# The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment

Maxim A. Batalin and Gaurav S. Sukhatme Robotic Embedded Systems Laboratory Center for Robotics and Embedded Systems Computer Science Department University of Southern California Los Angeles, CA 90089, USA maxim@robotics.usc.edu, gaurav@usc.edu

*Abstract*—In this paper we present the design and theoretical analysis of a novel algorithm (LRV) that efficiently solves the problems of coverage, exploration and sensor network deployment at the same time. The basic premise behind the algorithm is that the robot carries network nodes as a payload, and in the process of moving around, emplaces the nodes into the environment based on certain local criteria. In turn, the nodes emit navigation directions for the robot as it goes by. Nodes recommend directions least recently visited by the robot, hence the name LRV. We formally establish the following two properties: 1. LRV is complete on graphs, and 2. LRV is optimal on trees. We present some experimental conjectures for LRV on regular square lattice graphs and compare its performance empirically to other graph exploration algorithms.

Index Terms-Coverage, exploration, mobile robot, graph.

## I. INTRODUCTION

This paper addresses the design and analysis of an algorithm which simultaneously solves the robot coverage and exploration problems, as well as the sensor network deployment problem. These problems are complementary in that each is difficult to solve on its own. Our key insight is that an efficient algorithm exists which can solve them together.

*Coverage:* The coverage problem [3] is the maximization of the total area covered by a robot's sensors. *Static* coverage is the problem of deploying robot(s) in a static configuration, such that every point in the environment is under the robots' sensor shadow (i.e. covered) at every instant of time. Clearly, for complete static coverage of an environment the robot group should be larger than a critical size (depending on environment size, complexity, and robot sensor ranges). Determining the critical number is difficult or impossible if the environment is unknown *a priori*. *Dynamic* coverage, on the other hand, is addressed by algorithms which explore and hence 'cover' the environment with constant motion and neither settle to a particular configuration, nor necessarily to a particular pattern of traversal.

*Exploration:* The exploration problem [9] is the discovery of all traversable regions of the environment by the robot. A robot is said to have explored the environment completely, if at some time during its motion, no location

of the environment remains which has not been under the robot's sensor shadow at some prior time instant. Exploration may be thus viewed as the initial (startup or transient) phase of dynamic coverage.

Sensor Network Deployment: The network deployment problem [17], [6] is the emplacing of individual network 'nodes', each with limited communication range, into the environment by a process that guarantees certain communication properties of the resulting node ensemble. The simplest of these properties is that the emplaced nodes form a single network, in other words they form a graph with a single connected component where the nodes are the graph vertices, and an edge between vertices signifies that the corresponding nodes are within communication range.

Our domain of interest in this paper is a planar bounded environment, large enough to prevent complete static coverage by sensors on one robot. The robot must thus move in order to observe all points in the environment frequently. In other words we address the dynamic coverage problem with a single robot. Further, we assume that the environment is initially unknown and unexplored. Thus the robot's first complete tour of the environment is a solution to the exploration problem. Last, we assume the existence of a number of nodes, each capable of short range communication and local processing, and able to act as a constituent of an adhoc network. Our interest is in emplacing these nodes into the environment such that they form a single network.

We present the Least Recently Visited (LRV) algorithm, which is an efficient method to simultaneously address these three problems. The basic premise behind the algorithm is simple. The robot carries the network nodes as a payload, and in the process of moving around, emplaces the nodes into the environment based on certain local criteria. In turn, the nodes self-organize to form a network, and emit navigation directions for the robot as it goes by. The process by which nodes compute navigation directions depends on local frequency counts of which directions the robot has recently pursued. Informally, nodes recommend directions least recently visited by the robot, hence the name LRV. LRV is simple, decentralized and robust. It is a local algorithm and does not rely on a map or GPS and does not perform explicit robot localization. Moreover, the deployed network can be used for tasks other than coverage

and exploration: robot navigation [1] and Multi-Robot Task Allocation [4], [2] are two examples.

In past work [17], we have discussed an LRV-like algorithm informally and focused on its experimental characterization. The key contribution of this paper is the theoretical formulation of LRV and an analysis of its properties. For purposes of analysis we treat the deployed nodes as the vertices of a graph even though no explicit adjacency lists are maintained at each node. The graph is thus purely an aid to our analysis of the coverage and exploration algorithm, not an entity used by the algorithm itself. In particular we establish the following two properties:

- 1) LRV is complete on graphs (eventually terminates)
- 2) LRV is optimal on special graphs (i.e. trees)

Further, we present some experimental conjectures for LRV on regular square lattice graphs.

## II. RELATED WORK

The *static* coverage problem is addressed by algorithms [5], [6], [7] which are designed to deploy robot(s) in a static configuration, such that every point in the environment is under the robots' sensor shadow (i.e. covered) at every instant of time. *Dynamic* coverage, on the other hand, is a more general type of coverage and addressed by algorithms which explore the environment with constant motion and neither settle to a particular configuration [8], nor necessarily to a particular pattern of traversal.

Exploration, a problem closely related to coverage, has been extensively studied [9], [10]. The frontier-based approach [9] concerns itself with incrementally constructing a global occupancy map of the environment. The map is analyzed to locate the 'frontiers' between the free and unknown space. Exploration proceeds in the direction of the closest 'frontier'. The multi-robot version of the same problem was addressed in [11].

By contrast, LRV does not use a map, nor localization in a shared frame of reference. It is based on the deployment of static, communication-enabled, sensor nodes into the environment by the robot.

There is prior work on robot exploration based on the deployment of passive nodes (read-only devices), particularly using graph models [12], [13]. In both cases the authors studied the problem of dynamic single robot coverage on a graph world. The key result was that the ability to tag a limited number of vertices (in some cases only one vertex) with unique passive nodes dramatically improved the cover time. We note that [12], [13] consider the coverage problem, but in the process also create a topological map of the graph being explored. They also show that in certain environments exploration is impossible without tagging. There are four key differences between LRV and the work reported in [12], [13]:

 We do not assume the robot can navigate from one node to another in any reliable fashion. The robot does not localize itself, nor have a map of the environment (the structure of the graph corresponding to Algorithm 1 Least Recently Visited (LRV) Algorithm

#### **Robot Loop:**

if no sensor node within communication range then Deploy sensor node

#### else

Move in direction d suggested by nearest sensor node i

Notify node j of arrival in its vicinity

# Sensor Node Loop:

Emit	least	recently	visited	direction:					
$ANY\_OF(argmin_{\forall d \in D(i)}W(d))$									
Update	sensor no	de weight if n	ecessary: W	V := W + 1					

the environment is not known to the robot, nor does it construct it on the fly).

- 2) We assume the number of sensor nodes available for drop-off is unlimited; in [12], [13] a limited number of nodes is used.
- 3) We assume that each node being dropped off is capable of simple computation and communication the nodes are active; in [12], [13], the nodes are passive they neither compute nor communicate.
- 4) We do not assume that nodes need to be retrieved; in [12], [13] retrieval and reuse of nodes by the robot is implied.

Our work is closely related to the ant robots literature [14], [15], [16] where the idea of a node with decaying intensity (a semi-active node) is used. The robots sense the change in intensity and are able to change the direction of exploration to cover the environment efficiently. Our algorithm differs from the these approaches - we assume that each deployed node is capable of sensing, simple computation and communication. We exploit the computation and communication capabilities of the nodes to address problems beyond coverage and exploration.

## III. THE LRV ALGORITHM

As shown in Algorithm 1, LRV is the concurrent execution of two algorithms - one on the robot (Robot Loop) and another on every node (Sensor Node Loop). For every node i, let D(i) be the set of directions along which the robot can move away from i. Then  $\forall d \in D(i), W(i, d)$ is the weight (cached on node i) which stores the number of times direction d was actually traversed when the robot moved away from node i. In some cases, we will refer to the weight of direction d as W(d) if the node identifier is implicit. The function  $ANY\_OF(T)$  returns a single element of a set T according to some rule (i.e. in order, random, etc).

When a robot is placed into the environment initially, according to Algorithm 1, it deploys a node because there is no node within communication range. Over time the algorithm causes a network of nodes to be deployed since every time a new node is deployed it must be able to communicate with at least one other sensor node in the

network. Once deployed, each sensor node starts to emit the locally least recently visited direction (hence the name LRV), which is one of the directions with smallest weight W (if there are multiple directions of the same weight, one is arbitrary picked). In practice the number of directions per node is often bounded and application dependent. In our experimental work we set this bound to 4. Each node locally associates a weight with each direction of travel away from it. The weight is incremented in two cases: right before a direction is traversed and on the destination node right after a direction is traversed. Suppose the robot is in the vicinity of a node i and is directed by node i to move in direction d. The weight W(i, d) is incremented right before direction d is traversed. Suppose now that the robot enters the vicinity of node j through direction d. The weight W(j, d) is incremented right after direction d is traversed.

Several practical issues arise when the implementation of LRV is considered. These include the detection by the robot of which node vicinity it is in, and the problem of the robot actually following the directional suggestions it is given while avoiding obstacles, etc. For a detailed discussion of such practical issues see [17], [1].

## IV. THE GRAPH MODEL

For purpose of analysis, consider an open bounded environment with no obstacles. In this case, given our node deployment algorithm (LRV) described in the previous section, we can model the steady state spatial configuration of the nodes as a finite graph G = (V, E), where V is a set of vertices (the deployed nodes) and E is a set of edges such that  $\forall i, j \in V$  there is en edge between i and j iff 1. i and j are within communication range; 2. there is a physical path between i and j. Consider the schematic of the environment in Figure 1a. We represent the LRV-deployed network in this environment as a graph G = (V, E) (shown in Figure 1c). A graph model is a natural choice because of its flexibility and ubiquity of usage in such problems.

Before discussing the theoretical properties of LRV we provide working definitions for coverage and exploration on graphs and corresponding performance metrics.

**Definition** (Coverage on a graph) 3.1: Coverage on a graph is the act of visiting every vertex of a graph.

The performance of a coverage algorithm is measured using the *cover time* [18] metric defined as follows:

**Definition (Cover time) 3.2:** Cover time is measured in terms of the number of edges traversed such that every vertex of a graph is visited at least once, i.e. the graph is covered.

Note that in order to cover a graph, a robot needs to at least traverse one edge per node (consider a spanning tree of a graph). This notion is distinct from graph exploration or 'complete' graph coverage (where the robot needs to

Algorithm	<b>2</b> Least	Recently	Visited	(LRV)	Algorithm	on
Graph.						

if Covered/Explored the graph then
Exit
else
$n' = ANY\_OF(argmin_{\forall j \in E(n)}W(n, j))$
W(n,n') := W(n,n') + 1
n := n'

traverse every edge of a graph). This later notion is called graph exploration, defined as follows:

**Definition (Exploration on a graph) 3.3:** *Exploration on a graph is the act of traversing every edge of the graph.* 

An exploration algorithm is evaluated using the *Exploration time* metric defined as follows:

**Definition (Exploration time) 2.4:** *Exploration time is measured in terms of the number of edges traversed such that every edge is traversed at least once.* 

It follows from the above definitions that exploration is a superset of coverage. Therefore,  $Cover\_Time = O(Exploration\_Time)$ .

# V. RESULTS: COMPLETENESS AND ASYMPTOTIC COVER TIMES

Given the graph model we formally exhibit two important properties of LRV. First, we show that LRV is complete, and second we establish a relationship between its cover time and exploration time. For purposes of this analysis we are interested in the behavior of LRV in the 'steady-state' when all nodes have been deployed. In this special case one can consider a simple version of LRV on a graph as follows. For every vertex i, E(i) is the set of edges incident to *i*. For clarity we identify an edge in E(i) with the node this edge connects node i to. Then  $\forall e \in E(i) : W(i, e)$  is the weight (cached on node i) maintaining the number of times edge (i, e) was traversed from *i*. In some cases, we will refer to the weight of edge e as W(e) if the originating node is implicit. Note that in LRV the weight of an edge is incremented twice: before and after traversal, but on different nodes. This redundancy is required for practical purposes: the weights are cached on nodes and since the environment is dynamic, sensing and actuation are noisy, starting at the same node and traversing the same direction at different points in time does not guarantee that robot would arrive at the same node. In the graph model we study the steady state spatial configuration of the nodes on a finite unchanging graph. Hence, for clarity of presentation,  $\forall i, j \in V$  the weight associated with  $i \rightarrow j$  transition is stored on the edge  $e_{i,j} \in E$ . This weight is identical to the one associated with  $j \rightarrow i$  transition (e.g. W(i, j) = W(j, i). We increment the weight just in one case: right before an edge is traversed, and associate it with the edge  $e_{i,j} \in E$ .



(a) Initial environment

(b) Deployed sensor network and a mobile robot

of a)

Fig. 1. Modelling the network as a graph.



Fig. 2. Illustration for Theorem 1.

Algorithm 2 shows this simplified LRV on a graph. Note that the deployment function is removed since we are in the steady state.

# A. Completeness of LRV on Finite Graphs

Theorem 1 (Completeness): The exploration time of LRV on a finite graph is finite.

Proof: The goal is to show that LRV traverses every edge of any finite graph in finite time. The proof is by contradiction. Suppose the exploration time of LRV is infinite. Therefore, there is a time t after which LRV traverses only those edges that it traverses infinitely many times (edges of the graph  $G_{exp}$  in Figure 2). By definition, the weights of these edges grow without bound, including the edge that is considered for traversal infinitely many times but is never picked after time t (edge  $e_{ij}$  in Figure 2). By definition, LRV will be forced to traverse this edge after time t, which is a contradiction.

Note that the completeness result of Theorem 1 can be applied to a wider set of real-time search algorithms [19].

**Theorem 2:** For a graph G=(V,E) with maximum degree d, if Cover time = O(f(V)), then Exploration time = d \* O(f(V)).

*Proof:* Suppose LRV executes on a graph G until every vertex is visited at least once. It is obvious that at least one edge per vertex is traversed. Thus, after the first execution of the algorithm, the number of untraversed edges at every vertex is at most d-1. Note, that at a given vertex, while there are untraversed edges, LRV will choose one arbitrarily. Hence, after at most d executions of the algorithm every vertex would be covered and every edge would be traversed. Thus, if  $Cover\_Time = O(f(V))$ , then Exploration\_Time = d \* O(f(V)).

# B. LRV on a Square Lattice: Empirical Results from Simulation

In this section we consider the performance of LRV on the following special graph G:

- 1) G is undirected.
- 2) G has degree  $deg_G \leq 4$ . If all nodes have degree 4, then G is a square lattice i.e. a regular graph of degree 4.
- 3)  $|V| = \Theta(|E|).$

We consider this special graph because in practical implementations of LRV, a physical compass on the sensor node determines direction. If this compass has k bits of resolution, then each node is capable of identifying  $2^k$ directions resulting in a graph of degree  $\leq 2^k$ . In previous work [17] we performed experiments with k = 2, resulting in a square lattice-like deployments. Hence we analyze LRV on a square lattice. It has been shown [18] that the cover time of a random walk (RW) on a regular graph with V vertices is bounded below by  $V \ln V$  and above by  $2V^2$ . If we assume that passive nodes can be used, and the graph G = (V, E) is known (a topological map is available) and the robot can drop nodes of three independent colors, then the problem of coverage can be solved optimally by applying Depth-First Search (DFS) which is linear in V. DFS assumes that all resources are available - nodes, map, localization and perfect navigation.

In [12] the problem of coverage is considered in the context of mapping a graph-like environment with V vertices. Their algorithm explores the environment and constructs a topological map on the fly. The assumptions of the algorithm are that the robot has k(k < V) nodes, and perfect localization and navigation within the graph. The *cover time* of their algorithm is bounded by  $O(V^2)$ . It is important to note that the problem addressed in [12]





(a) Comparison of *Cover Time*  $n \ln n$  curve, DFS, RW and LRV

(b) A comparison between DFS and LRV. This graph is a magnified view of (a)



(c) Comparison of *Cover time* DFS, 1-LRTA\* and LRV.

Fig. 3. Comparison of graph coverage algorithms. The  $n \ln n$  curve is shown for reference.

is more complex than simple coverage, since they build a map while exploring.

We have conducted simulation experiments running RW, DFS and LRV on graphs with  $V \in [100^2, 1000^2]$  nodes. For every experiment the steady state *recurrent* cover time is reported. In other words, LRV performed multiple coverages of the graph until the average cover time reached a steady state and the weights of edges were not reset after a particular graph coverage had completed. The results of the experiments are shown in Figure 3. The figure also shows the  $n \log n$  curve for reference. These experiments lead us to

**Conjecture 1:** The asymptotic cover time of LRV is  $O(V^{1+\epsilon})$ .

From Figure 3 it is clear that cover time of LRV is less than  $V \ln V$ , however the bound is not tight. We suggest the following technique of determining an  $\epsilon$  for graphs with given number of nodes. Assuming that the function representing LRV cover time is monotonic we can analyze the sequence  $\frac{LRV(i)}{f(i)}$ . If this sequence is increasing then asymptotically  $LRV(i) = \Omega(f(i))$ , if it is decreasing then asymptotically LRV(i) = O(f(i)), and if it is constant then asymptotically  $LRV(i) = \Theta(f(i))$ . The following is the sequence for  $i \in [100^2, 1500^2]$  nodes and  $f(i) = i^{1.0005}, \frac{LRV(i)}{i^{1.0005}}$ :

[3.9406 3.9587 3.9639 3.9661 3.9672 3.9678 3.9682 3.9684 3.9685 3.9685 ... 3.9682]

Note that the sequence increases initially, then stabilizes at value 3.9685 for  $i \in [900^2, 1000^2]$  and finally decreases. Hence, the value of  $\epsilon$  for graphs with the number of nodes  $\leq 1500^2$  is  $\epsilon \leq 5 \times 10^{-4}$  and the asymptotic cover time of LRV is  $O(V^{1.0005})$ .

Using Theorem 2 and Conjecture 1, we get the following result for a square lattice.

Algorithm 3 1-LRTA* Algorithm on a graph.	
if Covered/Explored the graph then	
Exit	
else	
$n' = ANY\_OF(argmin_{\forall j \in E(n)}W(j))$	
W(n) := W(n') + 1	
n := n'	

**Corollary 1:** LRV explores the environment in asymptotic time  $O(V^{1+\epsilon})$ .

## C. LRV Tradeoffs

As mentioned earlier, the clear performance boundaries for the coverage task are given by RW (upper) and DFS (lower). The more interesting comparisons are between LRV and DFS and our algorithm and an algorithm with a limited number of passive node markers [12].

Figure 3b shows that the asymptotic performance of our algorithm is similar to DFS. Note that in order to determine the identity of neighboring vertices and to navigate perfectly from node to node, DFS assumes that a map of the environment is available and that the robot is perfectly localized. Our algorithm, on the other hand, does not have access to global information and the robot does not localize itself. The nodes used in our algorithm are more complicated than those used in DFS, and the cover times are asymptotically somewhat larger than the cover times of DFS.

In [12] the algorithm builds a topological map of the environment and assumes perfect navigation (and thus, localization) on the graph. The node *markers* are very simple (the only function is to mark the vertex) and the robot cannot differentiate between them. In addition the algorithm assumes that there exists a local enumeration of edges. The cover time of this algorithm, however, is bounded by  $O(V^2)$ . LRV, on the other hand, does not have a map and the robot does not localize itself. Another



Fig. 4. a) A map of the environment with an embedded sensor network with a tree-like topology. b) Tree isomorphic to embedded sensor network topology of a).

important difference is that we assume that the number of nodes available to us is equal to the number of vertices. In addition, the nodes used in our algorithm are more complex, since they keep a certain state per direction, and are uniquely identifiable. The cover time of our algorithm, however, is conjectured to be less than  $V \log V$ . Thus, the apparent trade off is using a large number of "smart" nodes (and no global information or localization) vs. a limited number of simple nodes (with mapping and partial localization within the graph). The cover time achieved by our algorithm is clearly better. However, if the nodes are a precious resource, the algorithm described in [12] would be preferred.

Another algorithm to which we compare LRV is 1-LRTA\* [20]. 1-LRTA\* is a well known graph search algorithm that can be applied to graph coverage. Algorithm 3 shows the details of 1-LRTA\*. In 1-LRTA\*, a weight is associated with a node. The edge to traverse is chosen on weights of neighboring nodes. The weight of a node is incremented with the weight of a node the robot transitions to. Hence, 1-LRTA\* requires nodes to communicate.

Figure 3c shows that generally 1-LRTA\* outperforms LRV. However, it should be noted that in practice [17] LRV is a deployment and exploration algorithm, whereas 1-LRTA\* is a graph exploration algorithm which assumes the graph is given.

#### D. LRV on Trees is Asymptotically Optimal

We now study the performance of LRV on trees. Figure 4a shows a map of the environment with an embedded sensor network. The sensor network has a tree-like topology. Figure 4b shows the tree which represents the



Fig. 5. Illustration for Lemma 1.

embedding. A tree differs from square lattice in two major ways: 1. The vertex degree is not bounded by 4; 2. A tree does not contain cycles. The next two Lemmas establish local properties of LRV needed for the main result of this section: performance of LRV on trees is linear or asymptotically optimal.

**Lemma 1:** An incoming edge  $e_{ij}$  is traversed twice iff every other edge incident to  $v_i$  is traversed twice.

Proof: Initially the weights of all edges are zero. Suppose a robot enters vertex  $v_i$  through an incoming edge  $e_{ij}$  (refer to Figure 5). The weight of  $e_{ij}$  is incremented and equal 1, whereas the weights of other edges incident to  $v_i$  are 0.

Next, LRV picks one of the 0-weighted edges, say  $e_{ik}$ , and traverses it. The weight of  $e_{ik}$  is incremented and equal 1, the weight of  $e_{ij}$  is equal 1 and the weights of other edges incident to  $v_i$  are 0. Due to Completeness Theorem, eventually robot returns back to  $v_i$  by traversing an edge  $e_{ik}$ . The weight of  $e_{ik}$  is incremented and equals 2, the weight of  $e_{ij}$  is equal to 1 and the weights of other edges incident to  $v_i$  are 0.

Apply the same reasoning to every other 0-weighted edge incident to  $v_i$ . The weight of  $e_{ij}$  is equal 1, whereas the weights of other edges incident to  $v_i$  are 2. At this point LRV is forced to pick  $e_{ij}$  as the only edge of minimum weight incident to  $v_i$ . Hence, an incoming edge  $e_{ij}$  is traversed twice iff every other edge incident to  $v_i$  is traversed twice.

It follows from Lemma 1 that if before traversing an edge  $e_{ij}$ , the weights of all edges incident to  $v_i$  are equal (initially all 0), then after an incoming edge  $e_{ij}$  is traversed twice the weights of all edges incident to  $v_i$  are equal and incremented by two.

**Lemma 2:** An incoming edge  $e_{ij}$  is traversed twice iff in a subtree  $T' = T - (T_{ij} + e_{ij})$  every edge is traversed twice.

Proof: Consider a subtree T' (refer to Figure 6). LRV starts at vertex  $v_i$ . Applying Lemma 1 to  $v_i$  results in every edge incident to  $v_i$  traversed twice. Applying Lemma 1 recursively to every vertex of T' results in every incident to



Fig. 6. Illustration for Lemma 2.



Fig. 7. Illustration for Theorem 3.

every vertex edge traversed twice. Hence, an incoming edge  $e_{ij}$  is traversed twice iff in a subtree  $T' = T - (T_{ij} + e_{ij})$  every edge is traversed twice.

**Theorem 3:** The exploration time of LRV on a tree is no more than 2|E|.

Proof: Consider a tree T (refer to Figure 7). Augment T with a vertex v' and an edge  $e_{v'v}$  connecting v' to vertex  $v \in T$ . Consider LRV on the augmented tree starting at v'. It follows from Lemma 2 that the robot executing LRV would traverse  $e_{v'v}$  twice when in tree T every edge is traversed twice. Hence, the exploration time of LRV on a tree is no more than 2|E|.

Theorem 3 asserts that the performance of LRV on trees is linear or asymptotically optimal.

# VI. IMPLICIT SENSOR NETWORK REPAIR AND MAINTENANCE

An emergent property of LRV is the ability to perform network repair and maintenance. Since the algorithm is shown to be complete, it is guaranteed to visit the same node over and over again. Suppose that one of the nodes, say node k, ran out of power or was damaged. Further consider a moment in time just before the robot traverses direction d towards the damaged node. Now, the robot is moving along direction d towards node k. According to the deployment function that is used, there should be a communication/sensing gap in the deployed sensor network (unless the network was overdeployed and does not require repair). Hence while facing the same deployment situation and using the same deployment function at the location where node k was deployed the robot simply deploys a new node, thereby solving the problem of sensor network repair and maintenance implicitly. Note that if the robot can recognize the nodes then it can attempt repairing the node first (or retrieving for later repair at the base) before deploying the new node.

# VII. SUMMARY

We presented an analysis of the Least Recently Visited algorithm for the problem of coverage and exploration. LRV is based on the idea of the robot deploying sensor nodes into the environment from time to time. Once deployed, every node acts like a signpost recording which directions the robot have explored recently. When a robot is in the vicinity of a node, it recommends to the robot a direction that has been least recently visited (hence, the name LRV). The algorithm is decentralized, scalable, robust, fault tolerant and can be used on simple robots.

We analyzed the characteristics of LRV theoretically, modelling the static steady state of the deployed network as a finite graph G. We proved that LRV is complete on G (i.e. the exploration time of LRV on a finite graph is finite). For a graph G=(V,E) with maximum degree d, if Cover Time = O(f(V)), then Exploration Time = d\*O(f(V)). We proved that Exploration Time is  $\leq 2|E|$  (twice the number of edges, or asymptotically optimal) for the special case, when G is a tree. For another special case, when G is a square lattice, we empirically conjectured that both cover and exploration times are asymptotically  $O(V^{1+\epsilon})$ . We suggested a technique to determine an  $\epsilon$  for graphs with given number of nodes. The special case of a square lattice is also interesting from practical perspective, because in our LRV implementation and experiments [17] we chose to maintain at most 4 directions, which results in a static steady state of the deployed sensor network resembling a square lattice.

We examined the tradeoffs that should be considered in choosing one exploration algorithm over another to solve the problem of coverage and exploration. The bounds for the coverage task are given by random walk (the robot has no information and explores randomly) and depth first search (a map of the environment is available in the form of a graph) which solves the problem optimally. The data shown in Figure 3, suggest strongly that our algorithm asymptotically outperforms the k node algorithm presented in [12]. In addition, it is shown in [12] that if the number kof available nodes reduces, the cover time increases rapidly. Therefore, in dynamic environments the performance of the algorithm decreases drastically even if one node is destroyed. Whereas in our algorithm such a problem does not exist, since a new node will be deployed in place of the destroyed one automatically.

We compared LRV to 1-LRTA\* [20]. 1-LRTA\* is a well known graph search algorithm that can be applied to graph coverage. In 1-LRTA\*, a weight is associated with a node. The edge to traverse is chosen based on weights of neighboring nodes. The weight of a node is incremented with the weight of a node the robot transitions to. Hence, 1-LRTA\* requires nodes to communicate. Figure 3c shows that generally 1-LRTA\* outperforms LRV. However, in reality LRV deploys the network in addition to exploring, whereas 1-LRTA\* requires the graph to operate on. In addition, LRV communicates only with a local node, whereas 1-LRTA\* communicates with the neighbors of the local node as well.

The theoretical analysis on graphs shows that trade offs in the assumptions can affect *cover time* significantly. Simple algorithms like RW or DFS can be used for coverage, but only in the extreme cases as described above. In case, where mapping and localization are not available, but the number of available nodes is unlimited, our algorithm appears to outperform others.

#### VIII. ACKNOWLEDGMENT

This work is supported in part by NSF grants ANI-0082498, IIS-0133947, EIA-0121141, and CCR-0120778.

## REFERENCES

- M. Batalin, G. Sukhatme, and M. Hattig, "Mobile robot navigation using a sensor network," in *Proc. IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, April 2004, pp. 636–642.
- [2] M. Batalin and G. Sukhatme, "Using a sensor network for distributed multi-robot task allocation," in *Proc. IEEE International Conference* on Robotics and Automation, New Orleans, Louisiana, April 2004, pp. 158–164.
- [3] D. W. Gage, "Command control for many-robot systems," in the Nineteenth Annual AUVS Technical Symposium, Huntsville, Alabama, USA, 1992, pp. 22–24.
- [4] M. Batalin and G. Sukhatme, "Sensor network-based multi-robot task allocation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003, pp. 1939–1944.
- [5] J. O'Rourke, Art Gallery Theorems and Algorithms. New York: Oxford University Press, 1987.
- [6] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems*, Fukuoka, Japan, 2002, pp. 299–308.
- [7] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Proc. of 6th International Symposium* on Distributed Autonomous Robotic Systems, Fukuoka, Japan, 2002, pp. 373–382.
- [8] —, "Sensor coverage using mobile robots and stationary nodes," in SPIE2002, vol. 4868, 2002, pp. 269–276.
- [9] B. Yamauchi, "Frontier-based approach for autonomous exploration," in *In Proceedings of the IEEE International Symposium* on Computational Intelligence, Robotics and Automation, 1997, pp. 146–151.
- [10] A. Zelinsky, "A mobile robot exploration algorithm," in *IEEE Transactions on Robotics and Automation*, vol. 8, 1992, pp. 707–717.
- [11] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun, "Collaborative multirobot exploration," in *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, vol. 1, 2000, pp. 476–481.
- [12] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," in *IEEE Transactions on Robotics and Automation*, 7-6, 1991.
- [13] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," in *Annual ACM Symposium on Theory of Computing (STOC '98)*, 1998.
- [14] R. T. Vaughan, K. Stoy, G. S. Sukhatme, and M. J. Matarić, "Lost: Localization-space trails for robot teams," *IEEE Transactions on Robotics and Automation, Special Issue on Multi-Robot Systems*, vol. 18, no. 5, pp. 796–812, Oct 2002.
- [15] S. Koenig, B. Szymanski, and Y. Liu, efficient and Inefficient Ant Coverage Methods. Annals of Mathematics and Artificial Intelligence, 31, 41-76, 2001.

- [16] I. Wagner, M. Lindenbaum, and A. Bruckstein, distributed covering by ant-robots using evaporating traces. IEEE Transactions on Robotics and Automation, 15(5):918 933, 1999.
- [17] M.A.Batalin and G. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, vol. 26, no. 2, pp. 181–196, 2004.
- [18] L. Lovasz, Random Walks on Graphs: A Survey, ser. Combinatorics, Paul Erdos is Eighty, Keszthely, Hungary, 1993, vol. 2, pp. 1–46.
- [19] S. Koenig and R. Simmons, "The influence of domain properties on the performance of real-time search algorithms," School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, Tech. Rep. CMU-CS-96-115, 1996.
- [20] —, "Easy and hard testbeds for real-time search algorithms," in Proceedings of National Conference on Artificial Intelligence, 1996, pp. 279–285.