On Accurate and Efficient Statistical Counting in Sensor-Based Surveillance Systems

Shuo Guo[§], Tian He[§], Mohamed F. Mokbel[§], John A. Stankovic[†] and Tarek F. Abdelzaher[‡]

[§]Department of Computer Science and Engineering, University of Minnesota
[†]Department of Computer Science, University of Virginia
[‡]Department of Computer Science,University of Illinois, Urbana-Champaign

Abstract

Sensor networks have been used in many surveillance systems, providing statistical information about monitored areas. Accurate counting information (e.g., the distribution of the total number of targets) is often important for decision making. As a complementary solution to double-counting in communication, this paper presents the first work that deals with double-counting in sensing for wireless sensor networks. The probability mass function (pmf) of target counts is derived first. This, however, is shown to be computationally prohibitive when a network becomes large. A partitioning algorithm is then designed to significantly reduce computation complexity with a certain loss in counting accuracy. Finally, two methods are proposed to compensate for the loss. To evaluate the design, we compare the derived probability mass function with ground truth obtained through exhaustive enumeration in small-scale networks. In large-scale networks, where pmf ground truth is not available, we compare the expected count with true target counts. We demonstrate that accurate counting within $1 \sim 3\%$ *relative error can be achieved with orders of magnitude reduction in computation, compared with an exhaustive enumeration-based approach.*

1 Introduction

Wireless sensor networks have been widely used to monitor many types of environments such as battlefields [1], buildings [2] and habitats [3, 4]. One of the key design objectives of these monitoring systems is to acquire and verify information about the number of targets/events within the system at any given point of time. For example, (i) in a battlefield, a commander needs to estimate enemy capability by counting different types of targets in an area to issue a counter-force attack strategically; (ii) in a building, a manager might want to turn off some facilities if the number of people in a certain area is less than a certain threshold; (iii) in geysers fields monitoring, the number of eruptions indicates the activity pattern underneath. In all these cases, although it is not necessary to have precise counting information, it is important to obtain reasonable total count values through a sensor network.

In general, there are two types of errors that would lead to inaccurate counts: miss-detection and double-counting. Miss-detection is normally addressed by using reliable sensing hardware [5] and/or robust detection algorithms [6, 7, 8], while double-counting is a more challenging problem, because it involves duplicates in both communication and sensing. Several excellent projects have investigated how to avoid the double-counting problem in communication. For example, synopsis diffusion [9] uses energy-efficient multi-path routing schemes to transmit order-and duplicate-insensitive (ODI) data aggregates. Recently, CountTorrent [10], uses Abstract Prefix Tree (APT) to ensure all values are counted once through distributive queries. We observe that these solutions work well by assuming original count values from each sensor is not duplicated. However, sensor nodes are normally densely deployed with a high-degree of redundancy (overlapping), therefore double-counting by adjacent sensors could be significant and should not be ignored. Although many researchers have studied the double-counting problem in communication, to our knowledge, this paper presents the first attempt to address the double-counting problem in the context of sensing in sensor networks. By avoiding double-counting in both communication and sensing, accurate statistic counting can be achieved.

To address double-counting in sensing, one straightforward solution is to use sophisticated identification sensors to differentiate targets by analyzing their signatures such as acoustic emission or thermal radiation. This approach requires high-cost sensor nodes and possibly introduces excessive energy consumption. Naturally, we raise the following question: *how to avoid double-counting statistically, using low-cost sensors without identification capacity*?

The main idea of our solution is to derive a probability mass function (*pmf*) of total target counts, using partition and compensation methods. With the *pmf* available, one can obtain the expected total count that approaches ground truth, i.e., the actual number of targets in the system. Specif-

978-1-4244-2575-4/08/$20.00

ically, the main contribution of this work lies in following aspects:

- Given separate counts from overlapping sensor nodes, we derive a probability mass function (*pmf*) of total target counts, from which various types of statistical information (e.g., expected value, variance, range, min and max) can be inferred accurately.
- We propose an accuracy-aware partitioning algorithm to reduce the computational complexity of calculating a system-wide probability mass function.
- Two algorithms are proposed to compensate for the inaccuracy introduced by the partitioning process. The first algorithm sacrifices certain accuracy in exchange of very fast computation, while the second algorithm achieves high accuracy with adjustable computation overhead.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the derivation of probability mass function, followed by the complexity analysis in Section 4. Section 5 describes how computation complexity can be significantly reduced by partitioning. Section 6 introduces two compensation algorithms for better accuracy. Simulation results are presented in Section 7. We conclude this paper in Section 8 with our summary and directions for future work.

2 Related Work

To obtain accurate target counts, a monitoring sensor system shall prevent miss-detection as well as double-counting. Miss-detection can be reduced by introducing reliable hardware design. For example, XSM motes [5] incorporate a band-pass filter to enhance the detection of acoustic emission, a digital potentiometer to detect a wide range of signals, and a polyethylene film to reduce the effect of sunlight. Besides hardware enhancements, advanced detection algorithms [6, 7, 8] have also been proposed to avoid misdetection with minimal energy consumption. VigilNet [7] utilizes a multi-level detection algorithm with in-situ adaptive thresholds to avoid both false positive and false negative detections in changing weather conditions. Feng et al. [8] propose a collaborative tracking algorithm with distributed Bayesian estimation to improve reliability based on current and previous estimation (beliefs) from sets of sensors.

Even with reliable detection at individual nodes, accurate total counts would not be obtained if a target is counted multiple times (double-counting). Double-counting problem has been investigated in the context of communication. The summarization of total counts without duplicates could be achieved by building a spanning tree rooted at the base. Individual counts are aggregated along a tree from leaves to the root as suggested in TAG [11]. However, this spanning-tree-based approach could suffer loss of counts severely, due to node or communication failures. For example, a single node failure could lose the count of a whole subtree beneath it. To address this limitation, synopsis diffusion [9] utilizes multi-path routing to deliver count information. The authors prove that duplicate-insensitive (ODI) count aggregation can be achieved by using Flajolet and Martin's algorithm (FM) [12], which counts distinct elements in a multi-set. Recently, CountTorrent [10] allocates binary labels to individual nodes using an Abstract Prefix Tree and disseminates the $(label, count)$ pairs through multi-path routing. Count values are aggregated only when two binary labels differ only in their last bit. Labeled aggregates ensure all values are counted only once during communication.

Although double-counting can be eliminated in communication, the final aggregated count could still be incorrect, if the targets within overlapping regions are counted more than once. According to [13], the percentage of overlapping region in sensor networks under random deployment is indeed significant. For example, with an average node density of 5, the overlapping percentage is 86% and with an average node density of 14, the overlapping percentage would be as high as 99.9%! Therefore, we argue double-counting is common in sensing and hence needs to be addressed accordingly.

3 Problem Definition and Assumptions

We consider a network model where counting sensor nodes are randomly deployed in a region (e.g., an open area or a room in a building) with known locations [14, 15]. They are used to monitor different types of targets, such as vehicles on the road, people in the room, or any other objects of interests. Counting capability is supported, using photoelectric-based sensors such as the one in [16]. The count values at individual sensors are reported to a base node, where the probability mass function (*pmf*) of the total number of distinct targets is calculated. Since the system-wide total count is the objective, a centralized solution at the base is a natural approach for sensor networks, which is also compatible with counting communication methods in TAG [11], Synopsis Diffusion [9], and CountTorrent [10].

To simplify the description, the sensing range of these nodes are treated as circles. It should be noted that the accuracy of our method only depends on the size of the area, not the shape of the area. In case of irregular sensing areas, methods proposed in [17] shall be used to obtain the size of sensing areas.

This work assumes spatial distribution of targets within the area is known (e.g., complete spatial randomness, spatial aggregation or spatial inhibition). Without loss of generality, we use Poisson distribution [18, 19, 20] as a concrete exemplary distribution to present our methodology through the paper. We expect our high-level idea can be applied to non-Poisson distributions, although mathematical derivation would be quite different.

Under the Poisson distribution, targets are uniformly distributed in the area of interest with intensity of λ. The λ value can be either known a prior or estimated online (as we explain later). The probability that there are k targets in

the region s of size S can be computed as follows:

$$P(\mathcal{N}(s)=k)=\frac{e^{-\lambda S}(\lambda S)^k}{k!}. \quad (1)$$

Suppose there are in total N sensor nodes. The i_{th} sensor node v_i whose sensing area is circle C_i has detected n_i targets in its sensing range, where $1 \le i \le N$. Suppose the N sensing circles of these nodes divide the whole area into M non-overlapping subareas. Each subarea, M_k, where $1 \le k \le M$, may belong to one or more circles. As a result, each circle is the union of a subset of all these subareas. We say $M_k \subset C_i$ if M_k is within the subset of the i_{th} circle C_i. If we further use $\mathcal{N}(M_k)$ to denote the number of distinct targets in subarea M_k, we will have the following equation:

$$n_i=\mathcal{N}(C_i)=\sum_{M_k \subset C_i}\mathcal{N}(M_k). \quad (2)$$

Since the subareas are non-overlapping, the total number of distinct targets detected by the N sensor nodes (denoted as T) will be equal to the sum of the number of targets in each subarea, which can be computed by the following equation:

$$T=\mathcal{N}(\bigcup_{i=1}^{N}C_i)=\sum_{k=1}^{M}\mathcal{N}(M_k). \quad (3)$$

The objective of this work is to find the probability mass function (*pmf*) of the total target count T in Equation (3), given the individual counts n_i from all nodes, with low computation complexity. Particularly, the probability that the total number of distinct targets equals to t given the count information can be expressed as

$$P(T=t|\mathcal{N}(C_1)=n_1,\mathcal{N}(C_2)=n_2,\cdots,\mathcal{N}(C_N)=n_N). \quad (4)$$

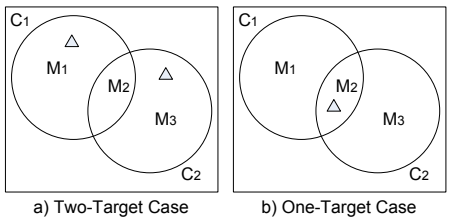

Figure 1. A simple example:Two-Circle Case

We start with a simple example as shown in Figure (1). Two sensor nodes v_1 and v_2 whose sensing circles are C_1 and C_2 divide the whole region into three subareas: M_1, M_2 and M_3. Suppose both of sensor nodes detect one target, there are two possible scenarios as shown in Figure(1a) and Figure(1b). The objective is to calculate the probability that there are in total two (or one) distinct targets in this area, respectively. For simplicity, we define notation $< m_1,m_2,m_3 >$ as the joint probability that M_1 has m_1 targets, M_2 has m_2 targets and M_3 has m_3 targets. An example is shown as follows:

$$<1,0,1>=P(\mathcal{N}(M_1)=1)P(\mathcal{N}(M_2)=0)P(\mathcal{N}(M_3)=1). \quad (5)$$

Using the definition of conditional probability, from (2), (3) and (4), we get,

$$\begin{aligned}
& P(T=2|\mathcal{N}(C_1)=1,\mathcal{N}(C_2)=1)\\
=\ & \frac{P(T=2,\mathcal{N}(C_1)=1,\mathcal{N}(C_2)=1)}{P(\mathcal{N}(C_1=1,\mathcal{N}(C_2)=1)}\\
=\ & \frac{P(\mathcal{N}(M_1)=1,\mathcal{N}(M_2)=0,\mathcal{N}(M_3)=1)}{\sum_{k=0}^{1}P(\mathcal{N}(M_1)=1-k,\mathcal{N}(M_2)=k,\mathcal{N}(M_3)=1-k)}\\
=\ & \frac{<1,0,1>}{\sum_{k=0}^{1}<1-k,k,1-k>}\\
=\ & \frac{<1,0,1>}{<1,0,1>+<0,1,0>}. \quad (6)
\end{aligned}$$

The penultimate equality holds because M_1, M_2 and M_3 are non-overlapping subareas and the numbers of targets in these subareas are independent random variables. Since each term in Equation (6) can be computed using Equation (1), we can finally compute the probability for any given t and thus compute the conditional *pmf* of the total number of distinct targets.

Noting that all the terms in the denominator of Equation (6) ($<1,0,1>$ *and* $<0,1,0>$) are the probability of possible solutions that satisfy the count condition of all circles ($\mathcal{N}(C_1)=1,\mathcal{N}(C_2)=1$) while the numerator ($<1,0,1>$) is the probability of the only solution that satisfies both the count condition and the total number of distinct targets condition ($\mathcal{N}(C_1)=1,\mathcal{N}(C_2)=1,T=2$). Similarly, we can derive the equation for a more general case, i.e., for a region that has been divided into M subareas by N sensor nodes, Equation (4) can be computed as

$$\begin{aligned}
& P(T=t|\mathcal{N}(C_1)=n_1,\mathcal{N}(C_2)=n_2,\cdots,\mathcal{N}(C_N)=n_N)\\
=\ & \frac{P(T=t,\mathcal{N}(C_1)=n_1,\mathcal{N}(C_2)=n_2,\cdots,\mathcal{N}(C_N)=n_N)}{P(\mathcal{N}(C_1)=n_1,\mathcal{N}(C_2)=n_2,\cdots,\mathcal{N}(C_N)=n_N)}\\
=\ & \frac{\sum_{\{m'_k\}\in(A\bigcap B)}<m'_1,m'_2,\cdots,m'_M>}{\sum_{\{m_k\}\in A}<m_1,m_2,\cdots,m_M>} \quad (7)
\end{aligned}$$

where $<m_1,m_2,\cdots,m_M>$ is defined similarly as before, but extended to a more general case, $\{m_k\}$ denotes the set of $\{m_1,m_2,\cdots,m_M\}$, $\{m_k\}\in A$ means for each term in the denominator, the corresponding $\{m_k\}$ satisfies the count condition of the N circles so that it is a solution to a set of equations A defined as follows:

$$A:\begin{cases}
\sum_{M_k\subset C_1}m_k=n_1\\
\sum_{M_k\subset C_2}m_k=n_2\\
\quad\vdots\\
\sum_{M_k\subset C_N}m_k=n_N\\
\mathcal{N}(M_k)=m_k\ge 0,\forall 1\le k\le M
\end{cases}. \quad (8)$$

Also, for each term in the numerator, the corresponding $\{m'_k\}$ satisfies both the count condition and the total number condition. As a result, each $\{m'_k\}$ is a solution to both A and B where A is defined in Equation (8) and B is defined as follows:

$$B: m_1+m_2+\cdots+m_M=t \quad (9)$$

Algorithm 1 Enumeration Algorithm

Input: G and C_i
Output: pmf of total target counts T

for k = 1 to M **do**
  $UB(m_k) = max\{n_i\}$ where $M_k \subset C_i$
end for
$D \leftarrow 0, N(t) \leftarrow 0, \forall$ possible t
for every possible $m_1, m_2, \cdots, m_M$ **do**
  if $m_1, m_2, \cdots, m_M$ is a valid solution to A **then**
    $d \leftarrow \prod_{k=1}^{M} P(\mathcal{N}(M_k) = m_k)$
    $D \leftarrow D + d$, $t_{sum} = \sum_{k=1}^{M} m_k$
    $N(t_{sum}) = N(t_{sum}) + d$
  end if
end for
$P(T = t | C_i = n_i) = \frac{N(t)}{D}$

4 Complexity Analysis

In order to determine the conditional *pmf* of the total number of distinct targets T, we need to compute the probability in Equation (7) for every possible value of t. Several interesting observations can be captured from Equation (7). First, the numerator is actually a subset of the denominator for a particular value of t. All these subsets are disjoint and sum to the denominator, which is consistent with the fact that all the values of the probability mass function sum up to 1. Second, in order to compute the denominator, we need to solve the equations of A and try to find all the solutions. As a result, the complexity of computing the probability for a single value of t is exactly the same as computing the whole probability mass function since we need to find all the solutions to A anyway. Third, since all the variables in A are non-negative integers, we have to exhaustively list all the solutions of A. Based on these observations, we develop an algorithm using an exhaustive enumeration-based method to compute Equation (7) as shown in Algorithm (1).

The complexity of finding the conditional *pmf* using Algorithm (1) can be computed as

$$f_1 = O(\prod_{k=1}^{M} UB(m_k)) \tag{10}$$

where $UB(m_k)$ is defined as the maximum possible number of targets in subarea M_k and can be computed as

$$UB(m_k) = max\{n_i\} \text{ where } M_k \subset C_i. \tag{11}$$

Obviously, f_1 is an exponential function of M. Remember M is the number of non-overlapping subareas divided by the N circles. As a result, M would be much greater than N. For example, even in a linear network where sensor nodes are uniformly deployed, M is almost twice of N. Exponential complexity makes it prohibitive for Algorithm (1) to obtain an accurate count in large-scale sensor networks, using a reasonable amount of time.

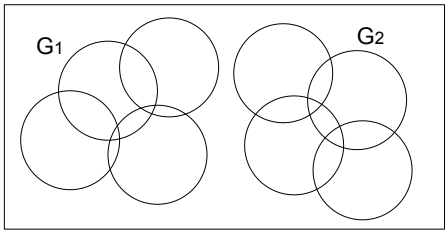

Figure 2. An Example of Natural Partitioning Case

5 Partitioning Design

In the previous section, we have concluded that a large M value makes the computation time intolerable, which also indicates that reducing M can significantly reduce computation complexity. Figure (2) shows that N circles belong to two disjoint groups G_1 and G_2 at initial deployment time. We note this deployment rarely happens in a dense network, however, we use this example to show the power of partitioning in reducing complexity. Suppose the numbers of subareas in G_1 and G_2 are M_{G_1} and M_{G_2}, respectively. Since G_1 and G_2 are disjoint, the total numbers of distinct targets in G_1 and G_2 (denoted by T_1 and T_2) are independent random variables. As a result, we can compute the *pmf* of T_1 and T_2 separately and then combine the two functions to compute the *pmf* of T which is equal to the sum of these two independent random variables: $T = T_1 + T_2$. The method used to combine two independent distributions can be found in textbooks [21] and will not be discussed. Here we are interested in how much complexity can be reduced by partitioning. It's clear that the complexity of the combination process is the product of the sizes of the sample space of G_1 and G_2. This value is negligible compared to the complexity of the enumeration process. Thus, the total complexity of this method can be computed in Equation (12).

$$f_2 = O(\prod_{k=1}^{M_1} (UB(m_k)) + \prod_{k=M_1+1}^{M_1+M_2} (UB(m_k)). \tag{12}$$

If the values of M_1 and M_2 are similar in G_1 and G_2, f_2 is much less than f_1, especially when M is large. Let's compare f_2 with f_1 using the example shown in Figure (2). Suppose all sensors detect n targets for simplicity. If we compute the conditional *pmf* using Equation (7) directly, the cost is $O(n^{22})$. If we compute the *pmf* for G_1 and G_2 separately and then combine them to get the total, the complexity according to Equation (12) is $O(n^{11})$. Generically, if multiple disjoint groups exist in the area and the maximum size of each group G_i is bounded, the computation complexity is:

$$f_3 = O(M \prod_{k=1}^{MAX} UB(m_k)) \tag{13}$$

where MAX is the maximum number of subareas of each

group. It's obvious that the f_3 is a polynomial function of M.

5.1 Deleting Zero Count Circles

In the previous section, we have shown that disjoint groups reduce the complexity significantly. However, given a space covered by sensor nodes, it's not always the case that the circles are disjoint. Therefore, it is necessary to partition the nodes into groups as well as compensate for the loss of accuracy caused by partitioning.

Recall that $< m_1, m_2, m_3 >$ is defined as the probability that M_1 has m_1 targets, M_2 has m_2 targets and M_3 has m_3 targets. Since $\mathcal{N}(M_k)$s are all independent due to the fact that M_ks are non-overlapping, the decomposability of $< m_1, m_2, m_3 >$ can be easily derived from Equation (5) as follows:

$$\begin{aligned} & < m_1, m_2, m_3 > \\ = \; & < m_1, m_2, * >< *, *, m_3 > \\ = \; & < m_1, *, * >< *, m_2, * >< *, *, m_3 > \end{aligned} \quad (14)$$

where the symbol "*" means the number within the corresponding subarea can be any value. Based on this property, the effect of eliminating a zero-count node can be studied.

Suppose node v_1, whose sensing circle is C_1 as shown in Figure (3), detects zero targets, which means $n_1 = 0$. It's possible that there still exist other zero-count circles besides v_1 but they would not affect the discussion here. From Equation (2) we have

$$n_1 = \sum_{M_k \subset C_1} m_k = 0. \quad (15)$$

Suppose there are z subareas in C_1. ($z = 5$ in the example shown in Figure (3)). For simplicity, these subareas are named as $m_1, m_2, \cdots, m_z$. Equation (15) then becomes

$$n_1 = \sum_{k=1}^{z} m_k = 0. \quad (16)$$

Note that Equation(16) is also an equation in A. Since all the m_ks are nonnegative, all the solutions to A satisfy the condition that $m_k = 0, \forall 1 \le k \le z$, which can be easily interpreted as *the number of targets in any subarea within circle C_1 should be zero*. Based on this, Equation (7) can be further rewritten as

$$\begin{aligned} & P(T = t | \mathcal{N}(C_1) = n_1, \cdots, \mathcal{N}(C_N) = n_N) \\ = \; & \frac{\sum_{\{m'_k\} \in (A \cap B)} < m'_1, m'_2, \cdots, m'_M >}{\sum_{\{m_k\} \in A} < m_1, m_2, \cdots, m_M >} \\ = \; & \frac{\sum_{\{m'_k\} \in (A \cap B)} < m'_1, \cdots, m'_z, *, \cdots, * >< *, \cdots, *, m'_{z+1}, \cdots, m'_M >}{\sum_{\{m_k\} \in A} < m_1, \cdots, m_z, *, \cdots, * >< *, \cdots, *, m_{z+1}, \cdots, m_M >} \\ = \; & \frac{< 0, \cdots, 0 > \sum_{\{m'_k\} \in (A \cap B)} < *, \cdots, *, m'_{z+1}, \cdots, m'_M >}{< 0, \cdots, 0 > \sum_{\{m_k\} \in A} < *, \cdots, *, m_{z+1}, \cdots, m_M >} \\ = \; & \frac{\sum_{\{m'_k\} \in (A \cap B)} < m'_{z+1}, \cdots, m'_M >}{\sum_{\{m_k\} \in A} < m_{z+1}, \cdots, m_M >}. \end{aligned} \quad (17)$$

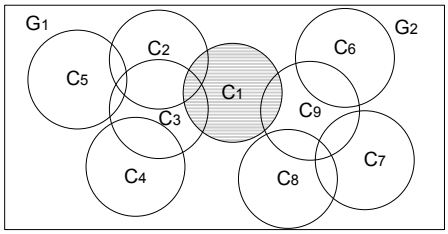

Figure 3. Deleting a Zero Count Circle

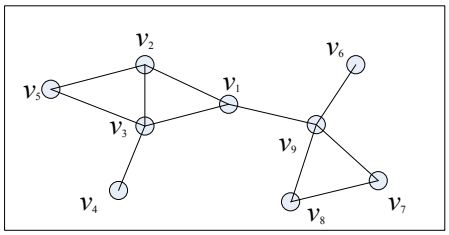

Figure 4. The Corresponding $G(V,E)$ of Figure (3)

The last equality holds because $< 0, \cdots, 0 >$ is a constant value which can be computed using Equation(1) and hence is canceled out in the function.

From Equation(17) we can conclude that deleting a zero-count node does not cause any loss of accuracy since the result in Equation (17) is the same as the computation in a similar network where C_1 is excluded. As a result, given a number of sensor nodes, the zero-count circles can be deleted first before computation. There are two major benefits from doing this. Firstly, by deleting the zero-count circles the number of subareas is reduced. Reducing M further reduces the complexity as we have discussed before. Secondly, the whole graph can sometimes be partitioned into groups by deleting these circles, especially when there are several zero-count circles. If the circles can be divided into groups that are not overlapping with each other as G_1 and G_2 shown in Figure (3), the complexity can be significantly reduced.

5.2 Partition with Balanced Minimal Cuts

In the previous section, we have shown that deleting a zero-count node simplifies the computation without losing accuracy. In this subsection we describe how to divide a sensor network into several balanced groups, each with bounded number of subareas, while incurring minimal loss of accuracy. Our solution is based on observation that we have less uncertainty in number of counts, if 1) the size of the overlapping area between different groups is small, and/or 2) the number of targets in the overlapping area is small.

With the consideration of the complexity of cutting and future compensation algorithm, our partition algorithm is recursive and pairwise optimal. The network is firstly di-

vided into two groups, one of which has a bounded number of subareas. If the size of the other group is still out of range, the partitioning algorithm is applied again until all groups have bounded number of subareas and their *pmf*s can be computed separately. The algorithm is described in more detail in the next few subsections.

5.2.1 Optimization Objectives

Based on the layout of overlapping areas, a sensor network can be modeled into a topology $G(V,E)$, where V is the set of the N sensor nodes $v_1, v_2, \cdots, v_N$ and edge e_{ij} exists between node v_i and v_j if and only if the sensing areas of the two nodes v_i and v_j overlap with each other. The weight of the edge e_{ij} is decided by both the size of overlapping area and the count values of nodes v_i and v_j. Formally,

$$e_{ij} = e_{ji} = \begin{cases} W(rs_{ij}, n_i, n_j) & C_i \bigcap C_j \neq \emptyset \\ 0 & C_i \bigcap C_j = \emptyset \end{cases} \quad (18)$$

where rs_{ij} is the percentage that the overlapping area between circles i and j out of the total area of circle i and circle j, W is an increasing function of rs_{ij}, n_i and n_j, respectively. A good example of W is $W(rs_{ij}, n_i, n_j) = rs_{ij} \times (n_i + n_j)$. Figure (4) shows the corresponding $G(V,E)$ of the sensor network in Figure (3).

If we partition G into two subgroups G_1 and G_2, we can define the objective function f_{obj} as the sum of the weights of all the edges cut by the partition. More precisely, f_{obj} can be expressed by the following equation:

$$f_{obj} = \sum_{\substack{v_i \in G_1 \\ v_j \in G_2}} e_{ij}. \quad (19)$$

The objective is to find a partition that minimizes f_{obj}.

5.2.2 Partition Algorithm

We develop a partition algorithm based on the Fiduccia-Mattheyses (FM) Algorithm [22]. For a given graph, the goal is to find a partition that divides the circles into two groups and minimizes f_{obj} as described before. We bound the size of the first group G_1 so that the *pmf* of T_1 can be computed directly. We apply the partitioning algorithm to G_2 recursively, until the size of G_2 is small enough and we can compute the *pmf* of T_2 directly as well.

We name the objective function f_{obj} as the *cutting size* since it denotes the total weight of *cutting edges* by a partition. The size of G_1 should always be smaller than the maximum size such that the *pmf* of T_1 can be computed directly. The size of each group should also be greater than a minimum size in order to maintain the accuracy of counting. These requirements on the size of the two groups are termed as the *balance requirements* (BR). The goal of our algorithm is to *find a partition that has the minimum cutting size while satisfying the balance requirements*.

As shown in Figure (5), the partition algorithm consists of several iterations, called *passes*. Each pass has several steps. In each step, a vertex that has the best *gain* is selected and moved to the other group. The *gain* of a vertex represents how much the cutting size can be reduced by moving

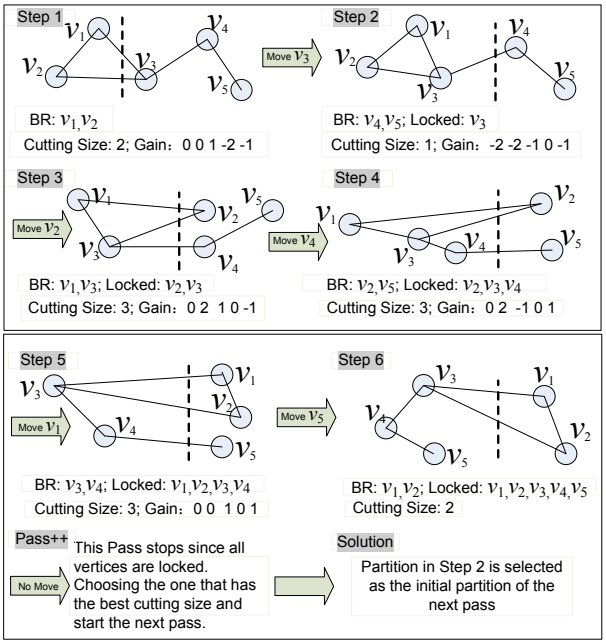

Figure 5. An Example of Partition Algorithm

this vertex to the other group. A vertex can be only moved once in a single pass and BR should always be satisfied. A single pass process stops when all vertices have been moved once or moving any unmoved vertex violates BR. Then the best partition (the one has the minimum cutting size) during the whole pass is selected as the starting partition of the next pass process.

An example of how the first pass works in a simple network is shown in Figure (5).

- **Step 1:** Initially, the vertices are divided into two groups randomly as shown in Figure (5) where v_1, v_2 belongs to one group and v_3, v_4 and v_5 belong to the other. This is the starting partition. Before performing any moving, the *gain* of each vertex is computed. Suppose all the edges in the figure have an equal weight of 1. Then for vertex v_3, the gain of moving it to the other group is 1 since the cutting size changes from 2 to 1. The gain of all the other vertices can also be computed in this way. We use (0, 0, 1, -2, -1) to denote the gain of the vertices v_1, v_2, v_3, v_4, v_5. Suppose the BR in this example requires the size of each group should be no less than 2. Due to this requirement, v_1 and v_2 can not be moved since it violates BR. Based on these observations, v_3 is selected to move to the other group since it has the best gain. It is also marked as *locked* after it is moved. A locked vertex can not be moved any more in the following steps during the current pass process.
- **Step 2:** In step 2, the gain of each vertex is updated. v_4, v_5 can not be moved in this step due to the balance requirement although v_4 has the best gain. v_3 can not be moved either since it has been locked. As a result, v_2 is selected to move to the other group although its

Algorithm 2 Partitioning Algorithm

Input: G and C_i
Output: Partition P_{new} : $\{G_1, G_2\}$

```
1: P_old ← {G1, G2}    %random initial partition
2: {V} ← all vertices in G1 or G2
3: repeat
4:    index ← 0; P_old = P_new
5:    repeat
6:       Compute the gain for all the vertices in V
7:       v ← An unlocked vertex satisfies BR and has the
         maximum gain
8:       if v ∈ G1 then
9:          G1 ← G1 − v; G2 ← G2 + v
10:      else
11:         G1 ← G1 + v; G2 ← G2 − v
12:      end if{record partition for each step}
13:      P_save[index++] ← {G1, G2}
14:   until no vertices in V can be moved
15:   P_new ← A minimum cutting partition in P_save
16:   Unlock all vertices
17: until P_old == P_new
```

gain is negative, i.e., moving v_2 makes the result worse.

- **Other Steps:** Similar process continues until Step 6 when all vertices are locked.
- **Selection and Unlock:** The partition in Step 2 is selected as the starting partition of next pass process since it has the smallest cutting size. All vertices are unlocked, ready for next pass.

A pass, which includes the above steps, repeats itself until there is no positive gain from moving any more, i.e., the partition selected at the end of a pass is the same as its starting partition at the beginning of this pass. Then this partition is the final partition of the algorithm.

In the example shown in Figure (5), the output of the second pass process is the same as its starting partition which is the partition in Step 2 in the figure. This partition is the final result. The whole process of the algorithm is shown in Algorithm (2). The complexity of this algorithm is $O(n^2)$.

6 Accuracy Compensation

In Section 5, we described how we can partition using minimal cutting. We can reduce the computational complexity to a certain level by setting the maximal size of each subgraph to a threshold *MAX*. However, partitioning leads to loss of accuracy. In this section, we propose two methods to compensate for the loss of accuracy caused by partitioning.

We start with an example shown in Figure (6). In this figure, there are 10 circles in total. Suppose there is no zero count circle as in this example. Using the partitioning algorithm described in Section 5, we can identify the best place of the first partitioning should be between C_1 and C_2, where there is only one overlapping subarea denoted as

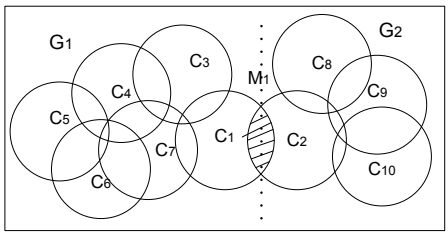

Figure 6. An Example of Partitioning

M_1 as shown in the figure. Then the whole graph will be divided into two groups with the six circles on the left belonging to G_1 and the four circles on the right belonging to G_2.

We can compute the *pmf* of the total number of distinct targets for G_1 and G_2 (denoted as T_1 and T_2) separately. If we estimate the final *pmf* by simply combining the *pmf* of T_1 and T_2 assuming that they are independent, there would be two main factors making the result inaccurate.

1. T_1 and T_2 are actually dependent since the two groups have an overlapping subarea M_1.
2. The targets in subarea M_1 are counted twice since they belong to both G_1 and G_2.

We propose two methods to compensate for the errors caused by these two factors. The first method compensates for errors by deducting the *pmf* counts in the overlapping area, called Partitioning Compensation Minus (P.C.Minus). The second method compensates by adding the *pmf* counts in the overlapping area, called Partitioning Compensation Plus (P.C.Plus). P.C.Minus is simple and efficient for complex topologies and extremely large-scale, while P.C.Plus achieves high accuracy with more overhead.

6.1 Partition Compensation Minus

A major factor that will cause the result to be inaccurate is that the targets in the overlapping area of the two groups have been counted twice. In order to eliminate such an error, we need to estimate the number of targets in the overlapping area and then subtract them from the final result.

As discussed in Section (4), the cost of computing the *true* pmf of the number of targets in the overlapping area (denoted as T_{ol}) is no less than the cost of computing the *pmf* of the total number of targets within the whole network. We only include a certain number of circles in the computation of T_{ol} in P.C.Minus. As shown in Figure (6), we can only include C_1 and C_2 in the computation of the *pmf* of T_{ol}. We can also include C_3, C_8 and other circles in the computation. The more circles that are included, the more accurate the result is and the more computation overhead. However, if we include circles that are too far away from the overlapping area, the computation cost increases much faster than the accuracy we gain. This is because the further circles are away from each other, the less correlated they are. The number of subareas included in the computation of T_{ol} should also be no greater than *MAX* which is

Algorithm 3 Partitioning Compensation Minus (P.C.Minus)

Input: G_1, G_2, M_{ol}
Output: *pmf* of total target counts T

1: $\{C_i\} \leftarrow$ minimum set of circles cover all overlapping subareas of G_1 and G_2
2: $M_{ol} \leftarrow$ sizeof $\{C_i\}$
3: **if** $M_{ol} < MAX$ **then**
4: $C_q \leftarrow$ the nearest circle from overlapping subareas
5: $M_{ol} \leftarrow$ sizeof $(\{C_i\} + C_q)$
6: **while** $M_{ol} < MAX$ **do**
7: $\{C_i\} \leftarrow \{C_i\} + C_q$
8: **end while**
9: **else**
10: compute T_{ol} by future partitioning
11: **end if**
12: Compute the *pmf* of T_1 directly
13: Compute the *pmf* of T_2 directly or by future partitioning
14: Compute the final *pmf* of T where $T = T_1 + T_2 - T_{ol}$

defined as the group size. In a small-scale network, this requirement can be satisfied since the size of overlapping subareas is usually small. However, in a large-scale complex network, more circles are clustered and more subareas will be included in a partition. The number of circles that are included in the computation of T_{ol} would be large. In this case, future partitioning would be needed and the *pmf* of T_{ol} will be computed recursively as the *pmf* of T_2. Based on these analyses, we develop the algorithm of P.C.Minus as shown in Algorithm (3). In essence, it first obtains the *pmf* of T_1, T_2, T_{ol} of G_1, G_2 and M_1 respectively. The *pmf* of total count T is calculated as $T = T_1 + T_2 - T_{ol}$.

6.2 Partition Compensation Plus

The P.C.Minus algorithm reduces the duplicate count in the overlapping area and gives a certain level of compensation to the final result. However, it doesn't solve the problem that T_1 and T_2 are not independent. Moreover, T_{ol} is also not independent with T_1 and T_2. In order to improve accuracy further, we develop the P.C.Plus method in this section.

We start with a simple assumption that both of C_1 and C_2 have detected one target. The number of targets in the overlapping subarea M_1 will have only two possibilities: $\mathcal{N}(M_1) = 0$ and $\mathcal{N}(M_1) = 1$.

We define $C_1' = C_1 - M_1$ and $C_2' = C_2 - M_1$. C_1' and C_2' are both partial circles excluding the overlapping subarea M_1. For a fixed $\mathcal{N}(M_1) = m_1$, the count information of C_1' and C_2' can be derived from the following equations:

$$\begin{aligned} \mathcal{N}(C_1') &= \mathcal{N}(C_1) - \mathcal{N}(M_1) \\ \mathcal{N}(C_2') &= \mathcal{N}(C_2) - \mathcal{N}(M_1). \end{aligned} \quad (20)$$

We further define $G_1' = G_1 - M_1$ and $G_2' = G_2 - M_1$ where G_1' and G_2' are disjoint as shown in Figure (7) and the

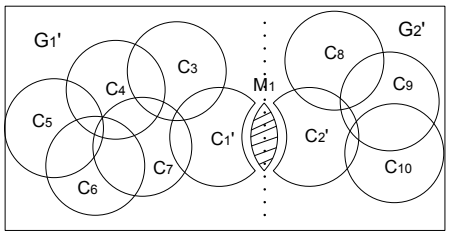

Figure 7. P.C.Plus Method

Algorithm 4 Partition Compensation Plus (P.C. Plus)

Input: G_1, G_2, M_k
Output: *pmf* of total target counts T

1: $G_1' = G_1 - M_1 - M_2 - \cdots - M_k$, $G_2' = G_2 - M_1 - M_2 - \cdots - M_k$
2: Compute the *pmf* of $\{\mathcal{N}(M_1), \cdots, \mathcal{N}(M_k)\}$
3: **for** each $\{m_1, \cdots, m_k\}$ **do**
4: Update count information for the remainder of overlapping circles
5: **end for**
6: Compute the *pmf* of T_1' and T_2'
7: Compute the *pmf* of $T(m_1, \cdots, m_k)$ where $T(m_1, \cdots, m_k) = T_1' + T_2' + m_1 + \cdots + m_k$
8: Compute the final *pmf*: $T = \sum P(\mathcal{N}(M_1) = m_1, \cdots, \mathcal{N}(M_k) = m_k) \times T(m_1, \cdots, m_k)$

corresponding T_1' and T_2' are independent. We use $T_1'(m_1)$ and $T_2'(m_1)$ to denote the counts in G_1' and G_2', respectively, under the condition that there are m_1 targets within M_1. The *pmf* of $T_1'(m_1)$ and $T_2'(m_1)$ are computed for each particular m_1, using the count and size information of C_1', C_2'. Similarly, $T(m_1)$ denotes the total count under the condition that there are m_1 targets within M_1. Finally, T can be expressed as follows:

$$\begin{aligned} T \quad = \quad & P(\mathcal{N}(M_1) = 0)(T_1'(0) + T_2'(0) + 0) \\ & + P(\mathcal{N}(M_1) = 1)(T_1'(1) + T_2'(1) + 1) \end{aligned} \quad (21)$$

where $P(\mathcal{N}(M_1) = m_1)$ denotes the probability that there are m_1 targets in subarea M_1. From Equation (21) we could see that if we find the *pmf* of $\mathcal{N}(M_1)$ we can compute the final *pmf* of T by enumerating all the possible m_1 values, computing the *pmf* of $T_1'(m_1)$ and $T_2'(m_1)$ for each m_1 and then combining them with different weights $P(\mathcal{N}(M_1) = m_1)$.

In the general case when there are k overlapping subareas $M_1, M_2, \cdots, M_k$, the process is quite similar. The *pmf* of $\{\mathcal{N}(M_1), \mathcal{N}(M_2), \cdots, \mathcal{N}(M_k)\}$ is computed first. Then, for a particular value $\{\mathcal{N}(M_1) = m_1, \mathcal{N}(M_2) = m_2, \cdots, \mathcal{N}(M_k) = m_k\}$, the count information of the overlapping circles are updated. $T(m_1, \cdots, m_k)$ can then be computed by simply combining the two independent random variables $T_1'(m_1, \cdots, m_k)$ and $T_2'(m_1, \cdots, m_k)$ plus the sum of $\{m_k\}$. The *pmf* of T can be computed as the sum of these $T(m_1, \cdots, m_k)$ values with different weight. The algorithm for the general case is shown as Algorithm (4);

It's not difficult to find that the P.C.Plus method has eliminated error caused by both of the two factors. The only inaccuracy comes from the estimation of the *pmf* of overlapping subareas. If we compute the *pmf* accurately, the complexity will be no less than the complexity of computing the final distribution. Like in P.C.Minus, only a subset of circles are included in the computation of the *pmf* of $\{\mathcal{N}(M_1), \cdots, \mathcal{N}(M_k)\}$.

Since the only error comes from the estimation of the *pmf* of the overlapping area, we can adjust the level of accuracy by increasing the number of circles in the computation, regardless of the maximum size of each group. The complexity also increases as the required level of accuracy becomes higher and higher. In the extreme case, when all circles are included to compute the *pmf*, there would be no loss of accuracy, however also no savings of computation. In practice the maximum size is still fixed to *MAX* in any *pmf* computation.

The P.C.Plus algorithm works well when the topology is simple and the intensity λ is small, i.e., there are not too many overlapping subareas and the numbers of targets within most circles are small. Otherwise the complexity will be too high because we need to compute T_1' and T_2' multiple times for every possible value of $m_1, m_2, \cdots, m_k$. The times we repeat on computing the *pmf* of the same block is exponential with respect to the number of overlapping subareas. It is also exponential with respect to the average of m_k. As a result, the complexity is too high when the topology is complex and when λ is large although we have already reduced the complexity to a polynomial function of M. However, P.C.Plus is a very good algorithm when used in a network with simple topology and moderate intensity as shown in the evaluation.

7 Evaluation

In this section we compare the performance of each algorithm in terms of both accuracy and computation overhead in different scales of networks consisting of 10 to 1000 nodes. If not specified, the nodes are randomly generated into a 2D unlimited space. Targets are also randomly generated with equal probability everywhere. We run all the simulations using Matlab on a computer with an Intel 2.13GHz Core2Duo processor. In subsections (7.1), (7.2) and (7.3) we assume that the intensity λ is already given in order to compare the accuracy of computation affected only by choosing a different algorithm. In (7.4) we show simulation results using an estimated λ instead of the real one. In (7.5) we study the scenario when target distribution is not uniform.

7.1 On Small-Scale Networks

In this experiment, we compare the performance of different algorithms in small-scale networks, where ten sensor nodes are overlapped with each other with a total number of subareas $M = 20$. The area of each sensing circle is set to 9 units and the total coverage of these ten sensor nodes is 63 units. The true number of distinct targets in the region is 12 which corresponds to $\lambda \approx 0.2$. The reason that we study this small-scale scenario is to compare the *pmf* estimated by each algorithm with the true *pmf* directly, which is impossible in large-scale networks since the true *pmf* of Equation (7) is always too complex to compute.

Figure (8) shows a typical example of the *pmf*s with the ground-truth count of 12, using four different methods: Direct Computation (D.C.) which corresponds to (i) the True *pmf* in the figure, (ii) Partition Method with P.C. Plus, (iii)Partition Method with P.C. Minus and (iv) Partition Method without any compensation at all (P.O.).

Compared with the True *pmf* in Figure (8), P.C. Plus gives the most accurate result that the *pmf* it computes is almost the same as the True *pmf*. P.C. Minus is less accurate than P.C. Plus, but is still a good estimation. The *pmf* computed by P.O. has a horizontal shift of about 2 due to the fact that the targets within the overlapping area have been counted twice. If we simply estimate the number of targets by adding the numbers reported by each node, we will get 17, which is worse than any of these methods.

We run similar simulations 200 times. In each simulation, λ is fixed to 0.2 and targets are randomly regenerated to make the count information different. The expected values of T (denoted as $E(T)$) are calculated from the *pmf* computed by the four methods. The average absolute error of $E(T)$ is compared in Figure (9). Note that the true expected value of T in each simulation is around 12, from which we can see the accuracy of P.C.Minus with a relative error of about 0.8% and P.C.Plus with a relative error of 0.3%.

Figure (10) shows the average runtime comparison (in log scale) between P.C.Plus, P.C.Minus and direct computation for various λs from 0.07 to 0.21. We don't include P.O. in the comparison here, because the complexity of P.O. is almost the same as that of P.C. Minus. From Figure (10), we can see when $\lambda \approx 0.2$, the runtime of P.C. Plus and P.C. Minus are less than 1% of the runtime of direct computation. Compared with the results shown in Figure (8) and Figure (9) we can conclude that P.C. Plus is the best choice in small-scale network.

7.2 On Large-Scale Linear Networks

We place 100 nodes linearly. We study linear networks in order to compare the performance of both P.C.Plus and P.C.Minus, which is similar to the reason we study small-scale networks. The size of each sensing circle is still set to 9 and the total coverage is around 600. We vary λ from 0.01 to 0.2 (The expected number of targets in the region varies from 10 to 130). Since this is a large-scale network where M is around 200, the True *pmf* is no longer available. As a result, the average expected value of T is compared in the evaluation.

For each λ, a fixed number of targets are generated randomly. Three partition methods (P.C. Plus, P.C. Minus and P.O.) are used to compute the *pmf* of T. Then $E(T)$ is calculated and recorded. We run similar simulations 200 times

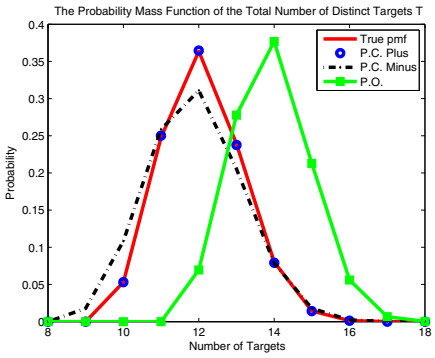

Figure 8. The *pmf* of a Small-Scale Network

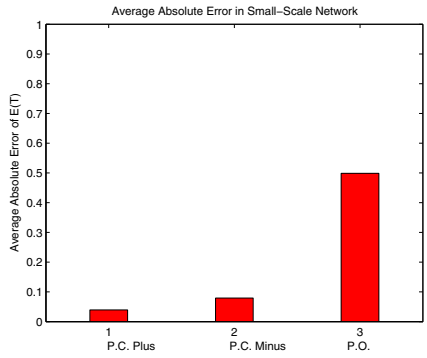

Figure 9. Average Expected Count Error

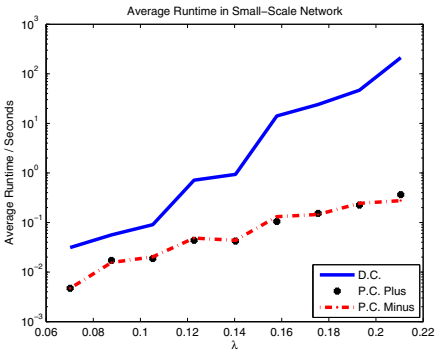

Figure 10. Average Runtime vs. λ

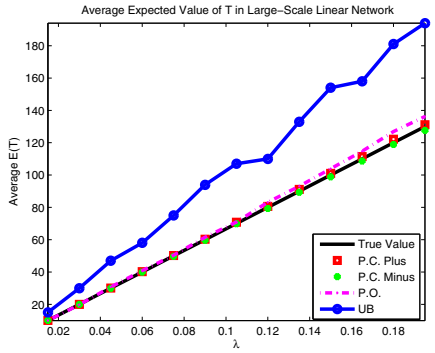

Figure 11. Average Expected Count vs. λ

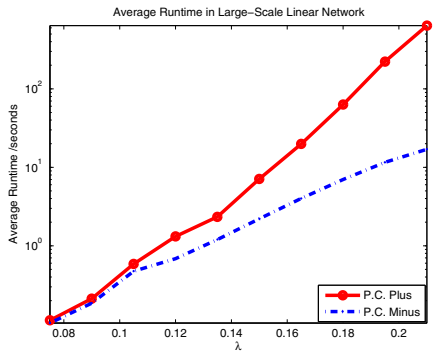

Figure 12. Average Runtime vs. λ

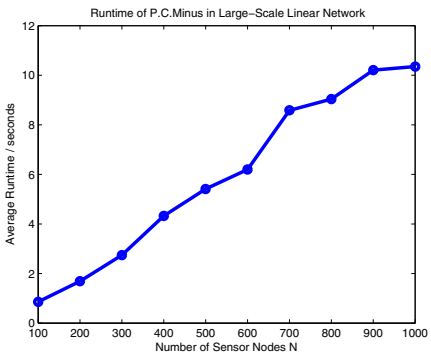

Figure 13. Runtime vs. Network Size

for each λ and compare the average of $E(T)$ with the true number of targets generated in the simulation.

Figure (11) shows the comparison of average $E(T)$ computed by each algorithm. It also shows the estimation by the naive method of simply adding up all the count information which corresponds to the "UB" in the figure. From the figure we can see that in terms of average expected value of T, both P.C. Plus and P.C. Minus give a very good estimation while P.O. has a higher expected value as expected. However, P.O. is still much better than the naive method.

Figure (12) shows the comparison of average runtime of P.C. Plus and P.C. Minus. Again P.O. is not included in the figure since it almost has the same runtime as P.C. Minus. From the figure we can see that as the intensity λ increases, P.C. Plus has a greater increase in runtime since it suffers too much from repeatedly computing the *pmf* of the same block. The runtime of P.C. Minus also increases when λ increases, but much slowly than P.C. Plus. From these observations we conclude that in large-scale linear networks, P.C. Plus is a good choice when λ is moderate while P.C. Minus is the best choice when λ is large.

Figure (13) shows the runtime of P.C. Minus for networks consisting of different numbers of sensor nodes from 100 to 1000 when λ is fixed to 0.1. From the figure we can see as the number of sensor nodes increases, the runtime of P.C.Minus increases linearly.

7.3 On 2-Dimensional Large-Scale Networks

We place 100 nodes randomly in a 2-Dimensional area. Since the topology this time is too complex, P.C. Plus is too complex to use since it needs too many iterations. As a result, only the performance of P.C. Minus and P.O. are compared here, using similar simulations as described in (7.2). Figure (14) shows the comparison of average $E(T)$ when λ varies from 0.01 to 0.25. It also shows the estimation of the naive method UB. From the figure we can see that P.C. Minus gives a very good estimation in terms of expected value of T. P.O. gives a higher expected value as expected, but the estimation is still better than UB.

7.4 Impact of Estimated λ

In (7.1), (7.2) and (7.3) we use a given intensity λ in the simulation in order to compare the performance of each algorithm. However, the intensity information is sometimes not available. As a result, λ should be estimated using just the count information. One way to estimate λ is to sum all the count information and then divide by the total area of the

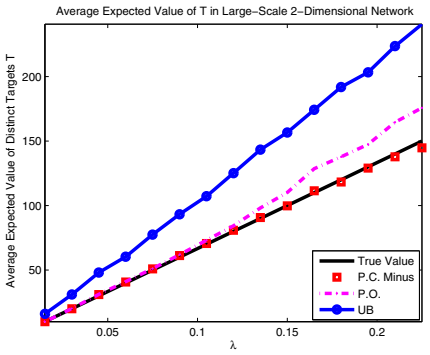

Figure 14. Average Expected Count vs. λ

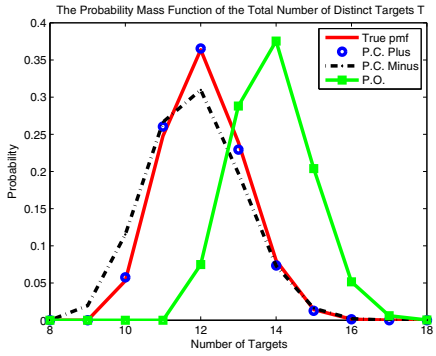

Figure 15. *pmf* with Estimated λ

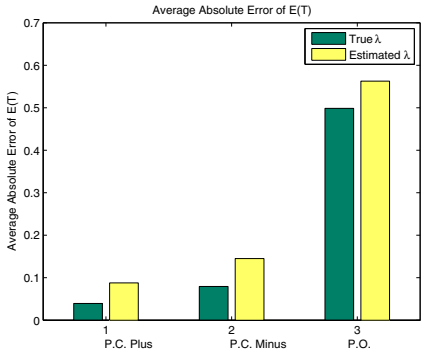

Figure 16. Average Absolute Count Error

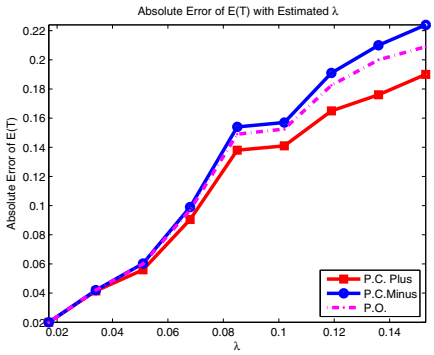

Figure 17. Average Absolute Count Error vs. Estimated λ

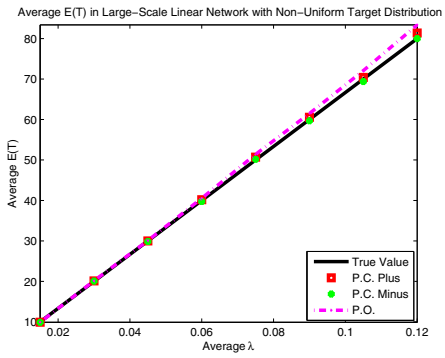

Figure 18. Expected Count with Non-uniform Distribution

sensing circles, assuming that they are not overlapping. For example, in the small scale network in (7.1), if the count information is $\{1,1,1,3,5,2,1,2,1,0\}$ and the size of each circle is 9, λ can be estimated as $\lambda = \frac{17}{9\times 10}$.

We repeat the simulation in (7.1) using estimated λ in P.C. Plus, P.C. Minus and P.O. while keeping an actual λ in the direct computation to obtain the result of the True *pmf*. Figure (15) shows the *pmf* computation result in the same scenario as in (7.1), where the count information is $\{1,1,1,3,5,2,1,2,1,0\}$ and the true number of target is 12. Comparing Figure (15) with Figure (8) we can see the *pmf* computed by the three algorithms are slightly changed, but P.C.Plus and P.C.Minus are still good approximations.

Figure (16) shows the average absolute error of the expected value of T in small-scale network. From the figure we can see the average absolute error increases when estimated λ is not so accurate. However, the difference is negligible due to the fact that $E(T)$ is around 12 and the relative error is extremely small especially for P.C. Plus and P.C. Minus.

Figure (17) shows the absolute error of $E(T)$ computed using an estimated λ with $E(T)$ computed using true λ in large-scale linear network. From the figure we can see P.C.Plus is the most robust when λ is inaccurate. The absolute error caused by the inaccuracy of λ only causes a very small difference of $E(T)$ which is negligible due to the fact that $E(T)$ is large according to Figure (11) and the relative error is extremely small.

Based on all comparisons we conclude that our algorithm can be easily applied to the real scenarios when the intensity information of λ is not available.

7.5 Impact of Non-Uniform Distribution

In previous evaluation we have shown that the P.C. Plus and P.C.Minus can approximate the *pmf* of T very well, under the assumption that the targets are uniformly distributed. However, it's not always the case that the targets are uniformly distributed with the same intensity λ everywhere. In this subsection we study the case that targets are non-uniformly distributed, i.e., λ in each subarea can be different. Again an estimated λ is used in the computation of each algorithm.

We study large-scale linear networks to compare the performance of all three partition methods. We divide the 100-node linear network into 10 different sub-networks each of which has a random intensity λ value when we distribute targets. In the simulation, λ is estimated using the same technique as in (7.4). Figure (18) shows the simulation re-

sults of average $E(T)$ in this scenario. From the figure we can see in terms of average expected value of T, P.C.Plus and P.C.Minus are both robust to change of distribution models.

8 Conclusion

The double-counting problem has been sufficiently addressed in the context of communication, however, to our knowledge, there is no existing solution for double-counting problem in sensing. This paper presents an efficient and accurate method to estimate the number of targets within a monitored area with duplicate counts among adjacent sensors.

Using a partition method, we significantly reduce the computation complexity of calculating the probability mass function of total counts. Using several compensation methods, we improve the accuracy with adjustable computational overhead. This work is theoretical in nature, however, it can be practically applied since most sensor systems do not require precise counts, but reasonable accurate estimations. As a result, statistical counting is a viable approach. To inspect whether our solution is practical with respect to several assumptions we make, we evaluate the design with estimated λ values as well as non-uniform distributions. Results reveal the accuracy of statistical counting degrades slightly when the true λ is unknown and targets are non-uniformly distributed. Through extensive simulation over various kinds of network settings, we demonstrate that accurate statistical counting within $1 \sim 3\%$ relative error can be obtained with orders of magnitude reduction in computation, compared with the exhaustive enumeration-based approach. Without loss of generality, we use Poisson distribution as a concrete example, we believe the ideas of partition with balanced minimal cuts and accuracy compensation are applicable to other target distributions as well.

Acknowledgements

This research was supported in part by NSF grants CNS-0626609, CNS-0626614 and CNS-0720465.

References

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*, 2004.

[2] Vipul Singhvi, Andreas Krause, Carlos Guestrin, Jr. James H. Garrett, and H. Scott Matthews. Intelligent light control using sensor networks. In *SenSys '05*, 2005.

[3] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[4] Geoff Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *EWSN '05*.

[5] Prabal Dutta, Mike Grimmer, Anish Arora, Steve Biby, and David Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05*, 2005.

[6] D. Min Ding; Terzis, A.; I-Jeng Wang; Lucarelli. Multi-modal calibration of surveillance sensor networks. *Military Communications Conference, 2006. MILCOM 2006*.

[7] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A.Tirumala, Q. Cao, T. He, J. A. Stankovic, T.Abdelzaher, and B. Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *SenSys'05*, 2005.

[8] F. Zhao, J. Shin, and J. Reich. Information-Driven Dynamic Sensor Collaboration for Tracking Applications. *IEEE Signal Processing Magazine*, March 2002.

[9] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[10] Abhinav Kamra, Vishal Misra, and Dan Rubenstein. Counttorrent: Ubiquitous access to query aggregates in dynamic and mobile sensor networks. In *ACM SenSys*, Sydney, Australia, November 2007.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Operating Systems Design and Implementation*, December 2002.

[12] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences*, 1985.

[13] Yong Gao, Kui Wu, and Fulu Li. Analysis on the redundancy of wireless sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 108–114, New York, NY, USA, 2003. ACM.

[14] Radu Stoleru, Tian He, John A. Stankovic, and David Luebke. High-Accuarcy, Low-Cost Localization System for Wireless Sensor Networks. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, November 2005.

[15] Juan Liu, Ying Zhang, and Feng Zhao. Robust distributed node localization with error management. In *MobiHoc '06*, 2006.

[16] B. Son, S. Shin, J. Kim, and Y. Her. Implementation of the real-time people counting system using wireless sensor networks. 2007.

[17] Joengmin Hwang, Tian He, and Yongdae Kim. Exploring in-situ sensing irregularity in wireless sensor networks. In *In Fifith ACM Conference on Embedded Networked Sensor Systems (SenSys 2007)*, 2007.

[18] Z. Vincze, R. Vida D. Vass, and A. Vidacs. Adaptive Sink Mobility in Event-driven Clustered Single-hop Wireless Sensor Networks. In *INC 2006: Proceedings of Sixth International Network Conference*, 2006.

[19] Laura Savidge, Huang Lee, Hamid Aghajan, and Andrea Goldsmith. QoS-Based Geographic Routing for Event-Driven Image Sensor Networks. In *Proc. of the 2nd International Conference on Broadband Networks*, 2005.

[20] Q Cao, T He, L Fang, T Abdelzaher, J Stankovic, and Sang Son. Efficiency Centric Communication Model for Wireless Sensor Networks. In *IEEE INFOCOM*, 2006.

[21] Morris H. DeGroot. *Probability and Statistics,3rd Edition*. Academic Internet Publishers Incorporated, 2006.

[22] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *DAC '82: Proceedings of the 19th conference on Design automation*, 1982.