decreasing the number of covered targets for TIA-CGA, CGA, and 2S-CGA, although the decrease seems to plateau after 6. As for the number of active sensors (Figure 8b), the increase in the number of possible sensor orientations results in the increase in the number



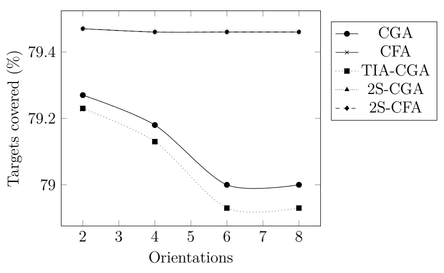
(a) Targets covered (b) Active sensors

(c) Syndromes (d) Utility

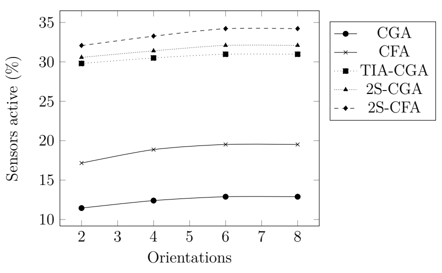
(e) Broadcasts

Figure 7: Effect of varying the number of targets on *distributed* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 4 sensing orientations, sensing radius = 10.

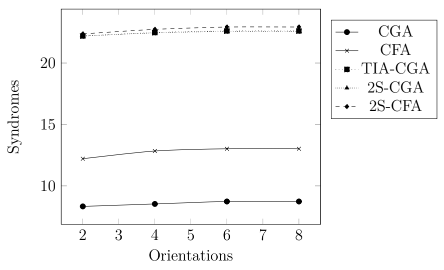
of active sensors for all algorithms. The number of syndromes for the three heuristic algorithms slightly increases as the number of possible sensor orientations increases (Figure 8c). The slight increase is brought about by the increase in the number of intersecting sensed regions, which is a consequence of the increased number of active sensors. The pattern seen in the number of syndromes is carried over to the system utility garnered by the algorithms (Figure 8d).



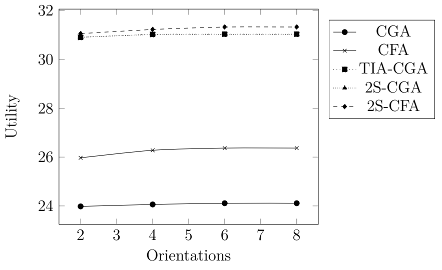
(a) Targets covered



(b) Active sensors



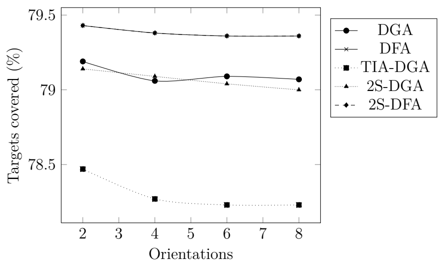
(c) Syndromes



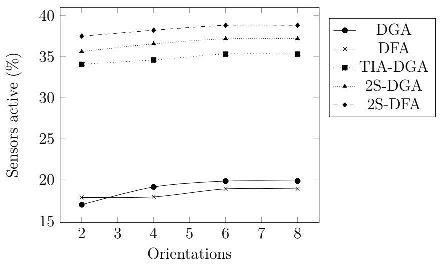
(d) Utility

Figure 8: Effect of varying the number of possible sensor orientations on *centralized* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, sensing radius = 10.

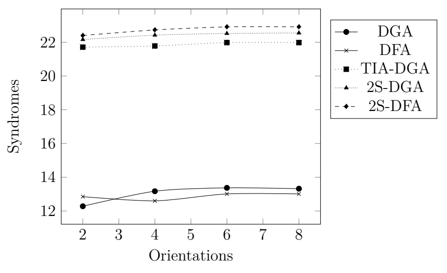
The results for the distributed algorithms are plotted in Figure 9. In Figure 9a it can be seen that the increase in the number of possible sensor orientations causes a slight decrease in the number of targets covered for all algorithms. The increase in the number of possible sensor orientations also has the effect of slightly increasing the number of active sensors (Figure 9b), which is to be expected as the area covered by each sensor decreases. The increase in the number of active sensors also causes an increase in the intersection of active sensed regions, resulting in an increase in the number of syndromes (Figure 9c). Similar to the centralized algorithms, even with the slight decrease in the number of covered targets, the increase in the number of syndromes more than compensates and the system utility garnered increases for all algorithms (Figure 9d). As for the number of broadcasts, the increase in the number of possible sensor orientations causes an increase in the number of broadcasts for all algorithms (partially to be expected due to the increase in the number of active sensors), although the increase is especially pronounced for TIA-DGA (Figure 9e).



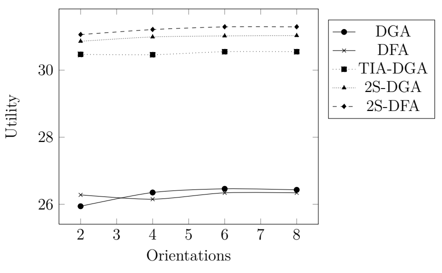
(a) Targets covered



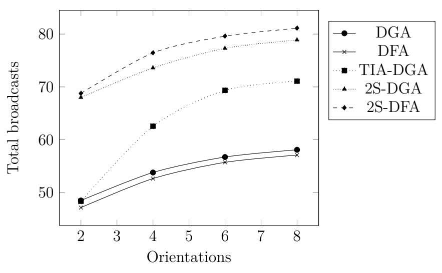
(b) Active sensors



(c) Syndromes



(d) Utility



(e) Broadcasts

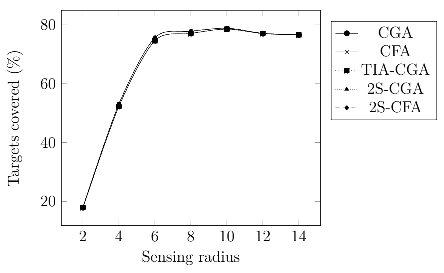
Figure 9: Effect of varying the number of possible sensor orientations on *distributed* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, sensing radius = 10.

### 6.3.4 Sensing radius

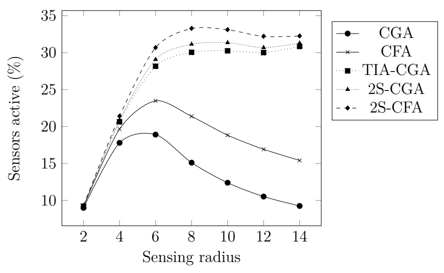
For the fourth simulation set, we vary the sensing radius from 2-8 and hold the number of sensors and targets constant at 50.

The results for centralized algorithms are plotted in Figure 10. In Figure 10a it can be seen that as the sensing radius increases, more targets are covered as sensors are able to reach previously unreachable targets. The increase however eventually flattens out. The trend holds true for all the algorithms. To cover the targets that can now be seen by the sensors, more sensors are activated (Figure 10b). Nevertheless, the increase in the number of active sensors eventually plateaus for TIA-CGA, 2S-CGA and 2S-CFA. In contrast, the number of active sensors actually *decreases* for CGA and CFA after some point - as the sensing radius of sensors grows, fewer and fewer sensors are needed to cover the targets (and coverage is solely the concern of CGA and CFA). The decrease is not seen in the TIA-CGA, 2S-CGA and 2S-CFA (at least not yet in the values tested) because the multiple coverage is used to increase the number of syndromes (Figure 10c). The increase in the number of covered targets (at least in the first few values) and the increase in the number of syndromes lead to increased system utility for all algorithms, although that for CGA and CFA eventually decreases (Figure 10d). The decrease in system utility for CGA and CFA is due to the decrease in the number of syndromes, which, in turn, is caused by the decrease in the number of covering sensors.

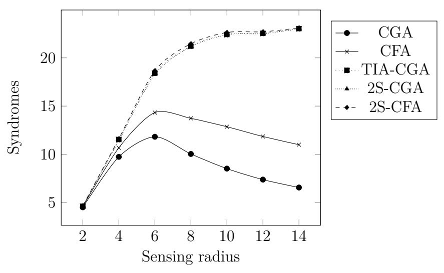
The results for distributed algorithms are plotted in Figure 11. The patterns seen in Figures 10a-10d can also be seen in their counterparts in



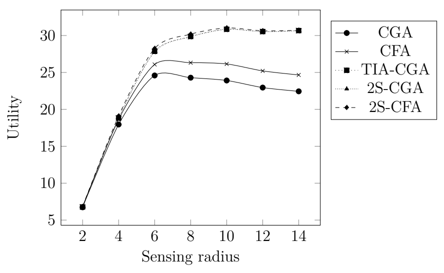
(a) Targets covered



(b) Active sensors



(c) Syndromes



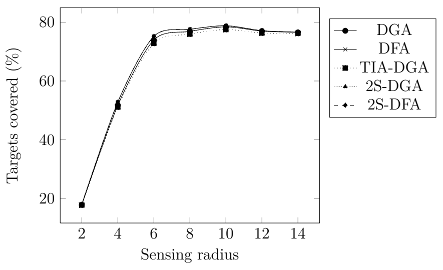
(d) Utility

Figure 10: Effect of varying the sensing radius on *centralized* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, 4 sensing orientations.

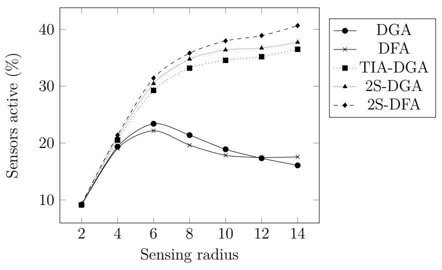
Figures 11a-11d. The increase in the number of targets that can be covered (Figure 11a), as well as the increase in the number of active sensors (Figure 11b), leads to the increase in the number of broadcasts (Figures 11e) - however, the increase for TIA-DGA is markedly less than that of 2S-DGA and 2S-DFA. The increases for DGA and DFA are even less than that of TIA-DGA, and also flatten out very early.

## 6.4 Comparison of execution times of centralized heuristic algorithms

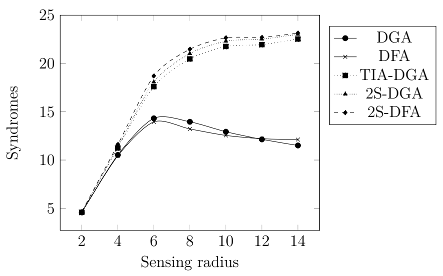
The primary advantage of (and rationale for) the centralized 2-stage algorithms are their lower execution times compared to that of TIA-CGA. To validate this, we plot the average algorithm execution times for the cen-



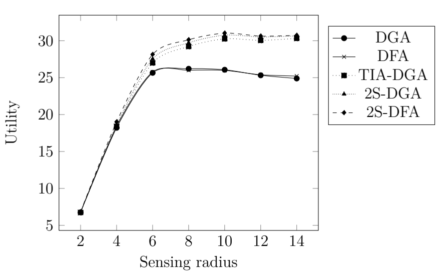
(a) Targets covered



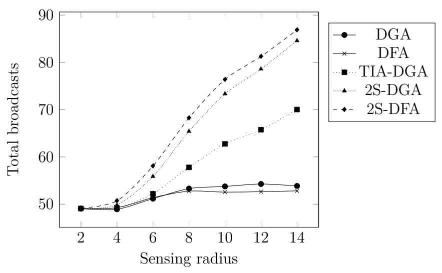
(b) Active sensors



(c) Syndromes



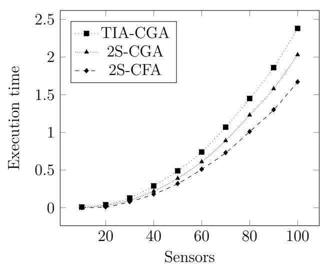
(d) Utility



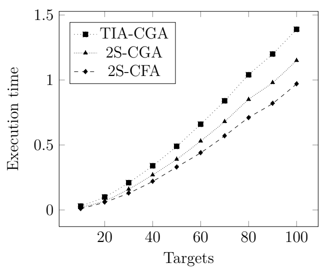
(e) Broadcasts

Figure 11: Effect of varying the sensing radius on *distributed* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, 4 sensing orientations.

tralized heuristic algorithms (as when they were used in Section 6.3.1 and Section 6.3.2) in Figure 12. Figure 12a shows the execution times for the simulation set where the number of sensors is varied, while Figure 12b shows the execution times for the simulation set where the number of targets is varied. Comparing Figure 12a and Figure 12b, it becomes apparent that the number of sensors has a greater effect on the growth of the execution time than the number of targets. It can also be seen in both plots that TIA-CGA consistently has longer execution times than the two 2-stage algorithms.



(a) Number of sensors varied



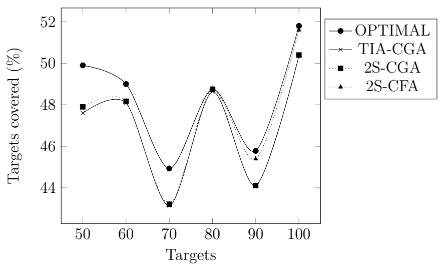
(b) Number of targets varied

Figure 12: Average execution times for the heuristic algorithms. $\alpha = 0.5$, 50x50 space, 4 sensing orientations, sensing radius = 10.

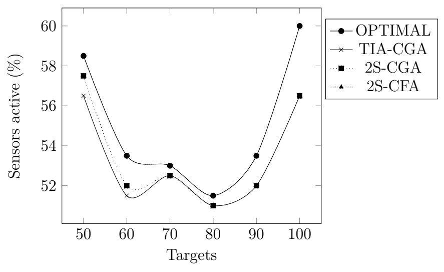
## 6.5 Limited comparison of heuristic solutions to optimal solutions

To provide validation (even if just a limited one) of how close the heuristic solutions are to the optimal solutions, we solve a limited number of problems (20 for each setup) using brute force. The number of targets is varied from 50 to 100, while the number of sensors is set constant to 10. The targets and sensors are distributed over a 30 x 30 space, with $\alpha$ set to 0.5, the number of sensing regions to 4, and the sending radius to 5. The results of the simulations are plotted in Figure 13 (centralized) and Figure 14 (distributed). The values plotted in Figure 13 and Figure 14 are the averages of 20 runs.

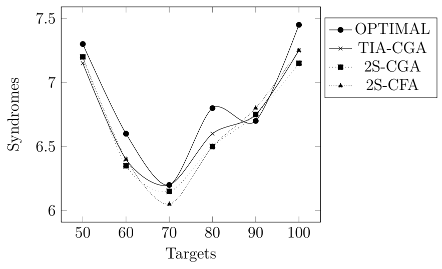
It can be seen in Figure 13 and Figure 14 that the heuristic solutions approximate the optimal solutions well. The system utility, which is the value being maximized, has two components: the number of targets covered and the number of syndromes generated. Figure 13 and Figure 14 show that it is possible for a heuristic algorithm to produce solutions that have more than the optimal solution in a certain component: for example, for targets = 90, all centralized heuristic solutions (Figure 13c) and 2S-DFA (Figure 14c) actually have more syndromes than the optimal solution. For all the setups tested, the best performance (in terms of system utility) by a centralized heuristic algorithm (Figure 13d) is by 2S-CFA (targets = 50), which comes within 0.3% of the optimal solution's system utility. The worst performance by a centralized algorithm is by TIA-CGA (targets = 50), which differs from the optimal solution by 4.0%. For the distributed algorithms (Figure 14d), the best performance is by 2S-DFA (targets = 80), with only 0.52% difference, while the worst is by TIA-DGA, with 8.25% difference.



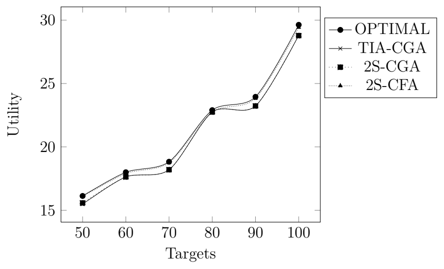
(a) Targets covered



(b) Active sensors

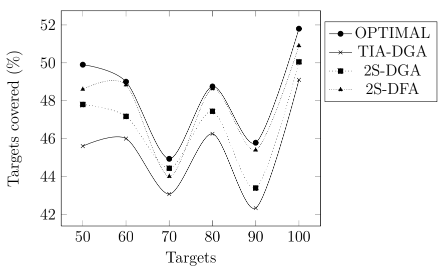


(c) Syndromes

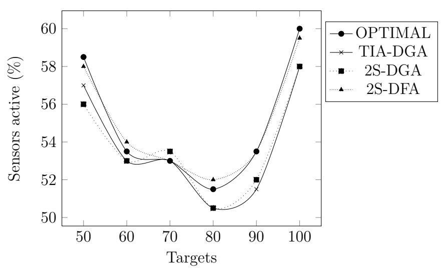


(d) Utility

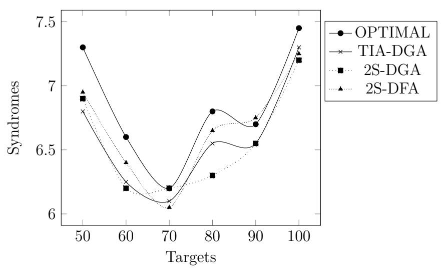
Figure 13: Validation of solutions produced by *centralized* heuristic algorithms against optimal solutions. $\alpha = 0.5$, 10 sensors, 30x30 space, 4 sensing orientations, sensing radius = 5.



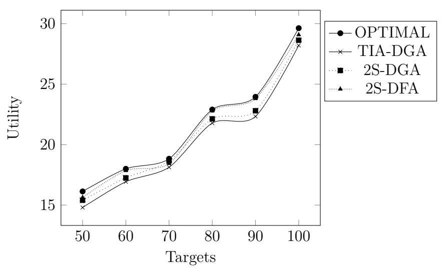
(a) Targets covered



(b) Active sensors



(c) Syndromes



(d) Utility

Figure 14: Validation of solutions produced by *distributed* heuristic algorithms against optimal solutions. $\alpha = 0.5$, 10 sensors, 30x30 space, 4 sensing orientations, sensing radius = 5.

## 7 Related work

The problem of maximizing target-based coverage for directional sensors was first formalized in [1] as the Maximum Coverage Minimum Sensors (*MCMS*) Problem. The MCMS problem's Integer Linear Programming (ILP) formulation was also given in [1], along with 2 heuristic-based solutions: a centralized algorithm (Centralized Greedy Algorithm, or *CGA*) and a distributed algorithm (Distributed Greedy Algorithm, or *DGA*).

Munishwar and Ab-Ghazaleh [10] also dealt with the problem of target coverage maximization, but in the specific context of visual sensor networks (cameras). The main difference between visual sensors and general sensors is the concept of $R_{Min}$ or the *minimum* distance required between the sensor and the target for the latter to see the former. Munishwar and Ab-Ghazaleh [10] proposed alternatives to [1]'s CGA and DGA, called Centralized Force-based Algorithm (*CFA*) and Distributed Force-based Algorithm (*DFA*), respectively.

Another common problem in directional sensor networks is finding and scheduling cover sets. Cover sets are sets of sensors (and their orientations) that ensure that all targets are covered. It must be noted that finding cover sets is equivalent to finding network configurations repeatedly, each time removing the cover set-comprising nodes from the set of viable nodes. The need for cover sets is primarily motivated by the limited lifetimes of nodes. By finding several such cover sets and scheduling their activation, the targets will be covered for a longer period of time than if only a single cover set is found. Notable studies on cover sets are [2] and [18].

This study is not the first study to go beyond maximization of the number of covered targets.

Fusco and Gupta [4] dealt with the problem of providing $k$-coverage to a given set of targets (or area) using a minimum number of sensors - a problem they called *SODkC*, or Selecting and Orienting D-sensors for $k$-Coverage. The problem was shown to be NP-hard, and they proposed two greedy solutions to the problem, a centralized algorithm and a distributed algorithm. Wang et al[17], on the other hand, dealt with the case where the targets have different coverage requirements.

It must be noted however, that the problem of maximizing coverage or meeting coverage requirements is different than that of maximizing target identifiability. For instance, if what is wanted is to satisfy a certain k-coverage requirement, when two sensors cover *exactly* the same set of targets, both should be activated since by doing so the coverage of each target covered increases. However, if what is wanted is the maximization of target identifiability, only one of the sensors should be activated, as activating both adds no new syndrome to the system.

## 8 Conclusion

It is important to recognize that the assumption of built-in target identifiability needs to be re-examined in directional sensor networks, as it does not always hold true. To this end, we introduce the concept of *syndromes* which facilitates the measurement of target identifiability in a directional sensor network - this leads to the definition of the Maximum Target Identifiability-Aware Utility with Minimum Sensors (MTIAUMS) problem, which we prove to be NP-hard. We also introduce heuristic algorithms for orienting sensors in a directional sensor network. The algorithms take into account not just the number of targets covered but also their identifiability when finding network configurations.

We introduce three centralized heuristic algorithms: Target Identifiability-Aware Centralized Greedy Algorithm (TIA-CGA), 2-stage Target Identifiability-Aware Centralized Greedy Algorithm (2S-CGA), and 2-stage Target Identifiability-Aware Centralized Force-based Algorithm (2S-CFA). TIA-CGA is parameterizable with $\alpha$ which lets users specify the desired level of trade-off between coverage and identifiability. Simulations show that $\alpha$ loses sensitivity or differentiability in the middle range of values because of the limited number of configurations possible for any sensor-target setup; nevertheless, the middle values still provide a middle ground, solutions-wise, between those given by extreme $\alpha$ values. The 2-stage algorithms on the other hand are not parameterizable, instead covering the targets first (using algorithms designed for MCMS) and then using the remaining sensors to increase the identifiability of covered targets. The 2-stage algorithms tend to have shorter runtimes than TIA-CGA. Of the three heuristic algorithms, 2S-CFA perform best in most of the simulations carried out, followed by 2S-CGA, and then TIA-CGA. 2S-CFA however, also has the tendency to use more nodes than 2S-CGA and TIA-CGA.

We also introduce three distributed heuristic algorithms: Target Identifiability-Aware Distributed Greedy Algorithm (TIA-DGA), 2-stage Target Identifiability-Aware Distributed Greedy Algorithm (2S-DGA), and 2-stage Target Identifiability-Aware Distributed Force-based Algorithm (2S-DFA). Similar to TIA-CGA, TIA-DGA is parameterizable with $\alpha$. 2S-DGA and 2S-DFA are the distributed counterparts of 2S-CGA and 2S-CFA, respectively. In most of the simulations carried out, 2S-DFA produces more syndromes and slightly higher system utilities than 2S-DGA and TIA-DGA.

## Acknowledgement

This work is funded in part by the *UK Technology Strategy Board (TSB) Emerging Technologies Programme*, Project 131187/26835-183208, *OPV-based Energy Harvesting for Urban Noise Pollution.* Author W. M. Tan is supported by the Republic of the Philippines' *Engineering Research and Devel-*

*opment for Technology (ERDT) Program.*

## Declaration of relevant previous publications

Some portions and aspects of this paper were published before elsewhere: we first introduced the centralized algorithms in [14] while we first presented TIA-DGA in [15]. Portions of Sections 6.2, 6.3.1 and 6.3.2 (sans data on total number of broadcasts) pertaining to the aforementioned algorithms were also previously presented in [14] and [15].

## References

[1] Jing Ai and AlhusseinA. Abouzeid. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization*, 11(1):21–41, 2006.

[2] Yanli Cai, Wei Lou, Minglu Li, and Mo Li. Target-oriented scheduling in directional sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1550–1558, May 2007.

[3] J. Djugash, S. Singh, G. Kantor, and Wei Zhang. Range-only slam for robots operating cooperatively with sensor networks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2078–2084, May 2006.

[4] G. Fusco and H. Gupta. Selection and orientation of directional sensors for coverage maximization. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, June 2009.

[5] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[6] I. Hakala, I. Kivela, J. Ihalainen, J. Luomala, and Chao Gao. Design of low-cost noise measurement sensor network: Sensor function design. In *Sensor Device Technologies and Applications (SENSORDEVICES), 2010 First International Conference on*, pages 172–179, 2010.

[7] Huadong Ma and Yonghe Liu. On coverage problems of directional sensor networks. In Xiaohua Jia, Jie Wu, and Yanxiang He, editors, *Mobile Ad-hoc and Sensor Networks*, volume 3794 of *Lecture Notes in Computer Science*, pages 721–731. Springer Berlin Heidelberg, 2005.

[8] Huadong Ma and Yonghe Liu. Some problems of directional sensor networks. *Int. J. Sen. Netw.*, 2(1/2):44–52, April 2007.

[9] Panasonic. WM-55A103/WM-56A103: Unidirectional Back Electret Condenser Microphone Cartridge, November 2013.

[10] Vikram P. Munishwar and Nael B. Abu-Ghazaleh. Coverage algorithms for visual sensor networks. *ACM Trans. Sen. Netw.*, 9(4):45:1–45:36, July 2013.

[11] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, pages 192–204, New York, NY, USA, 2005. ACM.

[12] Silvia Santini, Benedikt Ostermaier, and Andrea Vitaletti. First experiences using wireless sensor networks for noise pollution monitoring. In *Proceedings of the workshop on Real-world wireless sensor networks*, REALWSN '08, pages 61–65, New York, NY, USA, 2008. ACM.

[13] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 214–226, New York, NY, USA, 2004. ACM.

[14] W.M. Tan and S.A. Jarvis. Quality over quantity: Target identifiability in directional sensor networks. In *Wireless Sensor (ICWISE), 2014 IEEE Conference on*, Oct 2014.

[15] W.M. Tan and S.A. Jarvis. A distributed heuristic solution to the target identifiability problem in directional sensor networks. In *Computing, Networking and Communications (ICNC), 2014 International Conference on [TO APPEAR]*, Feb 2015.

[16] Dan Tao, Shaojie Tang, and Liang Liu. Constrained artificial fish-swarm based area coverage optimization algorithm for directional sensor networks. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 304–309, 2013.

[17] Jian Wang, Changyong Niu, and Ruimin Shen. Priority-based target coverage in directional sensor networks using a genetic algorithm. *Computers & Mathematics with Applications*, 57(11):1915 – 1922, 2009. Proceedings of the International Conference Bio-Inspired Computing-Theories and Applications BIC-TA 2007 Zhengzhou, China.

[18] Huiqiang Yang, Deying Li, and Hong Chen. Coverage quality based target-oriented scheduling in directional sensor networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5, May 2010.