2020

Optimizing Cluster Sets For The Scan Statistic Using Local Search

James Shulgan
jtshulgan@gmail.com

ABSTRACT

OPTIMIZING CLUSTER SETS FOR THE SCAN STATISTIC USING LOCAL SEARCH

James Shulgan, MS
Department of Electrical Engineering
Northern Illinois University, 2020
Dr. Benedito Fonseca, Director

In recent years, scattering sensors to produce wireless sensor networks (WSN) has been proposed for detecting localized events in large areas. Because sensor measurements are noisy, the WSN needs to use statistical methods such as the scan statistic. The scan statistic groups measurements into various clusters, computes a cluster statistic for each cluster, and decides that an event has happened if any of the statistics exceeds a threshold. Previous researchers have investigated the performance of the scan statistic to detect events; however, little attention was given to the optimization of which clusters the scan statistic should use. Using the scan statistic and a Gaussian sensor model, we present a local search approach for solving this optimization problem. Starting from multiple initial random cluster sets, our modified Gradient Ascent Search produces a cluster set that improves the worst-case detection performance of both grid and random sensor networks. By adding the best clusters to the worst emitter positions and removing the least valuable clusters, our search algorithm successfully produces a list of cluster sets that increase the minimum detection performance and outperforms baseline cluster sets by multiple standard deviations.

NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

AUGUST 2020

OPTIMIZING CLUSTER SETS FOR THE SCAN STATISTIC USING LOCAL SEARCH

BY

JAMES SHULGAN

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

Thesis Director:
Dr. Benedito Fonseca

DEDICATION

Dedicated to my parents for their love of learning and hard work and to my advisor for his passion for understanding.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION AND SUMMARY OF CONTRIBUTION

Section 1.0 – Introduction

A recurring objective in many engineering systems is to measure properties of an environment and react to them [1,2,3]. When detecting localized events occurring in a spacious region of interest (ROI), a fundamental approach consists of scattering wireless sensor nodes throughout the area and using their periodic measurements to detect the presence of an event [3,4,5,6]. Such wireless sensor networks (WSN) have become more practical with the development of smaller and cheaper microcontrollers and transceivers [1]. The compact and disposable nature of the sensor nodes makes them very suitable for inaccessible, hostile, or vast environments [7]. Wireless sensor networks have already been proposed for applications including intrusion detection, machinery fault detection [1,8], detection of chemical gases on a battlefield [7,9], unauthorized release of point radiation sources [10], or other threats [6].

Section 1.1 – Problem Statement and Objective

To reduce the effects of noise in WSNs, it is common to combine or fuse sensor measurements [2,3]. Once the sensors take their measurements, the data is transmitted to a fusion center where a fusion rule decides if an event is present, as illustrated in Figure 1. Here we will assume all sensors are found within a square region of interest indicated by a blue dotted line in

the figures. We use a powerful fusion rule known as the scan statistic [11,12] to fuse sensor measurements and decide between two hypotheses. Choosing hypothesis H_0 indicates that no event is present in the ROI while selecting hypothesis H_1 claims an event has been detected somewhere in the ROI. The scan statistic groups measurements from nearby sensors into clusters, computes a cluster statistic for each cluster, and decides that an event has happened if any of the statistics exceeds a threshold. The scan statistic has had significant success in detecting disease outbreaks [12], and many have proposed its use in WSN to detect point events [13-20]. However, limited research has been conducted to determine which clusters should be used by the scan statistic in the WSN. Because the set of clusters directly effects the sensor network's overall detection performance, our research goal is to optimize the cluster set of the scan statistic in order to maximizes its minimum detection performance.

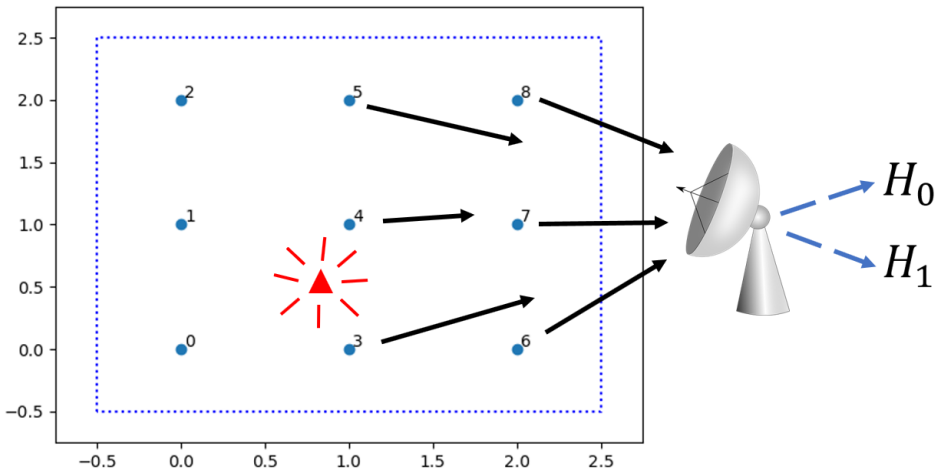

Figure 1. Measurements from sensors (blue dots) are sent to a fusion center so it can decide if the emitter (red triangle) is present in the region of interest (Hypothesis H_1) or not present in the region of interest (Hypothesis H_0).

Section 1.2 – Summary of Contribution and Results

Because the optimization of clusters in the scan statistic is a non-convex optimization problem, we propose a local search algorithm for choosing the scan statistic's cluster set. Using our modified Gradient Ascent Search, we start from an initial cluster set and add and remove specific clusters to successfully optimize cluster sets in networks were sensors are distributed in both a grid and random fashion. As part of our results, described in Chapter 6, we compare our final cluster set against other baseline cluster sets in various network sizes. We compare cluster sets using estimates for the Area Under the Receiver Operating Curve (AUC) when the emitter is at the worst location for each cluster set. In the scenario considered, we find that our proposed local search improves the estimated worst-case AUC from 0.781 ± 0.001 when using the best baseline cluster set to 0.811 ± 0.001 in the 5-by-5 grid network. To better appreciate the effect of this improvement, [13] has reported that improving the AUC by just 0.01 corresponded to improving the probability of detection from 23% to 26% (considering the same 5% probability of false alarm). In the scenario with 25 sensors randomly deployed, our proposed local search improves the worst-case AUC from 0.722 ± 0.001 to 0.814 ± 0.001.

CHAPTER 2

LITERATURE REVIEW

Designing clusters for WSNs has been proposed before [7,8,21-31]; however, the focus was not on developing which cluster should be in the cluster set for increased detection performance of the scan statistic. Instead, the focus was on connectivity; i.e., because the sensor nodes typically operate on limited battery power, papers like [7,8, 21-31] explore the communication, scalability, and energy conservation aspects of WSN. Clustering sensors is practical for the network's communication framework since, as discussed in [8], it is used to associate nearby clusters with a cluster head node. The cluster head is a special node that aggregates data from sensors within its cluster and routes the information throughout the network to the fusion center. Sending data to the closest cluster head and using only the cluster head to transmit over larger distances plays a key role in saving precious sensor battery life. For this reason, clustering algorithms proposed before focus on ensuring that each sensor is able to connect to a single cluster head. As illustrated by [13], if the clusters formed by such clustering algorithms are used in the scan statistic, the resulting detection performance is worse than the performance of many baseline cluster sets.

Other authors that do address detection in the network look at how to combine the sensor measurements with fusion rules different from the scan statistic [4,5,6,15,31,32]. If the individual sensor probability of detection P_D and probability of false alarm P_{FA} are known, Chair and Varshney have already developed the optimum fusion rule [2,15,32]. However, the emitter in

sensor network applications may be mobile and thus the P_D and P_{FA} of each sensor will vary depending upon the position of the emitter in the ROI at any given time [4]. A simple fusion rule presented by Niu and Varshney in [5] does not need the detection probabilities of each sensor, but rather just counts how many sensors indicate a detection. This counting rule was further developed in [4] to account for an unknown number of total effective sensors. Nevertheless, by just using the number of sensor detections, the WSN's P_D is harmed. As mentioned in [13], combining sensor measurements that are near and far from a given emitter will mix different signal strengths. The signal is typically weaker at sensor that are farther away, so measurements from these sensors will be more effected by noise. Likewise, the authors in [15], point out that the counting rule comes with spurious detection performance. This is unsurprising, since combining noisy sensor measurements with good sensor measurements will consequently degrade the network's P_D and P_{FA}.

The scan statistic, discussed in more detail in Chapter 3, was originally developed by statisticians focused on detecting anomalies in georeferenced data [11,12,33]. It has had significant success in detecting disease outbreaks and is currently used by CDC [33].

Since an emitter causes changes in the measurements of nearby sensors, it can be considered an anomaly, and Guerriero, Willett, and Glaz have proposed the use of the scan statistic in WSN [14]. In [14], the authors created clusters by moving a window across the ROI and formed clusters by grouping the sensors in the window. The authors then showed that an optimal window sizes exists when using a single sized window and also referred to creating cluster sets by scanning multiple window sizes over the ROI. Nevertheless, the authors in [14] were more focused on the application of the scan statistic to WSN's and did not consider whether

all the clusters found from the moving window are beneficial for the network's detection performance.

The optimization of which clusters should be present in a cluster set when using the scan statistic to improve the network P_D has been studied by Fonseca in [13]. The focus here was not on how to search for the best clusters but rather on highlighting the opportunity of designing cluster sets from a given set of all possible clusters for the network. Fonseca illustrated the benefit of designing cluster sets by showing that simply choosing the best cluster for all the worst emitter positions can improve the detection performance.

CHAPTER 3

MODEL DESCRIPTION AND BACKGROUND

Section 3.0 – Model Focus

As mentioned previously, many existing papers focus on the connectivity and networking aspects of wireless sensor networks; however, we will focus on the detection performance. Namely, we focus on a network's probabilities of detection (P_D) and false alarm (P_{FA}) through its Receiver Operating Curve (ROC) [34]. The core problem in sensor networks arises whenever a sensor takes a reading. Sensor measurements will always be corrupted by some noise which will cause the overall detection performance of the network to deteriorate. Noisy measurements increase the network's P_{FA} and thereby cause it to detect an emitter when no emitter is present. To mitigate the effects of noise, sensor measurements can be combined in clusters and/or a Fusion Center. This ultimately helps reduces the sensor noise through averaging. Because we will be focusing on the detection aspect of clustered sensors, we assume the connectivity of the network is not an issue and consider that all sensors are capable of transmitting their measurements to a Fusion Center.

Section 3.1 – Measurement Model

In our research, we will focus on sensor measurements with a Gaussian distribution: $N(\mu, \sigma^2)$. Considering a WSN with K sensors, let Z_k be the measurement reported by the sensor

with an index of k. We assume $Z_k \sim N(0,1)$ under H_0 (no emitter present). Furthermore, we will be assuming an emitter with a signal that decays with the square of the distance between the sensor and the emitter. Let the emitter be located at L_e, have a signal amplitude of A and a signal decay rate of $\gamma = 2$. The presence of the emitter will shift the sensor measurement mean. If the sensor is separated by a distance of d from the emitter at L_e, then, under H_1, the sensor's measurement distribution will be: $Z_k \sim N(\frac{A}{d^\gamma}, 1)$, where we assume that an emitter cannot be placed at the same location as a sensor. A similar model was adopted in [13,16].

Section 3.2 – Fusion Rule

When all the sensor measurements have been transmitted to the Fusion Center, a fusion rule is needed to process the sensor readings and make the final network detection decision. Namely, the fusion rule must decide if the emitter is present in the ROI (hypothesis H_1) or if the emitter is not in the ROI (hypothesis H_0). Fusion rules may come to a decision by looking at all the sensor individually (as in [4,5,32]) or at sensor groups i.e. clusters (as in [13,16-20]). In our research, the fusion rule we will be using is the scan statistic which has also been applied in epidemiological studies [11,12]. Thanks to developments in recent years, the combination of WSN's with this statistical tool has led to the valuable scan statistic fusion rule.

Section 3.3 – The Scan Statistic Fusion Rule

The scan statistic operates on a set of clusters in the ROI. Let C be the individual clusters in the WSN which make up the networks' cluster set $\boldsymbol{C}$. In its original conception, the cluster set $\boldsymbol{C}$ was created using a scanning process where a window is scanned over the region of interest.

Sensors found within the scanning window at any point in time are grouped into a cluster, and a cluster statistic is computed for that cluster using its contained sensor measurements. Here, our goal is to tailor the scan statistic's cluster set $\boldsymbol{C}$ to improve the WSN detection performance. More formally, given sensor measurements $\{Z_k\}_{k=1}^{K}$ from K sensors, the sensor measurements in a cluster C will be represented by $\boldsymbol{Z}_C \coloneqq \{Z_k : k \text{ belongs to } C\}$, and are used to calculate a cluster statistic. In here, we consider the following cluster statistic:

$$T(\boldsymbol{Z}_C) \coloneqq \frac{1}{\sqrt{|C|}} \sum_{k \in C} Z_k \tag{1}$$

(where $|C|$ is the number of sensors grouped in C and is included to normalize the cluster statistic). If the scanning window is used to create $\boldsymbol{C}$, the window would continue to move across the region of interest until it has covered the whole area.

For each cluster in $\boldsymbol{C}$, the cluster statistic is computed and aggregated into the scan statistic, which is the maximum among all cluster statistics. In other words, the scan statistic $S_{\boldsymbol{C}}$ is taken as the maximum of all cluster statistics for the given cluster set $\boldsymbol{C}$:

$$S_{\boldsymbol{C}} \coloneqq \max_{C \in \boldsymbol{C}} T(\boldsymbol{Z}_C) \tag{2}$$

The WSN then decides if an emitter is present by comparing the scan statistic against a threshold "t" using the following fusion rule:

$$\phi(S_{\boldsymbol{C}}) \coloneqq \begin{cases} H_1, \text{ if } S_{\boldsymbol{C}} > t \text{ (event present)} \\ H_0, \text{ if } S_{\boldsymbol{C}} \leq t \text{ (event not present)} \end{cases} \quad (3)$$

where the threshold "t" is chosen to keep the P_{FA} below a maximum P_{FA} requirement. Note that both the scan statistic and fusion rule explicitly depend on the set of clusters $\boldsymbol{C}$ that the WSN is using.

CHAPTER 4

RESEARCH GOAL AND MOTIVATION

Section 4.0 – Motivating Question

The design of the scan statistic's cluster set $\boldsymbol{C}$ to maximize the system's probability of detection is one aspect that has received little attention [13]. Essentially, the problem is: which clusters should be kept or removed from the cluster set $\boldsymbol{C}$ to improve the overall P_D while still keeping the P_{FA} below a maximum value when using the scan statistic?

Section 4.1 – The Multiple Hypothesis Testing Problem

One of the motivating issues for tuning the cluster set is the multiple hypothesis testing problem (MHTP). This problem occurs with the scan statistic because the scan statistic is equivalent to performing multiple individual hypothesis tests, once for each cluster [13]. In more details, as more clusters are formed in the network, the chance of having a false alarm in any of the clusters increases; and, to keep the P_{FA} low, one needs to increase the detection threshold *t*, which in turn causes the global P_D of the system to be reduced. At the same time, the network still needs to form enough clusters to adequately cover the region of interest and achieve the desired level of detection. A balance is thus needed to not have too many or too few clusters in $\boldsymbol{C}$. Furthermore, given the various possible cluster sets in a specific area, some clusters are better at detecting an event at a certain point in space than other clusters. Consider if an event occurs at

the center of a region, a cluster set with clusters at the center of the ROI will detect that event better than a cluster set with clusters positioned at the edges of the area. Ultimately, as illustrated by Fonseca in [13], the specific set of clusters used by the network will directly affect the WSN detection performance.

Section 4.2 – Worst Case Detection Performance

Typically, the WSN does not know where the emitter will be in the region of interest. This unknown factor brings further difficulty to designing the cluster set $\boldsymbol{C}$ since grouping sensors around an emitter is a key aspect in increasing the network's P_D. There are infinitely many possible positions for an emitter in a ROI; however, given a set of known sensor positions, it is reasonable to focus on the points that are farther away from the sensor positions since these points should reduce the detection performance in the neighborhoods of the various sensors. Following [13], we use the term "points of interest" (POIs) to refer to these locations. These points of interest can be found by generating a Voronoi diagram from the sensor positions, Figure 2. The vertices of all the Voronoi polygons (the pink diamonds in Figure 2) are the points that are farthest away from all their surrounding sensors. Given an emitter signal that decays with distance, it is thus expected that the worst-case emitter position will be found at a Voronoi vertex. We define the set $\boldsymbol{POI} \coloneqq \{poi_v : v \in \text{Voronoi vertices}\}$ to be the set of all individual Voronoi vertices in the region of interest, and $|\boldsymbol{POI}|$ to be the total number of individual poi_v in that region. For a cluster set $\boldsymbol{C}$, let its probability of detection with an emitter at a point of interest be $P_D(\boldsymbol{C}, poi_v)$. Fonseca has shown that optimizing the cluster set for the worst-case emitter positions will improve the worst-case cluster set $P_D(\boldsymbol{C}, poi_v)$ [13]. Note that while a cluster set should be optimized for all $poi_v \in \boldsymbol{POI}$ to ensure it covers all the points of interest in WSN,

there will be at least one poi_v that is worse than all other poi_v. Let poi^- be the poi_v at which the P_D of cluster set $\boldsymbol{C}$ is lowest: $P_D(\boldsymbol{C}, poi^-)$. Thus, a cluster set can be optimized for detection by maximizing its $P_D(\boldsymbol{C}, poi^-)$, while keeping its $P_{FA}(\boldsymbol{C})$ below a maximum P_{FA} requirement.

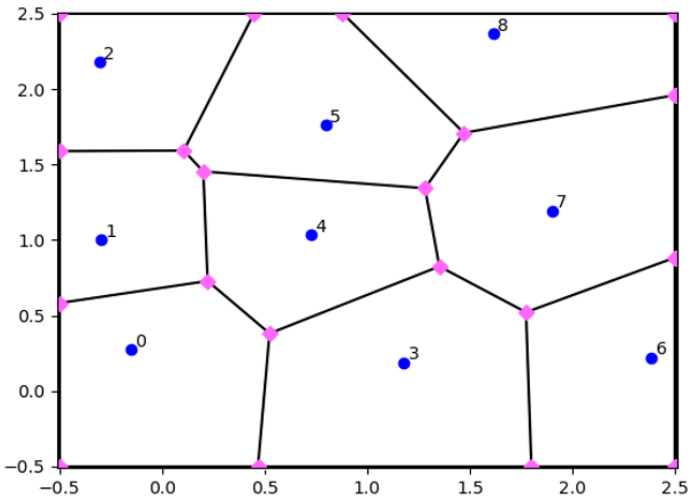

Figure 2. Voronoi polygons generated from sensor positions (blue dots) will have vertices (pink diamonds) at the worst possible emitter positions. We also call these vertex locations points of interest or POI.

Section 4.3 – Research Objective

The focus of our research is on optimizing the cluster set $\boldsymbol{C}$ to maximize its worst-case probability of detection $P_D(\boldsymbol{C}, poi^-)$ for $P_{FA}(\boldsymbol{C}) \leq 5\%$. As pointed out in [13], this is a non-convex problem. A further difficulty is that there are no closed form solutions for the worst-case probability of detection nor for the P_{FA} of the scan statistic. Furthermore, when optimizing the cluster set $\boldsymbol{C}$, the gradient of the worst-case probability of detection is not available since $P_D(\boldsymbol{C}, poi^-)$ is not differentiable with respect to $\boldsymbol{C}$. Therefore, our goal is to build a locally optimal cluster set by using an iterative local search approach [35].

The general procedure of local search to maximize a performance function without computing gradients is shown in Figure 3 [35]. In here, we use a one-dimensional problem to illustrate the search procedure. First, a random initial point in the search space is selected (point "A") and the value of the performance function is computed at that point. The function is then evaluated at the neighbors of the starting point (points "D" and "B"). The neighbor with the highest function value is selected as the new starting point for the next iteration (point "B" becomes the new highest point). The process repeats until a local optimum is found where the cost function does not increase for any neighbor (point "C").

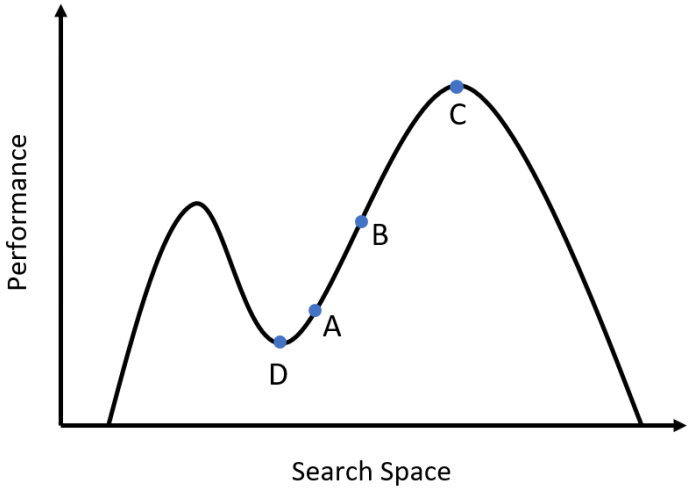

Figure 3. General local search procedure for maximizing a function without computing gradients.

In our case, we are trying to maximize the detection performance of the WSN at its worst poi_v, so our cost function is $P_D(\boldsymbol{C}, poi^-)$, and our search space consists of the different cluster sets that can be used in the WSN. Given a set of all possible clusters for our WSN: $\boldsymbol{C}_{all}$, any subset of $\boldsymbol{C}_{all}$ is a possible cluster set for the WSN. Note, in our implementation, we build $\boldsymbol{C}_{all}$ by scanning multiple windows of different sizes over the WSN and adding each window

grouping of sensor as a cluster in $\boldsymbol{C}_{all}$. Let $|\boldsymbol{C}_{all}|$ be the total number of clusters in $\boldsymbol{C}_{all}$, and let $\boldsymbol{C}_i$ be a cluster set that is a subset of $\boldsymbol{C}_{all}$. More formally: $\boldsymbol{C}_i \subset \boldsymbol{C}_{all}$ for $i = 1\ to\ |\boldsymbol{C}_{all}|$. The local search must thus search in the space of subsets of $\boldsymbol{C}_{all}$ to find the cluster set $\boldsymbol{C}_i$ with the highest $P_D(\boldsymbol{C}_i, poi^-)$ for the given $P_{FA}(\boldsymbol{C}_i)$. Let $\boldsymbol{C}_c$ be the current cluster set the local search is trying to improve, and let $\boldsymbol{C}_f$ be the final local optimum cluster set the search finds with the highest $P_D(\boldsymbol{C}_f, poi^-)$.

Section 4.4 – Problem Complexity

To illustrate the complexity of the optimization problem, note that there are two fundamental steps that must be performed during a local search: the first step is finding neighbor points around the current search point and the second step is evaluating the performance function at any given point. When applied to our cluster set search problem, both steps add their own complexity.

First, recall that we are optimizing the cost function at the worst poi_v. However, we don't know which poi_v is the worst for a given cluster set $\boldsymbol{C}_i$. The simplest way to find poi^- is to evaluate $P_D(\boldsymbol{C}_i, poi_v)$ at each poi_v in the WSN and set poi^- to be the poi_v with the lowest $P_D(\boldsymbol{C}_i, poi_v)$. This extra searching adds computation time to the overall algorithm search time.

Secondly, the cluster set search space grows very quickly with the size of $|\boldsymbol{C}_{all}|$. The more possible clusters a WSN can have, the more possible cluster sets are added to the search space. For example, if a WSN consisted of a simple 3x3 grid of sensors and only 25 possible clusters were created, the total number of unique cluster sets would be: $2^{25} = 33{,}554{,}432$. As the WSN grows and more clusters are added to $\boldsymbol{C}_{all}$, the number of possible cluster sets in the search space becomes enormous.

Thirdly, because cluster sets can have intersecting clusters (sensors may be shared by multiple clusters), the scan statistics $T(\boldsymbol{Z}_C)$ are statistically dependent on each other. Due to this dependent nature of intersecting clusters, we don't have exact expressions for the $P_D(\boldsymbol{C}, poi_v)$ or $P_{FA}(\boldsymbol{C})$ of such cluster sets. This means that performance evaluations have to be estimated through Monte Carlo simulations.

Finally, determining the neighboring cluster set for a given $\boldsymbol{C}_i$ is not straightforward and serves to add further complexity to the local search.

The difficulty of evaluating the detection performance of $\boldsymbol{C}_i$ combined with the extensive cluster set search space also indicates why an exhaustive search for this problem is infeasible.

Section 4.5 – Evaluating Cluster Set Performance

As mentioned above, we don't have an expression for our local search performance function $P_D(\boldsymbol{C}_i, poi^-)$ because $\boldsymbol{C}_i$ can have intersecting clusters with dependent cluster statistics. At the same time, estimating $P_D(\boldsymbol{C}_i, poi^-)$ using Monte Carlo simulation is impractical due to the extra computation required and the difficulty surrounding the threshold t needed to keep $P_{FA}(\boldsymbol{C}_i) = 0.05$. Therefore, we will not be using $P_D(\boldsymbol{C}_i, poi^-)$ as the cost function directly to compare neighbor clusters when performing a local search. Instead, we consider two alternative metrics for a cluster set's detection performance, namely the clusters set's lower bound for the worst case probability of detection ($P_{D_{LB}}(\boldsymbol{C}_i^o, poi^-)$) and the estimate for the worst-case Area Under the Receiver Operating Curve ($AUC(\boldsymbol{C}_i, poi^-)$).

Section 4.5.0 – The Lower Bound for P_D Metric

While we don't have an expression for a cluster set's $P_D(\boldsymbol{C}_i, poi^-)$, Fonseca in [16] presented a lower bound for $P_D(\boldsymbol{C}_i, poi^-)$ that can be computed analytically. Let $\boldsymbol{C}^o$ be a cluster set with strictly non-intersecting clusters. As shown in [16], given the threshold t and the cluster set $\boldsymbol{C}^o$, the probability of detection of $\boldsymbol{C}^o$ can be computed as:

$$P_{D_{LB}}(\boldsymbol{C}^o, poi^-) = 1 - \prod_{C \in \boldsymbol{C}^o} P_{H_1}[T(\boldsymbol{Z}_C) \leq t | L_e = poi^-] \tag{4}$$

where P_{H_1} refers to the probability under H_1. Because we don't have an exact expression for $P_{FA}(\boldsymbol{C})$, we can't compute the threshold t. However, Fonseca in [16] also provides the following upper bound for $P_{FA}(\boldsymbol{C})$:

$$P_{FA_{UB}}(\boldsymbol{C}) \coloneqq 1 - \prod_{C \in \boldsymbol{C}} P_{H_0}[T(\boldsymbol{Z}_C) \leq t] \tag{5}$$

where P_{H0} refers to the probability under H_0. Rearranging the $P_{FA_{UB}}(\boldsymbol{C})$ equation does provide a threshold formula, but the threshold depends on the total number of clusters in the cluster set $\boldsymbol{C}$. So, if two cluster sets have a different number of total clusters, they will require different thresholds to achieve $P_{FA}(\boldsymbol{C}) = 0.05$. Thus, equation 5 can be used to obtain a threshold for equation 4. If $\boldsymbol{C}^o$ is a subset of $\boldsymbol{C}_i$ with only non-intersecting clusters, then the lower bound computed for $\boldsymbol{C}^o$ will also be a lower bound for $\boldsymbol{C}_i$. Let $\boldsymbol{C}_i^o$ be a subset of $\boldsymbol{C}_i$ containing only non-

intersecting clusters. Thus for $\boldsymbol{C}_i^o \subset \boldsymbol{C}_i$, the lower bound for $P_D(\boldsymbol{C}_i, poi^-)$ is given by $P_{D_{LB}}(\boldsymbol{C}_i^o, poi^-)$.

When starting this research, we initially explored the use of the $P_{D_{LB}}(\boldsymbol{C}_i^o, poi^-)$ metric to compare cluster set detection performance. However, we found that the lower bound approach did not produce a tight enough bound to distinguish between two cluster sets well. Furthermore, the computation of the $P_{D_{LB}}(\boldsymbol{C}_i^o, poi^-)$ metric required that the cluster set $\boldsymbol{C}_i$ be simplified to a cluster set $\boldsymbol{C}_i^o$ that includes only non-intersecting clusters. This made it harder to compare very similar cluster sets since they could both be simplified to the same $\boldsymbol{C}_i^o$.

Thus, because of the difficulties in using the $P_{D_{LB}}(\boldsymbol{C}_i^o, poi^-)$ metric to compare cluster set detection performance, we used the Area Under the Receiver Operating Curve (AUC).

Section 4.5.1 – The Worst-Case Area Under the Curve (AUC) Metric

Another metric for evaluating the detection performance of a cluster set is the area under the receiver operating characteristic curve. When the P_D and P_{FA} for a cluster set $\boldsymbol{C}$ are computed for multiple thresholds t, the P_D can be plotted against P_{FA} for each threshold to obtain a receiver operating characteristic (ROC) curve. Thus, for a given desired P_{FA}, the ROC curve can be referenced to determine what is the corresponding P_D. The area under the ROC curve (AUC) indicates how good the overall detection performance of the cluster set is as a whole and a higher AUC generally indicates a higher P_D for a given P_{FA} constraint.

We can conveniently estimate the AUC of a cluster set using a Monte Carlo approach. If M samples of the scan statistic $S_{\boldsymbol{C}}$ are obtained for the cluster set $\boldsymbol{C}$ under both H_0 and H_1, the Wilcoxon-Mann-Whitney statistic can use these samples to generate an estimate for the AUC

[36]. While this estimate does take more computational time, it is simple to compute since it only requires samples of $S_{\boldsymbol{C}}$, does not require a threshold t, and can estimate the AUC of cluster sets with intersecting clusters. As mentioned in [13,36], given a cluster set $\boldsymbol{C}$ and M samples of $S_{\boldsymbol{C}|H_0}$ and $S_{\boldsymbol{C}|H_1}$, its estimated area under the ROC curve $AUC'(\boldsymbol{C})$ can be computed as follows:

$$AUC'(\boldsymbol{C}) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} 1\{S_{\boldsymbol{C}|H_1,i} > S_{\boldsymbol{C}|H_0,j}\}}{M^2} \tag{6}$$

where $1\{S_{\boldsymbol{C}|H_1,i} > S_{\boldsymbol{C}|H_0,j}\} = 1$ if $\{S_{\boldsymbol{C}|H_1,i} > S_{\boldsymbol{C}|H_0,j}\}$ is true and it equals 0 if the condition is false. A simple upper bound for the AUC standard deviation $\sigma_{AUC,UB}$ is also provided in [36] and is given as:

$$\sigma_{AUC,UB} = \frac{1}{2\sqrt{M}} \tag{7}$$

However, when comparing cluster sets, it is not enough to check if one cluster set has a higher AUC than another cluster set. Since the estimate of a cluster set's AUC is a random variable, we need to consider possible random variations when comparing the AUC estimates of the current cluster set and the neighbor cluster set. Following the approach suggested in [37], two cluster set AUCs can be compared by computing the z-score of their difference. After simplifying the z-score equation from [37], the following z-score formula is obtained for comparing two AUC values:

$$z = \sqrt{2M}(AUC_n - AUC_c) \quad (8)$$

where AUC'_n and AUC'_c are the estimated AUC values of the neighbor and current cluster sets respectively. The above equation (8) was achieved by reducing the formula presented in [37] in two ways. First, the samples used for the AUC estimation are uncorrelated, so the correlation coefficient in [37] reduces to zero. Second, to take a conservative approach the upper bound for the estimated AUC standard deviation: $\frac{1}{2\sqrt{M}}$, is used for the standard error. When the z-score is greater than a critical value of 1.96, we can claim with a confidence level of 95% that the neighbor cluster set estimated AUC is better than the current cluster set estimated AUC. On the contrary, a z-score less than -1.96 indicates that the current solution cluster set remains better than the neighbor cluster set. However, if |z| < 1.96, then we cannot claim that either cluster set is better than the other; i.e., a cluster set $\boldsymbol{C}_1$ having an AUC estimate better than the AUC of another cluster set $\boldsymbol{C}_2$ could have happened because of random variations in the Monte Carlo simulations. In this case, we say that the two cluster sets are "indistinguishable".

Because of the possible random variations in the estimate for the AUC of the cluster sets under analysis, our procedure will report not only the best cluster set found, but will also report all cluster sets that were investigated and were indistinguishable from the best cluster set found.

Section 4.6 – Local Search Success Evaluation

Regardless of the chosen metric for comparing cluster sets, this same metric will also be used to judge the overall performance of the local search. Once a locally optimum solution is obtained, its detection performance will be compared against several baseline cluster sets presented in Section 6.0. The performance of the baselines will be used to judge whether the

local search solution was able to significantly optimize the cluster set $\boldsymbol{C}$ to maximize $P_D(\boldsymbol{C}, poi^-)$ as determined by the chosen metric.

Section 4.7 – Searching Neighborhoods

Finally, before introducing our proposed local search in Chapter 5, note that an algorithm will have two primary approaches when deciding how the neighborhood cluster sets will be searched. In general, the algorithm could either evaluate all the neighbor cluster sets and move to the most improving cluster set, or it could evaluate the neighbor clusters sets sequentially and move to the first improving cluster set it encounters. While the most improving approach is desirable, since it makes the biggest improvement jumps, it is also computationally expensive because it must evaluate the entire neighborhood before it tries to make a single improvement. On the other hand, the first improvement method can make faster decisions at the cost of possibly using smaller improvement steps. The smaller search steps may also extend the total search run time if the local search requires too many steps to reach a significant improvement.

Nevertheless, in Chapter 5, we essentially merged the most improving and first improving approaches. We try to use the large improvement step of the first method while sequentially stepping through neighbor cluster sets as in the second method. Furthermore, while most of the search is performed by considering only one neighbor at a time, the algorithms goal is to try to end only if its final cluster set is the best in its entire neighborhood. This is mentioned as an advantage of the first improving approach in [35].

CHAPTER 5

THE PROPOSED LOCAL SEARCH

Section 5.0 – Gradient Improvement

Given the issues and motivation discussed in Chapter 4, we present an approach that resembles a Gradient Ascent Search. The approach described here is not exactly a Gradient Ascent search because the optimization function is not differentiable. In more details, recall that the goal is to optimize the worst case detection performance of a sensor network with respect to its cluster set: maximize $P_D(\boldsymbol{C}, poi^-)$, where we recall that poi^- indicates the worst point of interest, as described in Section 4.2. Since the function $P_D(\boldsymbol{C}, poi_v)$ is not continuous as a function of the cluster set, the objective function is non-differentiable. Thus, the gradient search presented here is not a true gradient search, but it follows a similar philosophy.

In our approach, we search along the direction of greatest improvement of $P_D(\boldsymbol{C}, poi^-)$; i.e., our search tries the cluster set changes that should provide the highest increase in the worst poi_v performance. Before presenting our proposed local search approach, it is first critical to understand how our gradient neighbor cluster sets are created and compared.

Section 5.1 – Neighbor Cluster Sets

The two most fundamental ways to build gradient neighbor cluster sets are to either add clusters or remove clusters from the current solution cluster set. Both ways of changing the

cluster set must be implemented in the search algorithm to allow dense and sparse initial cluster sets the opportunity to converge to a similar cluster set.

Furthermore, since the goal of adding or removing clusters is to improve the performance of the current cluster set at its worst point of interest (poi^{-}), it should be noted that both improvement methods may affect multiple point of interests (POI) at the same time. Adding or removing clusters to improve the performance at one poi_v could worsen the performance at another poi_v. So, gradient improvement approaches must take all the network POI into account when creating neighbor cluster sets.

In the next two subsections, we describe the basic principle behind adding and removing clusters that our algorithm uses.

Section 5.1.0 – Adding Clusters

When trying to improve the current cluster set by adding clusters to it, the new clusters should be added near the poi_v where the current cluster set has the lowest detection performance, i.e. near poi^{-}. Clusters containing sensors close to poi^{-} will have the highest probability of detecting an event at that POI.

To choose which cluster to add to a given poi_v, we create a ranked cluster list for each poi_v. If each cluster in $\boldsymbol{C}_{all}$ is considered individually, and assuming the Gaussian model discussed in Section 3.1, the probability of detection of a single cluster $P_{D,local}(C, poi_v)$ can be computed exactly for any given poi_v and a given threshold t'. For sensor measurements with a Gaussian distribution, the cluster static $T(\boldsymbol{Z}_C)$ will also be a Gaussian random variable, so a single cluster's $P_{D,local}(C, poi_v)$ can be computed using the Q-function:

$$P_{D,local}(C, poi_v) = P_{H_1}[T(\boldsymbol{Z}_C) > t' \mid L_E = poi_v] = Q\left(\frac{t' - \mu_C}{\sigma_C}\right) \quad (9)$$

$$P_{FA,local} = P_{H_0}[T(\boldsymbol{Z}_C) > t'] = Q\left(\frac{t' - \mu_C}{\sigma_C}\right) \quad (10)$$

The threshold t' used to compute the cluster's $P_{D,local}(C, poi_v)$ is found by rearranging the $P_{FA,local}$ equation: $t' = \mu_C + \sigma_C Q^{-1}(P_{FA,local})$. Note that this t' is a threshold used solely to rank the clusters. This is NOT the threshold t the final system would use since that threshold must account for the Multiple Hypotheses Testing problem. In our analysis, if we recall our model from Chapter 3, when no emitter is present we have $\mu_C = 0$ and $\sigma_C = 1$, so the threshold used to rank clusters is $t' = Q^{-1}(0.05)$. Once the $P_{D,local}(C, poi_v)$ is computed for each cluster at the given poi_v, the clusters are ranked from highest to lowest by their $P_{D,local}$ (see Appendix B, Table 1).

After the list is built for each poi_v, the cluster with the highest $P_{D,local}$ at poi^- (and not yet in the cluster set) is added to the cluster set. Since the addition of such a cluster will provide the highest cluster set $P_D(\boldsymbol{C}, poi^-)$ improvement, it can be considered the change of highest gradient improvement for the current cluster set.

However, it is possible that the addition of just one cluster will not improve the cluster set's $P_D(\boldsymbol{C}, poi^-)$, since the new cluster could make another poi_v worse than the current poi^-, or there could have been multiple poi^-. To see this, consider a 3x3 grid of sensors and the cluster set shown in Figure 4. This cluster set has four equal worst poi^- since its $P_D(\boldsymbol{C}, poi_v)$ will be lowest when the emitter is placed at any of the corner poi_v located at: (-0.5, -0.5), (-0.5, 2.5), (2.5, -0.5), and (2.5, 2.5).

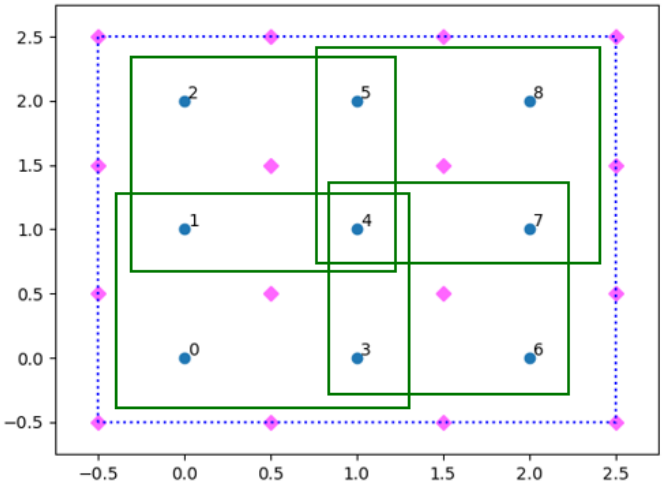

Figure 4. Cluster set with four equally worst points of interest (pink diamonds) located in the corners of the region of interest.

If just cluster {8} is added near the poi_v at (2.5, 2.5), the cluster set's $P_D(\boldsymbol{C}, poi^-)$ at the (2.5, 2.5) poi_v will improve, but the $P_D(\boldsymbol{C}, poi_v)$ at the other three corner poi_v will get worse. The reason the new cluster reduces the performance at the other three poi_v is because the extra cluster increases the cluster set's $P_{FA}(\boldsymbol{C})$ (due to MHTP). To maintain the desired $P_{FA}(\boldsymbol{C})$ a higher threshold t is required. However, the higher threshold reduces the $P_D(\boldsymbol{C}, poi_v)$ of the cluster set at the other three poi_v, and thus the $P_D(\boldsymbol{C}, poi^-)$ of the cluster set gets worse. Additionally, because there are multiple worst poi^-, they would all have to improve at the same time for the cluster set's $P_D(\boldsymbol{C}, poi^-)$ to increase. Overall, as shown in this example, the addition of just one cluster to the worst poi_v may not improve the $P_D(\boldsymbol{C}, poi^-)$ of the cluster set.

Nevertheless, even if the threshold is increased to maintain the desired cluster set $P_{FA}(\boldsymbol{C})$ and multiple worst poi_v existed, the addition of proper clusters can still improve the $P_D(\boldsymbol{C}, poi^-)$. A cluster set may be improved if we try adding multiple good clusters simultaneously. Adding the corresponding highest $P_{D,local}$ clusters to all the N worst poi^- allows each of these poi^- to

improve enough such that the $P_D(\boldsymbol{C}, poi^-)$ increases despite the higher threshold. In the above example, the best clusters for each of the four worst poi_v should be added simultaneously to improve the cluster set's $P_D(\boldsymbol{C}, poi^-)$. When this is done, the new $P_D(\boldsymbol{C}, poi^-)$ is still at the four corner poi_v, but it is higher than the original $P_D(\boldsymbol{C}, poi^-)$.

Section 5.1.1 – Removing Clusters

As explained above, a cluster set with many clusters will need a high cluster set threshold to keep a low $P_{FA}(\boldsymbol{C})$. Since the high threshold also reduces the cluster set's $P_D(\boldsymbol{C}, poi_v)$, a good cluster set must try to have as few clusters as possible to reduce the cluster set $P_{FA}(\boldsymbol{C})$, and thus increase its $P_D(\boldsymbol{C}, poi^-)$. However, we don't know if a given cluster set has too many clusters, so naturally our local search must try removing clusters as another possible cluster set improvement approach.

When removing clusters to improve the $P_D(\boldsymbol{C}, poi^-)$ of a cluster set, a different cluster ranking approach must be used. A gradient-like improvement by removal is achieved when removing the clusters that are worst for <u>all</u> POI, not worst for just a single poi_v. To see this, consider the simple cluster set show below for a 3x3 sensor grid (Figure 5).

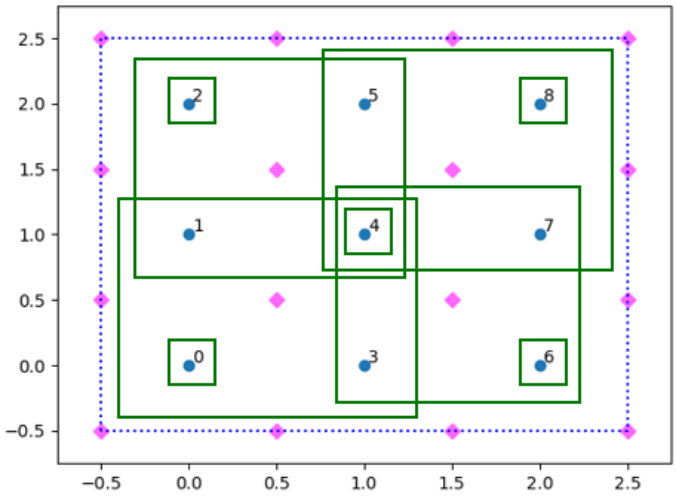

Figure 5. Cluster set where a single sensor corner cluster is best for only one POI and bad for most other POI (pink diamonds). Removing a cluster that is best for only one POI will hurt that POI, so the best cluster to remove in this cluster set is cluster {4}.

Assume we ranked all the clusters by the cluster's $P_{D,local}$ at each poi_v, and thus 16 ranked cluster lists were created (see Appendix B, Table 1). We then compute the cluster set's $P_D(\boldsymbol{C}, poi_v)$ at each poi_v and find that the poi_v at (-0.5, 2.5) is one of the worst poi^- since the cluster set has the lowest $P_D(\boldsymbol{C}, poi^-)$ when the emitter is at that poi^-. The ranked cluster list for this worst poi^- (see row 4 in Appendix B, Table 1) indicates that when an emitter is placed at this poi^-, cluster {2} has the highest $P_{D,local}$ and cluster {6} has the lowest $P_{D,local}$. Cluster {6} is the worst cluster for this poi^- because its sensors are furthest from this top left corner point of interest. However, if cluster {6} is removed, the cluster set's $P_D(\boldsymbol{C}, poi^-)$ would worsen because although cluster {6} is the worst cluster for the poi^- at (-0.5, 2.5) it is also the best cluster for the poi^- at (2.5, -0.5). Therefore, when deciding which cluster to remove, the cluster's $P_{D,local}$ rank at all POI must be considered.

To account for all the POI, we use a penalty function to indicate which clusters should be removed first. Clusters that have a low $P_{D,local}$ at most poi_v should have a small removal penalty, while cluster with a very high $P_{D,local}$ at any poi_v should carry a large removal penalty. The cluster with the smallest penalty is removed to create a gradient improvement neighbor cluster set.

Each cluster's penalty value is computed as a function of that cluster's $P_{D,local}$ ranks at all POI. To illustrate this, consider the cluster set shown in Figure 5. After ranking all the clusters by their $P_{D,local}$ at each poi_v, cluster $\{0\}$ has a rank $r = 0$ for the poi_v at (-0.5, -0.5), $r = 1$ for the poi_v at (-0.5, 0.5), $r = 6$ for the poi_v at (-0.5, 1.5), and so on (see Table 1 in Appendix B for all cluster rankings). A list is then created for each cluster C containing that cluster's ranks from all POI (see Appendix B, Table 2). Given cluster C, let $\boldsymbol{R}_C$ be the list containing the cluster's $P_{D,local}$ rank at each poi_v: $\boldsymbol{R}_C = [r_{poi_1}, r_{poi_2}, r_{poi_3}, \dots]$. In our example, cluster $C = \{0\}$ would have the ranks list: $\boldsymbol{R}_{\{0\}} = [0_{(-0.5,-0.5)}, 1_{(-0.5,0.5)}, 6_{(-0.5,1.5)}, \dots]$ or more simply $R_{\{0\}} = [0,1,6, \dots]$. A rank of 0 indicates that a cluster is the best cluster for a poi_v, and the largest rank value indicates the worst cluster for a poi_v. So, if a cluster has an $\boldsymbol{R}_C$ with many small rank values, then that cluster is very beneficial for the cluster set since it has high $P_{D,local}$ at many poi_v. If two clusters have the same $P_{D,local}$ for a poi_v (for example clusters $\{0\}$ and $\{8\}$ for the poi_v at (2.5, -0.5) in Figure 5), those clusters should have the same rank.

Before assigning a penalty value to each cluster, we partition the clusters into "primary" and "secondary" cluster groups. The primary clusters are those that have the highest $P_{D,local}$ (rank = 0) at any poi_v, and secondary clusters are all other clusters. This separation is done to ensure that the best cluster for a poi_v is removed last and to allow for slightly different penalty functions for the two cluster groups. Recall, we want to rank clusters by their removal penalty

and remove the clusters with the smallest penalty. When assigning a removal penalty to a secondary cluster, the cluster should have a higher removal penalty if it usually has high $P_{D,local}$ values at most poi_v when compared to other secondary clusters. However, primary clusters should have high removal penalties when their $\boldsymbol{R}_C$ ranks are seldomly better relative to other primary clusters. Primary clusters that are good at only a few poi_v are still very important to those poi_v, while clusters that are good at many poi_v are not as essential since other cluster may already be covering the same poi_v. An example of a primary cluster that has a high $P_{D,local}$ at only a few poi_v is cluster $\{8\}$ in Figure 5. This cluster is the best for the corner poi_v at (2.5, 2.5) but is outperformed by most other clusters at the remaining poi_v. So, primary clusters that have fewer ranks close to 0 must have a higher removal penalty.

Despite the "primary" and "secondary" clusters needing to be penalized differently, both of their penalty functions assign a penalty to each cluster based upon the ranks that cluster has at all POI, i.e. based upon $\boldsymbol{R}_C$. We used the following penalty function in equation 11 to assign a penalty score to clusters in the "primary" and "secondary" cluster groups.

$$G(C) = \begin{cases} \sum_{r \in \boldsymbol{R}_C} r \text{ if } C \in CS_{primary} \\ \sum_{r \in \boldsymbol{R}_C} \frac{1}{r} \text{ if } C \notin CS_{primary} \end{cases} \tag{11}$$

where $CS_{primary}$ indicates the group of primary clusters at this step.

Having given each cluster in both cluster groups a penalty score, clusters with the smallest removal penalty in the secondary group are removed first, followed by clusters with the smallest penalty in the "primary" group. Removing clusters in this order will create what we

consider gradient improvement neighbor cluster sets. Furthermore, just like in the adding clusters approach, it may be necessary to remove multiple clusters simultaneously to improve the cluster sets $P_D(\boldsymbol{C}, poi^-)$.

Section 5.2 – Finding Multiple Cluster Sets Solutions

Recall that we are trying to optimize cluster sets with intersecting clusters and exact formulas for the $P_D(\boldsymbol{C}, poi^-)$ or $P_{FA}(\boldsymbol{C})$ of a cluster set don't exist. Nevertheless, we can still estimate the detection performance of these cluster sets by using Monte Carlo simulations. We chose to use the AUC of a cluster set as our metric, since it does not require us to provide a threshold for detection and can handle intersecting clusters. As mentioned in Section 4.5.1, the AUC can be readily estimated with the Wilcoxon-Mann-Whitney statistic, so our gradient search will optimize $AUC(\boldsymbol{C}, poi^-)$ instead of $P_D(\boldsymbol{C}, poi^-)$. Furthermore, because our gradient search approach does not search the entire neighborhood of a cluster set, but rather only compares the current cluster set's performance against the most improving neighbor clusters set, the computation time when using the AUC becomes more manageable.

As explained in Section 4.5.1, because cluster sets are evaluated based on AUC estimates, some cluster sets can't be distinguished by their AUC detection performance. For a z-score confidence level of 95%, two cluster sets are indistinguishable if their z-score value falls between ± 1.96. Thus, when a search algorithm finds two indistinguishable cluster sets, it can't claim that one is better than another and should report both cluster sets as possible solution cluster sets. Consider for example the symmetric cluster sets in Figure 6. Even though the two cluster sets are unique, they still have the exact same worst-case AUC. Likewise, if two large cluster sets are exactly the same but differ in only one cluster, then their worst-case AUC will

most likely be indistinguishable as well. Given the various possible combinations of cluster shapes and sizes in $\boldsymbol{C}_{all}$, it is unsurprising that diverse cluster sets may still yield similar worst-case AUC estimates.

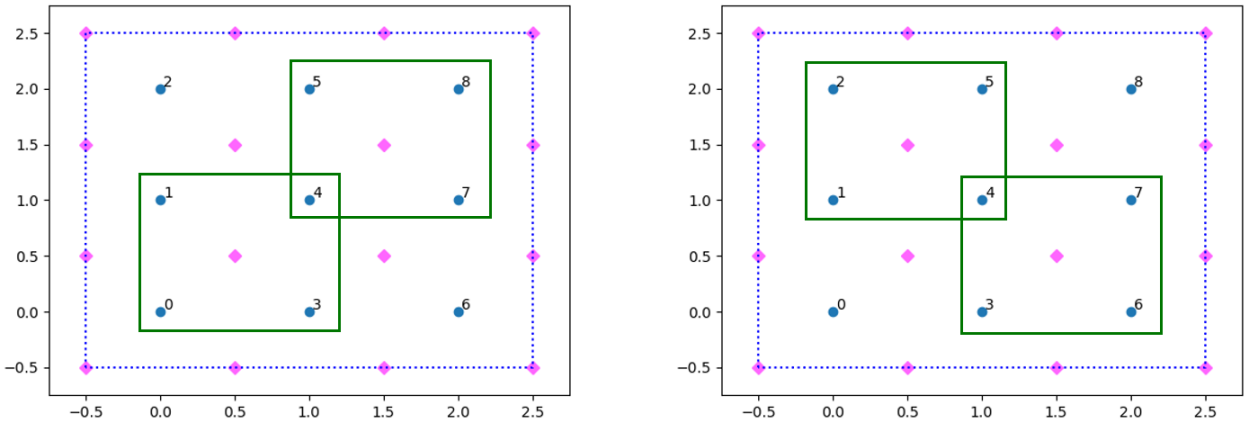

Figure 6. Two cluster sets that are unique but will still have the exact same worst-case detection performance at the POI (pink diamonds).

When a search algorithm finds at least two indistinguishable cluster sets, we say that the search has found an "AUC plateau" in the cluster set search space. The plateau of indistinguishable cluster sets may be more than two cluster sets long, and the WSN designer would benefit from examining the different resulting cluster sets. Thus, a search should continue looking at other neighbor cluster sets to determine if there are more cluster sets indistinguishable from the cluster set with the highest AUC. A local search algorithm that compares cluster sets based on Monte Carlo simulations should therefore return not only the final (locally optimum) cluster set found, but also a list of all cluster sets that are indistinguishable from the final cluster set.

While looking for more neighbors that are indistinguishable from the current solution cluster set, a gradient search can have three possible outcomes. First, if the new neighbor cluster set has an AUC that is much worse than the plateau AUC ($z < -1.96$), the search ends since the AUC is no longer improving in the direction of most improvement. In this case, all cluster sets in the plateau are part of a local AUC maximum. Secondly, if the new neighbor cluster set is much better than the current highest AUC ($z > 1.96$), the search must reset the list of plateau cluster sets and restart the search from its newly found solution cluster set. Finally, if the new neighbor cluster set AUC is indistinguishable from the plateau AUC, the neighbor set should be appended to the plateau list. However, even if the neighbor AUC is indistinguishable from the current highest plateau AUC, the neighbor AUC may still be higher than some of the current plateau cluster sets. For example, consider Figure 7 which shows cluster sets being compared and where the circles represent the cluster set AUC and the vertical lines passing through the circles represent the respective AUC variance. In figure 7, when the AUC of cluster set (CS) 1 is compared to CS 2, the AUC of CS 2 is clearly higher, so cluster set 2 becomes the current solution cluster set. When CS 2 is compared to CS 3, both cluster sets are indistinguishable due to AUC estimation randomness, so both are added to the plateau list. However, when CS 4 and CS 5 are compared to CS 3, they are found to be indistinguishable from CS 3 but clearly better than CS 2, and cause CS 2 to be removed from the list. Therefore, when indistinguishable clusters sets are being added to the list of solution cluster sets, the list must be refined with the addition of each new highest AUC cluster set. The final list of solution cluster sets must contain only the cluster sets that are indistinguishable from the final highest AUC cluster set.

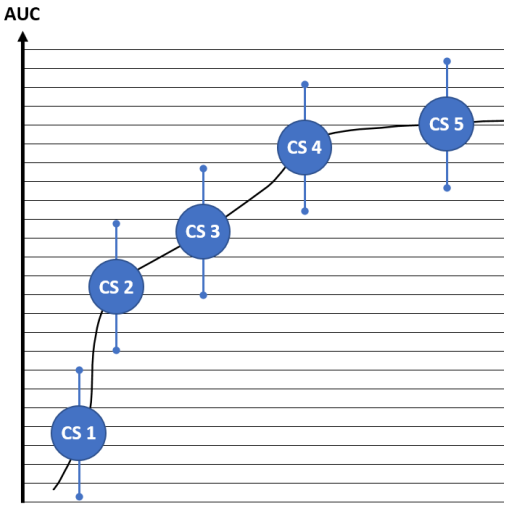

Figure 7. As cluster sets are search, some will be indistinguishable from each other due to estimation variance. However, as the search progresses, a new cluster set could be indistinguishable from the highest AUC cluster sets but distinguishable from the lowest AUC cluster set.

Once the gradient search finishes and returns its list of solution cluster sets, the WSN designer may choose to end the analysis or refine it further. If the AUC of the best cluster set in the final cluster set list is satisfactory, it can be selected as cluster set to use in the scan statistic. On the other hand, if additional analysis is desired, the best cluster set in the final cluster set list can also be used as a starting point for another gradient search using a higher number M of Monte Carlo drops, which reduces the AUC standard deviation and allows for better cluster set comparison.

Section 5.3 – Proposed Algorithm

In this section, we implement the gradient search principles presented above and describe our proposed gradient search algorithm in detail.

The core idea is to add and remove clusters to the highest AUC cluster set following the principles described in Sections 5.1.0 and 5.1.1 until the AUC stops improving. In general, the algorithm sequentially tries adding a group of 1, 2, 3, …, N clusters to the current solution cluster set until the AUC doesn't improve from adding any group of clusters. Once adding N clusters fails, the search tries removing a group of 1, 2 ,3, …, N clusters from the solution cluster set. After both improvement approaches stop increasing the AUC, the local search reports the cluster set with the highest AUC as well as all found cluster sets that are indistinguishable from the highest AUC. Whenever a better cluster set is found during the search procedure, the algorithm restarts the search from the new solution cluster set by adding and removing groups of 1 to N clusters from the new starting point. Finally, to give the local search more chances at finding the local maximum, the search is repeated multiple times with random initial cluster sets. The final lists from all the searches can then be filtered to obtain their highest AUC cluster set as well as all cluster sets indistinguishable from the highest AUC cluster set. The step by step search procedure is described below.

Algorithm Terms:

- Let SCS = Solution Cluster Set, the cluster set with the highest AUC so far
- Let NCS = Neighbor Cluster Set.
- Let N = the number of clusters to "add to" or "remove from" the SCS to create the NCS.
- Let ITT = "Improvements To Try", a list keeping track of which improvement approaches (such as adding clusters or removing clusters) have not been used to create a NCS.
- Let FL = "Final List", the list containing the highest AUC cluster set and all cluster sets indistinguishable from it.

Search Algorithm:

Initialize

1. Set initial cluster set as SCS.
2. Add SCS to FL.
3. Set $N = 1$.
4. Set N_{max} and FL_{max}.
5. Set ITT to all improvement approaches to try.

Search

6. Use first ITT entry to add/remove N clusters to SCS to create NCS.
7. Compute detection performance improvement metric.
8. Check for improvement.
    a. If NCS is better than or indistinguishable from SCS.
        i. Set $N = 1$ if NCS is better than SCS else set $N = N + 1$.
        ii. Reset ITT if NCS is better than SCS.
        iii. Update FL and SCS.
        iv. Return to step 6.
    b. If NCS is worse than SCS.
        i. Set $N = N + 1$.
        ii. Return to step 6.
9. If $N = N_{max}$ and $len(ITT) > 1$.
    a. Remove first ITT entry.
    b. Set $N = 1$.
    c. Return to step 6.
10. If $N = N_{max}$ and $len(ITT) = 1$.
    a. End search.
11. If $len(FL) = FL_{max}$.
    a. End search.

Recall that the search algorithm is repeated using multiple random initial cluster sets. After the search has been run with the desired number of random starts, the final cluster set lists from all the runs are combined and filtered to reveal the overall highest AUC cluster set as well as all cluster sets indistinguishable from it.

Section 5.4 – Notable Algorithm Details

The following are noteworthy details of the algorithm:

- Note that an ITT list is used to keep track of all the possible improvement approaches the algorithm should try when it builds a most improving neighbor cluster set. For example, the ITT list could be: $ITT =$ ["add_clusters", "remove_clusters"]. At any point in the search, the first element in the ITT list is the approach the algorithm uses to build the neighbor cluster set. Once the approach of adding N clusters doesn't improve the AUC, the first element in the ITT list is removed and the second element ("remove_clusters") becomes the current improvement approach. However, whenever a better cluster set is found, the ITT list should be reset, so that when returning to step 6 the search starts from the new solution cluster set. When resetting the ITT list, any tried improvement approaches should be appended to the end of the ITT list. This is to allow the search to finish searching along the current improvement approach before trying the other approach again. To illustrate this, assume the ITT list was initially: $ITT =$ ["add_clusters", "remove_clusters"]. After the algorithm finishes the "add_clusters" approach, the ITT list would become: $ITT =$ ["remove_clusters"]. If an improvement is found during the "remove_clusters" approach, the ITT list would be reset to: $ITT =$ ["remove_clusters", "add_clusters"] and if an improvement is found during the "add_clusters" approach, the list would be reset to: $ITT =$ ["add_clusters", "remove_clusters"]. Once all ITT approaches don't improve the current solution cluster set, the search ends.
- Another important aspect of the algorithm is when it decides to increment N. Recall that N is the number of clusters being added or removed to create the neighbor cluster set

(NCS). If the algorithm builds the NCS by adding or removing N clusters and the NCS is worse than the current solution cluster set (SCS), then N should be increased to try the next gradient improvement cluster set. However, if the NCS built by adding or removing N clusters is better than the SCS, the NCS become the SCS and N should be reset to 1 so that the search can restart at step 6 using the new SCS. A special case occurs when NCS is indistinguishable from SCS. Because we don't know if NCS is better or worse than SCS, we don't know if N should be increased or reset to 1. So, in the case of indistinguishable cluster sets both are added to the final list, but we consider the raw AUC (ignoring AUC standard deviation) of the cluster sets in order to decide which cluster set is better and whether N should be incremented. If the raw AUC of NCS is greater than the raw AUC of SCS, then the neighbor cluster set becomes the current solution cluster set, N is reset to 1, and the ITT list is reset. Otherwise, if the raw AUC of SCS is greater than that of the NCS, the SCS is not changed and N is incremented.

- Another detail worth mentioning is regarding how the final list (FL) is updated. During the search, the final list stores the cluster set with the highest AUC and all the cluster sets that are indistinguishable from it. As discussed in section 5.2, if a new cluster set is added to the final list, it should be indistinguishable from the highest AUC cluster set. However, if the new cluster set AUC is higher than the list's highest AUC and indistinguishable from it, the new cluster set AUC may be conclusively better than some of the other cluster sets in the list. So, whenever the final list is updated with a new highest AUC cluster set, every cluster set already in the list should be re-checked to ensure they are indistinguishable from the cluster set with highest AUC in the list. Recall that two cluster sets are indistinguishable if the z-score of the difference of their AUCs meets the

condition: $|z| < 1.96$. Thus, when a list is re-checked each of its cluster set AUCs must be compared against the highest AUC cluster set to ensure it still satisfies this condition.

Section 5.5 – Algorithm Optimizations

One of the most pressing issues when performing a local search over a non-convex space is ensuring the search can be completed in a reasonable amount of time. Thus, to expedite our local search, we recommend the following optimizations to improve the computation time and search performance of the core algorithm.

First, the most basic optimization is to save all clusters sets that are created along with their respective AUC in a look up table. When building neighbor cluster sets by adding and removing groups of clusters, the same neighbor cluster set might be created twice. So, to keep the algorithm from recomputing the AUC metric for the same neighbor cluster set multiple times, a look up table can be used to reduce the total computation time.

On another note, recall that neighbor cluster sets are built by adding or removing 1 to N_{max} clusters from the current solution cluster set. The N_{max} upper limit for the number clusters to add and for the number of clusters to remove should be set to different values. Because the neighbor cluster sets built by adding clusters are created by adding N clusters to the N worst poi_v, the max number of clusters that can be added is equivalent to the total number of POI in the WSN: $N_{max,adding} = |POI|$. On the other hand, when building neighbor cluster sets by removing clusters, the limit on the number of clusters that can be removed depends on the total number of clusters in the current solution cluster set. Since it is unreasonable to create an empty cluster set, the total number of clusters that can be removed should be the number of clusters in the current SCS less one: $N_{\text{max,removing}} = \text{len}(\text{SCS}) - 1$.

Similarly, when adding indistinguishable cluster sets to the final list (FL), an upper limit is needed for the final list size, FL_{max}. If a certain SCS has many neighbor cluster sets with very close AUC values, the final list may continue to grow and extend the search time. So, to help limit the algorithm's execution time, the search is terminated if the final list reaches its maximum size (see step 11 in the algorithm description). Given the total number of poi_v and the size of the initial cluster set, we recommend setting FL_{max} to the larger of the two: $FL_{max} = \max(|POI|, len(\boldsymbol{C}_{initial}))$. By setting $FL_{max} = |POI|$, the algorithm is guaranteed to try creating all neighbor cluster sets by adding clusters. Likewise, if $FL_{max} = len(\boldsymbol{C}_{initial})$, the algorithm is guaranteed to try building all neighbor cluster sets by removing clusters. Thus, setting FL_{max} to the larger of the two options ensures that both improvement approaches are fully explored.

Another recommended optimization is randomizing the elements in the ITT list. Because the main objective of the algorithm is to perform a local search, random initial clusters sets are used to set the start search point in the non-convex search space. To add additional randomness to the search, the improvement approaches in the ITT list can be randomized to vary the initial improvement direction the algorithm follows.

Finally, if the search ends due to its final list reaching maximum size, an important optimization would be to guarantee that the algorithm always tries both optimization approaches. Step 11 in the algorithm description conditions the search to stop once the final list size equals FL_{max}. However, if maximum FL size is reached while having tried only one improvement approach, the second approach is not given the chance to improve the SCS. Thus, as a final optimization, we recommend that if the final list reaches its maximum size and the ITT list contains more than one improvement approach, the FL_{max} limit should be doubled and the first

element in the ITT list should be removed. This allows the search to explore the second improvement approach and makes room in the final list for more indistinguishable cluster sets found using the second approach.

Section 5.6 – Understanding the Algorithm Output

Upon completing the local search, the presented algorithm returns the final list containing the highest AUC cluster set as well as all cluster sets it encountered that were indistinguishable from the highest AUC. For further analysis of the final list, the user should pay attention to a few key output variables to guide their decisions.

When the algorithm terminates due to the final list reaching it maximum size, the number of improvements that occurred during the search should be examined. An improvement takes place whenever a new cluster set is added to the final list and the list size either shrinks or is reset. Such a list change indicates that a new cluster set was better than the current solution cluster set or better than some of the other cluster sets already in the final list. So, if the search ends with a full final list, the number of improvements should be considered to see how much searching the algorithm actually did. If the final list is full and few improvements were made during the search, it indicates the algorithm found an area in the search space with many similar local AUC maxima. On the other hand, if the final list is full and many improvements occurred, it shows that the algorithm actively explored the search space to find the current highest AUC cluster set.

As mentioned previously, after analyzing the final cluster sets list, the user may simply choose to use the highest AUC cluster set as the final solution cluster set. However, if a more refined final list is desired, a higher number of Monte Carlo drops to estimate the AUC may be

selected, and the user can recompute the performance of all the cluster sets in the final list. Likewise, the user could also rerun the entire local search using the highest AUC cluster set as a new starting point with a higher number of Monte Carlo drops for better cluster set discrimination.

CHAPTER 6

ANALYZING ALGORITHM PERFORMANCE

Section 6.0 – Verifying the Results

To verify the performance of the proposed algorithm, the algorithm was implemented and used to optimize the scan statistic in sensor networks of different sizes with both grid and random sensor position distributions. The success in increasing the minimum detection performance was determined by comparing our proposed algorithm's final highest AUC cluster set against four baseline cluster sets. The four baselines are described below and illustrated in Figure 8 for the 3x3 sensor grid scenario.

- **Baseline #0:** The most fundamental cluster set a WSN can have is formed by simply making each sensor into an individual cluster. The baseline #0 thus consists of all single sensor clusters.
- **Baseline #1:** Adopted from a key paper by Guerriero [14] on the scan statistic, this cluster set is created by scanning a window across the sensors and forming a cluster from all the sensor encompassed by the window. The window's size is fixed, and it is confined to scanning within the WSN area of interest. In the implementation, a reasonable scanning window size was chosen such that the clusters created were two clusters wide and two clusters high. When the sensors are distributed randomly, only the clusters with exactly four sensors were used to keep the same general approach as in the grid case.

- **Baseline #2:** For the sensor grid case, the cluster set is created by selecting all the single sensor clusters in the corners of the sensor distribution, all the two sensor clusters along the edge of the sensor distribution, and all the clusters in the center of the sensor distribution that are two sensor wide and two sensor high. This cluster set can be formed by scanning a fixed size window across the network and allowing the window to start and end outside the WSN boundaries [13]. However, due to the difficulty in creating a reasonable cluster set for the random sensor distribution, all the clusters formed by the scanning window are used in the random scenario.
- **Baseline #3:** This baseline cluster set is formed by combining all clusters from $\boldsymbol{C}_{all}$ into one cluster set. Recall that $\boldsymbol{C}_{all}$ is formed by scanning multiple windows of different sizes across the WSN.

Section 6.1 – Simulation Setup

In the following analysis, all sensor placed in a grid were separated by consecutive integer spacing. Recall that when an emitter is present, a sensor's measurement depends on the sensor's distance to the emitter and on the emitter's amplitude and decay rate, i.e. $Z_k \sim N(\frac{A}{d^\gamma}, 1)$. The emitter in all scenarios was given a decay of $\gamma = 2$, and its amplitude A was adjusted to achieve an AUC between 0.70 and 0.75 in the Baseline #2 cluster set. Once fixed, this amplitude level was used in all AUC computations and comparisons. All AUC estimates were computed using $M = 100{,}000$ Monte Carlo drops to achieve an AUC standard deviation around 0.001 as given by the standard deviation upper bound $\sigma_{AUC,UB} = \frac{1}{2\sqrt{M}}$ in [36].

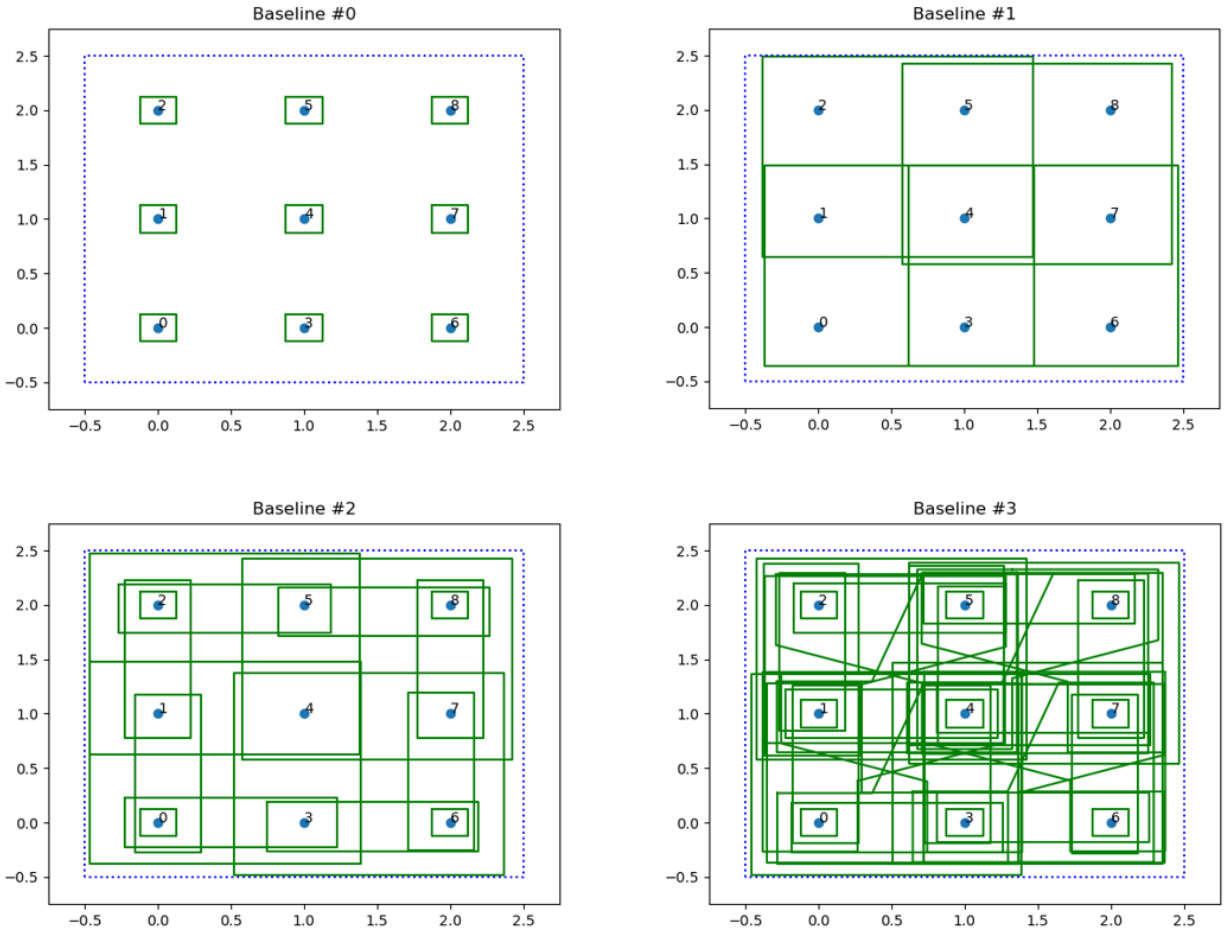

Figure 8. The four Baseline cluster sets used to evaluate the performance of the final local search cluster set. Baseline #0 is all single sensor clusters. Baseline #1 is all clusters of a fixed size. Baseline #2 is a scanning window cluster set. Baseline #3 is all clusters created using different sized scanning windows.

When finding the solution set of our proposed local search algorithm, the search was run multiple times and the highest local optimum was chosen. More specifically, our algorithm was repeated 12 times with 12 different initial cluster sets. The first search started from an empty cluster set and the second started from the Baseline #2 cluster set. The remaining 10 runs used random initial cluster sets. These random sets were built by starting with an empty set and sequentially adding a new cluster to a sensor not covered by the set until all sensors were included in the cluster set. The final lists from the 12 searches were then combined and the AUC difference z-score was used to compare all these cluster sets against their overall highest AUC cluster set. Once the final list is processed using the z-score, the list should contain only the overall highest AUC cluster set and all cluster sets indistinguishable from it. By running the local search multiple times, it was given more chances to find a better local optimum. Note that our local search implementation used all the optimizations described in section 5.5 and also used a critical z-score value of 1.96 to obtain a 95% confidence level.

Section 6.2 – The Grid Network Case

The local search approach was first tested with sensors placed in grids of varying sizes. For sensor network sizes ranging from 3x3 grids to 10x10 grids, the worst-case AUC estimate for the four baseline cluster sets and the cluster set produced by our local search algorithm were

computed. As mentioned previously, the local search was run 12 times for each grid size before choosing the highest local optimum. Comparing the relative worst-case AUC estimates of the four baselines with the local search cluster sets for each grid size in Figure 9, it is possible to observe that our local search outperforms all the baseline cluster sets.

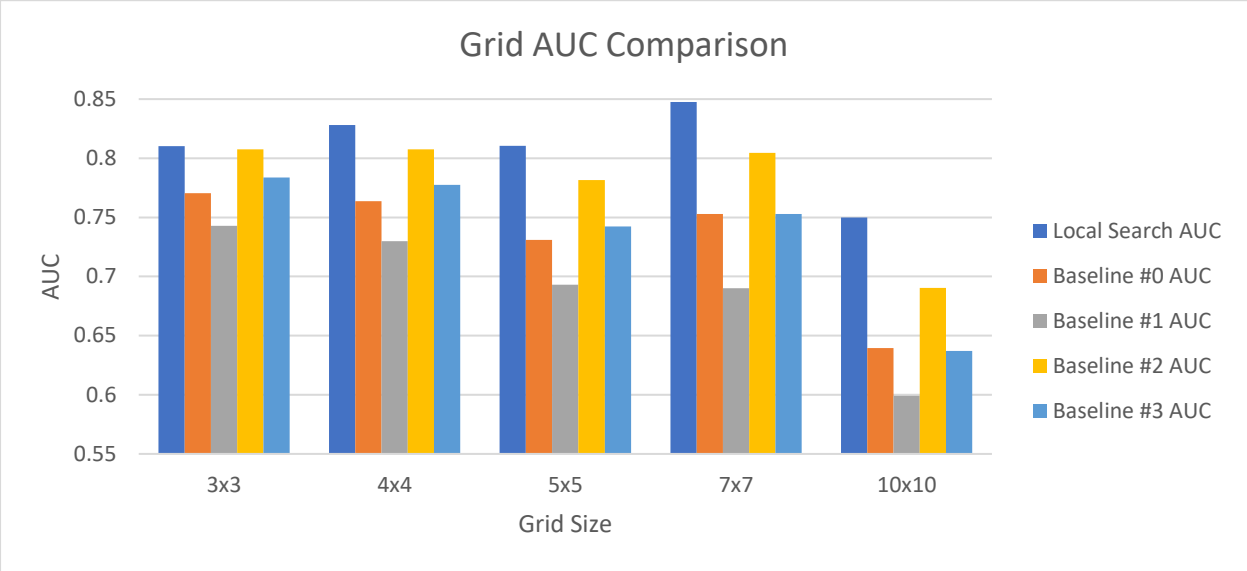

Figure 9. Comparison of detection performance of local search cluster sets against all Baseline cluster sets.

More precisely, in the 10x10 grid case, our proposed algorithm had an estimated AUC of 0.750 ± 0.001 while the best baseline (Baseline #2) had an estimated AUC of 0.690 ± 0.001 (see Figure 10 for a comparison of the ROC curves). This improvement in detection performance is also observed at all relevant emitter amplitudes (see Appendix C).

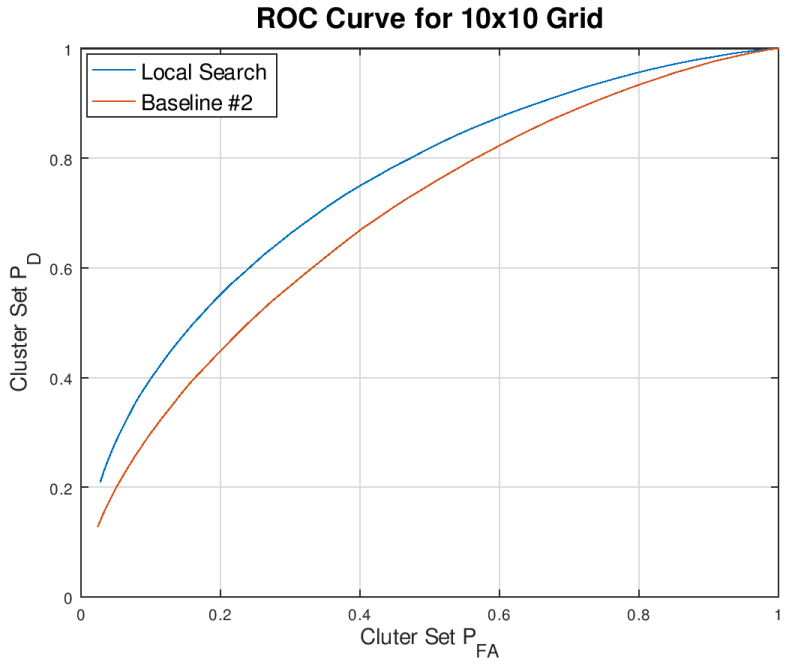

Figure 10. Comparison of ROC curves for 10x10 grid of sensors. Local search AUC is clearly shown to improve over the best baseline cluster set AUC at the worst POI.

For all grid sizes, Baseline #2 is the second best cluster set because it builds its cluster set by adding very reasonable clusters to account for all the grid poi_v. However, despite the reasonable clusters in Baseline #2, the local search cluster sets improved the minimum worst case AUC estimates for all grid sizes. Aside from the 3x3 grid, our local search approach exceeded the worst-case AUC estimate of Baseline #2 by at least 9 standard deviations, suggesting that the actual worst-case AUC is indeed better than the actual worst-case AUC of Baseline #2 and all other baselines. Furthermore, as shown in Figure 11, as the network grid size increases, the improvement of the local search cluster set over the Baseline #2 cluster set also increases. This is a promising trend which suggests the local search cluster set may offer even more improvement for larger grid sensor networks.

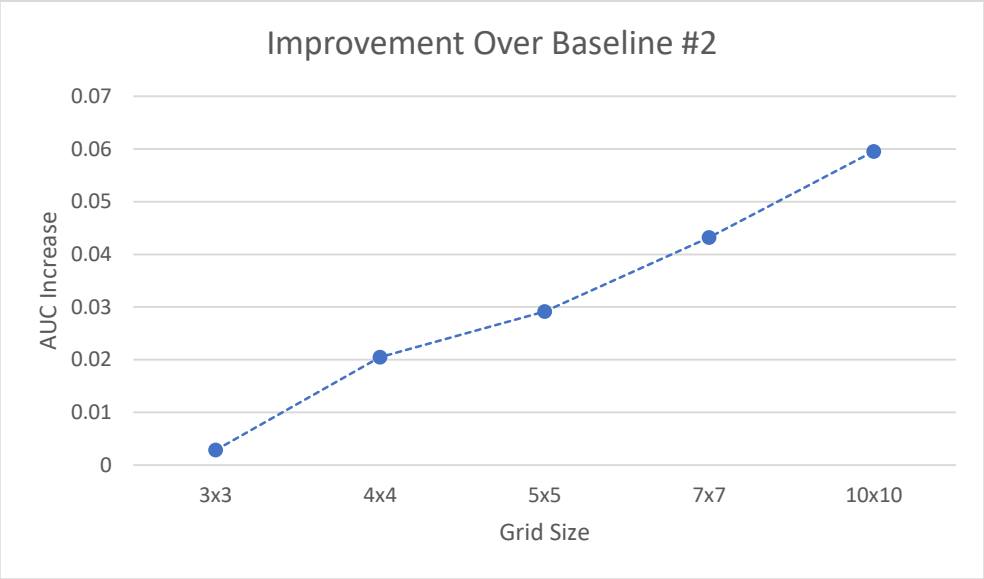

Figure 11. Local search cluster set detection performance improves over the Baseline #2 cluster set as the sensor network size increases.

To give a better insight on why the cluster set produced by our local search method is better than the baseline clusters sets, Figure 12 compares the Baseline #2 cluster set for a 7x7 sensor grid against the corresponding highest worst-case AUC cluster set found after 12 local searches. One of the most noticeable differences is that the local optimum cluster set has 30 clusters while Baseline #2 has 64 clusters. Using fewer clusters helped the local search increase its worst AUC from 0.804 ± 0.001 of the Baseline #2 cluster set to 0.848 ± 0.001 of the cluster set found by our local search because reducing the number of clusters served to reduce the effects of the Multiple Hypothesis Testing Problem (MHTP). Recall, the MHTP causes the P_D of a cluster set to suffer since using more clusters raises the scan statistic threshold and making it harder to detect the presence of the emitter [13]. Furthermore, notice that the local search cluster set prioritized using smaller clusters at the corners and edges of the region of interest while using larger clusters to cover the sensors at the center. Placing smaller clusters at the corners and edges

allows the cluster set to better detect an emitter placed at the most distant poi_v since the smaller and local clusters are the most relevant to the distant poi_v. Meanwhile, the poi_v in the center of the grid are surrounded by sensors and are more readily detected by practically any nearby clusters. In general, our proposed local search algorithm reduces the number of clusters near the center of the ROI and adds better clusters to the edges where the detection performance is worse. Essentially, the local search optimizes the cluster set by reducing the detection performance at the center of the WSN to improve the minimum detection performance at edge poi_v.

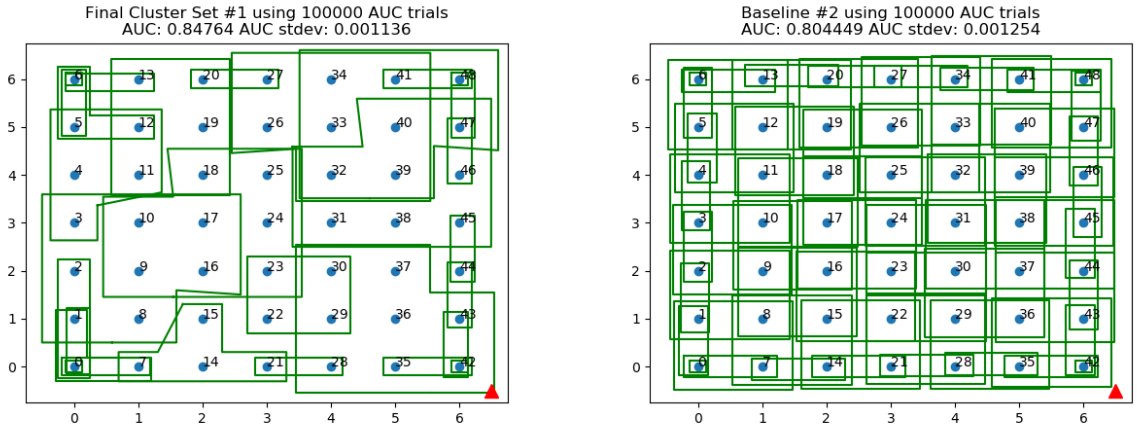

Figure 12. Local search cluster set (left) achieves a higher detection performance over the Baseline #2 cluster set (right) by reducing the total number of clusters which reduces the Multiple Hypothesis Testing Problem. The worst case emitter position is also shown by the red triangle.

Section 6.3 – The Random Network Case

While the grid scenario is easier to analyze, it may be harder to implement in the field. So, the local search algorithm was applied to a WSN with randomly scattered sensors. To evaluate the random case, ten different random sensor networks with 25 randomly located

sensors each were created. For each of the ten networks, the AUC of the four Baselines and the AUC of the local search cluster sets were computed. Once again, the highest AUC cluster set from 12 searches was used as the final local search cluster set. Figure 13 compares the AUCs of the Baselines and local search cluster sets from all ten random sensor networks.

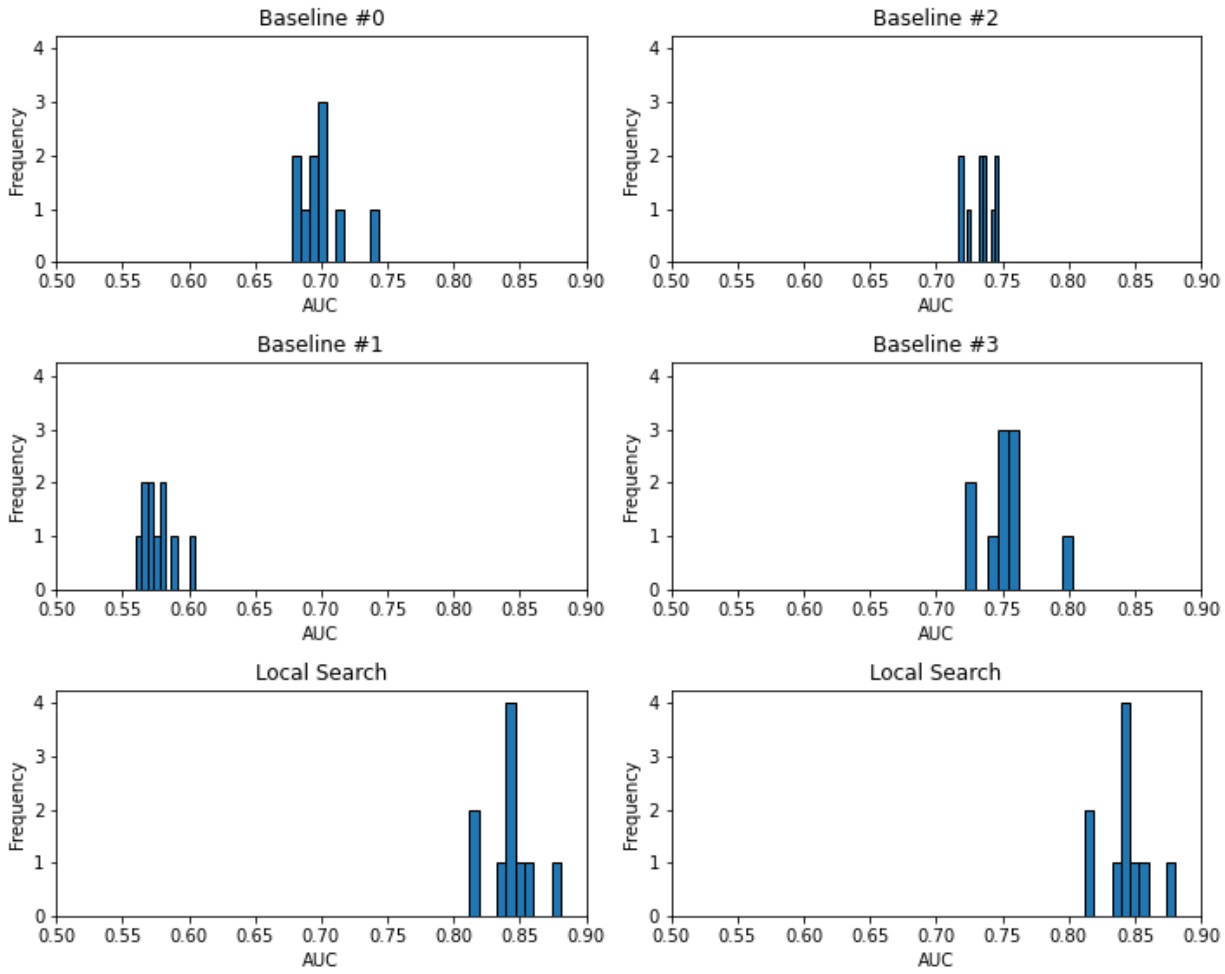

Figure 13. Comparison of local search cluster set against Baseline cluster sets when using different random sensor networks.

Just as in the grid scenario, the local search cluster set demonstrated definitive improvement over all baseline cluster sets. As shown by the histograms in Figure 13, the local search AUC mean is well above the mean of every baseline AUC. Likewise, the AUC improvement for a single random WSN can be observed by considering the ROC curve (Figure 14) for the local search cluster set and best baseline cluster set (Baseline #3). Just like in the grid scenario, the AUC improvement seen in a single random sensor network is also observed at all relevant emitter amplitudes (see Appendix C).

Due to the random placement of sensors, a WSN with 25 randomly placed sensors will have more than a WSN composed of a grid of 25 sensors (see Figure 15). Moreover, the in a random WSN also have a more complex spatial distribution. The added and unpredictable complexity of the sensor and locations makes it much harder to create reasonable cluster sets. Nevertheless, the focus of the local search to improve the minimum detection performance takes this complexity into account and drives its solution cluster set above the baselines.

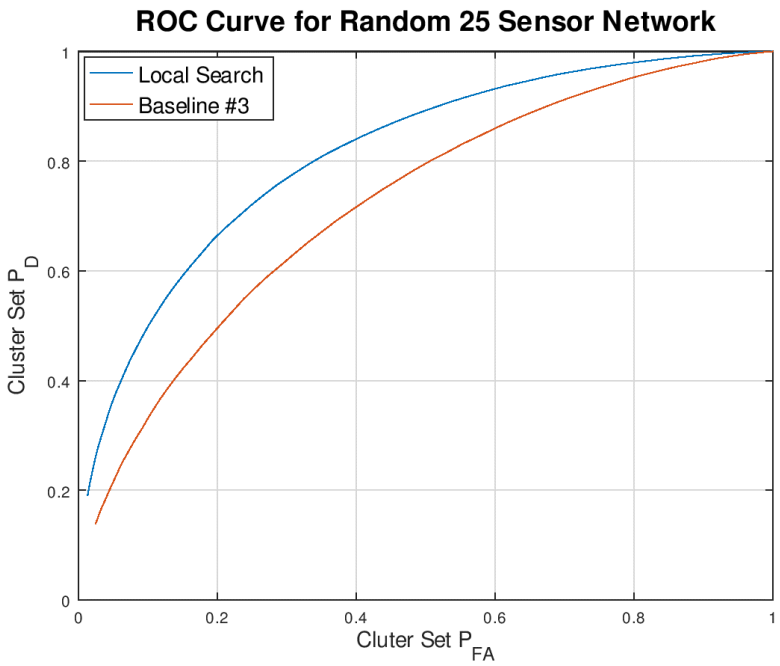

Figure 14. Comparison of ROC curves for a random 25 sensor network. The local search cluster set demonstrates once again its improvement over the best baseline cluster set.

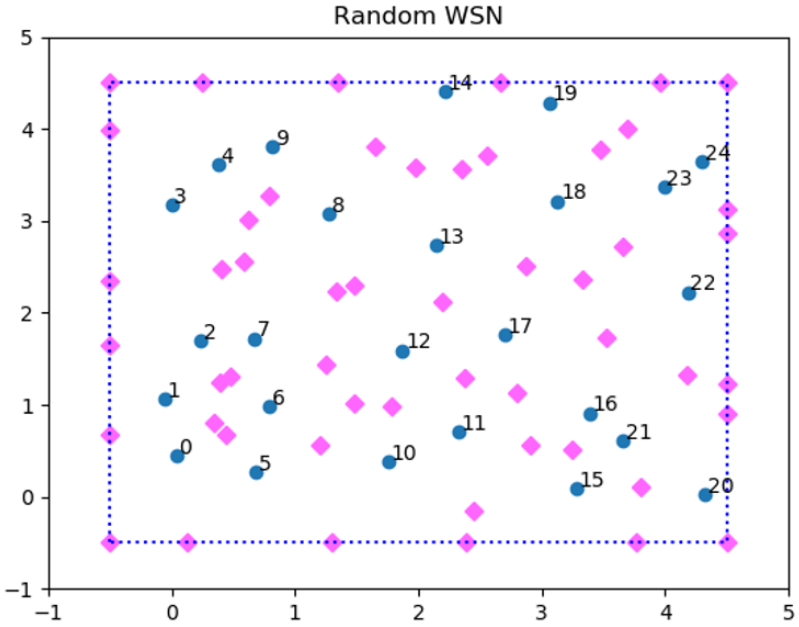

Figure 15. Random sensor (blue dot) placement leads to extra POI (pink diamonds) that are also positioned in random locations. The irregularity makes it harder for a human to create a reasonable cluster set by observation but is addressed by the local search algorithm.

CHAPTER 7

CONCLUSION

Section 7.0 – Final Remarks

In summary, while the detection performance of a wireless sensor network was known to be improved by using the scan statistic, there has been limited development to determine which clusters should be used by the scan statistic. We have presented a local search method that successfully optimizes the cluster set for the scan statistic in both grid and random sensor networks and thereby increases its worst-case detection performance. By sequentially adding clusters to the worst possible emitter positions as well as removing the least valuable clusters, our modified Gradient Ascent Search generated a list of optimized cluster sets that improved the scan statistic's worst-case detection performance over baseline cluster sets.

Section 7.1 – Future Work

Like any other local search, our local search algorithm naturally requires extensive computation time. Further research is needed to explore additional ways to improve the overall search time and performance. Possible areas for future work include:

- Implementing a swap improvement approach in order to build new neighbor cluster sets by removing and adding clusters simultaneously. This is another approach to try in

addition to the only adding clusters and only removing clusters approaches when building neighbor cluster sets.

- Exploring possible characteristics of unfavorable search paths to reveal early stop indicators to help the search change improvement directions sooner. For example, when removing clusters, there comes a point when the loss of N or more clusters will always make a worse cluster set.
- Improving the overall scalability of the algorithm. A local search can take a significant amount of time to find a local optimum, so it is crucial to ensure that efficient computation techniques are used. Two key areas worth exploring further are how to find the worst poi_v and how to compute the detection performance of a cluster set.
- Searching the first neighborhood completely before starting the first improvement approach. At the cost of extra initial computation time, the local search could be guaranteed to start its search in the direction of most improvement.

Finally, it should be especially noted that, while we used the cluster set's worst-case AUC estimate as our improvement metric, any cluster set detection metric could be applied to our local search. If a faster and more discriminating metric is found, it could further reduce the search time and lead to better optimized cluster sets. Nevertheless, as demonstrated above, our proposed local search has addressed the optimization of cluster sets and proved its ability to increase the sensor network's detection performance.

REFERENCES

[1] G. J. Pottie and W. J. Kaiser, ‘‘Wireless integrated network sensors,’’ Commun. ACM, vol. 43, no. 5, pp. 51–58, May 2000.

[2] P. K. Varshney, Distributed Detection and Data Fusion. New York, NY, USA: Springer-Verlag, 1996.

[3] R. Viswanathan and P. Varshney, ‘‘Distributed detection with multiple sensors I. Fundamentals,’’ Proc. IEEE, vol. 85, no. 1, pp. 54–63, Jan. 1997.

[4] R Niu, PK Varshney, Distributed detection and fusion in a large wireless sensor network of random size. EURASIP J. Wireless Commun. Netw. 2005(4), 462–472 (2005)

[5] R. Niu, P. K. Varshney, M. H. Moore, and D. Klamer, “Decision fusion in a wireless sensor network with a large number of sensors,” in *Proc. 7th IEEE International Conference on Information Fusion (ICIF ’04)*, Stockholm, Sweden, June–July 2004.

[6] A. H. Liu, J. J. Bunn, and K. M. Chandy, “Sensor networks for the detection and tracking of radiation and other threats in cities,” in Proc. IEEE 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw., 2011, pp. 1–12.

[7] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," in *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366-379, Oct.-Dec. 2004, doi: 10.1109/TMC.2004.41.

[8] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Maui, HI, USA, 2000, pp. 10 pp. vol.2-, doi: 10.1109/HICSS.2000.926982.

[9] A. Abbasi, M. Younis, “A survey on clustering algorithms for wireless sensor networks, Computer Communications,” in Computer Communications, vol. 30, Issues 14–15, pp 2826-2841, Oct. 2007, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2007.05.024. (http://www.sciencedirect.com/science/article/pii/S0140366407002162)

[10] N. S. V. Rao, M. Shankar, J.-C. Chin, D. K. Y. Yau, S. Srivathsan, S. S. Iyengar, Y. Yang, and J. C. Hou, ‘‘Identification of low-level point radiation sources using a sensor network,’’ in Proc. Int. Conf. Inf. Process. Sensor Netw. (IPSN), Apr. 2008, pp. 493–504.

[11] Kulldorff, Martin. (1997). A Spatial Scan Statistic. Communications in Statistics - Theory and Methods. 26. 1481-1496. 10.1080/03610929708831995.

[12] J. Glaz and M. V. Koutras, Handbook of Scan Statistics. New York, NY, USA: Springer-Verlag, 2017.

[13] B. Fonseca, "On the Design of Cluster Sets for the Scan Statistic in Sensor Detection Systems with Emitter Location Uncertainty," submitted to IEEE Access Journal (under review).

[14] M. Guerriero, P. Willett and J. Glaz, "Distributed Target Detection in Sensor Networks Using Scan Statistics," in *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2629-2639, July 2009, doi: 10.1109/TSP.2009.2017567.

[15] Aldalahmeh, S.A., Ghogho, M., McLernon, D. *et al.* Optimal fusion rule for distributed detection in clustered wireless sensor networks. *EURASIP J. Adv. Signal Process*. 2016, 5 (2016). https://doi.org/10.1186/s13634-016-0303-9

[16] B. J. B. Fonseca, "Designing Conservative Sensor Detection Systems With the Scan Statistic Under Emitter Location Uncertainty," in *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 711-722, Dec. 2019, doi: 10.1109/TSIPN.2019.2942130.

[17] E. Arias-Castro, E. J. CandŁs, and A. Durand, ‘‘Detection of an anomalous cluster in a network,’’ Ann. Statist., vol. 39, no. 1, pp. 278–304, Feb. 2011.

[18] J. Luo and Q. Wu, ‘‘Scan statistics with local vote for target detection in distributed system,’’ EURASIP J. Adv. Signal Process., vol. 2017, no. 1, p. 32, 2017.

[19] J. Luo and R. Zou, ‘‘Distributed decision fusion over nonideal channels using scan statistics,’’ IEICE Trans. Fundamentals, vol. E99.A, no. 11, pp. 2019–2026, 2016

[20] X. Song, P. Willett, J. Glaz, and S. Zhou, ‘‘Active detection with a barrier sensor network using a scan statistic,’’ IEEE J. Ocean. Eng., vol. 37, no. 1, pp. 66–74, Jan. 2012.

[21] S. Althunibat, A. Khalifeh, and R. Mesleh, ‘‘A low-interference decision gathering scheme for critical event detection in clustered wireless sensor network,’’ Phys. Commun., vol. 26, pp. 149–155, Feb. 2018.

[22] A. Peiravi, H. R. Mashhadi, and S. Hamed Javadi, ‘‘An optimal energy efficient clustering method in wireless sensor networks using multiobjective genetic algorithm,’’ Int. J. Commun. Syst., vol. 26, no. 1, pp. 114–126, Jan. 2013.

[23] Q. Tian and E. J. Coyle, ‘‘Optimal distributed detection in clustered wireless sensor networks,’’ IEEE Trans. Signal Process., vol. 55, no. 7, pp. 3892–3904, Jul. 2007.

[24] S. Bandyopadhyay and E. Coyle, ‘‘An energy efficient hierarchical clustering algorithm for wireless sensor networks,’’ in Proc. IEEE INFOCOM 22nd Annu. Joint Conf. IEEE Comput. Commun. Soc., vol. 3, Mar. 2004, pp. 1713–1723.

[25] G. Ferrari, M. Martalo, and R. Pagliari, ‘‘Decentralized detection in clustered sensor networks,’’ IEEE Trans. Aerosp. Electron. Syst., vol. 47, no. 2, pp. 959–973, Apr. 2011.

[26] C. Gherbi, Z. Aliouat, and M. Benmohammed, ‘‘A survey on clustering routing protocols in wireless sensor networks,’’ Sensor Rev., vol. 37, no. 1, pp. 12–25, Jan. 2017.

[27] L. Xu, R. Collier, and G. M. P. O’hare, ‘‘A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios,’’ IEEE Internet Things J., vol. 4, no. 5, pp. 1229–1249, Oct. 2017.

[28] D. N. Sandeep and V. Kumar, ‘‘Review on clustering, coverage and connectivity in underwater wireless sensor networks: A communication techniques perspective,’’ IEEE Access, vol. 5, pp. 11176–11199, 2017.

[29] S. Zafar, A. Bashir, and S. A. Chaudhry, ‘‘Mobility-aware hierarchical clustering in mobile wireless sensor networks,’’ IEEE Access, vol. 7, pp. 20394–20403, 2019.

[30] P. Ding, J. Holliday, and A. Celik, ‘‘Distributed energy-efficient hierarchical clustering for wireless sensor networks,’’ in Proc. Int. Conf. Distrib. Comput. Sensor Syst. Berlin, Germany: Springer, 2005, pp. 322–339.

[31] D. Kumar, T. C. Aseri, and R. Patel, ‘‘EECDA: Energy efficient clustering and data aggregation protocol for heterogeneous wireless sensor networks,’’ Int. J. Comput. Commun., vol. 6, no. 1, p. 113, Dec. 2015.

[32] Z. Chair and P. K. Varshney, "*Optimal Data Fusion in Multiple Sensor Detection Systems*," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-22, no. 1, pp. 98-101, Jan. 1986, doi: 10.1109/TAES.1986.310699.

[33] D. B. Neill, ‘‘Detection of spatial and spatio-temporal clusters,’’ Ph.D. dissertation, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2006.

[34] S. M. Kay, Fundamentals of Statistical Signal Processing: Detection Theory. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[35] C. H. Papadimitriou and K. Steiglitz, “Local Search,” in *Combinatorial Optimization Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982, ch. 19, sec. 19.6, pp. 470.

[36] C. Cortes and M. Mohri, “Confidence intervals for the area under the roc curve,” in Advances in neural information processing systems, 2005, pp. 305–312.

[37] Hanley JA, McNeil BJ. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*. 1983;148(3):839-843. doi:10.1148/radiology.148.3.6878708

APPENDIX A

ANOTHER CLUSTER SET REPRESENTATION

Representing Clusters as a Binary Strings

The way the cluster sets are represented is a key factor to determining what the neighbor clusters sets $\boldsymbol{C}_n$ of the current cluster set $\boldsymbol{C}_c$ will be. As an initial approach, we represented all cluster sets $\boldsymbol{C}_i$ with a binary string, as suggested in [13]. Assume a binary string with $|\boldsymbol{C}_{all}|$ digits so that each cluster in $\boldsymbol{C}_{all}$ corresponds to a digit in the binary string. For example, if $\boldsymbol{C}_{all} = \{C_1, C_2, C_3, C_4, C_5\}$ then a cluster set can be represented as a 5 digit binary number. Each binary digit would be 1 if the cluster C_x is present in the cluster set, or 0 if the cluster is not in the cluster set. So, a cluster set with clusters C_1, C_3, and C_4 would be written as: $\boldsymbol{C}_{134} = 10110$. Using this notation, neighbor cluster sets $\boldsymbol{C}_n$ could be created by taking all cluster sets that differ from $\boldsymbol{C}_c$ by d digits, i.e. have a Hamming distance of d from $\boldsymbol{C}_c$. If $d = 1$, the neighbors of $\boldsymbol{C}_{134}$ would be:

$$\boldsymbol{C}_{34} = 00110$$
$$\boldsymbol{C}_{1234} = 11110$$
$$\boldsymbol{C}_{14} = 10010$$
$$\boldsymbol{C}_{13} = 10100$$
$$\boldsymbol{C}_{1345} = 10111$$

Additional Hamming distance neighborhoods could also be created by varying d.

This cluster set and neighborhood representation proved to be impractical. By using a Hamming distance neighborhood with $d = 1$, each cluster set $\boldsymbol{C}_c$ would have $|C_{all}|$ neighbors. In our implementation, a 3x3 grid of sensors would have $|C_{all}| = 41$ neighbors while a 10x10 grid of sensors would have $|C_{all}| = 2{,}166$ neighbor cluster sets. These large neighborhoods did not restrict the search space enough and thus gave the current cluster set too many possible improvement directions. In chapter 5, we represent a cluster set simply as a collection of clusters and build neighbors by adding/removing specific clusters

APPENDIX B

CLUSTER SET TABLES

Table 1. Ranking clusters by their P_D at POI

	#0 Best ← Cluster Rank → #8 Worst								
POI Position	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
(-0.5, -0.5)	{0}	{0, 1, 3, 4}	{3, 4, 6, 7}	{1, 2, 4, 5}	{8, 4, 5, 7}	{4}	{6}	{2}	{8}
(-0.5, 0.5)	{0, 1, 3, 4}	{0}	{1, 2, 4, 5}	{3, 4, 6, 7}	{8, 4, 5, 7}	{4}	{2}	{6}	{8}
(-0.5, 1.5)	{1, 2, 4, 5}	{2}	{0, 1, 3, 4}	{8, 4, 5, 7}	{3, 4, 6, 7}	{4}	{0}	{8}	{6}
(-0.5, 2.5)	{2}	{1, 2, 4, 5}	{8, 4, 5, 7}	{0, 1, 3, 4}	{3, 4, 6, 7}	{4}	{8}	{0}	{6}
(0.5, -0.5)	{0, 1, 3, 4}	{0}	{3, 4, 6, 7}	{1, 2, 4, 5}	{8, 4, 5, 7}	{4}	{6}	{2}	{8}
(0.5, 0.5)	{0, 1, 3, 4}	{3, 4, 6, 7}	{1, 2, 4, 5}	{4}	{0}	{8, 4, 5, 7}	{6}	{2}	{8}
(0.5, 1.5)	{1, 2, 4, 5}	{8, 4, 5, 7}	{0, 1, 3, 4}	{4}	{2}	{3, 4, 6, 7}	{8}	{0}	{6}
(0.5, 2.5)	{1, 2, 4, 5}	{2}	{8, 4, 5, 7}	{0, 1, 3, 4}	{3, 4, 6, 7}	{4}	{8}	{0}	{6}
(1.5, -0.5)	{3, 4, 6, 7}	{6}	{0, 1, 3, 4}	{8, 4, 5, 7}	{1, 2, 4, 5}	{4}	{0}	{8}	{2}
(1.5, 0.5)	{3, 4, 6, 7}	{8, 4, 5, 7}	{0, 1, 3, 4}	{4}	{6}	{1, 2, 4, 5}	{8}	{0}	{2}
(1.5, 1.5)	{8, 4, 5, 7}	{3, 4, 6, 7}	{1, 2, 4, 5}	{4}	{8}	{0, 1, 3, 4}	{6}	{2}	{0}
(1.5, 2.5)	{8, 4, 5, 7}	{8}	{1, 2, 4, 5}	{3, 4, 6, 7}	{0, 1, 3, 4}	{4}	{2}	{6}	{0}
(2.5, -0.5)	{6}	{3, 4, 6, 7}	{8, 4, 5, 7}	{0, 1, 3, 4}	{1, 2, 4, 5}	{4}	{8}	{0}	{2}
(2.5, 0.5)	{3, 4, 6, 7}	{6}	{8, 4, 5, 7}	{0, 1, 3, 4}	{1, 2, 4, 5}	{4}	{8}	{0}	{2}
(2.5, 1.5)	{8, 4, 5, 7}	{8}	{3, 4, 6, 7}	{1, 2, 4, 5}	{0, 1, 3, 4}	{4}	{6}	{2}	{0}
(2.5, 2.5)	{8}	{8, 4, 5, 7}	{3, 4, 6, 7}	{1, 2, 4, 5}	{0, 1, 3, 4}	{4}	{6}	{2}	{0}

Table 2. Cluster ranks at all POI

Cluster	**Cluster's Rank at each POI**
{0}	0, 1, 5, 5, 1, 2, 4, 6, 5, 4, 5, 7, 5, 6, 7, 6
{2}	5, 5, 1, 0, 6, 4, 2, 1, 7, 5, 4, 5, 6, 7, 6, 5
{6}	5, 6, 7, 6, 5, 4, 5, 7, 1, 2, 4, 6, 0, 1, 5, 5
{8}	6, 7, 6, 5, 7, 5, 4, 5, 6, 4, 2, 1, 5, 5, 1, 0
{4}	4, 5, 5, 4, 5, 2, 2, 5, 5, 2, 2, 5, 4, 5, 5, 4
{0,1,3,4}	1, 0, 2, 2, 0, 0, 1, 3, 2, 1, 3, 4, 2, 3, 4, 3
{1,2,4,5}	2, 2, 0, 1, 3, 1, 0, 0, 4, 3, 1, 2, 3, 4, 3, 2
{3,4,6,7}	2, 3, 4, 3, 2, 1, 3, 4, 0, 0, 1, 3, 1, 0, 2, 2
{4,5,7,8}	3, 4, 3, 2, 4, 3, 1, 2, 3, 1, 0, 0, 2, 2, 0, 1

APPENDIX C

VARYING P_{FA} AND EMITTER AMPLITUDE

Improved Detection Performance for Varying Emitter Amplitudes and Network P_{FA}

The results shown in Chapter 6 demonstrated the improved detection performance of the local search cluster set over the best baseline cluster set. However, because the final probability of detection of the cluster set will also depend on the emitter signal amplitude, the network designer must analyze the final selected cluster set at multiple emitter amplitudes to establish a minimum amplitude for acceptable detection performance. The final P_D of the cluster set will also depend on the desired maximum scan statistic P_{FA}. Nevertheless, the local search algorithm presented is robust regarding the emitter amplitude and desired scan statistic P_{FA}. As shown in Figure 16, the highest AUC cluster set found using the local search approach will outperform the best baseline cluster set AUC.

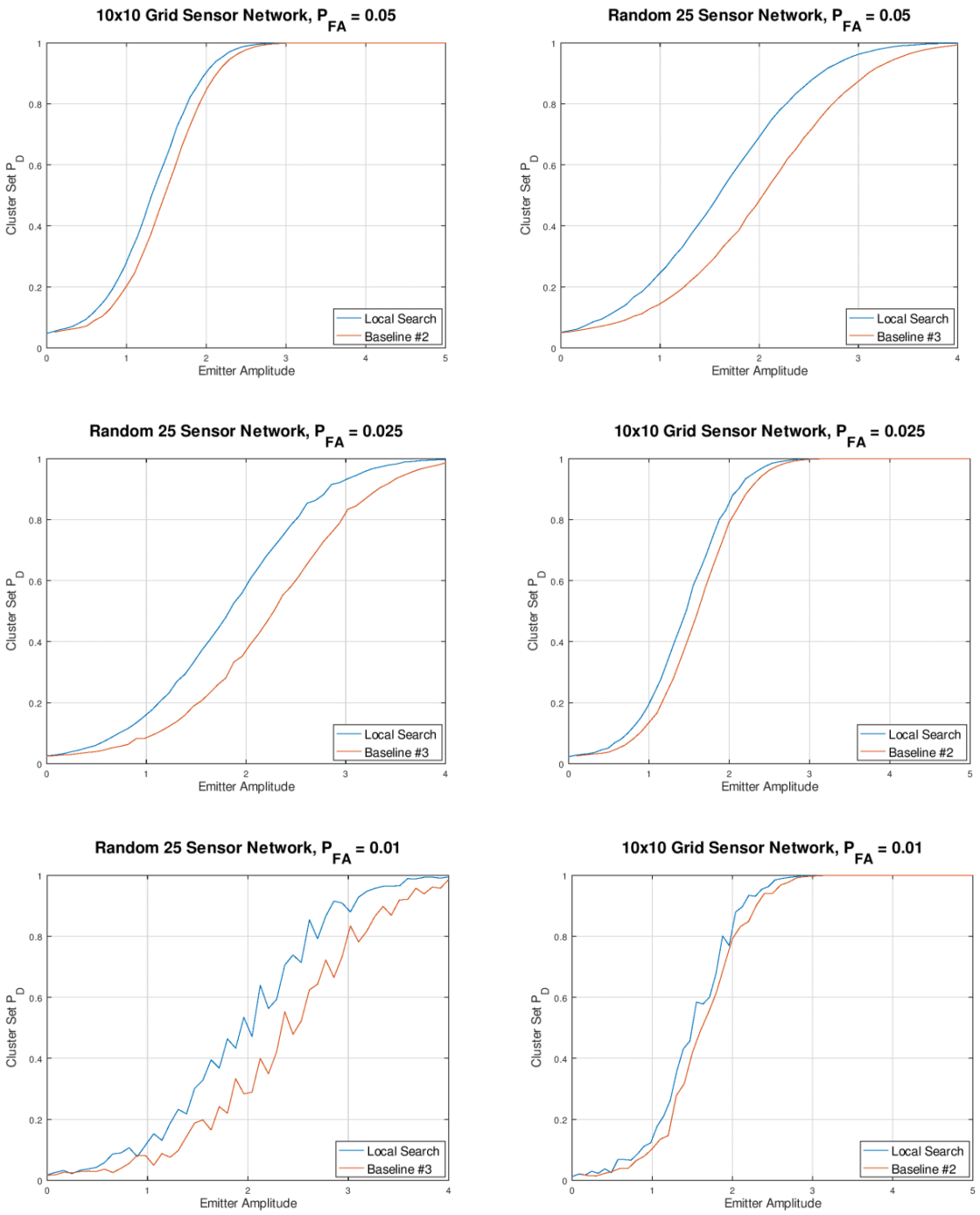

Figure 16. Comparison of the local search cluster set and the corresponding best baseline cluster set in a 10x10 grid and 25 sensor WSN. For various emitter amplitudes and desired maximum network P_{FA}, the local search P_D still shows improvement over the best baseline P_D.