Matching and Fairness in Threat-based Mobile Sensor Coverage

Chris Y. T. Ma, David K. Y. Yau, Jren-chit Chin, Nageswara S. V. Rao, Mallikarjun Shankar

Abstract—Mobile sensors can be used to effect complete coverage of a surveillance area for a given threat over time, thereby reducing the number of sensors necessary. The surveillance area may have a given threat profile as determined by the kind of threat, and accompanying meteorological, environmental, and human factors. In planning the movement of sensors, areas that are deemed higher threat should receive proportionately higher coverage. We propose a coverage algorithm for mobile sensors to achieve a coverage that will match – over the long term and as quantified by an RMSE metric – a given threat profile. Moreover, the algorithm has the following desirable properties: (1) stochastic, so that it is robust to contingencies and makes it hard for an adversary to anticipate the sensor’s movement; (2) efficient; and (3) practical, by avoiding movement over inaccessible areas. Further to matching, we argue that a fairness measure of performance over the shorter time scale is also important. We show that the RMSE and fairness are in general antagonistic, and argue for the need of a combined measure of performance, which we call efficacy. We show how a pause time parameter of the coverage algorithm can be used to control the tradeoff between the RMSE and fairness, and present an efficient offline algorithm to determine the optimal pause time maximizing the efficacy. Lastly, we discuss the effects of multiple sensors, under both independent and coordinated operation. Extensive simulation results – under realistic coverage scenarios – are presented for performance evaluation.

Index Terms—Wireless sensor network, mobile application, distributed systems.

1 INTRODUCTION

A network of sensors can be used to protect people, livestock, or the environment against harmful substances in a geographical region. For example, Oak Ridge National Laboratory (ORNL) personnel have deployed a sensor network at the Port of Memphis to protect the area’s population against the exposure to known chemical, biological, and radiological threats. A variety of sensor modalities is used to detect the presence of pollutants with sufficient accuracy and sensitivity. The project report [1] states that in choosing where to place the sensors, a pragmatic consideration is to select locations that are accessible. Besides accessibility, the report concludes that the primary factor in deciding the placement of a limited quantity of sensing resources is to assess its impact on the area’s population distribution, since “human effects represent the true consequences of failure” to detect a harmful agent and subsequently evacuate the affected population. It states that densely populated areas should receive priority attention relative to unpopulated or sparsely populated areas. Moreover, historical meteorological information, such as wind rose data characterizing the predominant seasonal distribution of wind speeds/directions, should be considered. This is because the spread of a chemical/biological/radiational plume is affected by wind conditions, and sensors in the wind’s direction will be able to monitor the most vulnerable areas and detect the plume with the smallest delay.

The Memphis Port deployment exemplifies the need and benefits to provide differential sensor coverage of different geographical areas based on a concept of threat level. Intuitively, the threat level of an area quantifies the relative danger of exposing the area due to non- or under-coverage. An area may have a high threat level because it is under high risk or because a realized risk in the area will produce severe consequences. In the Memphis Port experiment, static sensors are used. Because of the limited number of sensors, the Port area cannot be fully covered [1]. In this paper, we consider the use of mobile sensors to cover a whole surveillance area over time. Because the area coverage occurs over time, and does not have to be complete all the time, a significantly smaller number of sensors can be used compared with static sensors. We assume that the movement of a sensor can be under program control. For example, the sensor is carried by a robot supporting programmable movement. We then consider the design of a mobility algorithm to control the sensor’s movement, such that it can effect a coverage profile that matches a given threat profile.

It is clear that the economical savings of using fewer sensors have to be balanced against the costs of supporting the movement. We do not attempt to answer

References:

- Chris Y. T. Ma, David K. Y. Yau, and Jren-chit Chin are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907. E-mail: {ma18, yau, jccchin}@cs.purdue.edu.
- Nageswara S. V. Rao and Mallikarjun Shankar are with the Oak Ridge National Lab, Oak Ridge, TN. E-mail: {raons, shankarm}@ornl.gov.
- Research was supported in part by the Office of Naval Research under grant number ONR 00012275, in part by the U.S. National Science Foundation under grant number CNS-0305496.

Manuscript received June 27, 2007; revised August 26, 2008; accepted April 27, 2009.

For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-06-0189.

Digital Object Identifier 10.1109/TMC.2009.117
the question of economic tradeoff definitively one way or the other, as it depends in part on the difficulty of the sensing task relative to the movement task and in part on future technological developments. We do notice, however, that commodity robots [2], for instance, are available that are rather inexpensive, and believe that it is interesting to explore such a tradeoff. Moreover, there may be other reasons to use mobility besides economics. For example, robots may be used because they can carry sensors over a deployment field that is hard to get to for installing a static sensor (e.g., an underwater environment or the accessibility placement condition of the Memphis Port deployment). Also, mobility can be more robust against an unplanned sensor failure (e.g., an area left uncovered by a failed sensor can be covered later by another sensor that moves into the area) or an unexpected contingency (e.g., an obstacle unexpectedly appears that obstructs the sensing path, and a mobile sensor is able to navigate around the obstacle).

Our contributions in this paper are as follows:

- To the best of our knowledge, this paper represents a first effort to investigate general threat-based coverage by sensors that move during deployment.
- We identify matching and fairness as the major performance criteria in evaluating the effectiveness of coverage. We show that the two performance measures are in general antagonistic, and discuss their tradeoffs. We show how the two metrics can be unified to give a combined metric of efficacy, by adopting a common view of utility.
- We present the development of a coverage algorithm for sensors to achieve effective matching and fairness simultaneously under realistic deployment scenarios. The algorithm provides a tunable parameter to control the tradeoff between matching and fairness. The optimal parameter maximizing a given efficacy metric can be computed by an efficient offline algorithm.
- We present simulation results to quantify the performance of our algorithms. In particular, our main results show that
  - A complement of techniques contribute to a coverage algorithm that can match the coverage profile of a mobile sensor to a given threat profile with excellent accuracy.
  - There is an inherent tradeoff between matching and fairness.
  - Using more sensors can significantly improve the fairness of coverage, although the marginal improvement due to an additional sensor decreases slightly as the number of sensors increases.
  - Using more sensors – either independently or under basic coordination – does not significantly improve the accuracy of matching. Rather, the matching deteriorates slightly due to possible redundancy of coverage by multiple sensors.

- For multiple sensors, basic coordination approaches do not improve performance over independent operation. Rather, independent operation of the sensors in stochastic movement is viable, because it is simple, is shown to be effective, and is robust to sensor failure.

2 Related Work

Static sensor coverage. A significant amount of research has studied the placement problem of static sensors for optimal area coverage [3], quality of surveillance [4], or energy efficiency [5]. The relationship between coverage and connectivity, given communication range \( c \) and sensing range \( s \), has been studied for the cases of \( c/s \leq 1 \) [6] and \( c/s < \sqrt{3} \) [7]. The generalization to any \( c/s \) and to \( 2 \) \( k \)-connectivity is given in [8]. The tracking of a moving target by a network of static sensors has been studied in [9], [10]. In [9], a protocol is presented to enable the sensors to transition to a low-power state to conserve energy, without compromising the quality of surveillance. In [10], the number of sensors needed to track a radiational source under uncertainty is analyzed. The strategy for a moving target to evade a static network of sensors with minimum exposure is discussed in [11].

In the area of static sensor placement, the work closest to our problem is the Memphis Port sensor network deployment [1]. They present an iterative algorithm to place a given number of \( n \) sensors around the Port of Memphis to maximize the protection of the area’s population. At each step of the algorithm, they use a search procedure to place the next sensor at a position that will maximize the marginal gain in risk coverage. Our work addresses similar threat-based coverage, but in the context of mobile sensors.

Mobile sensor coverage. Previous work on sensor mobility has focused on moving the sensors to deployment locations that optimize the area of coverage [12], [13]. The sensors do not move during the sensing task. Other hybrid mobile/static networks have used moving relay nodes to collect data from static information sources [7]. They show that using the mobile relay as the sink is the most efficient, and that, for a dense network, the improvement in network lifetime of one relay is upper bounded by a factor of four over a static network. In [14], optimal algorithms are presented to move the sink adaptively according to the flow of current events, to minimize the communication energy or the maximum load on a specific sensor.

Due to trends in robotics and embedded sensor technologies in vehicles, there is a growing amount of work on sensors that move during deployment. The area coverage of mobile sensors is characterized in [15] under the assumption of a uniform node density. They show that mobility can significantly reduce the number of sensors needed to detect a randomly located stationary target in a given amount of time. If the target can also move and is
intelligent, it can plan its movement to avoid detection. In that case, a pursuit-evasion game can be defined. A greedy policy for directing a group of moving agents to “swarm” locations with the highest probabilities of finding an evader is proposed [16], and is shown to find an evader in finite time. The implementation of the theoretical game on unmanned aerial/ground vehicles is discussed in [17], and the use of mobile sensors orbiting in space to help minimize the time-of-capture of the evaders is considered in [18].

In the area of mobile sensors during deployment, the work closest to our problem is [19]. In their work, they study the capture of transient events by a mobile sensor. The events arrive/depart according to given stochastic processes at given points of interest (PoI) in a circular space. They show how a sensor moving in a circle at variable speeds can optimize its movement to detect the largest fraction of events. Their PoIs can be viewed as a specialized threat profile, since they are the positions where interesting targets are likely to appear compared with non-PoIs. They use a variable speed but restrict the path to be circular around what is essentially a one-dimensional space, whereas we use a fixed speed but randomness in the path selection over a 2D space.

Motion planning in robotics. Motivated by applications such as monitoring of pollutants and biomass in a lake, motion planning of robots to optimize the collection of information in an exploration task has received consideration attention [20], [21], [22], [23]. In the simultaneous localization and mapping (SLAM) problem, a robot’s sequence of actions is designed to minimize the uncertainty about the pose of the robot [24], the map of the environment being sensed [25], or both of them in an integrated manner [22]. Existing solutions aim to minimize the entropy of the posterior distribution of the pose and/or the map, thereby maximizing the information gain due to the exploration (which is terminated on resource exhaustion). In deciding its next movement, the robot typically has to trade off between branching to an unknown area in order to learn more about the map, and revisiting a previously visited location in order to “close the loop” and relocalize the robot’s pose, and such a decision can be guided either by local information [26] or global information about the whole trajectory [20]. On-line estimations of the posterior map/pose distributions can be efficiently maintained using Rao-Blackwellized particle filters [27] or the extended Kalman Filter (EKF) [28]. For path planning with multiple robots, a sequential allocation technique is proposed in [21] in which near-optimal single-robot-path planning (e.g., branch-and-bound [21] or pruned breadth-first-search based on A-optimality [20]) is successively applied to get the paths of n robots, with an approximation guarantee on optimality. In the SLAM problem, the robot’s pose is deemed uncertain, the map is deemed static, and the geography of the field under exploration is deemed unknown a priori. Because the map is static, there is no reason to revisit a location once that location has been learned, except to relocalize the robot. Hence, the main notions of threat-based coverage and fairness in this paper (i.e., to check/recheck higher-threat areas longer/more frequently for any newly appearing event) do not apply. In addition, we assume that the mobile sensor knows its own position as well as the locations of the PoIs under monitoring.

Another important problem in robotics motion planning is the optimal searcher path (OSP) problem [29]. In OSP, multiple robots plan their motion in order to (1) maximize the probability of detecting a non-adversarial target under a finite time budget, or (2) minimize the time until detecting the target. It is shown in [29] that explicit coordination by the robots will lead to a search space that grows exponentially in the number of robots. Hence, a linear-time implicit coordination approach is proposed in which each robot plans individually but shares information about its path for the other robots to consider. Our work in this paper similarly evaluates the effectiveness of coordination approaches that are simple and can be efficiently implemented. However, there is no notion of a single specific target in our problem context. Rather, we assume that simultaneous target events may appear at different PoIs, and the events may arrive or depart dynamically.

Markov Chain Monte Carlo (MCMC) algorithms. MCMC algorithms [30], [31] represent a general technique to determine the transition probabilities of a Markov Chain that will result in a desired equilibrium distribution of the state probabilities. It appears that MCMC is applicable to the matching component of our problem, by formulating the states to be the set of cells in the surveillance area and the equilibrium state probabilities to be the threat profile. The challenge would be, however, to account for the side effects of coverage, i.e., the intermediate cells covered as the sensor travels from cell i to cell j. When the fairness component of our problem is also considered, the applicability of MCMC for the conjoint problem is not clear and the algorithm design problem under MCMC would be highly non-trivial.

3 Problem Formulation

We consider the surveillance of a network area, which we call the map, for a given threat by one or more mobile sensors. For simplicity, we assume that the map is a two-dimensional rectangular space, of dimensions $x \times y$, where $x$ and $y$ are in distance units. The map is partitioned into an $m \times n$ array of cells, each of dimensions $S\times S$ (in distance units), and $S$ divides both $m$ and $n$. The cells are enumerated by their unique integer ids, 0, 1, $\ldots$, in top-to-bottom row order, and left-to-right column order within each row. Square or rectangular cells do not match the sensing regions of real-life sensors (although hexagonal cells could be used to better approximate circular cover regions, as in cellular networks), but they
are an effective approximation device widely used in the literature for sensing-based motion planning [20], [21], [22].

The distribution of threat in the map area is characterized by a threat profile, denoted by \( \Phi \). The threat level of a cell, say \( i \), is given by \( \Phi(i) \), and quantifies the risk of not covering the cell relative to the other cells. As motivated in Section 1, the threat profile should be determined according to the application, namely the kind of threat, and any relevant meteorological, environmental, and human factor. In addition, we allow certain cells to be marked as inaccessible, meaning that a sensor cannot monitor nor travel over such a cell due to physical limitations or policy decisions. An inaccessible cell, say \( i \), has a threat level of \( \Phi(i) = \text{NaN} \). Mathematically, \( \Phi \) is a probability distribution: \( 0 \leq \Phi(i) \leq 1, \{ \forall i : \Phi(i) \neq \text{NaN} \} \) and \( \sum_j \Phi(j) \neq \text{NaN} \). We assume that cells in the map are connected, i.e., it is possible to travel from any cell \( i \) to any other cell \( j \).

In solving the coverage problem, areas that are deemed higher threat should receive priority attention in the form of proportionately higher coverage. The goal is achieved by a mobility algorithm that controls the movement of the sensors (see Section 4). As a sensor moves, it will enter different cells. For the purpose of bookkeeping, we assume that a cell is covered, in the sense that any threat event present in the cell is detected, whenever a sensor is inside the cell. By bookkeeping on a per-cell basis, the bookkeeping costs can be kept low, although there may be some loss of precision in the matching, i.e., it is possible for a cell to be partially covered within range of the sensor. In general, a smaller cell size allows a finer granularity of accounting for the coverage times received by different parts of the map, and is therefore more accurate. To balance between accuracy and overhead, we assume that the cell dimension \( S \) is set comparable to the sensing range of the sensor. For example, if the sensing region is circular and of radius \( R \), then if \( S = \sqrt{2}R \), the sensor moving to and staying at the center of the cell is guaranteed to cover the whole cell without further movement inside the cell. The fraction of time that a sensor, say \( l \), spends in each cell up to time \( t \) is given by the sensor’s coverage profile, denoted by \( \Pi_l \). Specifically, \( \Pi_l(i) \) gives the fraction of time that the sensor \( l \) spends in the cell \( i \) up to time \( t \). Similar to \( \Phi \), \( \Pi_l \) is a probability distribution. When the context of the sensor is clear, we drop the superscript \( l \) for simplicity.

**Performance measures.** For a mobile sensor, the matching between the given threat profile and the achieved coverage profile at time \( t \) is quantified by the following root mean square error (RMSE) measure:

\[
RMSE_t = \sqrt{\frac{1}{m \times n} \sum_{i=1}^{m \times n} (\Phi(i) - \Pi_l(i))^2}
\]

The main intent is to measure the dissimilarity between two distributions, namely the threat and coverage profiles, represented as vectors of numbers; other quantitative measures that fit the objective could be used without affecting the effectiveness. If the sensor’s movement is a stationary stochastic process, the coverage profile will reach a steady state distribution, and the limit \( \lim_{t \to \infty} \Pi_l(t) \) exists, which will in turn determine the steady state matching performance of the algorithm.

The matching measure alone does not fully evaluate the performance of a coverage algorithm. Consider the monitoring of a cell whose threat level is 0.1. A coverage algorithm may achieve a 10% coverage of the cell in the steady state, but does so by spending one month in the cell once every 10 months. The average exposure time of the cell, i.e., the average duration of the continuous time interval over which the cell is not covered, is 9 months, and would be unacceptable if, say, the application is to monitor a residential area for flooding. In contrast, another algorithm that visits the cell (i.e., the residential area) for one minute every 10 minutes achieves the same 10% coverage, but never leaves the cell uncovered for more time 9 minutes. Any flood event can then be detected and reported in a timely manner. To further quantify the time scale over which a certain matching is achieved, we define an unfairness measure, denoted by \( F \), of the algorithm, as follows:

\[
F = \sum_i \epsilon(i) \times \Phi(i)
\]

where \( \epsilon(i) \) is the average exposure time of cell \( i \). Notice that the unfairness is defined as the weighted average of the exposure time of each cell by the threat of that cell, and is a time quantity. We therefore assess fairness by its dual unfairness measure. A fair algorithm is then one that achieves a low unfairness. For a persistent threat event, i.e., an event that remains present once it appears, the unfairness measures the weighted average of the delay until the event is detected after occurrence. For the transient events discussed in [19], both the unfairness and the matching determine the weighted average fraction of the events that will be missed.

### 4 Mobility Algorithms

In this section, we develop algorithms for a mobile sensor to determine its movement and effect coverage that matches a given threat profile with high accuracy. We target the following desirable properties of the algorithm in our design:

- **Accurate.** The algorithm should achieve a low RMSE of coverage against the threat profile (see Section 3).
- **Fair.** The algorithm should be fair (i.e., have a low unfairness value) in the sense of Section 3.
- **Stochastic.** The random movement makes it harder for an adversary to anticipate the sensor’s movement and hence avoid detection, although optimal strategies by the sensor against an intelligent adversary would require a game-theoretic analysis [32] that is beyond the scope of the paper. Also, random movement enables \( n \) sensors to be deployed independently without advance schedule planning.
or runtime coordination, but still with good performance benefits (as shown in Section 7).

- **Efficient.** The algorithm should have low space and time complexities, so that it can be efficiently executed on the mobile sensor.
- **Practical.** The algorithm should admit and obey given accessibility constraints for the coverage area. For example, a sensor carried on a terrestrial vehicle will not be able to enter sea areas in a geographical region. The algorithm should avoid movement over the inaccessible areas.

As a starting point of our design, we use a weighted random waypoint (WRW) algorithm. The random waypoint formulation in [33] has been used widely to model user/device movement in a mobile network, in which case there is a significant debate about whether the model is realistic or not. Notice that the concern of realism does not apply in our problem context, since our objective is not to model a mobile network, but to develop an algorithm for determining the sensor movement. In our algorithm, a sensor moves in a sequence of trips. The $t^{th}$ trip, $t = 0, 1, \ldots$, starts at the center of cell $s_t$ and ends at the center of cell $d_t$, the $(t+1)^{st}$ trip starts at the center of cell $s_{t+1} = d_t$ and ends at the center of cell $d_{t+1}$, and so on. For simplicity, we assume henceforth that when we say a trip starts/ends at a cell, it is understood that the trip starts/ends at the center of the cell. The movement from $s_t$ to $d_t$ occurs in a direct, straightline path at speed $v_t$, if the path does not cross an inaccessible area. Otherwise, necessary “detours” are made around the inaccessible areas while keeping the total trip length as short as possible. The details of such detours are not important, as long as the path between any two cells is fixed, so that its effects on covering intermediate cells can be considered as precomputed knowledge in the mobility algorithm. In standard random waypoint [33], the speed $v_t$ is selected uniformly randomly from a range $[v_{\text{min}}, v_{\text{max}}]$, and each destination $d_t$, also called a waypoint, is selected uniformly randomly from the whole map. In our algorithm, we let the sensor move at a fixed speed $v$ specified for that sensor. Moreover, to be threat-aware, our algorithm will consider the given threat profile in choosing a waypoint, and select a cell $i$ as the waypoint with probability $\Phi(i)$, which is the threat level of $i$.

The WRW algorithm is simple and probabilistic, thus meeting the third and fourth design objectives. Moreover, it attempts to achieve a coverage that matches the threat profile, by considering the profile in selecting the waypoints. The basic algorithm, however, fails to achieve an accurate match because it fails to consider the effects of a trip on covering the intermediate cells between the source and destination. For example, consider a map with a few high threat hotspots. In attempting to move between the hotspots to give them sufficient coverage, the sensor will also visit frequently all the cells between the hotspots, thereby overcovering the intermediate cells.

To overcome the weaknesses of the basic algorithm, WRW can be used in conjunction with the following complementary techniques:

**Maximum trip length.** In this variation, we do not allow the distance of a trip to exceed a given parameter $L$ (in distance units). Hence, in choosing the next waypoint $d_{t+1}$ after the $t^{th}$ trip, we constrain the candidate cells to be within the disc centered at $d_t$ of radius $L$. The choice of the waypoint among the restricted set of candidate cells occurs as in the basic algorithm. Limiting the trip length helps to decouple the intermediate cells visited from a set of high threat cells that require frequent visits. For example, consider two hotspots, say $i$ and $j$, in a map. A suitable maximum trip length will force the sensor to consider more possible paths to move between $i$ and $j$, thereby reducing the possibility of “warming up” the intermediate cells as a side effect.

**Adaptivity to prior coverage.** Because of the stochastic nature of the WRW algorithm, and the correlations between cells visited due to their physical positions, the algorithm’s actual coverage at any point in time may deviate from the given threat profile. To avoid such deviations from accumulating to an unacceptable level, we propose to use the sensor’s prior coverage as an input in selecting the next waypoint. Specifically, we compute the undercoverage of each cell, say $i$, as

$$C_t(i) = \max \{ 0, \Phi(i) - \Pi_t(i) \}$$

where $\Pi_t(i)$ is the fraction of time that cell $i$ was visited by the sensor up until the end of the $t^{th}$ trip. Then, the probability that a candidate cell, say $i$, is chosen as the next waypoint $d_{t+1}$ is proportional to $C_t(i)$. Considering undercoverage as a selection criterion has the obvious advantage of ramping up visits to cells that have been neglected relative to their threat level, at the expense of cells that have received too much prior coverage. Notice that if all of the candidate cells (i.e., those allowed by the maximum trip length) have $C_t(i) = 0$ (i.e., none of them is undercovered), then our algorithm picks candidate cell $i$ as the next waypoint with probability proportional to the threat level $\Phi(i)$ of the cell.

**Random pause time.** To raise the coverage of an undercovered cell, say $i$, in order to improve matching with the threat profile, the most efficient approach is for the sensor to stay in $i$ for long enough to correct the undercoverage. The approach is extremely efficient because it requires zero overhead of movement and there is no possibility of inadvertently changing the coverage of other cells due to the (now avoided) movement. However, by staying at the current cell longer, clearly the sensor will take longer before it can return to a previously visited cell. Hence, fairness suffers, showing that there is an inherent tradeoff between improving matching efficiently/accurately and being fair. The issue is not unlike scheduling in traditional systems areas. For example, in CPU scheduling, improving fairness requires increased context switching between processes,
which reduces the efficiency of the global system. To enable a useful and controllable tradeoff between the RMSE and unfairness metrics, the sensor, on reaching the destination of a trip, will stay at the destination for a pause time \( p \) (in time units) before selecting the next waypoint. The time \( p \) is drawn randomly from a distribution determined by a pause time parameter denoted by \( P \) (in time units). Specifically, at the end of the \( t \)th trip at destination cell \( i, p \sim U(0, \Omega_t(i)) \),

\[
\Omega_t(i) = \frac{P \times \Phi_t(i)}{\sum_{j \in C} \Phi_t(j)}
\]

for the basic WRW algorithm, and

\[
\Omega_t(i) = \frac{P \times C_t(i)}{\sum_{j \in C} C_t(j)}
\]

for the WRW variant that is adaptive to prior coverage, and \( C \) is the set of cells that are candidates as the next waypoint.

**Family of algorithms.** Notice that the complement of features augmenting the WRW algorithm can be orthogonally combined, thereby offering a family of algorithms for threat-based mobile coverage. We will denote a particular augmented algorithm by WRW-\textit{feat}, where \textit{feat} is a list of letters enumerating the augmentations in alphabetical order, and the letters \( \text{L}, \text{a}, \text{and} \text{P} \), are for the “maximum trip length,” “adaptivity to prior coverage,” and “random pause time” features, respectively. For example, WRW-L denotes the WRW algorithm with the maximum trip length constraint, and WRW-aLP denotes the algorithm with all the three features enabled. The experimental results in Section 7 show that each feature contributes positively to accurate matching, and hence the WRW-aLP algorithm is the most powerful in the matching respect. Additionally, the pause time parameter in WRW-aLP enables a useful tradeoff between matching and fairness, an issue that we will address in Section 4.1.

The WRW-aLP algorithm is specified in Fig. 1. In the specification, the WRW-aLP program takes as input the threat profile \( \Phi \), and the \( L \) and \( P \) parameters of the WRW-aLP algorithm. The function \textbf{SelectWaypoint}, takes five input parameters, in which the parameter \( x_t \) is the current position of the sensor, and returns the destination and pause time of the next trip. The \textbf{Accessible} function (whose specification is not shown) checks if all the intermediate cells connecting a given pair of cells are accessible, and can be precomputed for each given pair. Either for-loop in \textbf{SelectWaypoint} has a complexity of \( O(L^2 \alpha) \), where \( L = L/s \). Hence, WRW-aLP requires \( O(L^2 \alpha) \) computation after every trip of length \( O(L) \). The space costs of storing either the map of cells or the precomputed \textbf{Accessible} function is \( O(m \times n) \). Hence, WRW-aLP can handle given inaccessibility constraints and has an efficient implementation. The experimental results in Section 7 evaluate the algorithm’s effectiveness in also matching the fairness and accuracy objectives.

**4.1 Matching, Fairness, and Efficacy**

The dual concerns of matching and fairness means that coverage algorithms must be compared in a two-dimensional space. Moreover, the inherent tradeoff between the two concerns means that it will be impossible to rank many interesting algorithms in a total order. Rather, in comparing two algorithms, say \( A \) and \( B \), \( A \) may perform better in one respect, but less well in the other. Whether \( A \) or \( B \) is preferred in a given situation should depend on the context of the situation, such as the preferences of the user, or the characteristics of the application. We seek an approach to rank algorithms by a single, unifying metric, after appropriately considering the specifics of the situation.

The major difficulty in unifying the two metrics is that they are of completely different natures: Matching is measured as an RMSE, which is a percentage quantity, whereas unfairness is the threat-weighted average exposure time, a time quantity, between successive visits to the same cell. How do we combine a percentage value and a time quantity, while addressing the issue of user preferences? Our approach recognizes that a
user, in the context of a given application, derives a certain level of "satisfaction" from an achieved level of performance, in either performance aspect. For example, in monitoring a residence for flooding, the user may be quite satisfied (i.e., have a 100% level of satisfaction) if each room is checked at least every two hours, on average, but is completely dissatisfied (i.e., have a zero level of satisfaction) if a room may be left unchecked for a whole day, on average. Between two hours and 24 hours, the user’s level of satisfaction decreases linearly from one to zero. The example can be expressed as a utility function, $U(f)$, similar to the one in Fig. 9, where the utility, a number between zero and one, is given as a function of the unfairness of coverage. A similar utility function for matching, denoted by $U_M(\cdot)$, characterizes the utility as a function of the achieved RMSE.

After mapping both RMSE and unfairness values to utility quantities, we can define the efficacy of an algorithm as a weighted sum of the utilities; i.e.,

$$\text{efficacy}(f, m) = \alpha \times U(f) + (1 - \alpha) \times U_M(m)$$

where $f$ and $m$ are, respectively, the unfairness and RMSE achieved by the algorithm, and $0 \leq \alpha \leq 1$ expresses the importance of fairness relative to matching in the deployment context. Henceforth, we will say that an algorithm has a better performance if it has a higher efficacy than another algorithm.

Notice that the example utility functions in Fig. 9 are linear. Their choice is solely due to their simplicity, which facilitates illustration of our solution methodology. Real-life utility functions should reflect the preferences of users, and cannot be prescribed by any hard-and-fast rule regarding their shape or detailed parameters. In particular, concave functions might be suitable to model the law of diminishing return, i.e., the marginal increase in user satisfaction becomes smaller as the user acquires more of a performance measure, and other utility functions might be suitable for other purposes. Our discussion and solution methodology in this paper are independent of the specific utility functions, as long as such functions are available and supplied by the user.

To apply Brent’s method, we first need to find two abscissa $a$ and $b$ that bracket the optimal $P$. To do so, we start with a zero $P$ and magnify the bracket by the golden ratio until we overshoot the optimal input of the efficacy function. During the magnification, we also shift the bracket to eliminate intervals that are known not to contain the optimal point. Given the two abscissa $a$ and $b$, we then compute the optimal point $c$ within $a$ and $b$.

Two methods of searching for the optimal point are used: (1) the inverse parabolic interpolation technique, and (2) the golden section search. The first method is more efficient and converges faster than the second method or a linear interpolation technique. However, it may not always succeed for certain types of function. Therefore, we fall back on the golden section search in the case that the inverse parabolic interpolation fails to produce a solution.

## 6 multiple sensors

When the surveillance area is large, one sensor may not be sufficient to cover the area with good performance. Increasing the speed of the sensor may help to some extent, in that the sensor can move more quickly between cells that require attention and apportion its service more efficiently. A more effective approach is, however, to fundamentally increase the amount of sensing resources available by deploying $n$ sensors.

The most simple strategy of deploying $n$ sensors is to deploy them independently, each working according to the WRW-aLP algorithm. The stochastic property of the algorithm makes it likely for these sensors to distribute their service well over the network area, even without advance schedule planning or close coordination at run time. There is a concern, however, if the number of sensors is large relative to the size of the surveillance area. As the area becomes relatively smaller, it is more likely for the sensing ranges of the sensors to overlap.

The overlapped coverage is wasteful because one sensor would be sufficient to detect a threat event by our problem formulation. As a result, we also investigate two preliminary approaches of deploying multiple sensors in a coordinated manner:

- **Knowledge of global coverage profile.** In this approach, we assume that each sensor knows, at time $t$, the fraction of time that a cell is covered by any sensor up to time $t$. In adapting to the effects of the prior coverage, each sensor will then determine, independently but based on the undercoverage of each cell by all the sensors, the cells that should receive priority attention in the future coverage. Two observations are in order. First, although the sensors use the global coverage history as information, they will not communicate in order to avoid visiting the same cell at the same time. Hence, redundant coverage is not eliminated. Second, there is clearly a need to disseminate the coverage information of individual sensors to the global network in an
implementation of this approach. The information exchange can be readily supported if we assume, for example, the existence of a cellular phone infrastructure and the sensors are equipped with the necessary cellular communication interfaces. Nevertheless, our goal in this paper is not to consider how such information exchange should occur nor its runtime overhead. Instead, we are interested in the benefits of having the global information assuming that it is available.

- **Static division of responsibilities.** This approach seeks to eliminate the redundancy of coverage by partitioning the responsibilities for covering different cells between the sensors in a disjoint but complete manner. In essence, each sensor, say $i$, is assigned a job as a connected set of cells, denoted by $J_i$, such that $J_i \cap J_j = \emptyset$ for $i \neq j$, and $\bigcup_i J_i = \{\text{full set of accessible cells}\}$. Each sensor then uses the WRW-aLP algorithm to cover its set of cells in a threat-based manner. There are different ways to perform the partitioning into jobs. In particular, the division by equal area produces jobs that have an equal area of cells that are accessible. The division by equal threat produces jobs whose cells have the same aggregate threat level. In general, multiple actual divisions exist that can achieve either objective.

7 Simulation Results

We report simulation results to illustrate the performance of our algorithms. We consider coverage of a number of metropolitan cities, including San Francisco, Los Angeles (LA), Daly City, Atlanta, Paris, London, and Tokyo. The boundary longitudes/latitudes of the cities and the sizes of their populations are shown in Table 1. The maps of Atlanta, LA and Daly City are also shown in Fig. 2. As formulated in Section 3, each city area is divided into a two-dimensional grid of cells. The division (except Daly City) is according to the LandScanTM 2004 database of global population data [35]. LandScanTM provides population data in a cellular grid format with each cell corresponding to 1/120 degree of longitude in width and 1/120 degree of latitude in height. For ease of interpretation, we project the LandScan data to cartesian coordinates, according to World Geodetic System '84 (WGS84). The projection gives us $s \approx 0.75$ km, or a cell size of about $0.75 \times 0.75$ km. For Daly City, which is of size 8 cells by 8 cells, a cell size is $250 \times 250$ m, and the threat level is estimated from the building structures in each cell. Due to space constraints, we present selected experimental results in this section. The presented results are representative.

We assume that terrestrial mobile sensors are used over the cities to monitor, say, air pollutants with health impact on people. Hence, the threat level of a cell is defined as the size of the population inside that cell, because a more densely populated area will endanger more people if left uncovered. Since we model terrestrial sensors, water areas in a map (e.g., the Pacific Ocean part of the LA map) are defined to be inaccessible. In the maps used in our experiments, the water areas do not partition the land areas. Hence, it is possible for one sensor to cover all the land areas given enough time.

**Parameters and performance measures.** We use the performance measures of matching and unfairness as defined in Section 3. For matching, we scale the RMSE by the population size of the city, which gives a mismatch measured in number of people. For the unfairness, we report the weighted average exposure time in time units, where each time unit is 180 seconds. Unless otherwise specified, the following parameters are used in the experiments: (1) A mobile sensor moves at a speed of $3S$ / time unit, or about 34.8 mph; and (2) Where applicable, the maximum trip length parameter is set to be $L = 10 \times S$. Results are reported as averages of 50 simulation runs. The 25- and 75-percentiles are reported in certain experiments, in which case they are shown to be close to the means, and are omitted in the other experiments because of their small deviations from the means.

7.1 Matching by one sensor

In this experiment, we use one sensor to cover a city area, using various instances of the WRW family of algorithms in Section 4. Because the cities are large, it takes one sensor a significant amount of time to cover an entire city area. In particular, the unfairness numbers are of the order of several hours, which represent an inherent limitation due to constrained physical resources, and not due to the coverage algorithms. As the results in Section 7.5 show, the unfairness can be significantly decreased by using multiple sensors. Nevertheless, the results in this section illustrate the major performance properties of the coverage algorithms.

Fig. 3(a) and Fig. 3(f) give the threat profiles of Atlanta and LA, respectively. Figs 3(b)–(e) show the achieved steady-state coverage profiles of the WRW, WRW-a, WRW-aL, and WRW-aLP algorithms, respectively, for Atlanta. Figs 3(g)–(j) show the corresponding achieved steady-state coverage profiles for LA. For the WRW-aLP algorithm in this experiment, the pause time parameter is set to be one time unit. Visually, the matching with the threat profile improves as we progress from Fig. 3(b) to Fig. 3(e), or from Fig. 3(g) to Fig. 3(j). The visual observation can be quantitatively confirmed by referring to Fig. 4(a), in which we show the RMSE achieved by each algorithm normalized to the RMSE of the WRW algorithm (i.e., the RMSE of the WRW algorithm is shown as one, and the normalized RMSE of each algorithm shows the algorithm’s percentage improvement over WRW.) For the five cities shown, including Atlanta and LA, the normalized RMSE consistently decreases from left to right. Additionally, Fig. 4(b) shows the unfairness of each algorithm normalized to the unfairness of the WRW algorithm. Observe that the unfairness numbers of WRW-a, -aL, and -aLP are about the same, and are
<table>
<thead>
<tr>
<th>City</th>
<th>Northeast Latitude</th>
<th>Northeast Longitude</th>
<th>Southwest Latitude</th>
<th>Southwest Longitude</th>
<th>Average Width (km)</th>
<th>Height (km)</th>
<th>Dimension</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>34.033333</td>
<td>-84.025000</td>
<td>33.700000</td>
<td>-84.616667</td>
<td>54.75</td>
<td>36.97</td>
<td>40 x 70</td>
<td>1982086</td>
</tr>
<tr>
<td>Daly City</td>
<td>37.705621</td>
<td>-122.477195</td>
<td>37.700044</td>
<td>-122.483783</td>
<td>0.61</td>
<td>0.61</td>
<td>8 x 8</td>
<td>957</td>
</tr>
<tr>
<td>London</td>
<td>51.600000</td>
<td>0.100000</td>
<td>51.400000</td>
<td>-0.308333</td>
<td>28.35</td>
<td>22.25</td>
<td>24 x 48</td>
<td>4228314</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>34.195833</td>
<td>-118.120833</td>
<td>33.895833</td>
<td>-118.570833</td>
<td>41.55</td>
<td>33.28</td>
<td>36 x 54</td>
<td>4599286</td>
</tr>
<tr>
<td>Paris</td>
<td>49.029167</td>
<td>2.687500</td>
<td>48.729167</td>
<td>2.012500</td>
<td>49.51</td>
<td>33.36</td>
<td>37 x 81</td>
<td>8449465</td>
</tr>
<tr>
<td>San Francisco</td>
<td>37.820833</td>
<td>-122.304167</td>
<td>37.687500</td>
<td>-122.5458</td>
<td>21.30</td>
<td>14.80</td>
<td>16 x 29</td>
<td>802056</td>
</tr>
<tr>
<td>Tokyo</td>
<td>35.812500</td>
<td>139.962500</td>
<td>35.545833</td>
<td>139.504167</td>
<td>41.49</td>
<td>29.59</td>
<td>32 x 55</td>
<td>12072968</td>
</tr>
</tbody>
</table>

Table 1

Position and population data of seven cities.

Fig. 2. Maps of the cities under surveillance.

Fig. 4. RMSE/unfairness of mobility algorithms for six cities.

significantly smaller than the WRW unfairness. We conclude that the progression of features, namely, a, aL, and aLP, each contributes to increased matching accuracy, and WRW-aLP is the most powerful algorithm in the matching respect. Moreover, the more accurate matching is achieved without hurting the fairness.

7.2 Impact of pause time parameter

We illustrate the impact of the pause time parameter \( P \) on the matching and unfairness measures, for the case of one mobile sensor using the WRW-aLP algorithm. In this set of experiments, we vary the pause time parameter to be \( P = 1, 2, 4, 8, 16, 32, \) and 64 time units. Fig. 5 show combined plots of the RMSE (left y-axis) and the unfairness (right y-axis) as a function of \( P \), for Atlanta and LA. Notice that for both figures, as the pause time increases, (1) the unfairness increases, in a partly constant, partly linear manner; and (2) the RMSE decreases like \( 1/(P + c) \), where \( c \) is a small constant. We also show the 25- and 75-percentiles of the RMSE in Table 2 for the set of runs for Atlanta. Notice that the values deviate little from the averages. We will omit the 25- and 75-percentiles of the data distributions for the future sets of experiments, due to their closeness to the means. From this set of experiments, we conclude that there is an inherent tradeoff between the matching and fairness of coverage, and that the pause time parameter provides a means to control this tradeoff for the WRW-aLP algorithm.

7.3 Convergence time of WRW-aLP

We report experiments to illustrate the convergence of the WRW-aLP algorithm. We define convergence as the
IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. X, NO. X, MONTH 20XX 10

Fig. 3. Threat profiles and steady-state coverage profiles of mobility algorithms for Atlanta (a)–(e) and LA (f)–(j).

Table 2
Average and 25-/75-percentiles of WRW-aLP RMSE of population distribution for Atlanta, as a function of $P$.

<table>
<thead>
<tr>
<th>$P$ (time unit)</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE average</td>
<td>258.44</td>
<td>201.83</td>
<td>136.35</td>
<td>76.98</td>
<td>37.69</td>
<td>17.3</td>
<td>7.32</td>
</tr>
<tr>
<td>25-percentile</td>
<td>258.32</td>
<td>201.57</td>
<td>136.12</td>
<td>76.61</td>
<td>37.6</td>
<td>17.28</td>
<td>7.28</td>
</tr>
<tr>
<td>75-percentile</td>
<td>258.76</td>
<td>201.89</td>
<td>136.59</td>
<td>77.21</td>
<td>37.75</td>
<td>17.34</td>
<td>7.35</td>
</tr>
</tbody>
</table>

RMSE being smaller than an RMSE threshold $x$ over five consecutive time intervals, each of length 500 time units. Convergence is hence defined as a small enough deviation from the threat profile, and not simply as the convergence of the coverage profile to an arbitrary value. We use the Daly City map in Figure 2(c) for one sensor. Unless stated otherwise, $P = 4$ time units, $L = 4$ distance units, and $x = 0.00075$, which roughly corresponds to 2.5% mismatch between the threat and coverage profiles.

7.3.1 Impact of $P$ and $L$

We illustrate the impact of the $P$ and $L$ parameters on the convergence time. $P$ is varied to be $P = 1, 2, 4, 8, 16, 32, 64$ time units, and $L$ to be $L = 4, 6, 8$ units. We repeat each simulation 10 times, and report the averages and error bars showing the 25-th and 75-th percentiles. Figure 6 shows the results. Notice from Figure 6(a) that $L$ has little effect on the convergence time provided that the algorithm is able to converge to within the threshold. However, if $L$ is too large and $P$ is small, convergence for small RMSE thresholds may not be achieved, shown as the missing data points in the figure. This is consistent with the matching results earlier and demonstrates the need for the $L$ parameter. Figure 6(b) shows that the convergence time generally increases with $P$, since the sensor moves less quickly between the cells. However, if the algorithm runs long enough, then a larger $P$ will allow convergence to smaller RMSE thresholds, similar to the results in Figure 5.

7.3.2 Impact of RMSE threshold

Figure 7(a) shows the convergence time of WRW-aLP as a function of the RMSE threshold. The results show that it takes longer to converge to a smaller threshold, which is well expected. More interestingly, as the threshold exceeds 0.00075, the convergence time drops sharply, showing that converging to a reasonably small error (e.g., 2.5% deviation) is efficient but converging to an extremely small error may take significantly longer. This is due to the RMSE’s dependence on the ratio of the mismatch time to the coverage time so far, and hence even if the mismatch time is not too large, a large coverage time is still needed for the ratio to become very small.

7.3.3 Impact of map size

We evaluate the convergence time as the surveillance region increases in size and complexity. To do that, we
use different sub-maps of Daly City as the surveillance region in a set of experiments. Each sub-map is the part of the Daly City map consisting of $m \times m$ cells beginning at the lower right corner, where $m = 2, 3, \ldots, 8$. Figure 7(b) shows that the convergence time increases with $m$, i.e., a larger and more complex surveillance region leads to a longer convergence time.

### 7.3.4 Dynamic threat profiles

We show that WRW-aLP is able to adapt its coverage profile to a changing threat profile. To do so, we run WRW-aLP so that it converges to an original threat profile denoted as $\Phi'$. After the convergence, we change the threat profile to $\Phi'$ by randomly changing some of the threat levels in $\Phi$, and measure the time until the coverage profile converges to $\Phi'$. We report the cases when the percentage difference between $\Phi$ and $\Phi'$ are 1%, 5%, 6%, and 10%. Each simulation is repeated 20 times, and the averages and standard deviations are reported. Notice from Table 3 that there is no simple correlation between the % difference and convergence time. Figure 8 shows an example trace of the convergence first to $\Phi$, and then from $\Phi$ to $\Phi'$. The spike in the middle of the figure shows the increased RMSE after the threat profile change, but the algorithm is able to adapt to the new threat profile and bring down the RMSE afterwards.

### 7.4 Efficacy and Optimal Pause Time

We evaluate the method in 4.1 to compute the efficacy of single sensor coverage for Atlanta and Los Angeles. We use the utility functions, $U_M(\cdot)$ and $U_F(\cdot)$, shown in Fig. 9 for the RMSE (scaled by the population size) and unfairness, respectively. Because the cities are of different sizes, we use $t_m = 320$ and $t_f = 3500$ time units for Atlanta, and $t_m = 1100$ and $t_f = 3700$ time units for Los Angeles. Fig. 10 plots the efficacy measure achieved against the pause time parameter $P$, for $\alpha = 0.5, 0.7, 0.9$. Notice that in general, the efficacy increases initially as $P$ increases, because of improved matching. The efficacy reaches a single peak and then decreases afterwards, because a further increase in $P$ causes the unfairness to become too high.

We apply Brent’s method (see Section 5) to find the optimal $P$ that maximizes the efficacy. For each of the efficacy functions shown for each city, Brent’s method converges in less than 12 iterations. For LA and $\alpha = 0.7$, Fig. 11 shows the computed $P$ and the corresponding efficacy achieved, after each iteration of the algorithm. As shown in the figure, the first three iterations are used to bracket the optimal $P$, and the next seven iterations identify that optimal. Table 4 summarizes the optimal $P$ computed for the two cities, for $\alpha = 0.5, 0.7, 0.9$. They agree with the highest corresponding efficacy shown in Fig. 10. We conclude that Brent’s method can compute the optimal efficacy parameter accurately and efficiently.

### 7.5 Multiple sensors

This set of experiments illustrates the effects of multiple sensors. Figures 12(a) and 12(b) show the unfairness...
Fig. 10. Efficacy functions.

Fig. 11. Computed $P$ and corresponding Efficacy achieved after each iteration of Brent’s method for LA, $\alpha = 0.7$.

and RMSE of the WRW-aLP algorithm, respectively, for Atlanta. The number of sensors, $n$, is varied to be 2, 4, and 8. We compare the cases when the sensors operate independently (the case labeled “nc”) or when they have access to the global coverage profile (the case labeled “gk”), as defined in Section 6. Notice from Fig. 12(a) that for both nc and gk, the unfairness roughly halves each time we double the number of sensors, showing that increasing the sensing resources will reap roughly proportionate benefits, for up to 8 sensors and for a large city like Atlanta. However, Fig. 12(b) shows that in contrast to fairness, the steady-state matching does not improve as we use more sensors. This is because, for both nc and gk, the global coverage profile of all the sensors will, over the long term, approach the global coverage profile of each individual sensor. Hence, the additional sensors do not fundamentally benefit a long-term performance measure such as matching. Conversely, multiple sensors actually introduce the possibility of inefficiency when more than one sensor visit the same cell at the same time, which may hurt the matching. In the case of up to 8 sensors for Atlanta, the degree of redundant coverage is small. Hence, the RMSE increases slowly.

We further evaluate the impact of the pause time parameter on the RMSE and unfairness results for multiple sensors. The results for Atlanta are shown in Fig. 13, for 1, 2, and 4 sensors and both cases of nc and gk. The results show that the nature of tradeoff remains the same in the multiple sensor case as in the single sensor case.

We evaluate the coordination strategies presented in Section 6. We use four sensors for Atlanta. The coordination strategies are as presented in Section 6 and include: (1) independent operation (case “nc”), (2) knowledge of global coverage profile (case “gk”), (3) static partitioning by equal accessible area, and (4) static partitioning by equal total threat. For (3) and (4), we implement two different actual partitions that satisfy each of the equal area (cases “ea-1” and “ea-2”) and equal threat (cases “et-1” and “et-2”) objectives. The achieved unfairness and RMSE of the different coordination approaches are shown in Fig. 14. First, notice that for static partitioning, the performance can be dependent on the actual partition used. The largest difference, though still small, is for the unfairness between ea-1 and ea-2 (134 vs 137 time units). Second, independent operation has highly competitive performance against the coordinated approaches. In fact, it performs the best in all the cases except for fairness under ea-1. This shows that while independent operation can cause redundant coverage, the performance penalty is not larger than the loss of efficiency due to a non-optimal partitioning of the coverage areas, which restricts the ability of one sensor to help monitor a cell assigned to another sensor. Notice that while we have studied only basic coordination approaches, the almost best-case fairness gain with small

<table>
<thead>
<tr>
<th>City</th>
<th>Atlanta</th>
<th>Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Optimal $P$ (time unit)</td>
<td>10.00</td>
<td>6.50</td>
</tr>
<tr>
<td>Efficacy</td>
<td>0.70</td>
<td>0.68</td>
</tr>
<tr>
<td># iterations</td>
<td>12</td>
<td>9</td>
</tr>
</tbody>
</table>

TABLE 4
Optimal $P$ found by Brent’s method for Atlanta and LA, $\alpha = 0.5, 0.7, 0.9$. 

Fig. 12. Performance of WRW-aLP for varying number of sensors in Atlanta.
RMSE for independent operation suggests that, unless the map is small relative to the number of sensors, it will be difficult for any coordination approach to significantly outperform no coordination.

In summary, we conclude that (1) using more sensors can significantly improve the fairness of coverage (as quantified by a lower unfairness value), although the marginal improvement due to an additional sensor decreases slightly as the number of sensors increases; (2) using more sensors reduces the accuracy of matching slightly, under either independent operation or knowledge of the global coverage profile; and (3) for multiple sensors, basic coordination approaches – based on either knowledge of the global coverage or a static division of responsibilities – do not improve performance over independent operation. The almost best-case performance of \( n \) sensors without coordination shows that the stochastic movement enables the benefits of multiple sensors to be largely realized, without additional schedule planning/runtime coordination overheads. Moreover, Fig. 13 shows that if some of the sensors fail in a stochastic, uncoordinated deployment, the system of sensors will achieve a graceful degradation in performance without explicit recovery/replanning actions.

8 CONCLUSIONS

We have formulated the problem of covering a surveillance area by one or more mobile sensors based on a general threat profile. We proposed matching and fairness as basic though antagonistic performance measures of the problem. We showed how a complement of techniques can be combined orthogonally to give a WRW-aLP algorithm that can achieve excellent matching and good fairness at the same time. Moreover, a pause time parameter in WRW-aLP enables a controlled tradeoff between the fairness and matching, and the optimal parameter that maximizes the combined efficacy metric can be efficiently computed.

We showed that the achievable fairness can be limited by the availability of too few sensors for too large an area. In that case, the use of more sensors is effective. Our multiple sensor results, while preliminary, suggest that a simple deployment strategy of independently operating the sensors in stochastic movement, is viable, because it is largely effective while requiring no customized planning based on the number of sensors. Moreover, the independent operation approach degrades gracefully when a subset of the sensors fail, even without explicit recovery/replanning actions.

We have studied how our algorithm can adapt to dynamic threat profiles. However, we have assumed that the threat profiles are learned by an out-of-band mechanism (e.g., published population data) and they can only change slowly. In the case that the threat profile is learned by the sensing process itself, the Gaussian Process [36] is a useful tool of analysis. If changes to the threat profile can occur quickly, such as the effects of a sudden storm on plume propagation, the challenge is significantly harder and has not been solved in this work.

We are building a campus-scale sensor testbed based on the proposed mobile coverage algorithms. In the testbed, radiation/chemical sensors are carried by low cost robots that support wireless communication and programmable movement.

REFERENCES

Fig. 14. Performance of WRW-aLP with multiple sensors under different coordination approaches in Atlanta.


Chris Y. T. Ma is a Ph.D. student in the Department of Computer Science at Purdue University. He received his B.Eng. in Computer Engineering and M.Phil. in Computer Science and Engineering from the Chinese University of Hong Kong in 2004 and 2006, respectively. His research interests include performance study of wireless networks and mobile sensor networks.

David K. Y. Yau for a photograph and biography please see p. 287 of the February 2009 issue of this TRANSACTIONS.

Jren-Chit Chin for a photograph and biography please see p. 287 of the February 2009 issue of this TRANSACTIONS.

Nageswara S. V. Rao received his B.Tech from National Institute of Technology, Warangal, India in Electronics and Communications Engineering in 1982, M.E. in Computer Science and Automation from Indian Institute of Science, Bangalore, India in 1984, and PhD in Computer Science from Louisiana State University in 1988. He is currently a Corporate Fellow in Computer Science and Mathematics Division, Oak Ridge National Laboratory, where he joined in 1993. He has been on assignment at Missile Defense Agency as Technical Director of C2BM Knowledge Center since 2008. He was an Assistant Professor in Department of Computer Science in Old Dominion University during 1988-1993. He published more than 250 technical conference and journal papers in the areas of sensor networks, information fusion and high-performance networking. He is a Fellow of IEEE and received 2005 IEEE Technical Achievement Award for his contributions in information fusion area.

Malikarjun Shankar received his Ph.D. in Computer Science from UIUC in 2000, and is currently a Research Staff member in the Computational Science and Engineering Division of Oak Ridge National Laboratory (ORNL). He has participated in the Sensor-Cyber Networks (SCN) Research Team as the technical point of contact. His research interests are in the area of real-time and embedded systems and systems support for wireless sensor networks.