Abstract—Autonomous drones are employed with ever-increasing frequency in applications ranging from search and rescue, detection of forest fires, and battlefield/civilian surveillance. In this paper, we study the effects of limited mobility in such mobile sensor platforms, from the perspective of the effect limited mobility has on coverage effectiveness. We define a problem that we call Exploratory Coverage in Limited Mobility Sensor Networks, wherein the objective is to move a number of mobile sensors to fully explore (and hence, sense every point in) a target area in order to detect any critical event that has already occurred in the area. Further, we provide a taxonomy of problems within exploratory coverage as identified by the relationships between sensor range, coverage area, number of sensors, and mobility (range). We then design a purely localized and distributed approximation algorithm for our problem, and provide simulation results to demonstrate the effects of limited mobility on exploratory coverage.

Keywords—Mobile Sensors; Sensor Networks; Exploratory Coverage; Algorithms

I. INTRODUCTION

In this paper, we study the problem of Exploratory Coverage in Limited Mobility Sensor Networks. The notion of Exploratory Coverage is different from traditional notions of coverage such as Blanket, Sweep, or Barrier Coverage [15]. Blanket Coverage involves deploying sensors such that every point in the coverage field (the area to be searched) is covered by one or more sensors, as if waiting for future events to occur. Sweep Coverage has been described as a moving wave of sensors and is sometimes solved by periodically polling points-of-interest that are known a priori. And in the latter, Barrier Coverage, every crossing path between the boundaries of the coverage field must be covered by one or more sensors, as if sensors lie in wait for a mobile intruder to pass by. Mobility (and limited mobility) has been studied in these types of coverage problems as well. These various notions fall under the class of proactive coverage, wherein sensors are proactively deployed to sense events of interest. Exploratory Coverage focuses on sparse deployment models, wherein ensuring full blanket coverage of the field would require too many sensors to be deployed. Thus, we leverage mobility (albeit limited) in sensors to enhance quality of exploratory coverage with fewer sensors. In Exploratory Coverage, the strategy is to move a number of mobile sensors from an initial deployment to fully explore (and hence sense, or cover) every point in the coverage field at least once in order to detect any critical event that has already occurred in the area.

There are numerous factors driving increased attention to usage of automated drones. The human safety factors mentioned above have long been a primary motivator for interest in employing robots for a variety of tasks where humans would prefer not to be. Also, mobile sensor hardware platforms have seen many recently advances such as higher computation power, relatively lower weight, and lower power requirements that allow drones to carry much more computational capacity and payload or stay deployed and functional for longer periods of time. More and more, these platforms are more readily available, as the benefits of mass production of commercially designed systems are realized.

Research into advances for robotic platforms is occurring on many fronts [25]. UAV Platforms such as the DraganFlyer™X6 [24] as seen in Figure 1 show great promise as both an academic as well as practical model in real-world commercial applications. Less recently, numerous efforts have focused on building self powered miniature mobile robot sensor platforms as shown in Figure 2. The Robomote (70mm x 45mm x 35mm) [12], Khepera series of robots (70mm x 30mm) [18] [26], and the XYZ platform [23] are prototypes of mobile sensors that are battery powered and motor driven. In each of these platforms, sensors (to sense events) and communication devices create mobile communicating sensors. A rather novel approach to mobility in sensor platforms utilizes a fuel-powered hopping mechanism [9] used in Intelligent Mobile Land Mine Units (IMLM) developed by DARPA.

Our solution is intended to utilize limited mobility sensor platforms such as those mentioned above by conducting an
initial deployment, then employing an algorithm to direct the mobility of the sensors in order to locate events in the coverage field. We first review related work in this area, then formally define our problem including a taxonomy of the problem space and the subset to which we will devote our focus. Given the complexity of the problem, we analyze approximation algorithms, two centralized and one distributed. One of the centralized algorithms constrains the number of mobility choices that a sensor can make, where the distance traveled per choice is considered to be constant, and the other places the constraint on the distance traveled by each sensor. The distributed algorithm takes the constrained choices approach, but each sensor uses a localized heuristic to make mobility decisions. Finally, we analyze the performance of the distributed solution, fixing certain variables and varying others to understand the effects on coverage. Lastly, we present our conclusions about the effects of mobility on coverage.

II. RELATED WORK

Our problem has a number of close relatives in the class of Traveling Salesman Problems (TSP) [19], where an agent, given a graph representing cities at the vertices, and distances between cities as the edges, is tasked with visiting all the cities while traveling the shortest distance.

A variant known as the Prize-Collecting Traveling Salesman Problem (PC-TSP) [3], charges the agent a cost for traversing each edge, and pays the agent a reward upon arrival to each city, while loosening the requirement that the agent visit all cities. The goal is to maximize the overall reward. Another variation of PC-TSP that includes reducing rewards over time is known as the Discount-Reward TSP (DR-TSP) [6]. Modeling an agent-controlled sensor as a salesman, the coverage field as a graph, and assigning costs/rewards for touring the graph and visiting nodes, where rewards diminish over time and by how many sensors have already visited a node makes these techniques closely analogous.

In a similar fashion, we can say our problem is also closely related to $k$-TSP [14] and Vehicle Routing Problem (VRP) [7]. In $k$-TSP, $k$ agents can start at various cities and must solve for $k$ tours such that the total distance traveled between cities is minimized. A key difference between this problem and ours is that in $k$-TSP, no single agent has any bounds on its travel distance as long as the overall distance is minimized, where our agents each have limited mobility, or a per-sensor budget, if you will. Often $k$-TSP and VRP variations involve multiple agents originating at a common (or set of common) point(s). However, we believe that our problem shows promise for numerous real-world example applications where the initial distribution is uncontrolled. Limited mobility is analogous to TSP subtour elimination constraints in the $k$-TSP problem [4] and capacitation or time-window constraints in VRP [27].

One differentiating factor between the problems is the construction of the coverage field. While analogous, TSP and VRP problems typically apply to problem sets involving graphs representing roads, whereas our problem applies to a continuous coverage field (conceding that we approach the problem with a discretized implementation by dividing the area into tiles that can be considered a graph where each node is connected to nodes representing adjacent tiles). The characteristics of the graphs could be seen to vary only in terms of degree of connectivity.

Another close relative is the problem of ORIENTEERING [8]. In the Orienteering problem, the challenge is to find a walk between two nodes that maximizes the total number of nodes visited within a travel budget. In our problem, in contrast, the goal is to maximize coverage, but there is no destination (referred to as the unrooted variant of the problem), simply that as much of the coverage field is visited as possible before mobility limits are reached.

Blanket coverage has been approached using other methods, such as potential fields [16], where navigation is directed based on a model of repulsive forces and friction to encourage equilibrium. Another approach used in dense deployment configurations is to identify as many disjoint sets of sensors as possible that cover the area completely and activate them successively [28]. Other approaches, such as ant colony optimization technique, have also been employed [20].

Barrier and sweep coverage have been studied using a TSP approach [13] [21], as well as by other techniques, including a technique where the coverage field is decomposed into
subregions [17]. Other approaches, such as machine learning techniques, have also been employed [22].

III. PROBLEM DEFINITION

We formally define the problem of Exploratory Coverage in Limited Mobility Sensor Networks as follows. First, we are given a planar area defined as the coverage field \( A \). Next, we are given a set of mobile sensors \( K \) each with sensor range \( r \) and mobility range \( d \). For the distributed algorithm, we further assume sensor capabilities to include a communication range that provides awareness of neighboring sensors. Sensor range \( r \) specifies the radius of a disc within which a sensor can detect an event, and mobility range \( d \) is the total distance that a mobile sensor can travel. Mobility range sometimes referred to as overall distance and sometimes in number of units of distance traveled for each mobility choice, without loss of generality.

Further, we establish an initial deployment function \( f(K, A) \) that produces an initial deployment location for each sensor in \( K \). As we will see later, we focus on a function that uses a Gaussian distribution around a point for this function, such as might occur when sensors are dropped from an aircraft.

We define a navigation to be a set of transitions (using mobility) of a sensor from an initial deployment location to a final resting location, when either that sensor’s mobility range would be exceeded by another move, or the sensor has decided to stop navigating (say because \( A \) has been covered and additional mobility would only waste resources). From a given initial deployment configuration, we must produce a set of navigations (one each for sensors in a set) such that the execution of those navigations results in maximal coverage of \( A \). The number of all possible navigations from all possible starting locations to all ending locations can be very large. The subset that would actually result in complete coverage of \( A \) can also be large. On the other hand, many potential navigations that would result in complete coverage could be invisible if they require exceeding the mobility constraints of one or more sensors.

Next, we define a plan \( L \) to be a set of navigations. Consider the set of all plans to be \( C = \{L_1, L_2, \ldots\} \), and \( D \) is the subset of plans that result in complete coverage of \( A \) without exceeding mobility constraints for \( K \). We seek to map an initial deployment to a plan such that we maximize the area covered by the sensors while minimizing the total distance traveled by that set of sensors. In other words, we seek to select a set of mobility choices for sensors such that navigating according to this plan results in maximizing event detection in \( A \) while minimizing the number of moves.

The decision problem can be expressed as: given \( C, A, K, r, d, \) and \( f \), is there a plan \( L \in C \) that would result in covering all points in \( A \)? We note that there exist instances of the problem for which there are no such plans, and so we also discuss the problem of finding a plan that maximizes coverage of \( A \) while minimizing total distance traveled by sensors.

IV. TAXONOMY OF SUB-PROBLEMS

Coverage problem instances can be partitioned into a number of sub-problems, where each sub-problem is identified by the relationship between the number of sensors, the ratio of the sum of the potential coverage area of all sensors combined to the size of the coverage field, and the mobility constraints of the sensors.

Partition 1: There are problem instances for which there is no solution whereby the mobility of the sensors will allow for even 1-coverage to be achieved. In these cases, the best we can hope to achieve is a solution that improves coverage from the initial deployment to covering as much of the coverage field as possible given the available mobility of the sensors. In this case, we consider feasible solutions to be those that result in maximal coverage.

Partition 2: There are instances where a plan exists (there exists at least one \( L \in D \)) where all of \( A \) can be covered within mobility constraints.

Partition 3: There are instances where the number of sensors is high enough, and settings for \( r \) and \( d \) provide for the sensors to achieve \( n \)-coverage, where we are able to show that plans exist such that multiple sensors can pass within \( r \) of every point in \( A \) (or even achieve blanket coverage of \( A \)). Sometimes this is a beneficial problem space to explore, e.g., in cases where the coverage field need be covered for a time that can exceed the lifetime of the sensor. In this case, the goal is to provide redundant coverage so that as sensors fail, 1-coverage can be achieved multiple times (or preserved for a maximal period of time).

Partition 4: There are instances where the settings for \( r \) and \( d \) can provably be shown to allow all plans in \( C \) to feasibly cover \( A \), regardless of the initial deployment (\( D = C \)).

A visualization of this simple taxonomy of coverage problem instances is shown in Figure 3.

For this paper, we focus on partitions 1 and 2 which we categorize as Sparse Coverage problems. In these problems, plans may exist to allow coverage of \( A \) but are not guaranteed that such plans exist, but rather we are dependent upon the initial deployment and a thoughtful mobility strategy to achieve success.

In partition 3 where the number of agents or the sensor range of those agents becomes such that much of the coverage
field can be covered by multiple sensors simultaneously, then the problem shifts closer to the realm of other coverage problems such as Blanket coverage, and interesting solutions to explore focus on \( n \)-coverage rather than 1-coverage.

In Figure 3, a solution to partition 4 cases can be constructed in a straightforward manner. When the mobility budget is no longer an issue, then the problem becomes \( k \)-TSP. First, we pick a set \( V = \{K\} \) points in \( A \) where each point in \( A \) is within \( r \) of at least one of the points in \( V \). Each sensor is assigned a point from \( V \) and navigates from its origin to that point. Thus if not for time, mobility, nor other factors such as hazards and detection, we would simply loosen mobility constraints and wait until each sensor navigated to its assigned post. Since we are considering cases where the budget for mobility is a real constraint, however, more care must be taken in choosing the tour for each sensor to maximize coverage while minimizing distance traveled.

V. APPROXIMATION ALGORITHMS

Approximation algorithms have been developed for many variations of the problems discussed above. There is no known polynomial time approximation scheme (PTAS) for general TSP unless \( P = NP \), however the \((1 + \epsilon)\)-approximation for Euclidean-TSP in \( \mathbb{R}^2 \) and \( \epsilon > 0 \) is relevant to the discussion [2] and as well the 3/2-approximation algorithm for TSP where the triangle inequality holds [11] are well known. However, these algorithms are not practical for our problem as they rely on the location of events to be known a priori. Bounds on approximation schemes for PC-TSP are explored in [5]. For the problem of ORIENTEERING, a 2-approximation algorithm exists when the graph representing the coverage field is undirected [8].

Here we present three algorithms, two centralized and one distributed, for an approximate solution to the Exploratory Coverage in Limited Mobility Sensor Networks problem. For solutions involving constrained hops, we refer to a “hop” as a mobility choice that results in a sensor choosing to move a constant distance from its current location in a chosen direction to a new location where it waits to make another mobility choice.

A. Centralized Algorithm - Constrained Hops

The centralized algorithm assumes a centralized processor evaluates the initial deployment and selects a plan for all sensors. First, we choose a target configurations from \( C \) such that the points in \( L_x \) form a grid of squares of width/height \( 2^{-1/2}r \). This size is chosen so that any mobile sensor located within this square can sense any point within and thereby covers that square.

From this set of squares, we construct a graph \( G = (V, E) \) where \( V = v_0, v_1, ... \) is a set of vertices created for each corner of a square, and \( E \) is a set of edges created by connecting each vertex in \( v_i \in V \) to adjacent vertices to the left, right, above, and below \( v_i \). Each edge cost is thus \( 2^{-1/2}r \).

In this way, we have constructed a graph where sensors may navigate to cover any portion of the graph by way of hopping from one vertex to another, traveling a constant distance with each hop, and completing their route when they cannot make another hop without exceeding its mobility budget. Thus, we have translated the problem into one where the number of hops by each sensor is constrained to \( d/2^{-1/2}r \) hops.

B. Centralized Algorithm - Constrained Distance

For the constrained distance problem, we can first solve the constrained hops problem and then solve ORIENTEERING to go from the constrained hops solution to the constrained distance solution.

The ORIENTEERING problem is the following: given a set of nodes and distances between each pair of nodes, and a budget \( K \), find a tour with maximum number of hops such that the total distance covered is at most \( K \).

With the 2-approximation algorithm for undirected graphs in hand, and a \( O(\log^2 n) \)-approximation for directed graphs, we can construct a possible solution for the constrained distance problem:

1) Given a constrained distance problem instance \( X_d \), let \( OPT_d \) be value of the optimal solution
2) Let \( X_h \) be the corresponding constrained hop problem and \( OPT_h \) the optimal solution of \( X_h \)
3) Solve \( X_h \) optimally using the MAX FLOW formalization
4) For each sensor \( i \), let \( H(i) \) be the (optimal) set of hops found by the MAX FLOW solution
5) For each \( i \), define an ORIENTEERING problem as follows: \( ORIENT_i = \) Given sensor node \( i \)'s original position and distances to hops in \( H(i) \), find a graph with the vertices and distances equal to the shortest path distances. Now this is an instance of the ORIENTEERING problem where the goal is to cover as many vertices as possible using at most distance \( h \)
6) For each \( i \), solve \( ORIENT_i \) using the approximation algorithm [8]. For each \( ORIENT_i \), the solution \( A_i \) is a path (of length at most \( h \)) over \( \{i\} \cup H(i) \). Let \( OPT_i \) be the optimal solution for \( ORIENT_i \)
7) Output \( A_1, A_2, ..., A_n \)

To argue that this is indeed a good approximation, we need the following:

\[
OPT_d \leq OPT_h \leq \sum_i OPT_i \leq \sum_i 2 \ast A_i \leq 2 \ast valueofthesolution
\]

All these inequalities hold, except the last one. The reason is that the \( A_i \) instances could overlap (in terms of tiles they cover), so the value of the solution could be significantly less than \( \sum_i A_i \).

C. Distributed Algorithm

We can use a MAX FLOW-like solution for the distributed algorithm also. This would be considered distributed MAX
Flow and examples of such a solution can be found in [1]. We can also use heuristic-based distributed algorithms with synchronization. A number of reinforcement-learning techniques such as EA or learning classifiers, and even ant colony optimization approaches would also work.

Another approach to solving the distributed approximation scheme is to employ a heuristic and a form of matching to direct the navigation of the mobile sensors:

1) Given a constrained hops problem instance $X_d$, let $S_0$ be the initial state having deployed the mobile sensors
2) For each sensor $s_i$, use a heuristic to map sensors within $s_i$’s tile to moves that are predicted to produce the highest reward
3) Employ a protocol for $s_i$ and other sensors that occupy the same tile to decide which sensor will perform which move
4) All sensors make a hop
5) Repeat until hop distance would exceed mobility constraint

Next, we discuss the heuristic that the sensors will use to direct their decision making process. The heuristic produces tile colorings defined as:

- White = the tile appears to never have been occupied, and no sensor is in a position where it could move there next turn
- Gray = the tile appears to never have been occupied, and one or more sensors in adjacent tiles could occupy it next turn (we make a tally)
- Black, unoccupied = we remember the tile has been visited, but isn’t occupied now
- Black, occupied = it has been visited, and in fact is occupied now (we make a tally)

Additionally, each sensor is mobility-limited and has a remaining number of hops counter.

Once all tiles within each sensor’s range is examined and colored, we decide moves based on a simple algorithm.

The basic pseudocode for the distributed solution is shown in Figure 4, which can be stated in two ways: first, how it is implemented, and second, from the perspective of an agent working in distributed fashion.

How it is implemented: A turn (or tick) is defined as an opportunity for each agent to move, if it is able to and decides to move.

How it works from the sensor’s point-of-view: The main difference is that we simply state that agents employ a protocol whereby they communicate in order to decide who should move first in a neighborhood, and that once that sensor makes a move, it notifies the others and now they recalculate what they know and decide again amongst themselves whose turn it is. The logic is based around the premise that the fewer choices a sensor has, the earlier it moves.

VI. Performance Evaluations

For a set of sensors to perform optimally, the sensors deployed to a coverage field would navigate the least distance to cover maximal space in the minimum time possible. Because sensor deployments can result in concentrations of sensors to the same area, and with mobility being limited, it could require more sensors or more hops per sensor could be required in order to assure the desired expected level of coverage.

As a side note, in selecting a communication protocol between sensors, the number of messages transmitted is also a factor we desire to minimize, as sending messages costs energy and increases risk of detection (as well as betraying the sensors’ location to enemies). Also, although sensor technology options are increasing [10], real-world sensors typically are subject to processing and memory limitations, and so minimizing the utilization of these resources is also important to consider.

In order to gather data to support our hypotheses, we developed a simulation program written in C#. Input parameters to the program include number of trials to run, number of turns to play for each trial, the width and height of the coverage field (as measured in tiles), the count of sensors, the number of times that a sensor is allowed to hop (its mobility constraint), the agent type which designates which heuristic to use, the deployment method, and a value for $\sigma$ that is used when the deployment type is Gaussian.

We implemented several heuristics for comparison purposes. The first had each sensor stochastically choose a neighboring tile to navigate to. As expected, this performed poorly with respect to mobility conservation, given that each sensor always chose to navigate to an adjacent tile. The second heuristic also allowed sensors to navigate to a random adjacent tile, but only if that tile was not occupied by another sensor. This also performed poorer than desired with respect to conserving mobility, but did rather well with respect to coverage. We used these heuristics to judge the feasibility of our WGB heuristic, constructed according to the distributed algorithm described previously.

The program begins by assigning each sensor to a tile (its initial deployment). We experimented with initial deployment methods, including stochastic and statistical distributions. Results from stochastically assigning sensors to tiles did not produce interesting results, due to the fact that regardless whether we varied the size of the coverage field, or the number of sensors, the only parameter that was changing significantly was the density of the sensor deployment. We experimented with Gaussian distribution of sensors around a point and Gaussian distribution along a line, with satisfactory results. However, the results were similar enough between the two types that we chose Gaussian around a point for the results presented here. Thus, given a point and a specified value for $\sigma$, sensors are more densely located nearer the point, and more sparsely as distance increases from the point.

The tiles are constructed as a grid with no edges, where sensors that travel North from the top wrap around to the South side, and vice-versa. Likewise, sensors that travel West from the western edge wrap around to the East, and vice-versa.

The simulation begins, and for each trial, agents select
Fig. 4. Distributed Solution to WGB \( \dagger \) around a point.

Initial coverage at deployment time using Gaussian distribution for a fixed number of 100 sensors. Note: Hops0 indicates the size of the coverage field and number of hops per sensor, using the White-Gray-Black (WGB) algorithm, varying the average level of coverage resulting from numerous simulations complete coverage. The data presented in Figure 5 shows the more sensors or more hops per sensor in order to assure

A. Coverage for various sized coverage areas

Because sensor deployments can result in concentrations of sensors, and with mobility being limited, it could require more sensors or more hops per sensor in order to assure complete coverage. The data presented in Figure 5 shows the average level of coverage resulting from numerous simulations using the White-Gray-Black (WGB) algorithm, varying the size of the coverage field and number of hops per sensor, for a fixed number of 100 sensors. Note: Hops0 indicates the initial coverage at deployment time using Gaussian distribution around a point.

B. Coverage for various number of sensors

As shown in Figure 6, as we increase the hops per sensor, the resulting coverage increases. With a 20x20 grid (400 tiles) and 16 hops, these 100 sensors were nearly able to visit all tiles within the entire region. Twenty-five evenly distributed sensors making 5 hops each could have toured the entire coverage field, but due to the Gaussian concentration of sensors around a point, some sensors had to travel a considerable distance to visit outlying tiles.

C. Coverage when the deployment concentration (\( \sigma \)) varies

Our deployment implementation uses Gaussian sets of coordinate values that produces numbers with a mean of 0,
and a standard deviation of 1. That is normalized, scaled using value $\frac{\text{Size}}{\sigma}$, where $\text{Size}$ is the width/height of tiles in the grid. The higher the value for $\sigma$, the less concentrated the initial deployment. For this experiment, we looked to see what happened when we vary the value of $\sigma$ and analyze the resulting coverage performance. We fix the grid size to 50x50. Results are presented in Figure 9.

Notice that in Figure 10, the best initial deployment results in a value of $\sigma = 2$, and that by increasing $\sigma$, we observe a drastic reduction in the coverage performance after the second hop. In fact, even with $\sigma = 2$, it takes 4 hops per sensor to achieve 90% coverage performance.

D. Effects of mobility on coverage performance

Using the data from the experiments above, we would now like to examine the effects of mobility on coverage. We will do this using a metric $e = \frac{H}{\%}$ where $H$ is the number of hops taken by sensors, and $e$ is the hops incurred per percentage increase in coverage. In other words, how much mobility is expended to achieve additional levels of coverage in a variety of scenarios. Figure 11 shows the data for this metric when varying the size of the coverage area.

What we conclude is that the effects of mobility on coverage are greater in more sparse deployments than in concentrated deployments. The results illustrated in Figure 12 are intuitive considering that as sensors become more concentrated, the overlapping coverage increases. Increasing mobility adds to the quality of coverage, but with a diminishing return on investment. Results were consistent for scenarios where the number of sensors and the values for $\sigma$ were varied, which was likewise expected.

VII. CONCLUSION

Practical algorithms for employing networks of unmanned, autonomous mobile devices with limited mobility, to sparse regions where it is crucial that events are located, identified, and communicated is becoming an ever more commonplace facet of our world. As sensor technology improves and becomes more mobile, computing power expands while requiring less and less energy, and production of drones and robotic platforms become more robust, algorithms for directing the behavior of these devices will only become more of a focus.

We have explored a variety of problems related to coverage of a coverage field. Within this we have focused on the problem of Exploratory Coverage in Limited Mobility Sensor Networks with sensors that have limited mobility capabilities. We believe the taxonomy of the various subproblems within Exploratory Coverage should prove helpful for further study to allow focus on certain facets of the problem.

A range of prior work exists and has provided a wealth of conclusions from which to inform our work in the area of the different approaches to the problem of Exploratory Coverage, and we have shown several algorithms for finding approximate solutions.
We then design a purely localized and distributed approximation algorithm, and provide simulation results conducted to demonstrate the effects of limited mobility that show that the effects of mobility constraints on is to vary the level of coverage achieved more in sparse deployments than when sensors are deployed in a more concentrated fashion.

**REFERENCES**


**ACKNOWLEDGMENTS**

This work was supported in part by National Science Foundation (NSF) under Grant No. 1254117 and 1205695. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.