Multi-Agent Area Coverage Using a Single Query Roadmap: A Swarm Intelligence Approach

Ali Nasri Nazif*, Alireza Davoodi, and Philippe Pasquier

Faculty of Communication, Art and Technology
School of Interactive Art and Technology
Metacreation, Agents and Multiagent Systems (MAMAS) Group,
250 - 13450 102nd Avenue, Surrey, BC, Canada, V3T 0A3
{anasri}@aut.ac.ir, {alireza_davoodi,pasquier}@sfu.ca
http://www.siat.sfu.ca

Abstract. This paper proposes a mechanism for visually covering an area by means of a group of homogeneous reactive agents through a single-query roadmap called Weighted Multi-Agent RRT, *WMA-RRT*. While the agents do not know about the environment, the roadmap is locally available to them. In accordance with the swarm intelligence principles, the agents are simple autonomous entities, capable of interacting with the environment by obeying some explicit rules and performing the corresponding actions. The interaction between the agents is carried out through an indirect communication mechanism and leads to the emergence of complex behaviors such as multi-agent cooperation and coordination, path planning and environment exploration. This mechanism is reliable in the face of agent failures and can be effectively and easily employed in cluttered environments containing narrow passages. We have implemented and evaluated the algorithm in different domains and the experimental results confirm the performance and robustness of the system.

Key words: Multi-agent, Single query Roadmap, Environment Coverage, Swarm Intelligence, Indirect Communication

1 Introduction

The area coverage problem in robotics deals with the use of one or more robots to physically sweep [8] or visually sense [5]the free space of an area. The aim of this research study is to propose a robust mechanism to cope with the problem of multi-agent visual environment coverage.

To realize this purpose, we apply an emergent coordination approach and introduce a bio-inspired technique which utilizes a roadmap called `WMA-RRT` and employs a group of multiple autonomous agents to cover a given environment. To this end, agents should be capable of performing in the area and cooperating with each other to achieve the mutual objective of covering an area.

* The first and second authors have equally contributed in the paper.

In accordance with the *Swarm intelligence* principles originally introduced by Beni and Wang [3], the proposed system is made up of several simple autonomous agents. These agents locally interact with the environment and pursue their own goals by following some simple and explicit condition-action rules and performing the their corresponding actions. In such a scenario, the multi-agent cooperation is an invaluable by-product of a special communication mechanism. In the other words, the direct agents' interaction with the environment leads to the emergence of indirect communications and consequently teamwork among the agents.

In this research study, we assume that the structure of the environment and a 2D map of the area are available. Furthermore, the robots are considered to be point robots for the sake of simplicity, and it does not affect the solution. If the robots are circular robots of radius r, Minkowski sum of the static obstacles and the robots are computed and considered instead of the original static obstacles in the environment[11]. Moreover, the robots are assumed to have 360 degree field of view and they are able to observe part of the area which is not further than a pre-defined visibility range.

The approach to the environment coverage problem, presented in this paper, can be divided into two processes. Firstly, a special roadmap called `WMA-RRT` is constructed and embedded in the environment to discretely represent the free space. Secondly, `WMA-RRT` is distributed among all the agents. Each agent independently bears the responsibility of locally traversing its associated part of the `WMA-RRT` roadmap.

The agents employ the `WMA-RRT` both as a roadmap and an information exchange channel. This channel is used by the agents to interact with the environment and indirectly communicate to one another. In the other words, the agents change the situation of the environment by altering the states of the edges and nodes of the roadmap while traversing the roadmap tree. Meanwhile, this information is used by the agents in order to decide which action to take, where to go next, how to return to the origin, how to support other working agents and handle the agents' failures during the operation. We have implemented our approach and evaluated it considering some criteria such as the number of robots, different visibility ranges and maps.

The rest of the paper is organized as follows. First, in Section 2, we review some preliminary concepts and related works. The `WMA-RRT` roadmap is introduced in Section 3. The agents architecture is addressed in Section 4 and followed by the `WMA-RRT` traversing algorithm in Section 5. Also, the implementation of the mechanism and experimental results are presented in Section 6. Finally, Section 7 concludes with a short summary of the research and introduces some future works.

2 Preliminaries and Related Works

2.1 Roadmaps

A roadmap is a graph constructed in the environment to capture the connectivity of the free space of the area. Generally, two types of roadmaps have been introduced by the researchers. The first group of roadmaps requires and utilizes the exact structure of the environment. `Visibility Graphs` and `Generalized`

`Voronoi Diagram` belong to this group. Kai and Wurm [19] introduce an approach for environment exploration using `Voronoi Diagram`. They use `Voronoi Diagram` to decompose the environment into several regions and assign each of them to a robot to explore. Also, Jiao and Tang [17] employ a visibility-based decomposition approach for multi-robot boundary coverage in which the robots are supposed to collectively observe the boundary of the environment.

On the contrary, there are sampling-based roadmaps which do not need the explicit geometrical representation of the workspace. Instead, they use some strategies for generating sample points throughout the free space and connecting them to build the roadmap. Basically, two types of sampling-based approaches have been introduced by the researcher: `Multiple Query` and `Single Query`. In multiple-query planners [9], a roadmap is constructed by simultaneously expanding some trees from several randomly distributed starting points and merging them to prepare the entire roadmap. It is likely that the roadmap is not connected in a cluttered environment. Probabilistic roadmaps, `PRM`, introduced by Kavraki is the most popular multiple-query planner.

Single-query planners, on the contrary, have been introduced to construct a connected tree which spreads over the environment. They use an incremental approach to build the roadmap by expanding the tree from a random starting point. `RPP`, `Ariadnes Clew`, `2Z`, `EST`, `Lazy RPM` and `RRT` are single-query planners. Since the planner introduced in this paper is a variation of Rapidly-Exploring Random Tree `RRT` [10], this section briefly explains how `RRT` algorithm works.

The roadmap constructed by the `RRT` planner is a tree which expands throughout the free space of a given environment. This planner is probabilistically complete meaning that the probability of covering every location of the environment converges to one, if the algorithm keeps running for a long enough time.

In order to construct `RRT` roadmap, first a uniform distribution is used to insert a sample point q_{init} in the environment. This point is added as a node to the `RRT` tree if collision-free. In each iteration another sample point, q_{rand}, will be placed randomly in the free space and its nearest neighbor node, q_{near}, among previously added nodes will be selected for further expansion. A new node, q_{new}, is produced as a result of moving q_{near} by ϵ, a predefined value called `step size`, toward q_{rand} and it will be added to the roadmap if collision-free. Figure. 1 illustrates a snapshot of the expansion process.

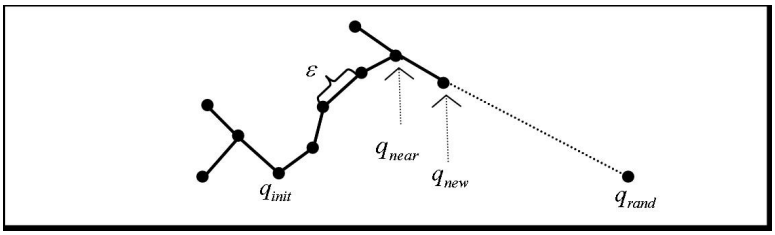

Fig. 1. This figure illustrates how a `RRT` tree is expanded given the current state of the tree.

2.2 Multi-Agent Environment Coverage

The Swarm Intelligence model, `SI`, relies on emergent coordination and is inspired by the collective behaviors observed in a multitude of species which exhibit social life such as ants and honey bees. As an AI technique, `SI` discusses the systems consisting of a population of simple agents interacting locally with one another or the environment [15, 18, 1] and these interactions often lead to the emergence of complex and global behaviors.

In one of the oldest attempt, Gordon and Wagner [6] employed honeybees' foraging behavior to coordinate a group of agents with no elaborate language and advanced communication device. Svennebring and Koenig [16] introduce a bio-inspired algorithm for covering an environment decomposed to a number of cells in which the robots use an artificial pheromone to mark the visited cells. Also Bayazit [2] discusses a multiple query roadmap used as a means of indirect communication between agents for covering an environment. Another application of indirect communications between agents is the work of Halasz [7] in which he deals with the dynamic redistribution of a swarm of robots among multiple sites to cover the environment.

Moreover, the swarming behavior model observed among birds flocks, fish schools and sheep herds motivated Reynolds [14] to introduce a bio-inspired technique to steer a group of rule-based autonomous agents, called Boid *birds-like object*. The Boids are endowed by some simple behaviors like seeking, fleeing, obstacle avoidance, wandering etc. The Boids make use of these actions to navigate the environment and represent more complex actions.

On the contrary, in intentional cooperation approaches, the agents deliberately communicate or negotiate with each other in order to cooperatively achieve a mutual goal. Zlot [20] addresses a market-based mechanism for environment exploration in which they use an auction based method to assign exploration targets between the agents. Furthermore, Choset [4] surveys different approaches such as approximation, grid-based and decomposition approaches. In one of the recent attempt Packer [12] applies a computational geometry approach in order to compute multiple Watchman routes which can cover inside a given polygon. In this paper, she first solves the art gallery problem and finds a set of guards which can collectively cover the entire area. Then, the visibility graph of the obstacles and static guards is constructed and decomposed into several routes. Finally, each route is allocated to one agent to traverse.

Grid-based approaches are also used in environment coverage. Hazon and Kaminka [8] propose a grid-based mechanism in which the environment is decomposed into several same size cells on the basis of the size of the robots. Considering the initial positions of the robots, they find a spanning tree of the free cells which can be decomposed to some balanced sub-trees. Finally each sub-tree is assigned to a robot.

3 Weighted Multi-Agent RRT

`WMA-RRT` is an extension of the `RRT` planner and consequently, categorized as a single-query planner. Moreover, `WMA-RRT` roadmap is an infrastructure which

supports simultaneous movements of many agents throughout the cluttered environments which contain narrow passages.

Construction of WMA-RRT begins by figuring out the location of the root of the roadmap which is called Main Root and continues to specify the nodes that are adjacent to the Main Root and located at the next level of the tree. These nodes are called Secondary Roots. The WMA-RRT is then constructed by expanding all the edges between the Secondary Root and Main Root in all directions throughout the free space.

In the construction of WMA-RRT, it is assumed that all the agents in the environment are situated close to each other and consequently can be surrounded by a simple polygon. A simple polygon is a polygon whose boundary does not cross itself. In order to construct this polygon, the convex hull algorithm [13] is applied to build a polygonal boundary enclosing all the agents. The convex hull algorithm considers the locations of the agents as points and constructs the minimal convex hull of the points regardless of the obstacles in the environment. Then the intersection point of the two longest diameters of the polygon is calculated and considered as the starting point for the construction of the WMA-RRT and called Main Root of the roadmap. If the intersection point is not collision-free, the intersection points of other next longest diameters of the polygon are considered. If the convex hull is a triangle, the intersection point of two longest edges of the triangle is considered as the Main Root.

In the second step, the goal is calculating and assigning one particular node of WMA-RRT to each agent. To this end, the algorithm considers all the edges, which connect the agents initial locations to the Main Root. A set of candidate nodes is produced as a result of moving out from the Main Root by the value of ϵ towards the agents' locations along the edges. The value of ϵ is specified by the user and must be equal or less than the value of the step size as defined in the previous section. If a candidate node is collision-free, it is called a Secondary Node and assigned to the corresponding agent whose location point is an endpoint of the edge. If the candidate node is not collision-free, the algorithm finds the nearest agent which has been assigned a Secondary Node and assign it to the agent. Figure. 2 illustrates the Main Root and Secondary Roots of a particular configuration.

From this point on, the algorithm is similar to the RRT planner algorithm with two differences. The first one is in the number of initial points, q_{init}. While RRT there is only one q_{init} node, which is randomly located in the free space, the WMA-RRT includes as many q_{init} nodes as there are Secondary Roots. The other difference is observed in the initial expansion of the roadmap. In RRT the roadmap expands through the root of the roadmap, q_{init}, while in WMA-RRT the Main Root is exempted from expansion and instead, the Secondary Roots are considered for expansion. The initial roadmap expands very rapidly throughout the free space of the environment and yields a tree with one Main Root, which has as many sub-trees as there are Secondary Roots. Each sub tree is called a Branch.

The edges' weight of the roadmap is initialized to 0 so the roadmap is considered as a weighted tree. During the covering operation, at-least one agent is assigned to each Branch. The weights of the edges are updated while traversing the roadmap. The agents use WMA-RRT roadmap to move throughout the

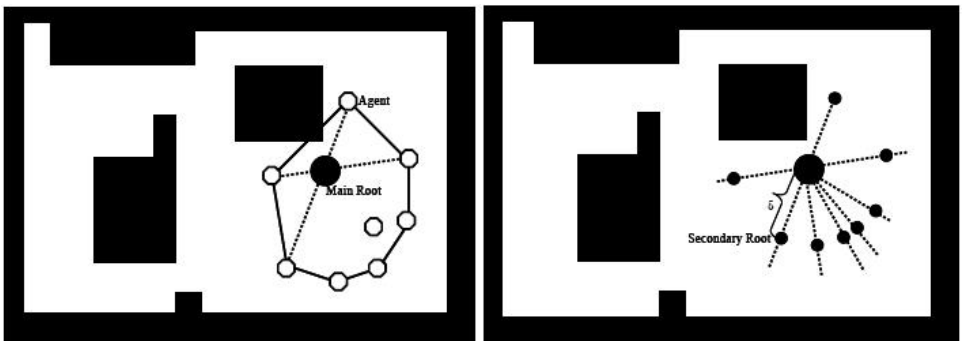

Fig. 2. Illustrates a situation in which eight agents (small white polygons) are available. The figure on the left represents the polygonal boundary enclosing the agents, the diameters and the `Main Root` (the black circle). The figure on the right illustrates the corresponding `Secondary Roots`

free space, interact with the environment and indirectly communicate with one another. In the other words, while traversing the `WMA-RRT` roadmap, the agents perceive information available through the roadmap and update their knowledge about the environment. Further, the agents change the state of the environment by depositing information into the nodes of the `WMA-RRT`. There is no information available through the edges. Table 1 represents the information that may be deposited by the agents and available to them in the `WMA-RRT` roadmap.

information	Main Root	Secondary Root	Intermediate Nodes
e_o: label of each outgoing edge	No	Yes	Yes
e_i: label of the incoming edge	No	Yes	Yes
w_o: weight of the incoming edge	No	Yes	Yes
w_i: weight of the outgoing edge	No	Yes	Yes
c: shows all the sub-trees of a node have been completely explored	Yes	Yes	Yes
l_o: label of the outgoing edge which is temporarily locked and is not accessible	No	Yes	Yes
c_o: label of the first edge of a sub-tree which has been completely explored	No	Yes	Yes

Table 1. Represents all the information available in the nodes of the `WMA-RRT` roadmap.

In some situations, it is impossible to have as many branches as there are agents. This is specially the case when the environment is highly cluttered and contains many narrow passages since some of the `Secondary Roots` may not be collision free and consequently, have to be deleted from the roadmap. In such

situations, in order to make use of the maximum capabilities of the available agents, the algorithm assigns more than one agent to each branch. Figure. 3 represents an environment and its corresponding `WMA-RRT` roadmap containing two `branches` and there are four agents available in the environment. As illustrated, the algorithm assigns two agents to each branch.

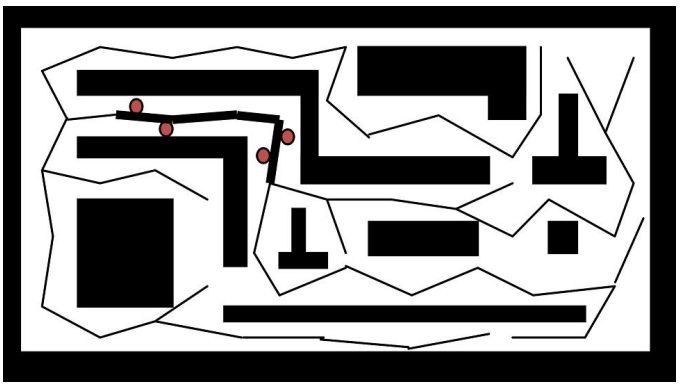

Fig. 3. A cluttered environment with narrow passages. Two agents have been assigned to each branch.

Finally, we compare `WMA-RRT` tree as a single query roadmap with the Probabilistic Road Map, `PRM`, as a multiple query roadmap. As mentioned above, a multiple query planner might lead to a disconnected roadmap and consequently, some parts of the environment are not accessible by any agents. Figure. 4 represents the results of running `PRM` and `WMA-RRT` algorithms on a sample environment.

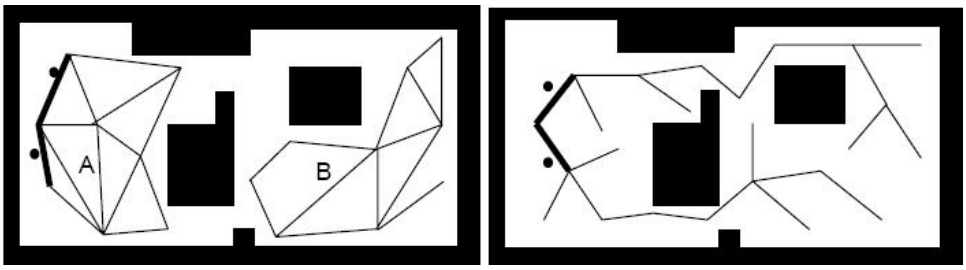

Fig. 4. The figures on the left and right represent the results of running PRM and `WMA-RRT` algorithms respectively on an environment.

4 Agent Architecture

As mentioned in previous sections, the agents in our system are simple autonomous individuals capable of following some explicit condition-action rules and performing the corresponding actions. To this end, our agents are modeled based on reactive architecture and are implemented in such a way that supports `beliefs`, `actions` and `Plans` of the agents. Furthermore, they are utility based entities, which try to maximize their own utilities independently. The utility function of an agent is defined as the average over the weight of all the edges, the agent traverses. The utility function of an agent motivates it to select the edges with minimum weight.

Additionally, the agents perceptions are represented as beliefs in the agents' belief bases and each agent contains some pre-defined plans stored in its plan library. Each plan is an instruction which tells the agent what to do in a particular circumstance. Each agent has a goal and uses a plan to approach the goal. In the rest of this section, we discuss some basic beliefs, actions and plans used by the agents during the covering operation. We use Agent Speak, a multi-agent programming languages standard syntax and format in order to explain the agents beliefs, actions and plans. As discussed in previous section, the agents update their knowledge base about the world when they reach a node and perceive the information available at that particular node. In the other words, while traversing the `WMA-RRT` tree, an agent detects the information available in the nodes of the roadmap and updates its belief base accordingly. The followings are some beliefs of the agents:

1. `subtreeState(e, w)`: Standing at each node, an agent could detect some information about all the outgoing and incoming edges, including the edges' labels ,`e`, and weights, `w`.
2. `currentEdge(e,w)`: When an agent decides which outgoing edge to take, it can remember the edges label and weight until it reaches the endpoints of the edge and updates the related information about the traversed edge.
3. `completed(e)`: It means that the agent believes the sub-tree of the current node that starts with edge `e`, has been completely explored.
4. `currentBranch(b)`: It represents the current branch, b, which is being explored.
5. `agentName(ag)`: Each agent has a unique name which is stored in the belief base of the agent.

Furthermore, each agent is capable of performing some basic external and internal actions to change the state of the environment and its internal state. The followings are some of the actions:

1. `allocateBranch`: Standing at a `Secondary Root`, an agent updates the state of the current roadmap's node, by adding information about the label and weight of the incoming edge to the current node, which is a `Secondary Root`
2. `traverse`: An agent can traverse the selected edge by moving from the start node of the edge and reaching the end point of it.

3. `departNode`: When an agent decides which edge to traverse next, it increases the weight of the selected edge by one and updates its corresponding information, including the label and new weight of the selected edge, in the current node.
4. `reachNode`: As an agent reaches a node, it updates the state of the current node to represent the label and weight of the incoming edge to the current node.
5. `comeback`: When an agent is to return one level back to the parent of the current node, it should set the current node as completed. It means that the entire sub-tree of the current node has been completely explored. Then, as soon as the agent arrives in the parent node, it updates the state of the parent node to indicate that one of its sub trees has been completely explored.
6. `lock(Edge)`: When an agent enters an edge, it locks the edge temporarily to prevent other agents enter the edge simultaneously. In order to do this, the agent adds some information in the start node of the edge representing that the edge with label `Edge` is locked. Each locked edge, gets unlocked after a specific period of time automatically. This period of time equals to the twice the maximum amount of time an agent needs to completely traverse an edge, plus some amount of time the agent requires to detect the information available and update its belief base accordingly. This time can be specified at the beginning of the covering operation. Remember that the maximum length of each edge of the `WMA-RRT` equals to the step size, ϵ,which has been already defined in `WMA-RRT` construction algorithm.
7. `detect`: The agents use their sensors in order to percept all the information available in the current node. Standing at a node, an agent also can detect the outgoing edges of the current node.
8. `scan`: Standing at a node, each agent can access a local roadmap containing the current node, incoming edge and outgoing edges of the current node. The agents cannot store these local roadmaps in their limited memories but an agent can temporarily employ the local roadmap in order to enumerate the outgoing edges and assign them temporary labels.
9. `enumerate`: Standing at a node, an agent changes the state of the current node by updating information relevant to the incoming and outgoing edges of the current node. To differentiate the outgoing edges, the agents locally enumerate them and assign each of them a label. To this end, all the agents assume that, the labels of all the edges going out of the `Main Root`, are zero. As mentioned in previous sections, the roadmap is locally available to the agents so they can detect all the outgoing edges and label them by enumrating all the outgoing edges in a clock-wise order starting from the incoming edge and incrementally assigning each of them a label.To this end, the algorithm assumes that the label of all the `Secondary Roots` are zero. In other words, suppose that the weight of the incoming edge of the current node is w and there are two outgoing edges. Then, by doing a clock-wise search starting from the incoming edge, the label of the first and second edges are $w + 1$ and $w + 2$ respectively. Figure 5, illustrates how the edges of `WMA-RRT` are locally enumerated by the agents.

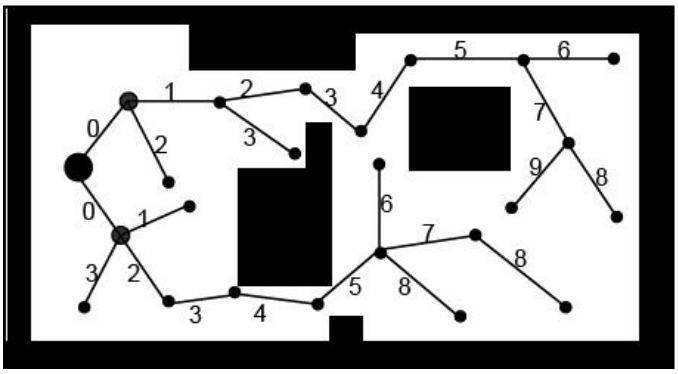

Fig. 5. This figure represents how the edges of the WMA-RRT are locally enumerated and labeled by the agents. Here, the biggest black circle is the Main Root and two second biggest are the `Secondary Roots`.

Furthermore, each agent maintains a plan library including some built-in plans. Some of the simplified versions of the plans available in the plan library of the agents are as followings:

1. `findingBranch`: Each agent applies this plan to find a branch to explore. This plan starts by performing the action of `scanning` around the `Main Root` to build a local roadmap. The local roadmap contains the `Main Root`, `incoming edge` and `outgoing edges`. The `incoming edge` to the `Main Root` is exclusive to each agent since the agent considers the edge it takes to reach the `Main Root` from its initial position as mentioned in Section 3. The local map is then used by the agent to locally enumerate and label the outgoing edges. Then, the agent traverses the outgoing edges sequentially in order to find the first outgoing edge with minimum weight. This is done through detecting the information available in the `Secondary Roots`.

```
!+findingBranch : constructed(wma-rrt) <-
scan;
internalAction.enumerate;
internalAction.getTheUnCheckedEdgeInOrder(Edge);
traverse(Edge);
detect;
!allocation(Edge).

!+allocation(Edge) : isMinimum(Edge) <-
!findingBranch.

!+allocation(Edge) : not isMinimum(Edge) <-
allocateBranch(Edge);
!nextDestination(Edge).
```

2. `nextDestination`: This plan is used when an agent reaches a node. Each agent is simply able to temporarily remember the label and weight of the

incoming edge to the current node. This plan tells the agent to scan around the current node, locally enumerate the outgoing edges to assign each of them a label, detect all the information available in the current node and find the next best edge to traverse. The best edge is an edge which is not currently locked and has not been set as completed and also has the minimum weight among all other outgoing edges of the current node.

```
+!nextDestination : true <-
scan;
internalAction.enumerate;
detect;
internalAction.findNextEdge(NextEdge);
!traversing(NextEdge).
```

3. `traversing`: This plan is invoked when the agent has made a decision on which outgoing edge to traverse. This plan makes the agent lock the selected edge, depart from the current node, traverse the edge and reach the other endpoint of the selected edge.

```
+!traversing(Edge) : true <-
departNode;
lock(Edge);
traverse(Edge);
reachNode.
```

4. `return`: This plan is used when an agent reaches a leaf or blind node, a node whose sub-trees have been traversed and set as completed, so the agent cannot move forward and has to return one level back to the parent of the current node.

```
+!return(Node) : leaf(Node) | blind(Node) <-
internalAction.nextEdge(Edge);
comeback(Node);
traverse(Edge);
detect.
```

5 Exploration of WMA-RRT Roadmap

As discussed earlier, the `WMT-RRT` roadmap, is considered as a set of several sub-trees called `Branch`. The ultimate goal of our algorithm is to assign the branches of the `WMA-RRT` roadmap to the agents effectively and let the agents traverse the tree completely. To achieve this objective, each agent is supposed to independently find a branch and commit itself to traversing it. An agent start performing the coverage mission as soon as it arrives at the `Main Root`.

When an agent arrives at the `Main Root` the only things it can detect are the edges going out of the `Main Root`. Using the `findingBranch` plan the agents find a branch and commit to traversing it. Since there is no information maintained in the `Main Root`, the agents have to check the `Secondary Roots` to figure out

which one has not been yet assigned to one agent. Using the same enumeration approach each agent can exclusively enumerate all the outgoing edges from the `Main Root` and visit them sequentially. The information available at the `Secondary Roots` tells the agent whether the current branch has been already assigned to any other agent.

Each agent continues to search for a branch to eventually find a branch and commit to exploring it. The utility function of an agents motivates it to select unallocated branches. If there is no remaining unallocated branch, they look for a branch in which the minimum numbers of agents are working. Running the `findingBranch` plan directs each agent to reach its corresponding `Secondary Root`. The agents then update the state of the environment accordingly by changing the states of the nodes.

At each node, the agent executes the `nextDestination` plan to find the next edge to traverse. Basically, The outgoing edge with minimum weight is selected for exploration since edges with higher weights have been already visited more so it is reasonable to select those edges with fewer weights. Then the agent performs the `trversing` plan to update the state of the start node of the edge, traverse the edge and update the state of the end node of the edge. If an agent reaches a leaf or blind node of the current branch, it applies the `return` plan in order to return to the parent node and the leaf or blind node is marked as completed. An agent sets a node as completed if and only if the node's entire sub-trees have been marked as completed. When an agent is going to perfom the `return` plan, it should know which edge is the incomming edge. Since the states of the incoming edge and outgoing edges of the current node are available, the agent can figure out which edge is the incoming edge since the incoming edge is the only edge which has not been marked as completed.

Eventually, when an agent returns to its corresponding `Secondary Root`, it marks the `Secondary Root` as completed and returns one level back to the `Main Root` to contribute in exploration of other branches if there is any remaining. As soon as an agent realizes that all the branches have been set as completed, it marks the Main Root as completed to indicate that the exploration process is done.

It is very likely that one agent completes the navigation of its corresponding branch and returns to the origin while some other agents are still working in other branches. In such situations, an agent manages to maximize its contribution by helping some other agents in their exploration missions. As an agent returns to its corresponding `Secondary Root` and consequently, reaches the `Main Root`, it applied its findingBranch plan to find the branch in which the minimum number of agents are working in and commit to exploring it. Consequently, the agent could help other agents working in the same branch to finish the exploration task. When an agent arrives in a node, which has been marked as completed, it does not go further and therefore, returns to the parent node by executing the `return` plan. This process guarantees the exploration of the environment even in the case of occurring massive failures in the agents. This feature makes our system reliable and robust.

6 Implementation and Experimental Results

6.1 Implementation

Since the agents are modeled according to the `BDI` architecture, we have applied the same design pattern used in the Agent-Speak multi-agent programming language to implement the system. According to this pattern, there is an environment which is shared between all the agents in the environment and is implemented in the class `SharedEnvironment`. All the required data structures including the data structures for `WMA-RRT` have been defined and implemented in a class called `EnvironmentModel` which models the environment.

Each agent is a thread which is started in the `SharedEnvironment` class as soon as the offline process of constructing `WMA-RRT` is completed. Since all the agents are homogeneous, each agent is implemented as an object of the `Agent` class. In this class, variables such as max-speed, max-force, current-active-branch, current-position, current-velocity and target-position are defined. This class also includes all the required data structures for representation of beliefs, belief base, plans library and all necessary methods like deliberation, plan selection and belief update. Also, the actions of each agent have been implemented as methods in the Agent class.

Each plan is also implemented as an object of the Plan class. This class includes some data structures in order to maintain the plans pre-conditions, event triggers and a list of references to the actions listed in the plans.

6.2 Experimental Results

We have evaluated the mechanism discussed in this paper, in a variety of environment in order to capture the effect of the number of sample points, step size (distance between two adjutant nodes of `WMA-RRT`) and the sensor range, which is the visibility range of the agents. All the experiments have been conducted in a Pentium 4 CPU 3.00GHz with 1.00 GB of RAM machine. Figure **??** represents a sample environment including three agents initially located near each other and its corresponding `WMA-RRT` tree and areas covered by the agents.

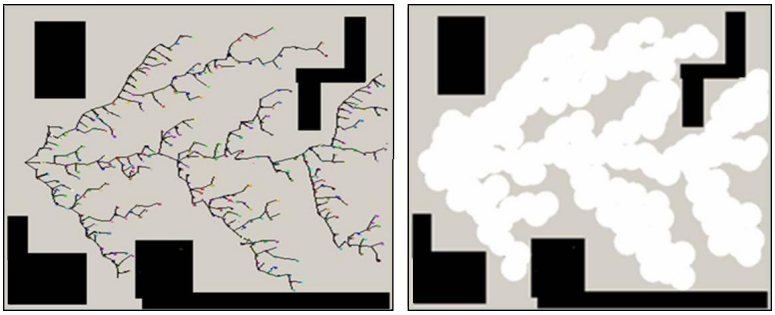

Fig. 6. Shows the results of running the `WMA-RRT` tree and the coverage mechanism in the presence of three agents.

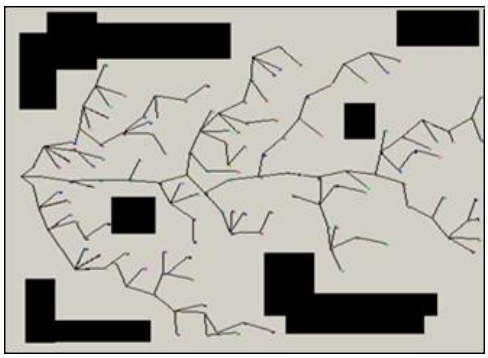

Fig. 7. Represents another sample environment and the corresponding `WMA-RRT` tree in the presence of three agents.

The Tables 2 and 3 summarize the results of running the proposed area coverage mechanism in the presence of three agents on two environments represented in Figure 6 and 7 respictively. According to the results represented in Tables 2 and 3, it can be concluded that as the number of samples, step size and sensor range increase, the coverage percentage of the environment improves. As the results show, in cluttered environments the step-size should be as small as possible.

7 Future Works and Conclusion

In this paper, we discuss the problem of visual environment coverage by means of a group of many simple agents. In the proposed mechanism, a special roadmap called `WMA-RRT` is constructed in the environment and used by the agents both as roadmap and a communication channel. We use an emergent coordination mechanism in which the agents cooperate with each other to achieve a final goal while pursuing their own individual objectives. The proposed approach is specially effective in situations in which the exact structure of the environment is not known and it is possible to use many numbers of simple agents each of which is only capable of performing some simple actions and there is no communication and negotiation devises. While the `WMA-RRT` can be easily used in an environment with narrow passages but the covering time increases as the number of narrow passages increases. This is due to the effect of the value of `step size` in `WMA-RRT`. In order to cover the narrow passages of environments, the value of `step size` should be small enough to let the `WMA-RRT` tree expand throughout the narrow passages. As further works, we are interested to extend our study in order to apply a group of heterogeneous agents with a different visibility range to cover the environment. Also, we are interested in finding a more appropriate roadmap which decreases the total time of the environment coverage in the presence of many narrow passages.

number of nodes	step size	run/per each group of agents	visibility range	coverage percentage
400	20	10	r	74
400	20	10	$2r$	87
400	20	10	$4r$	90
400	20	10	$infinite$	99
800	20	10	r	83
800	20	10	$2r$	88
800	20	10	$4r$	92
800	20	10	$infinite$	99
200	40	10	r	88
200	40	10	$2r$	93
200	40	10	$4r$	95
200	40	10	$infinite$	99
400	40	10	r	93
400	40	10	$2r$	95
400	40	10	$4r$	97
400	40	10	$infinite$	99

Table 2. Represents all the information available in the nodes of the `WMA-RRT` roadmap.

number of nodes	step size	run/per each group of agents	visibility range	coverage percentage
150	20	10	r	68
150	20	10	$2r$	80
150	20	10	$4r$	88
150	20	10	$infinite$	97
300	20	10	r	83
300	20	10	$2r$	92
300	20	10	$4r$	95
300	20	10	$infinite$	99
600	40	10	r	93
600	40	10	$2r$	95
600	40	10	$4r$	97
600	40	10	$infinite$	99
150	40	10	r	68
150	40	10	$2r$	80
150	40	10	$4r$	88
150	40	10	$infinite$	97

Table 3. Represents all the information available in the nodes of the `WMA-RRT` roadmap.

References

1. A. Abraham, H. Guo, and H. Liu. Swarm intelligence: Foundations, perspectives and applications. In *In: Swarm Intelligent Systems, Studies in Computational Intelligence*, pages 3–25. Springer, 2006.
2. O. B. Bayazit. Roadmap-based flocking for complex environments. In *Proc. 10th Pacific Conference on Computer Graphics and Applications , PG02*, pages 104–113, 2004.
3. G. Beni and J. Wang. Swarm intelligence. *Proceedings Seventh Annual Meeting of the Robotics Society of Japan, Tokyo: RSJ Press*, pages 425–428, 1989.
4. H. Choset. Coverage for robotics - a survey of recent results. *Ann. Math. Artif. Intell.*, 31(1-4):113–126, 2001.
5. A. Davoodi, P. Fazli, P. Pasquier, and A. K. Mackworth. On multi-robot area coverage. In *7th Japan Conference on Computational Geometry and Graphs, JCCGG2009*, 2009. To Appear.
6. N. Gordon, I. A. Wagner, and A. M. Bruckstein. Discrete bee dance algorithm for pattern formation on a grid. In *IEEE International Conference on Intelligents Agents Technologies, Toronto, CA*, pages 545–549, 2003.
7. Á. M. Halász, M. A. Hsieh, S. Berman, and V. Kumar. Dynamic redistribution of a swarm of robots among multiple sites. In *International Conference on Intelligent Robots and Systems, IROS*, pages 2320–2325, 2007.
8. N. Hazon and G. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 2008.
9. L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. volume 12, pages 566–580, 1996.
10. S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *4th Workshop on the Algorithmic Foundations of Robotics, Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.
11. E. Oks and M. Sharir. Minkowski sums of monotone and general simple polygons. *Discrete & Computational Geometry*, 35(2):223–240, 2006.
12. E. Packer. Computing multiple watchman routes. In C. C. McGeoch, editor, *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, volume 5038 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2008.
13. F. P. Preparata and S. J. Hong. Convex hulls of finite sets of poin ts in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977.
14. C. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.
15. E. Sahin, T. H. Labella, V. Trianni, J. louis Deneubourg, P. Rasse, D. Floreano, L. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo. Swarm-bot: Pattern formation in a swarm of self-assembling mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Hammamet*, pages 6–9. IEEE Press, 2002.
16. J. Svennebring and S. Koenig. Trail-laying robots for robust terrain coverage. In *ICRA*, pages 75–82, 2003.
17. L. J. Tang. A visibility-based algorithm for multi-robot boundary coverage. *International Journal of Advanced Robotic Systems*, 5:63–68, 2008.
18. P. Tarasewich and P. R. McMullen. Swarm intelligence: power in numbers. *Commun. ACM*, 45(8):62–67, 2002.
19. K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. 2008.
20. R. Zlot, A. Stentz, A. tony Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. pages 3016–3023, 2002.