Multiple Target Counting and Tracking using Binary Proximity Sensors: Bounds, Coloring, and Filter

Lei Song
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
leisong03@gmail.com

Yongcai Wang
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
wangyc@tsinghua.edu.cn

ABSTRACT

Binary proximity sensors (BPS) provide extremely low cost and privacy preserving features for tracking mobile targets in smart environment, but great challenges are posed for tracking multiple targets, because a BPS cannot distinguish one or multiple targets are in its sensing range. In this paper, we at first address the counting problem by presenting a maximum clique partition model on unit disk graph, which leads to a tight lower bound for estimating the number of targets by a snapshot of sensor readings. Then, to more accurately count and track the multiple targets by sequential readings of sensors, we state the key is to comprehensively infer the states behind the events. Therefore, at each event we infer which target may trigger the event via a dynamic coloring technique (DEC) and predict the potential regions of the multiple targets by a colorful area shrinking and expanding approach. Such an approach generates multiple potential scenarios containing different colors to interpret the sequential events, where the number of colors indicates the different estimations of the target number. Then we designed multi-color particle filter (MCPF), which is run in parallel in each scenario to enumerate and evaluate the potential trajectories of the targets under the color constraint. The likelihoods of the trajectories are evaluated by each target's movement consistence. The overall best trajectory over all scenarios is voted to provide not only the most possible target number, but also the trajectories of the targets. Extensive simulations were conducted using a multi-agent simulator which show good accuracy of the proposed multi-target tracking algorithms.

1. INTRODUCTION

This paper investigates the problem of tracking multiple targets using a network of binary proximity sensors (BPS). A BPS is a low-cost sensor, which provides binary detection to the mobile users in its proximity. It outputs "1" when one or more targets are presenting and "0" otherwise. It cannot distinguish individual or multiple targets, nor provide any moving direction or location specific information. But BPS can well preserve users' privacy, which make it highly suitable for occupancy sensing in smart environments. It detects users' demands without leaking users' privacy, which is the foundation for smart control, such as on-demand lighting or HVAC control. The typical BPS sensors include Passive Infrared sensor, ultrasound sensor, and microwave radar sensor etc.

MobiHoc'14, August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632959.

Although a BPS provides very limited information, prior work [1] has shown that a collaborative network of BPSs can yield respectable single target tracking accuracy. In [1], the authors showed that a target can be localized with a location error proportional to $\frac{1}{\rho R^{d-1}}$, where ρ is the sensor density, R is the sensing range, and d is the dimension of the space. But significant difficulties will be encountered for tracking multiple targets, because the sensors can't tell how many targets are presenting, which leads to the difficulty of disaggregating the motion trajectories of the multiple targets.

Only in 1-Dimension network, the counting problem was investigated in[2], which presented a lower bound of the presenting targets as the maximum number of *positively independent sensors*[2], where the positively independent sensors are "on" sensors sufficiently far apart, or separated by at least one "off" sensors [2]. They presented a greedy algorithm to calculate the lower bound. However, this lower bound is shown conservative in 2-D space in this paper, and a novel unit disk graph (UDG) model for the BPS network is proposed, which leads to a tighter lower bound, i.e., *minimum number of cliques that partition the UDG formed by the "on" sensors.* This new lower bound is valid in both 1-D and 2-D networks.

Then how the sequential readings overtime can further improve the target counting and tracking is investigated. The sequentially state changing events reported by distributed sensors introduce temporal and spatial constraints, i.e., some timely close by, but spatially far away events must be triggered by different targets because the targets must move continuously. Based on this intuition, despite the little information provided by each event, we comprehensively infer and evaluate possible target distributions and trajectories that may coincide the sequential events to improve the counting and tracking accuracy. We define a *time-spatial distance* between events, which can help more accurately counting the targets than that of the snapshot-based counting. Then for tracking and disaggregating the trajectories of the multiple targets, we propose dynamic edge coloring (DEC) and Multi-color Particle filter (MCPF). The DEC generates and maintains a set of target distribution scenarios of different colors. The number of colors in a scenario indicates the estimation to the number of the targets. In each scenario, from the sensor readings before and after an event, shrinking and expanding methods are presented to predict the feasible regions of the targets at the occurred event.

Then in each scenario, we further developed a multi-color particle filter (MCPF), which generates random particles according to the possible target distributions to enumerate and evaluate the potential trajectories of the targets. The MCPF calculates the likelihood of the multi-target trajectory by the consistence of the moving speeds of each target. So that, the overall best trajectory among all trajectories over all scenarios is voted at each event time, which provides not only the estimation to the most possible number of the targets, but also the disaggregated trajectories of these targets. Extensive evaluations by multi-agent simulations showed that the proposed methods can provide satisfactory counting and tracking accuracy for multiple, anonymous, uncooperative, simultaneously moving targets, despite the very limited information provided by the BPSs.

The remaining part of this paper is organized as follows. Background and problem model are introduced in section 2. Lower bound for target number counting is presented in section 3. Interplay of DEC and MCPF is presented in Section 4. Simulation results are presented in section 5, with conclusions drawn in Section 6.

2. BACKGROUND AND PROBLEM MODEL

We introduce the problem model and the most related works in this section.

In this work, we consider an idealized sensing model for the BPS. Let's consider n sensors are deployed in a two-dimensional area-of-interest. The locations of the sensors are assumed known by self-localization techniques [3] and the sensors are assumed timely synchronized [4], so that the concurrent readings of the sensors can be measured to form a snapshot. Each sensor monitors the targets (e.g. people) within its sensing range and outputs "1" when one or more targets are in its sensing range, and "0" otherwise. Let's R denote the sensing radius of a sensor and we assume targets within this range can be detected by the sensor without error. Each sensor only reports the state transition events, i.e, the timestamp and the event type ("1" to "0", or "0" to "1") to a central collector. The collector infers the concurrent states of all the sensors and conducts information processing to estimate the number and locations of the presenting targets. We assume the data collection is well addressed by routing and MAC protocols.

Under above sensing model, the sensing regions of the n sensors will partition the area-of-interest into m *patches*. A patch is a close area bounded by the sensing boundaries of sensors. Each patch can be uniquely coded by a length n vector $\mathbf{P}_i = \{b_{i,1}, \cdots, b_{i,n}\}$, where $b_{i,j} = 1$ if sensor j covers patch i and $b_{i,j} = 0$ otherwise. Since the locations in a patch have the same code, conventionally, the discriminability resolution of the BPS network is considered to be determined by the size of the patches[1]. Whereas, when taking the time attributes of the events into consideration, we can further characterize the possible locations of the target which triggers the event by the arc it is crossing. This can further narrow down the freedom of the location estimation than

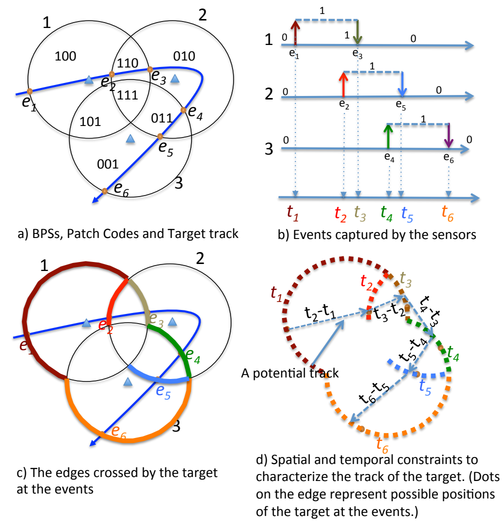


Figure 1: Spatial and temporal information provided by the BPS sensors for motion track estimation when a target moves through the sensing regions of three BPSs.

the patch-based representation. For targets not triggering the event, we can still estimate their locations by the patch indices. Therefore, as introduced in Section 4, by shrinking and expanding based approach, we can interactively use edge-based and patch-based location representation to more accurately estimate the possible locations of the targets.

Fig.1 shows an example of tracking a mobile target by three BPS sensors. Fig.1a) shows the formed patches and the patch codes. Fig.1b) shows the state transition events reported by the sensors. Fig.1c) shows the crossing edges of each event by different colors. Fig.1d) illustrates the spatial and temporal information provided by the sensing events, where the crossing edge at each event is discretized to dots to represent the possible positions of the target at the event time. Tracking multiple targets is more complex, because when an event occurs, we don't know who triggers it and don't have clue for other targets' locations.

2.1 Existing Counting Limits and Bounds

The problem becomes difficult when we consider tracking multiple targets by the BPS network. By assuming the sensor can distinguish the targets, i.e., can distinguish which event is triggered by which target, FindingHuMo [5] proposed adaptive Hidden Markov Model (HMM) to disaggregate the motion trajectories of the multiple targets. By assuming knowing the total number of the targets, [6] proposed multiple pairs shortest path algorithm to search the most possible trajectory using walking speed variance to disaggregate target trajectories. But the problem becomes extremely difficult when the number of targets is unknown and when the sensors cannot distinguish the targets, which is however, the most practical scenario, because in real applications, neither the Infrared sensor, ultrasound sensor nor the microwave sensor can distinguish the targets, and we generally don't know the number of the presenting targets.

For this practical case, only in one-dimensional space where the targets move along a line, a method for evaluating the lower bound of the number of targets was presented in [2]. In the method, if the “on” sensors can be partitioned into at most X positively independent sets, where the positively independent sensors are “on” sensors whose sensing regions don’t overlap, or separated by at least one “off” sensors, Theorem 4 in[2] stated that the number of targets in the sensing field is not less than the cardinality of X, i.e., $|X|$.

But we found this lower bound was conservative when applied to 2-D spaces. Fig.2 explains this lower bound in several scenarios. In Fig.2a) and Fig.2b), the two “on” sensors (red segments) form two positively independent sets, so the lower bound of the target number is two in these cases. In Fig.2c), the two “on” sensors form only one positively independent set, so the lower bound of target number is one. But in the 2-D example, as shown in Fig.2d), the three “on” sensors are not independent, forming only one positively independent set, so the lower bound of target number is one. But it is not hard to verify that it needs at least two targets presenting in the positive sensing field (white area) to trigger the three sensors to “on”. So the lower bound is conservative. Because tracking multiple targets in 2-D spaces is a more general scenario in practice, tighter lower bound, and efficient target counting and tracking algorithms are needed.

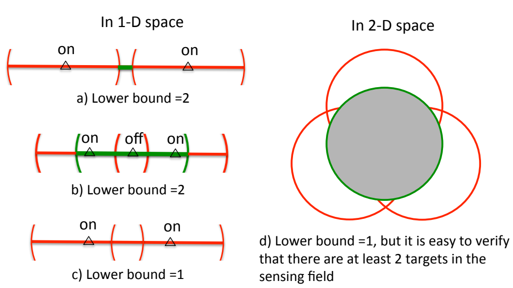


Figure 2: Calculate lower bound of target number in 1-D and 2-D spaces by the maximum number of positively independent sensors.

2.2 More related works

BPS-based target counting and tracking has attacked many studies. In [7], Busnel et al. explored the limits of multiple object tracking using a network of binary motion sensors. They showed that the multiple objects cannot be identified by BMSs on a general graph even by an omniscient external observer if all the objects can freely move on the graph. They showed that priori restrictions on the graph or priori restrictions on object movements can help to make the problem solvable. Further than FindingHuMo [5], to overcome the difficulties of training the HMM model, *MiningTraMo* was proposed in [6] to infer the most possible trajectories by multiple pairs shortest path algorithm based on walking speed variance. In [8], Jeswani et al. proposed an approach for target tracking using sparsely deployed binary sensor network. The trajectory of a target is approximated by a piece-wise curve, where each piece is an estimated as tangent between sensing regions of two sensors. In [9], cluster-based decentralized variational filtering was proposed for target tracking in binary sensor networks. In [10], He et al. proposed a hybrid multiple target tracking scheme, which conducts coarse-scale tracking by binary proximate sensors to narrow down search area, and uses high-end sensors for fine-grained tracking. In [11], Cao et al. presented collaborative scheme for tracking groups of targets using BMSs. In [12], Wang et al. presented a distributed energy efficient target tracking scheme using binary sensor networks. The algorithm exploited the successive detection events of neighboring binary sensors to infer the trajectory of the target. In [13], Rao-BlackWellised Particle filters are introduced to tracking and identify target, which is implemented in a device-free people tracking and identification system. In System presented in [14], motion character is introduced to improve locating accuracy in netwokred BPS. A family of bound on performance for multiple target tracking is given in [15], whose core idea is locating error is introduced by uncertain origin of measurement. Based on this uncertainty, error in optimal solution can be bounded.

3. COUNTABILITY BY A SNAPSHOT

To investigate the countability problem by a snapshot of sensor readings, we firstly present sufficient and necessary conditions for precisely target counting in 2-D space.

3.1 Conditions for Precisely Target Counting

Let’s consider an arbitrary snapshot at time t. Let $\mathbf{S}_t \in (0,1)^n$ be a length-n vector representing the “on/off” states of the n sensors at time t. Let $\mathbf{A}_t$ denote the set of the “on” sensors. Let $\mathbf{E}_t$ denote the set of the “off ” sensors. Let function $P(s)$ return all the patches covered by the sensor set s. Let F_t be the *feasible target space* in which the targets may present. Then:

$$\mathbf{F}_t = P(\mathbf{A}_t) - P(\mathbf{E}_t) \quad (1)$$

Let S^i be the set of positive sensors that sense the target i. Let function $F(S^i)$ return the *feasible target space* contributed by these positive sensors in set S^i, then

$$F(S^i) = P(S^i) - P(S^i) \cap P(\mathbf{E}_t) \quad (2)$$

Then a sufficient condition for precisely target counting can be stated as:

THEOREM 1. *(Sufficient condition). If S^i is the set of sensors that sense the target i, the sufficient condition for precisely target counting is that $\forall i \neq j, F(S^i) \cap F(S^j) = \emptyset$.*

PROOF. Suppose there are m targets. Since $\forall i \neq j$, $F(S^i) \cap F(S^j) = \emptyset$, the feasible target spaces are naturally partitioned to m isolated feasible areas, one per target. Therefore, we can precisely determine the m isolated feasible target spaces, which precisely indicates there are m targets. □

Note that this sufficient condition requires the groups of sensors triggered by the different targets are enough separated. The sufficient condition presented in [2] which requires the pairwise distances between any two targets to be larger than $4R$ is a special case of this sufficient condition.

THEOREM 2. *(Necessary condition). If S^i is the set of sensors that sense target i, the necessary condition for precisely target counting is that $\forall i \neq j, F(S^i) \neq F(S^j)$.*

PROOF. We prove by contradiction. If $F(S^i) = F(S^j)$, then we can never determine whether there is one or two targets in the area $F(S^i)$. Therefore, for precisely target counting, there must be $\forall i \neq j, F(S^i) \neq F(S^j)$. □

The necessary and sufficient conditions give properties but are not practical. A more general requirement is to estimate the number of targets by a given snapshot of the sensor readings, for which, we present a new tight lower bound for estimating the number of targets.

3.2 UDG Model for Feasible Target Space

We firstly present a unit disk graph (UDG) model to model the geometrical structure of the feasible target space. For a given snapshot, the feasible target space in which the targets may locate is determined by $\mathbf{F}_t = P(\mathbf{A}_t) - P(\mathbf{E}_t)$. This area may be partitioned by the sensing areas of the "off" sensors into some isolated *islands*, which are denoted by $\mathbf{L}_t = \{L_1, L_2, \cdots, L_v\}$, where v is the total number of the isolated islands. Note that two feasible areas L_i and L_j are isolated if they are spatially separated by the sensing regions of the "off" sensors.

An example is shown in Fig.3, in which the feasible target space (the space in white) is separated by the "off" sensors (space in grey) into three isolated islands, denoted by L_1, L_2, L_3 respectively. A trivial lower bound for the number of target is therefore the number of the isolated islands, because each island contains at least one target. But this lower bound is not tight. We can improve this lower bound by considering the topology of each island. Note that each island is a union region of the sensing regions of some "on" sensors subtract the regions covered by the "off" sensors. If we normalize the sensor radius R to 1, we can construct a *Unit Disk Graph* (UDG) by connecting the vertexes of the intersected "on" sensors in each feasible island. The construction rules are:

- The "on" sensors in the isolated islands are chosen as the vertexes of the UDG.
- For any two "on" sensors, if they have intersected sensing region and the intersected region is not fully covered by the regions of the "off" sensors, an edge is added between these two vertexes.

By examining all the "on" sensors by above rules, A UDG is constructed in each island. Examples of UDG construction are shown in Fig.3. Note that the UDG in each island is connected and the UDGs in different islands are disconnected.

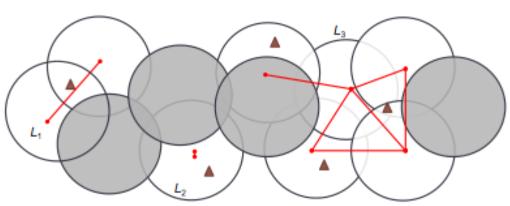


Figure 3: Unit disk graph for target counting

3.3 Lower Bound of Target Number

By investigating the topologies of the UDGs, a tighter lower bound for the number of targets can be calculated.

Definition 1 (clique). *In graph theory, a* clique *in an undirected graph $G = (V, E)$ is a subset of vertices $C \in V$, such that every two vertices in the subset C are connected.*

Definition 2 (clique partition). *Given a UDG $G = (V, E)$, a clique partition is a partition of V into nonempty, disjoint sets $\{C_1, C_2, \cdots, C_u\}$, such that each set $C_i, (i = 1, \cdots, u)$ induces a clique. The minimum clique partition (MCP) is to partition the vertex into the minimum number of cliques.*

Let's consider a clique partition on a UDG, which is denoted by $\{C_1, C_2, \cdots, C_u\}$. If $\mathbf{S}_j$ represents the sensors partitioned into the clique C_j, because the cliques are disjoint, $\forall C_i \neq C_j, \mathbf{S}_i \cap \mathbf{S}_j = \emptyset$. All the sensors in the same clique can be triggered "on" by one sensor (because these sensors are connected, i.e., have a common feasible target region), therefore, finding the lower bound of the target number is equal to find the minimum clique partition of the UDG.

Lemma 1. *For a clique containing k sensors, one target located in the k-intersected sensing area of the k sensors is sufficient to make all the k sensors be in "on" state.*

Proof. From the construction of UDG, two sensors are connected if they have intersected sensing area in the feasible target space. Because k sensors in a clique are fully connected, they share a common k-intersected feasible target space. Only if one target is presenting in the feasible k-intersected area, all sensors in the clique will triggered "on" to coincide their readings. $\square$

Theorem 3. *(Lower bound of the target number by a snapshot). Given a snapshot consisting of v isolated islands. Let N be the number of targets that matches the sensor readings, then there must be $N \geq \sum_{i=1}^{v} l_i$, where l_i equals to the minimum number of cliques partitioning the UDG of the ith island.*

Proof. Since the isolated islands are independent, we only need to prove that the number of targets in an island i is at least l_i, i.e., the least number of targets equal to the minimum number of cliques that partition the UDG.

1). For sufficiency aspect, from Lemma1, one target placed at the most intersected area in each clique is sufficient to trigger all sensors in the clique to "on" state, so l_i targets is sufficient to make the states of all the sensors in the UDG be coincide with their "on" readings.

2). For necessity aspect, if $l_i - 1$ targets can make all sensors' states to be "on", we can repartition the UDG by selecting the sensors detecting the same target into the same clique. Because the sensors detecting the same target must have intersected sensing region, i.e., connected. This new partition will partition the UDG into $l_i - 1$ cliques. This is contradict to that l_i is the minimum number of cliques partitioning the UDG, so that there is at least l_i targets in the sensing field. $\square$

Corollary 1. *The lower bound of target number in Theorem 3 is tight, because it can be exactly achieved by placing a target in the most intersected sensing region in each clique.*

However, because the MCP of the UDG is not unique, there are different versions of minimum clique partitions. Even if we know exactly that the least number of targets are presenting in the sensing field, it is still ambiguous to determine the locations of the targets.

Moreover, regarding the computation cost for estimating the lower bound of the targets, the MCP problem on UDGs

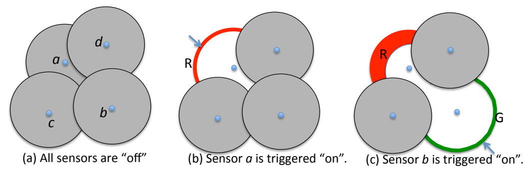


Figure 4: Example to illustrate the idea of time-space distance and FCA coloring

is a well-known NP-hard problem[16]. Recent work in [17] provided a polynomial time approximation algorithm, which is guaranteed to be within $(1+\epsilon)$ ratio of the optimum when the input is a UDG. For the special characters of target tracking problem, the number of vertexes in an feasible island are generally limited. Therefore, we can calculate a rather accurate lower bound for the number of targets given the snapshot readings of the sensors.

An example to estimate the lower bound of the number of the targets by a snapshot is shown in Fig.3. In the three isolated islands, UDGs are constructed. The minimum number of cliques in the islands L_1, L_2 and L_3 are 1, 1, 3 respectively, so that the least number of targets estimated by this snapshot is totally five. The figure gives an example distribution of the five targets. Note that the minimum clique partition in L_3 is not unique, so that the locations of the three targets are still ambiguous, even if there are exactly three targets.

4. COUNT AND TRACK DYNAMICLY

Although Theorem 3 presents a tight lower bound for the number of targets at a snapshot. The exact number of the targets and their locations are still difficult to be estimated from the snapshot. But this doesn't exclude the possibility to count and track the targets more accurately by further exploring the temporal and spatial constraints from the sequential events of the sensors.

In this section, we investigate how the temporal and spatial information introduced by the state changing events can help to improve the target counting accuracy. In particular, we propose a concept of *space-time separation* between the events. We show that when the events triggered by different targets are enough separated in *space-time distance*, they can be successfully attributed to different targets.

4.1 Space-time Separation Between Events

The sequential readings of the sensors provide more contexts for multi-target counting and locating. In an instance when a sensor's state switches from "off" to "on" (or from "on" to "off"), it must because some targets are entering (leaving) its sensing region from its sensing boundary.

DEFINITION 3 (FEASIBLE CROSSING ARC (FCA)). *When a sensor's state change is detected, the feasible crossing arc indicates the arc segments where the targets are traversing to trigger the event without violating the states of the other sensors.*

The FCA provides temporal location constraint to a moving target. Each event maps to a corresponding FCA. To characterize whether two successive events are possibly triggered by the same target, we measure the distance between the FCAs of these two events. If the occurrence times of the two events are close, but the distance between their FCAs is large, we can distinguish confidently that the two events are triggered by different targets.

DEFINITION 4 (DISTANCE BETWEEN FCAS). *The distance between two FCAs f_a and f_b is the shortest distance between any two points on the two FCAs, where the shortest path connecting the two points must be within the feasible target space. We denote the distance between FCAs as $\|f_a - f_b\|_2$.*

Let's suppose the maximum moving speed of the targets is V_{max}, then:

DEFINITION 5 (TIME-SPACE DISTANCE). *The time-space distance between two event E_a, E_b happened at time t_a, t_b, with FCAs f_a, f_b respectively is defined as:*

$$\mathcal{D}_{a,b} = \|f_a - f_b\|_2 - |t_b - t_a| V_{\max}$$

THEOREM 4 (DYNAMIC TARGET SEPERATION). *Two events E_a and E_b must be triggered by different targets, if $\mathcal{D}_{a,b} > 0$.*

PROOF. $\mathcal{D}_{a,b} > 0$ means $\|f_a - f_b\|_2 > |t_b - t_a| V_{\max}$, which means that even the shortest distance between the two FCAs is larger than $|t_b - t_a| V_{\max}$. Because $|t_b - t_a| V_{\max}$ is the largest distance a target can travel from t_a to t_b, a target triggers E_a hasn't enough time to move from f_a to f_b to trigger E_b, so that E_a and E_b must be triggered by different targets. $\square$

Dynamic target separation can count target more accurately than the snap-shot based lower bound. An example is shown in Fig.4. Initially, all sensors are "off" as shown in Fig.4(a). The red arc in Fig.4(b) shows the FCA when sensor a turns to be "on". The green arc in Fig.4(c) shows the FCA when sensor b turns to be "on". The red area in Fig.4(c) shows the reachable area of the target who triggered E_a when the event E_b happens. Since the red target cannot reach the green arc, these two events must be triggered by different targets. If using MCP-based method, the UDG contains only one clique, the lower bound provided by the snapshot is one, which is smaller than dynamic lower bound. Therefore, dynamic counting can be more accurate in practice by considering both time and location constraints. Based on this intuition, we propose *dynamic edge coloring (DEC)* and *multicolor particle filter (MCPF)* to dynamically color the edges to count the number of targets and to disaggregate the trajectories of targets by the presenting colors and also by evaluating the consistence of the targets' movement patterns.

4.2 DEC and MCPF

DEC and MCPF are jointly designed. DEC is event driven, which is used to assign color to the FCA (feasible crossing arc) according to occurring event, which is to specify which target may trigger the event. The edge coloring is based on the time-space distance from the historical events, therefore, it is based on the historical states of the targets, which are stored and tracked by the MCPF. The MCPF stores a set of parallel *scenario trajectories*. Each scenario trajectory has different estimation to the number of the targets. For an example, a scenario trajectory considers that all the previous events are triggered by 3 targets (having three colors), but another trajectory considers the previous events are triggered by at least 4 targets (having four colors). Further, each scenario trajectory maintains a dynamic set of *feasible target trajectories*. A feasible target trajectory contains the historical positions of the targets from beginning up to now. The feasibility constraint of the target trajectory is that at each event's time, the positions of the targets must make the states of the sensors coincide with their readings. The likelihood of each feasible target trajectory is evaluated by evaluating the variance of the moving speeds of the targets. The feasible target trajectory with the overall highest likelihood provides not only the most likely number of the targets, but also the disaggregated historical trajectories of these targets.

This method depends on the consistence of the target number and the consistence of the targets' moving patterns during a monitoring period, therefore we assume the number of the target doesn't change within the short monitoring period and the targets move smoothly.

4.2.1 Scenario Generation

A snapshot of sensor readings is obtained at each event. For the ambiguity of the sensor readings we cannot determine exactly the target distribution, but we can generate a set of possible distributions by Monte-Calo method. Given a snapshot, we generate distribution scenarios with different number of targets, then these scenarios can be evolved and be evaluated lately by the following events. Since determining the minimum number of presenting targets is NP-hard, we propose an efficient method to generate the distribution scenarios on a proposed *positive patch graph*, induced by a snapshot of the sensor readings.

Definition 6 (Patch Graph). *refers to the directional neighboring graph of patches. Each positive patch (i.e., in the feasible space of targets) is treated as a vertex. Edge $v \rightarrow u$ exits iff corresponding patch P_v and P_u have a common arc and P_v is inside the arc. Each arc separates the area into two regions; the inside region refers to the one close to the circle center (the circle where the arc is on).*

The problem of target distribution generation is then to place some targets into these positive patches, so that all the patches can be in "on" state, i.e., all sensors' states coincide their readings.

Theorem 5. *When we place one target in a patch, all its offspring patches and its ancestor patches on the patch graph can be "overlooked". The occupied patch and the overlooked patches can be deleted from the patch graph when we consider the placement of the next target. After all the patches are deleted, the placement of the targets coincide the sensor readings.*

Proof. In the patch graph, if P_o is an offspring of P_v, then the sensors covering P_o must be in a subset of the sensors covering P_v. If P_a is an ancestor of P_v, there must be one common sensor covers both P_a and P_v. Therefore, when we place a target at P_v, a sensor covering P_o and a sensor covering P_a must be triggered "on", so that both P_a and P_o can be overlooked. $\square$

Further, when the targets are uniformly distributed, the patch with larger size has higher probability to contain a target. Therefore, we also took the patch size into consideration, and developed algorithm 1 to efficiently generate the possible target distribution scenarios.

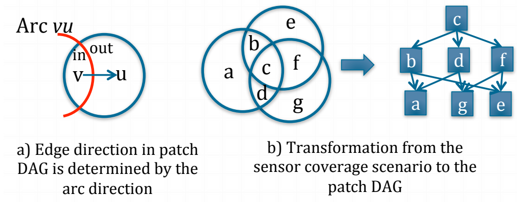


Figure 5: Explanation of Patch DAG

Algorithm 1 Generate-Target-Distribution-by-Snapshot()

Require: $\mathbf{S}$ is the snapshot. $\mathbf{G_p(S)}$ is the patch graph.
Ensure: $O_{1:m}$, Occupancy state for each Patch coincides with $\mathbf{S}$.

$O_{1:m} \leftarrow 0$.
while $\mathbf{G_p(S)} \neq \emptyset$ **do**
  Select v form $\mathbf{G_p(S)}$ randomly with probability according to the size of the patch v
  $O_v \leftarrow 1$.
  $G_p(S) \leftarrow G_p(S) \setminus \mathbf{O}(v) \setminus \mathbf{A}(v)$, where $\mathbf{O}(v)$ are the offsprings of v and $\mathbf{A}(v)$ are the ancestors of v on $G_p(S)$
end while
output $O_{1:m}$

Algorithm 1 output the most likely target distribution in positive patches. Since the estimation may be incorrect, we generate many alternative target distributions by running the algorithm multiple times. Then we categorize the target distributions by putting the distributions that contain the same number of targets into a common scenario. After the categorizing, we suppose there are Γ scenarios and each scenario contains different number of target distributions as shown in Fig.6.

4.2.2 Handle an Event

Scenario generation is carried out by the first snapshot after initialization. Now suppose we know the latest scenarios and possible target trajectories at event $t-1$. Once a new on/off event happens at event t, we try to estimate the target distributions at time T_t. Note that the events are indexed by $1, 2, , t-1, t$. It is not necessarily the interval between two events are equal. We denote T_t as the time of event t. The target trajectory updating scheme contains four steps: 1) determining the FCA, 2) dynamically edge coloring, 3) location updating for all the targets, and 4) evaluate the likelihood of particles.

1). Determining the FCA: When an event happens, the feasible crossing arc is easily determined by comparing the two snapshots before and after the event. The FCA are the intersected edges of the state changing sensor with other positive sensors surrounding it at the event time.

2) Dynamically edge coloring: After determining the FCA, for each feasible target distribution of each scenario, we evaluate the color of the FCA. This is based on the target distribution estimations at the event $t-1$. This step judges two things:

1. It helps to verify the feasibility of the guessed target

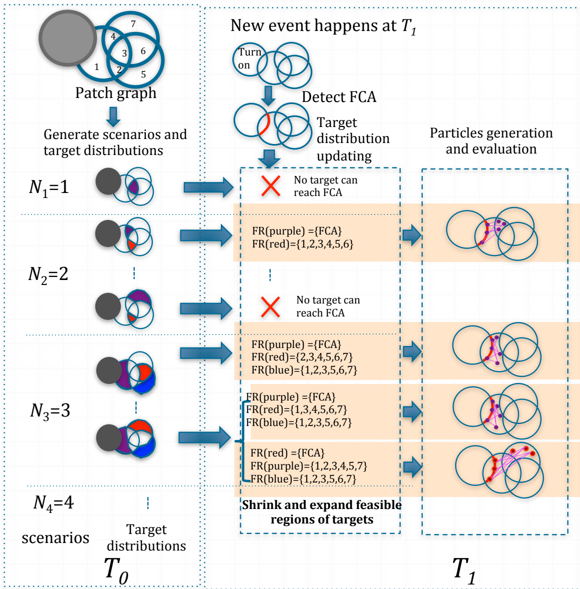


Figure 6: Example of scenario generation and particle generation

distributions at the last event. If based on a distribution at event $t-1$, all the targets cannot reach the FCA at event t, the distribution is judged infeasible and will be deleted.

2. For feasible distributions, we estimate which target is on the FCA at event t, i.e, the color of the arc, which is the basis for generating new target distributions.

To assign color to the FCA, this step needs to calculate the shortest distance from the feasible patches to the FCA, i.e, patch-arc distance, which is complex because both the patches and the FCA are in irregular shapes. We designed an efficient Monte-Calo algorithm to carry out this task. Δ points are randomly generated in the feasible patch, whose shortest distances to the FCA are calculated, and the overall shortest distance to the FCA is used as the shortest distance from the patch to the FCA. The shortest distance from a point to an arc can be easily calculated via geometric method. For a considering patch, let's denote d_{min} be the shortest distance from this patch to the FCA, then if

$$d_{min} - \varepsilon > V_{max}(T_t - T_{t-1}) \tag{3}$$

the target in the patch at T_{t-1} cannot reach the FCA at T_t, where ε is a small value to compensate the calculation error of d_{min} by Monte-Calo. If in a target distribution at T_{t-1}, all the targets cannot reach the FCA, the target distribution is judged invalid and will be deleted. If a target can reach the FCA, we update the target distribution as following.

3) Feasible Region Shrinking and Expanding: Suppose a target distribution in a scenario at T_{t-1} has Θ targets, among which, Θ_1 targets can reach the FCA at T_t and $\Theta_2 = \Theta - \Theta_1$ targets cannot reach the FCA. In this case, we generate Θ_1 new possible target distributions at T_t. In each generated target distribution, a target which can reach the FCA is put on the FCA (i.e., *shrinking*), while the feasible regions of the other targets will be expanded based on the elapsed time from T_{t-1} to T_t to a broad area where they may reach. So that by shrinking and expanding, we update the location estimations of all the targets at event T_t. It helps us to update not only the possible locations of the targets that may trigger the event, but also the possible locations of the unobservable targets at T_t.

The feasible region of a target is maintained by a patch vector $\{P_{i,1}, \cdots, P_{i,k}\}$, which are a set of patches that the target i may locate at time t. Expanding of feasible region is carried out on the patch graph. From T_{t-1} to T_t, the maximum distance a target can move is $d_{max,t} = V_{max}(T_t - T_{t-1})$, so all its connected positive patches within $d_{max,t}$ will be added into its feasible patch vector at T_t. Different methods can be used to calculate the patch-patch distance: 1) Since the patches are static, the patch-patch distances can be evaluated offline only once by Monte-Calo method, i.e., generating random points in two patches and calculating the average distance between the points in the two patches; 2) An more efficient method of expanding the feasible region is to simply add all direct neighboring positive patches of the current feasible patches into the feasible region when a new event happens. By knowing the feasible region of targets at time T_t, dynamic coloring can be carried out by (3) at time T_{t+1} when an new event happens.

4) Particle Generation and Likelihood Evaluation: In each generated target distribution, a target is on FCA and some other targets are predicted to be in their feasible regions. In particle generation step, we generate the possible positions of the target on the FCA and generate the possible positions of other targets in their feasible regions. All particle generation and the likelihood evaluation are for the same color target. Multiple targets' particle filters work in parallel.

Let's consider only the kth target in a distribution of Θ targets at time T_{t-1}. Its MCPF stores: 1) M most possible trajectories for the target up to time T_{t-1} and the possible end positions denoted by $\{x_{k,j}(T_{t-1})$, where $k = 1, 2, \cdots, \Theta$ is the target index. 2) the probability distribution functions (pdf) of the moving speed of the target in trajectory j, i.e., $\{p_{k,j}(v)\}, j = 1, \cdots, M$; 3) the likelihood of each trajectory, denoted by $\{l_k(\text{track}_j(T_{t-1}))\}$.

So that, at time T_t, we generate Q random positions for the target and connect these points to the M end points at time T_{t-1}, which generates $P = M \cdot Q$ particles representing the possible movement trajectories of this target from T_{t-1} to T_t. For each particle, we can evaluate the moving speed of this target by:

$$v_{k,i,j}(t) = \frac{|x_{k,i}(T_t) - x_{k,j}(T_{t-1})|}{T_t - T_{t-1}}, \tag{4}$$

where $x_{k,j}(T_{t-1})$ is an ending location at time T_{t-1} and $x_{k,i}(T_t)$ is a generated location at T_t. The likelihood of the particle from $x_{k,j}(T_{t-1})$ to $x_{k,i}(T_t)$ is evaluated by $p_{k,j}(v_{k,i,j}(t))$. To evaluate the likelihood of the trajectory goes through $x_{k,j}(T_{t-1})$ and ends at $x_{k,i}(T_t)$ by multiplying $p_{k,j}(v_{k,i,j}(t))$ with $l_k(\text{track}_j(T_{t-1}))$. So that we can get the likelihoods of the MQ particles. We sort these particles by likelihood and only preserve the most possible M trajectories for the next step. The pdfs of the target speed are updated and the M most possible trajectories are stored at T_t. Algorithm 2 presents the pseudo code of the MCPF for a target in one scenario. Note that parallel MCPFs are run for multiple targets in each parallel scenario. Fig.6 shows particle generation scenarios at T_1. Because no historical trajectory information exist, both $x_{k,j}(T_0)$ and $x_{k,i}(T_1)$ are randomly generated. For clarity, only the particles generated on FCA are plotted. The particles generated for the other targets are not visually shown.

Algorithm 2 MCPF-for-A-Target-in-A-Scenario

Require: m trajectories up to T_{t-1}; pdf of the moving speed of the target in each trajectory $\{p_{k,j}(v)\}, j = 1, 2, \cdots, m$; likelihood of each trajectory $\{l_{k(track_j(T_{t-1}))}\}$; feasible region of the target
Ensure: Updated trajectories, pdf of speed and likelihood of trajectories up to T_t;
Generate Q points in feasible region of target;
Connect Q points to m trajectories' ending points to generate mQ particles;
for $i = 1; i <= Q; i++$ **do**
  for $j = 1; j <= m; j++$ **do**
    Evaluate $v_{k,i,j}(t)$;
    Evaluate $c_{i,j} = p_{k,j}(v_{k,i,j}(t))l_k(track_j(T_{t-1}))$;
  end for
end for
Sort $\{c_{i,j}\}$;
Preserve the most likely m trajectories;
Update pdfs of the target speed in the trajectories;

4.3 Interplay of DEC and MCPF

The top level interplay of DEC and MCPF is explained in Algorithm 3. In initialization phase, scenarios and possible target distributions are generated by placing targets on the patch graph. We generate a large amount of target distributions (denote by I) to increase the probability that the distributions close to the ground truth can be generated. In online phase, the interplay of DEC and MCPF is splitted into different scenarios. In each scenario, the algorithm conducts dynamic edge coloring, feasible region shrinking and expanding for each target trajectory and generate new trajectories based on the new event. Then for each target, MCPF is run to rank the trajectories to find the top-M trajectories that can better interpret the event sequence. At last, all the scenarios are ranked by the likelihood of their most possible trajectory. The overall best trajectory provides not only the estimation of the target number but also the location estimations of the targets.

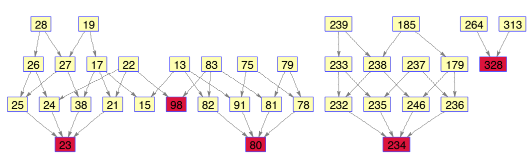


Figure 7: Random sample on patch DAG

5. EVALUATION

We developed a multi-agent simulator to evaluate the MCPF and DEC algorithms. In this platform, trace of targets are generated by random walk, which is updated frame by frame. In each frame, the reading of BPS is calculated according to the location of targets. The velocity of target is draw from normal distribution. The size of target is omitted

Algorithm 3 Interplay of DEC and MCPF

Require: Readings of sensors at T_t; Sensors' positions and sensing radius; Patch indices;
Ensure: Estimation of the most possible number of targets; most possible locations of the targets;
Initialization:
for $i = 1 : 1 : I$ **do**
  Generate-Target-Distribution-by-Snapshot ();
end for
Categorizing the generated target distributions to form Γ scenarios; suppose the ith scenario contain Θ_i targets and Γ_i target trajectories.
Online Phase:
Determine FCA by an event;
for $i = 1 : 1 : \Gamma$(number of scenarios) **do**
  for $j = 1 : 1 : \Gamma_i$(# of trajectories in scenario i) **do**
    Dynamic edge coloring;
    if (n_1 targets can reach the FCA) **then**
      Generate n_1 new target distributions;
      Shrink and expand targets' feasible regions;
    end if
    if (0 targets can reach the FCA) **then**
      delete the target distribution;
    end if
  end for
  for $i = 1 : 1 : \Theta_i$(number of targets in scenario i) **do**
    run MCPF-for-A-Target-in-A-Scenario();
  end for
end for
Rank the scenarios and trajectories by likelihood;
Output the estimation of target number and target's positions based on the most possible scenario and the most possible trajectory in the scenario;

and each target is represented by a point. Setting of simulation is presented in Fig.9, where 36 BPSs are placed in grid topology. The sensing area of BPS is denoted by black circles. The target is denoted by small star with different color. The reachable region of targets are constrained in a square denoted by red dashed line to prevent target entering region uncovered by BPS. In this setting, sensing range of BPS

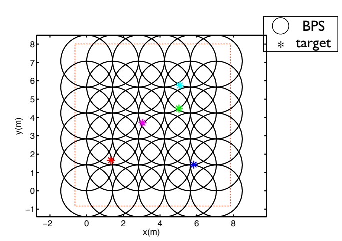
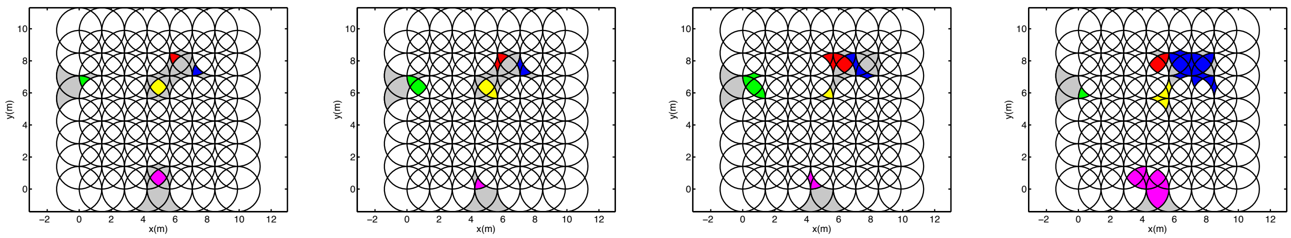


Figure 9: Simulation setting and area distribution

separate the area into 211 patches as shown in Fig.9. The maximum and the minimum number of BPS covering one patch is 4 and 1. The average of patches' size is $0.447m^2$.

The DEC and MCPF are evaluated from several aspects. We firstly evaluate the counting accuracy. Then the tracking accuracy is evaluated in patch level by comparing the located patch band with the true trajectory. After that, the accuracy is evaluated by point to point distance from



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Figure 8: Illustration of DEC and the feasible region shrinking and expanding process

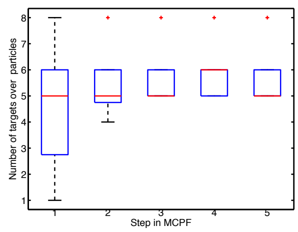


Figure 10: Convergence of number estimation

estimated trajectory to true trajectory, which is called fine-grained accuracy evaluation. Finally, the accuracy regarding to the different target density is investigated.

One of the difficulties to efficiently implement DEC is how to efficiently store the irregular patch areas and the arcs. In our implementation, we stored the starting and ending point of each arc segment, so that each FCA was represented by a set of arc segments. Then, by giving every arc and every patch an index, we can refer their locations and size in the tracking process and specify their colors and size to visualize the result of tracking. Some snapshots of the output of the multi-agent simulator in tracking five targets are illustrated in Fig.8 and 7. Fig. 7 shows the random target distribution generation on the patch graph. Fig.8(a)-8(d) shows the DEC and feasible region shrinking and expanding process at four successive events. For an example, in the Fig.8(d) when an event happens, the feasible region of the green target shrinked its feasible region onto the region of FCA, while the other targets expanded their feasible regions. We have provided our multi-agent simulator for the multi-target tracking as an open-source project, which is hosted at: https://bitbucket.org/thufresh/multi-object-tracking-simulation-platform

5.1 Counting Accuracy

In our tracking algorithms, the estimation of the number of targets is maintained by the parallel possible scenarios. Different scenarios represent different estimations of the target number. Each scenario contains a set of potential target distributions under the target number estimation. Initially, all possible scenarios and their target distributions are generated randomly by the first event. As sequential events are processed, some invalided target distributions will be deleted by DEC. When all target distributions in a scenario are deleted, the scenario is deleted. The scenario with the feasible estimation of the target number is more likely to survive as time going, which makes the counting process converge. As an example shown in Figure 10, for tracking 5 targets, our initial guess of target number was 1-8. The distribution of feasible target distributions was plotted by the box graph, which indicated the scenarios with 3 to 6 targets containing more target distributions. As more events were processed, wrong target distributions with less than 5 targets were cleaned up. Only scenarios with 5, 6 and 8 targets survived after step 3 and the possible target distributions with 8 colors had been very limited. The wrong scenarios with a little higher number estimation than the ground truth can survive for long time, which can only be corrected when targets are enough separated.

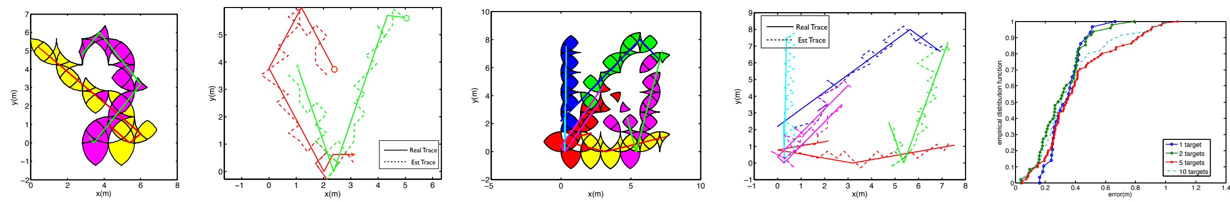
5.2 Patch Level Tracking Accuracy

Patch is generally treated as the minimum representing unit of the target position, therefore we firstly evaluate the patch level accuracy. The located patches of a target overtime will form a band of the same color. We visually plot two tracked bands of two targets, which are shown in figure 11(a). Note that the bands are plotted for the most likely scenario's most likely target distribution, whose estimation of the target number is correct. The ground truth of the targets' moving tracks are represented by the solid lines. It shows that the detected most possible patch locations well covered and followed the movement the targets. Some errors happened when two traces intersect, but the errors could be recovered soon by the particle filter.

5.3 Trace Level Tracking Accuracy

We have also drawn samples in patches to form estimated traces of the targets. The trace is timestamped, such that the distance between the estimated location and the real location at the same time is counted as the locating error. Contrast between the real traces and the estimated traces is shown in Fig.11(b). We can see the estimated trace followed well to the real traces even when the two targets are moving closely. When real traces intersected to each other, estimated traces were overlapped in segments. The cumulated distribution of the locating error is presented in Fig.11(e), which tells that more than 90% tracking error in tracking one or two targets is less than $0.5m$. This result is benefited by the small size of patches.

Increasing the density of targets would improve probability of individual target been overlooked, we can expect accuracy deterioration with number of targets increasing. Bands and traces formed by 5 targets are shown in Fig.11(c) and 11(d). Fig. 11(e) shows how target density affect locating accuracy. When number of target was 5 or 10, the tracking accuracy degraded, but it was still acceptable considering the very limited information provided by the sensors.



(a) Bands of two targets (b) Traces formed by two targets (c) Bands formed by five targets (d) Real and estimated trace of five targets (e) CDF under different number of target

Figure 11: Patch and Trace Level Accuracy of Multi-target Tracking

6. CONCLUSION

This paper has investigated the problem of counting and tracking multiple, anonymous targets by binary motion sensors, which provide only binary proximity information about the target's presenting. A tight lower bound for target counting in two-dimensional scenario based on snapshot readings of sensors has been presenting by a novel UDG model on the sensing graph of sensors. Then we investigated how to more accurately count and track the multiple targets by exploiting the temporal and spatial constraints captured by the sequential snapshots of sensors. We present the interplay of dynamic edge coloring and the multi-color particle filter to generate different color scenarios, and rank the most potential trajectory in a scenario which can best interpret the sequential sensor readings to output the estimation of the most possible target number and the trajectories of the targets. In future work, the dynamic tracking resolution, the conditions for precisely dynamic counting all need further investigations; multi-target counting and tracking when sensors are noisy should also be further investigated.

Acknowledgment

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61202360, 61033001, 61361136003.

7. REFERENCES

[1] N. Shrivastava, R. Mudumbai U. Madhow, and S. Suri. Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms. SenSys '06, pages 251–264, New York, NY, USA, 2006. ACM.

[2] Jaspreet Singh, Upamanyu Madhow, Rajesh Kumar, Subhash Suri, and Richard Cagley. Tracking multiple targets using binary proximity sensors. In *IPSN '07*. ACM, April 2007.

[3] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. MobiCom '01, pages 166–179, New York, NY, USA, 2001. ACM.

[4] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, December 2002.

[5] D. De, Wen-Zhan Song, Mingsen Xu, Cheng-Liang Wang, D. Cook, and X. Huo. FindingHuMo: real-time tracking of motion trajectories from anonymous binary sensing in smart environments. In *ICDCS*, pages 163–172, 2012.

[6] Chengliang Wang, Debraj De, and Wen-Zhan Song. Trajectory mining from anonymous binary motion sensors in smart environment. *Knowledge-Based Systems*, 37:346–356, January 2013.

[7] Yann Busnel, Leonardo Querzoni, Roberto Baldoni, Marin Bertier, and Anne-Marie Kermarrec. Analysis of deterministic tracking of multiple objects using a binary sensor network. *ACM Trans. Sen. Netw.*, 8(1):8:1–8:27, August 2011.

[8] D. Jeswani, A. Kesharwani, S. Chaudhari, V.P. Sadaphal, and R.K. Ghosh. A practical approach for target tracking in sparsely deployed binary sensor network. In *(MASCOTS)*, pages 153–160, 2012.

[9] Jing Teng, H. Snoussi, and C. Richard. Decentralized variational filtering for target tracking in binary sensor networks. *IEEE Transactions on Mobile Computing*, 9(10):1465–1477, 2010.

[10] Ting He, C. Bisdikian, L. Kaplan, Wei Wei, and D. Towsley. Multi-target tracking using proximity sensors. In *MILCOM*, pages 1777–1782, 2010.

[11] Donglei Cao, Beihong Jin, Sajal K. Das, and Jiannong Cao. On collaborative tracking of a target group using binary proximity sensors. *Journal of Parallel and Distributed Computing*, 70(8):825–838, August 2010.

[12] Zijian Wang, Eyuphan Bulut, and Boleslaw K. Szymanski. Distributed energy-efficient target tracking with binary sensor networks. *ACM Trans. Sen. Netw.*, 6(4):32:1–32:32, July 2010.

[13] Dirk Schulz, Dieter Fox, and Jeffrey Hightower. People tracking with anonymous and id-sensors using rao-blackwellised particle filters. In *IJCAI*, pages 921–928, 2003.

[14] R Gupta and S R Das. Tracking moving targets in a smart sensor network. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, pages 3035–3039, 2003.

[15] Frederick E Daum. Bounds on performance for multiple target tracking. *Automatic Control, IEEE Transactions on*, 35(4):443–446, 1990.

[16] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[17] Adrian Dumitrescu and János Pach. Minimum clique partition in unit disk graphs. *Graphs and Combinatorics*, 27(3):399–411, 2011.