2013 10th IEEE International Conference on Control and Automation (ICCA)
Hangzhou, China, June 12-14, 2013

Efficient, Swarm-Based Path Finding in Unknown Graphs Using Reinforcement Learning*

M. Aurangzeb, F. L. Lewis, and M. Huber

Abstract— This paper addresses the problem of steering a swarm of autonomous agents out of an unknown maze to some goal located at an unknown location. This is particularly the case in situations where no direct communication between the agents is possible and all information exchange between agents has to *occur indirectly through information "deposited" in the* environment. To address this task, an ε –greedy, collaborative reinforcement learning method using only local information exchanges is introduced in this paper to balance exploitation and exploration in the unknown maze and to optimize the ability of the swarm to exit from the maze. The learning and routing algorithm given here provides a mechanism for storing data needed to represent the collaborative utility function based on the experiences of previous agents visiting a node that results in routing decisions that improve with time. Two theorems show the theoretical soundness of the proposed learning method and illustrate the importance of the stored information in improving decision-making for routing. Simulation examples show that the introduced simple rules of learning from past experience significantly improve performance over random search and search based on Ant Colony Optimization, a metaheuristic algorithm.

I. Introduction

This paper presents a randomized, distributed approach to steer a swarm of agents out of any type of unknown maze to a goal located at some unknown location using only locally stored information and no direct communication between the agents. This is an important problem not only for groups of autonomous robots but also for minimum overhead distributed routing and graph search problems for a wide range of applications. The approach presented here employs a collaborative reinforcement learning (RL) framework and is based on formal results underlining the soundness of the approach.

The problem of robot learning to escape a maze is not new to the machine learning research community; it was originally posed many decades back by H. Abelson and A. A. diSessa in [1]. Since then, there has been a great deal of research in robots learning to navigate in and escape from a maze. In [5] an architecture for autonomous mobile agents is proposed that maps a two-dimensional environment, and provides safe paths to unexplored regions. In [6], algorithms are proposed for two heterogeneous robots searching for each other in an unknown environment. In [7] an ultrasonic sensor localization system for autonomous mobile robot navigation in an indoor semi-structured environment is presented. To efficiently navigate mazes, various approaches for automated maze search have been implemented and several testing environments have been proposed [2], [8]. In [3] a knowledge-guided route search is proposed based on obstacle adaptive spatial cells. Similarly, a neural network based approach is used in [4] for a robot to solve a maze while avoiding concave obstacles.

Besides these applications in robotics and route planning, maze exploration is also used as a standard test benchmark for artificial intelligence and machine learning techniques [9]. Along those lines, there is also some study showing that antibodies in an immune system use a mechanism of learning from their surroundings to efficiently fight antigens. This has led scientists to use machine learning for the development of artificial immune systems [10], [11], [12] which, in turn have been tested on moving robots in mazes [10].

While most of the robotics learning work on maze navigation deals with single or small groups of robots, swarm intelligence (SI) is a class of decentralized algorithms based on the cooperative behavior of a large number of agents to achieve a common goal. These algorithms are based on simple rules inspired by biological systems in nature. There are a significant number of SI algorithms proposed in literature [13], [14]. These include Ant Colony Optimization (ACO) [15], [44], Artificial Bee Colony (ABC) [16], Artificial Immune System (AIS) [17], Charged System Search (CSS) [18], Cuckoo Search (CS) [19], [20], Firefly algorithm (FA) [21], [22], Gravitational Search Algorithm (GSA) [23], Intelligent Water Drops algorithm (IWD) [24], Particle swarm optimization (PSO) [25], Multi-Swarm Optimization (MSO) [26], River Formation Dynamics (RFD) [27], Self-propelled particles (SPP) [28], and Stochastic Diffusion Search (SDS) [29], [30]. These algorithms can be applied to flocking behavior in discrete surroundings [32] and to solve mazes. E.g. in [31], ACO is deployed to unknown mazes. However, many of these algorithms have some centralized elements and rely on empirical metaheuristics. Unlike most of these algorithms, this paper presents an SI approach with rigorous mathematical base that efficiently addresses the problem of steering a swarm of agents through an unknown maze to an unknown goal using only local information exchange.

The fundamental maze exploration algorithm is random search. In many cases this is the only available exploration method. Other maze exploration algorithms are generally deterministic in nature. The wall follower method [34], [35] which coarsely corresponds to a depth-first search strategy, works well with 2D perfect mazes [33], [34], [35], i.e. mazes that do not contain loops and thus form a tree when

M. Aurangzeb and F. L. Lewis are with the University of Texas at Arlington Research Institute (UTARI), 7300 Jack Newell Blvd. S., Fort Worth, TX 76118 USA (phone/fax: +817-272-5938; e-mail: {aurangze, lewis}@uta.edu). M. Huber is with the Department of Computer Science and Engineering, The University of Texas at Arlington, TX; email: (huber@cse.uta.edu).

*This work was supported by the National Science Foundation ECCS-1128050, the Army Research Office W91NF-05-1-0314, the Air Force Office of Scientific Research FA9550-09-1-0278, and China NNSF 61120106011.

represented as a graph [36], but does not work for imperfect mazes or to find a goal within the maze [33], [34]. An improvement to the wall follower algorithm is the Pledge Algorithm [34], [35], [1] which works with the help of a compass. Another extension is Trémaux's algorithm which is a sure algorithm for solving a maze that works where passages are well-defined and where there is a provision to draw lines on the floor [34]; it is based on bidirectional double tracing. Similarly, there are some other algorithms to explore a maze given complete knowledge of the maze [34], [35]. However, these algorithms generally rely on geometric properties of the maze or on extensive information storage and are thus often not applicable to general mazes in unknown spaces.

In this paper a randomized approach is presented to steer a swarm of agents out of any type of unknown maze. The approach uses distributed computations and principles of RL [10], [37], [38] to achieve the goal and improve on random and metaheuristic-based search. RL is a type of real-time machine learning which refers to modifying one's actions or control policies based on learning from one's experience. It is inspired by learning mechanisms that occur in nature, where living beings modify their control actions based on indirect, potentially sparse feedback [10], [37], [38]. In a distributed environment the scope for RL is wide. In this paper, the agents not only learn from their own experiences, but also from the experiences of their peers.

In this paper the structure of the maze is unknown to the agents and the goal is positioned at some anonymous location. A location is defined as an intersection point where a routing decision is needed. The maze is represented as a static graph [36] with locations in the maze taken as nodes and adjacency of two locations represented as an edge between the corresponding nodes in the graph. Agents here do not possess any means of communicating directly with each other and the only way information can be exchanged is through limited information left at the nodes of the graph. The swarm of agents is sent into the maze at a prescribed location and all of these agents are free to move from one node to the other in the graph along its edges, seeking for the goal. As these agents move they jointly develop a distributed database by updating information at memory nodes placed at explored nodes of the graph based on their own local experiences. The memory node network built in this form helps an agent to intelligently choose an edge from one node to another, using principles of RL [10], [37], [38]. The algorithm of this paper is compared to ACO in Section III. Simulation results show that the algorithm in this paper outperforms ACO.

The contribution of this paper is to provide a rigorous theoretical framework for an intelligent distributed search of a maze by a swarm of agents. In Section II, a collaborative RL based routing algorithm for path finding in a maze is presented and two theorems are derived that show the soundness of the learning and exploration approach and illustrate the appropriateness and sufficiency of the local information stored at the maze nodes by the proposed RL approach. Based on this basic RL framework, a routing scheme is presented in Section III. Simulation results are presented in Section IV and show the superiority of the proposed scheme over random search and the search based on ACO and a combination of both of them. These simulation results also show that the swarm of agents achieves the goal as a swarm and not as separate agents. Conclusions are presented in the Section V.

II. Formulation, Routing Results on finite Graphs, and Data Structures

This section formally defines the problem of routing in an unknown maze used in this paper by explaining the system architecture and information structures on the agent and in the memory nodes. For this, a graph theoretic formulation is used [36]. Two theorems concerning routing in finite mazes provide the basis for the data structures and show the soundness of the collaborative reinforcement learning approach.

In the problem definition used here, a swarm of agents is sent into an unknown finite maze from a single location, and the agents are required to reach an unknown goal location in the maze by using only information available locally to each agent. To do so, on visiting a node, an agent has to make a routing decision about which edge to follow when leaving the node. In this, it is desired to minimize information storage and the number of data exchanges while significantly improving upon the performance of random exploration by each agent.

A. Representing a Maze as a Graph

To formalize the routing problem and objectives, graph theory [36] is used. The unknown maze is represented as an unknown undirected graph $G(V,E)$, where vertices V correspond to the set of locations in the maze and $|G|=|V|$ is the number of locations in G. A location or node is defined as an intersection in the maze where a route decision is needed. There is an edge $\{i,j\} \in E$ if and only if the locations $i \in V$ and $j \in V$ are adjacent in the maze. The degree d_i of node i is the number of edges incident on node i. A swarm of N agents is sent into the graph G at some starting node $1 \in V$ and the agents are required to reach a goal located at some unknown node $g \in V$. Each agent must make routing decisions using only information available locally. It is assumed that at least one path exists from the starting node $1 \in V$ to the goal $g \in V$. A path from node $1 \in V$ to the goal $g \in V$ is defined as a finite sequence of distinct vertices $\{1 = v_1, v_2, ..., v_n = g\}$ with an edge existing in G between each pair of consecutive vertices in the sequence. The goal is reachable from node i if there exists a path from node i to the goal.

Memory Nodes and Explored Subgraph

Any agent arriving at an unvisited node for the first time places a memory node with a memory at that node, which is then termed visited. This produces a growing network of visited nodes V^* that forms a graph $G^*(V^*, E^*)$. Since G^* is grown by the agents starting from the same initial point $1 \in V$ and by traversing paths on G, $G^* \subseteq G$ is a connected subgraph of G. Each node in G^* refers to the

corresponding node in G and G^* is referred to as the explored subgraph of G.

B. Formal Routing Results for Finite Mazes

Here, two theorems are presented which show that certain local information stored at each node $i \in V^*$ is sufficient for routing in an unknown maze and yields performance far better than random exploration of the maze by the agents. This information at node i includes the number of agent traversals, $N_{ij}(t)$, of an edge $\{i, j\}$ prior to time t, and the number of times, $\eta_{ij}(t)$, that agents that traversed the edge have subsequently returned to node i prior to t. Accumulation of this minimal information, which is accelerated as the number of agents in the maze becomes large, provides routing information that allows the agents to reach the unknown goal with probability tending to 1. This makes the distributed, intelligent pursuit of the unknown goal by each agent possible using only information related to $N_{ij}(t)$ and $\eta_{ij}(t)$, stored locally at the nodes and agents.

To demonstrate this, the first theorem shows that in any finite graph without non-traversable edges, if $N_{ij}(t)$ becomes large while the number of agents $\eta_{ij}(t)$ returning to node i is equal to zero, then the probability that the goal is reachable from node i through a path containing edge $\{i, j\}$ tends to 1.

Theorem 1. Let $N_{ij}(t)$ be the number of agents who have passed node i along an edge $\{i, j\}$ prior to time t. Assume that $\eta_{ij}(t)$, the number of agents who have returned to node i subsequently, is equal to 0, and that all edges are traversable and have non-zero probabilities of being chosen by the agents. Then, as $N_{ij}(t)$ and t increase, the probability that there exists a path containing $\{i, j\}$ from node i to the goal tends to 1.

Proof: The maze is represented as a connected graph G. Consider a node i from where $N_{ij}(t)$ agents have gone along an edge $\{i, j\}$ prior to time t. Let us assume that there is no path leading to the goal from node i containing edge $\{i, j\}$. Split each edge of the graph into a pair of directed edges and consider the graph as a Markov chain. The chain does not have self-loops at any node other than the goal which is an absorbing node. Let the probability of going along any edge be larger than 0. Then there is a nonzero probability p that an agent will come back to node i in finite time t. Thus there is some nonzero average rate of return, λ, of the agents back to node i. As the number of agents $N_{ij}(t)$ increases the probability that k agents will come back in time t, given that there is no path leading to the goal from node i containing edge $\{i, j\}$ is given by the Poisson distribution [39].

$$p(k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}; k = 0,1,2,... \quad (1)$$

From (1), for $k = 0$ $p(0) = e^{-\lambda t}$, which implies that

$$P(k = 0 \mid \neg s) = e^{-\lambda t} \quad (2)$$

where S=There exist a path from node i containing $\{i, j\}$ leading to the goal. This leads to

$$P(\neg s \mid k = 0)\frac{P(k = 0)}{P(\neg s)} = e^{-\lambda t} \quad (3)$$

or

$$P(s \mid k = 0) = 1 - \frac{P(\neg s)}{P(k = 0)} e^{-\lambda t} \quad (4)$$

In (4), as time t increases, $P(k = 0)$ reaches 1 (due to $\eta_{ij}(t)$ = 0) and $P(\neg s)$ is fixed. Thus $P(s \mid k = 0)$ reaches 1. ■

The second result shows that if $\eta_{ij}(t) \neq 0$ at a node i, then the steady state probability of finding the goal is maximized by following an edge $\{i, j\}$ with the minimum ratio $\frac{\eta_{ij}(t)}{N_{ij}(t)}$.

Theorem 2. Let $N_{ij}(t)$ be the number of agents who have passed an intermediate node i along edge $\{i, j\}$ prior to time t, and $\eta_{ij}(t)$ the number of agents out of $N_{ij}(t)$ who have returned to node i. Then as $N_{ij}(t)$ increases, the steady state probability $p_s = P(s)$ that there exists a path containing $\{i, j\}$ from node i to the goal is maximum for the edge $\{i, j\}$ for which the ratio $\frac{\eta_{ij}(t)}{N_{ij}(t)}$ is minimum.

Proof: For an agent who is present at an intermediate node i at some time t, the whole maze can be viewed as d systems reachable from it where d is the degree of node i. This view is shown in Fig. 1, where the systems are states of a Markov chain, numbered 1… d, and the state g represents the goal node g.

Supposing that these systems are randomly selected by the agent at node i, let p_j be the probability of return of an agent from system j to node i. Also, let p_{S_j} be the probability of going from the system j to the goal node, g, which is an absorbing node. The probability that an agent within system j remains within this system is p_{T_j} .Also, let p'_j be the probability to select a particular system from node i so that the probability $p_S = P(s)$ of reaching the goal is maximized.

Using conditioning this probability can be written as

$$p_s = \sum_{j=1}^{d} P(s \mid j) P_{ij} \quad (5)$$

where

$$P(s \mid j) = p_{sj}, P_{ij} = p'_j \quad (6)$$

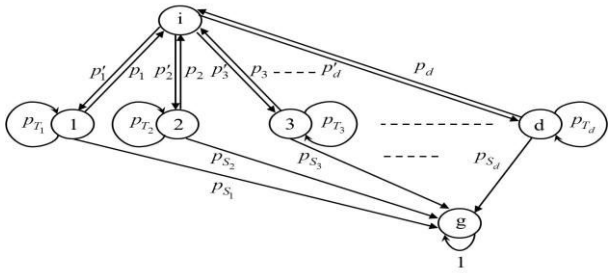

Fig. 1: Visualization of maze as d systems

Substituting the values from (6) into (5), yields

$$p_s = \sum_{j=1}^{d} p_{s_j} p'_j \tag{7}$$

For each system $j = 1, 2, ..., d$,

$$p_{sj} + p_{T_j} + p_j = 1 \tag{8}$$

Substituting the value of p_{sj} from (8) into (7) yields

$$p_s = \sum_{j=1}^{d} (1 - p_{T_j} - p_j) p'_j \tag{9}$$

or

$$p_S = 1 - \sum_{j=1}^{d} p_{T_j} p'_j - \sum_{j=1}^{d} p_j p'_j \tag{10}$$

Given that all edges have non-zero probabilities to be taken by the agent, the steady state transition probability p_{T_j} of keeping an agent within the same system is 0. The problem of maximizing the probability p_s of an agent to reach the goal from the node i is thus transformed into the problem of minimizing $\sum_{j=1}^{d} p_j p'_j$ on the right hand side of (10). Since

$$\sum_{j=1}^{d} p'_j = 1 \tag{11}$$

the summation $\sum_{j=1}^{d} p_j p'_j$ is convex and has the minimum

$$\min \sum_{j=1}^{d} p_j p'_j = p_{j_0} \tag{12}$$

where $p_{j_0} = \min(p_j)$, when $p'_j = 0 \forall j \neq j_0 \wedge p'_{j_0} = 1$. This means that p_s is maximized if the edge with minimum probability of an agent coming back to node i is taken at time t. Under the given assumption of equal complexity of the systems, and given $N_{ij}(t) \forall j : \{i, j\} \in E$ is large enough, as the time passes, the ratio $\frac{\eta_{ij}(t)}{N_{ij}(t)}$ reaches the steady state probability of return of an agent to node i. Therefore, the greater the value of $\frac{\eta_{ij}(t)}{N_{ij}(t)}$, the smaller are the chances of finding the goal while going along the edge $\{i, j\}$. ■

C. Reinforcement Learning Definition

Theorem 1 and Theorem 2 in the previous section showed the importance in routing of $\frac{\eta_{ij}(t)}{N_{ij}(t)}$, the ratio of the number of returns to node i after traveling along edge $\{i, j\}$ and the number of times that an agent traveled along that edge. This section capitalizes on these results to derive a reinforcement learning (RL) definition that will be used to build the collaborative maze traversal algorithm proposed here.

To formulate the RL problem, the maze task can be treated as a Markov Decision Problem with the nodes of the maze graph representing states, edge choices being the actions of the agent, and transitions being defined deterministically through the edges of the graph. Based on the results of Theorems 1 and 2, a reward function for the agent m when traversing edge {i,j} in state i can be defined as

$$r_j(i) = \begin{cases} -1 & \text{if } \{i, j\} \text{ was previously traversed} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

This reward function basically produces a negative reward whenever the agent has looped back and no reward otherwise.

Using this reward function and using an average reward utility function, the agent can estimate the utility of traveling along an edge in the form of a Q-function where

$$Q_m(\{i, j\}) = E\left[r_j(i) \right] \tag{14}$$

Since all agents have the same objective and the same action space (and thus the same utility function), this utility function can be maintained collectively by accumulating all the rewards of the agents at the nodes, leading to a collective utility function:

$$Q^{(t)}(\{i, j\}) = E\left[r_j(i) \right] = \frac{-\eta_{ij}(t)}{N_{ij}(t)} \tag{15}$$

From this it can be seen that based on Theorem 2, maximizing this utility by selecting edges that have maximum utility is equal to maximizing the likelihood of moving along edges that lead to the goal, illustrating the soundness of the proposed RL scheme.

D. Data Structures of Agents and Nodes

To maintain and efficiently update the reward and utility data used by the reinforcement learning agent, utility data has to be maintained at the nodes of the network and sufficient information to determine rewards have to be maintained by the agents. The data structures presented here contain parameters used to represent these values and to allow for routing decisions by the agents for maze exploration. Upon its arrival at a memory node, an agent exchanges data with the memory node, and based on the principles of RL presented in the previous section and supported by Theorem 1 and 2 updates the data structures and makes a decision about which edge to follow on leaving the node.

Data Maintained by Visited Nodes

Each memory node $i \in V^*$ at the visited locations maintains a time-varying matrix $R_i(t)$ of size $d_i \times 2$, where d_i is the degree of the node i. This matrix is given by

$$R_i(t) = [\underline{N}_i(t) \;\; \underline{\gamma}_i(t)] \tag{16}$$

Here, $\underline{N}_i(t)$ and $\underline{\gamma}_i(t)$ are vectors of length d_i, each having one entry corresponding to each edge incident on node i.

The vector $\underline{N}_i(t)$ has the j-th entry equal to the number of agents $N_{ij}(t)$ who have travelled along the edge $\{i, j\}$ prior to time t. The j-th entry in the vector $\underline{\gamma}_i(t)$ is the negative of the number of agents, out of $N_{ij}(t)$, who have returned to it. That is to say, $\gamma_{ij}(t) = -\eta_{ij}(t)$. If none of the agents who have gone along edge $\{i, j\}$ have returned to node i, then the j-th entry in vector $\underline{\gamma}_i(t)$ is equal to zero.

According to Theorems 1 and 2, routing decisions that select edge $\{i, j\}$ on leaving node i are beneficial if the j-th entry in the vector $\underline{\gamma}_i(t)$ is equal to zero and detrimental as the j-th entry in the vector $\underline{\gamma}_i(t)$ becomes more negative. The vector $\underline{\gamma}_i(t)$ here represents accumulated rewards, with element j of $\underline{\gamma}_i(t)$ reflecting the rewards of all the agents when traveling along $\{i, j\}$ accumulated up to time t.

Data Maintained by Agents

Nodes $i \in V^*$ in the explored graph G^* are identified by EUI-64 IDs [40]. To compute its rewards and to be able to update the information at the nodes, every agent $m \in N$ keeps track of the edges it has visited in the form of a finite list L_m of ordered pairs of EUI-64 IDs of nodes it has visited. An entry of L_m, $L_m(l)$, is equal to $(i(m,l), i(m,l+1))$, the edge of V^* previously traversed by agent m.

E. Node and Agent Data Updates

In this section the computational details for updating the system data parameters, and thus the collective utility function of the agents used for the RL algorithm, are discussed. The node parameters are given in (13) and the agent parameters are the list L_m of edges it has visited. These data parameters are updated based on only the local information passed between an agent and the node at which it currently resides. Based on the data updates, an agent makes a routing decision by selecting an edge along which to proceed on leaving the node.

The system parameter update mechanism works through update algorithms operating at the agents and at the nodes. Suppose that at time t there are n agents $m_1, m_2, \ldots m_n$ at some node $i(t) \in G^* : i(t) \neq g$. All these agents receive the EUI-64 ID of the node, which is also referred to as $i(t)$, and the parameter matrix $R_i(t)$ from the node $i(t)$.

The following update algorithm is performed by an agent m_k who is at node $i(t)$ at time t. In the following algorithm $\underline{r}_{m_k}(t)$ is a vector updated by the agent m_k who is at node $i(t)$ at some time t and of length equal to the degree of $i(t)$. The algorithm has three steps: data update, routing decision, and communicate to node.

Algorithm 1- Agent Update and Routing Decision

1) Data Update: Define,

$$\underline{r}_{m_k}(t) : r_{m_k j}(t) = 0 \;\; \forall j \in V^* : \{i(t), j\} \in E^*$$

Search L_{m_n} for $i(t)$

$$[r_{m_k}(t)]_{L_{m_k}(l+1)} = -1 \;:\; l < t \text{ and:}$$

$$L_{m_k}(l)(\text{first coordinate}) == i(t)$$

The update specified here implies that at most one component of $\underline{r}_{m_k}(t)$ is nonzero, which is -1. This nonzero component corresponds to the edge the agent used when it visited node i the previous time. The agent update to its RL vector is

$$\underline{\gamma}_i(t) = \underline{\gamma}_i(t) + \underline{r}_{m_k}(t) \tag{17}$$

2) Decision and Further Update: Select edge $\{i, j\}$ to follow on leaving node i based on the ratio of the accumulated reward $\gamma_{ij}(t)$ and the number of agents $N_{ij}(t)$ (i.e. based on the average reward utility Q-value in (12)). This decision algorithm is given in Section III. Moreover, if the agent reaches a previously visited node, it updates the second coordinate of the corresponding entry in L_{m_n} to the one it is going to visit; otherwise it will create a new entry in L_{m_n}.

3) Communicate: Form $\underline{e}_{m_k}(t)$, a vector of length $d_{i(t)}$ consisting of zero entries, with only one entry equal to 1 at the position corresponding to the edge $\{i, j\}$ selected.

Communicate $\underline{e}_{m_k}(t)$ and $\underline{r}_{m_k}(t)$ to $i(t)$. ■

In Algorithm 1, Step 1, an agent gives a negative reward of -1 to the edge incident at $i(t)$ which it followed when it visited the node the previous time. This setback is communicated to the node $i(t)$ in the form of vector $\underline{r}_{m_k}(t)$. Also, the agent communicates the edge taken at time t in the form of $\underline{e}_{m_k}(t)$.

Upon receiving this information from the visiting agents, node $i(t)$ adds all rewards to $\underline{\gamma}_i(t)$ to get the new RL setback vector $\underline{\gamma}_i(t+1)$. The vector $\underline{N}_i(t)$ is also updated according to the decisions of the agents to compute $\underline{N}_i(t+1)$. These actions are summarized in the following algorithm.

Algorithm 2: Node Update Algorithm

Update RL gain and number of agents visiting node i.

$$\underline{\gamma}_i(t+1) = \underline{\gamma}_i(t) + \sum_{k=1}^{n} \underline{r}_{m_k}(t) \quad (18)$$

$$\underline{N}_i(t+1) = \underline{N}_i(t) + \sum_{k=1}^{n} \underline{e}_{m_k}(t) \quad (19)\blacksquare$$

Equations (15) and (16) elaborate the updates of $R_i(t)$ of node i on the basis of the RL feedbacks given by the visiting agents to the incident edges and the edges taken by agents at time t.

III. Routing in a Maze Using Reinforcement Learning

This section describes the routing decision algorithm used by each agent in deciding which edge to follow when leaving a node i at time t. This corresponds to Step 2 of Algorithm 1, the agent update and routing decision algorithm. This algorithm uses exploitation of the data about the explored graph to reach the goal. It also uses exploration to obtain data about the unexplored portion of the maze. The importance of balancing exploitation and exploration is well known in reinforcement learning [37].

A. Exploitation of Information for Intelligent Routing

Routing in an unknown maze requires a balance between using or exploiting available information to select the route most likely to reach the goal, and exploring the unknown portion of the maze. This balance has been formalized as the 'exploitation vs. exploration' dilemma [37].

According to Theorem 1 and 2, the data contained in vectors $R_i(t) = [\underline{N}_i(t) \;\; \underline{\gamma}_i(t)]$ contains information that can be used to make intelligent routing decisions. Specifically, as the j-th entry $N_{ij}(t)$ of $\underline{N}_i(t)$ becomes large while the absolute value of the j-th entry $\gamma_{ij}(t) = -\eta_{ij}(t)$ of $\underline{\gamma}_i(t)$ is small, the probability is high that the goal node is reachable by following edge $\{i, j\}$, which corresponds to exploiting the available information. This RL based system proposes for an agent located at node $i(t)$ to randomly take one of the edges $\{i(t), j^*\}$ that maximize the collaborative utility, $\gamma_{ij}(t) / N_{ij}(t)$.

Taking the edge with the highest Q-value (or equivalently the lowest ratio $\frac{\eta_{ij}(t)}{N_{ij}(t)}$ is a greedy form of routing decision. Always following a greedy action, however, is not a good strategy since there should always be a finite probability that the agent will proceed along any edge in order to maintain a finite probability of finding other, better paths. Allowing a finite probability, say ε, of following a different edge is termed an ε-greedy exploration policy and it has been shown that ε-greedy actions can preserve a good balance between exploitation and exploration [37]. Here, it is proposed that an agent follows the edge suggested by the RL system, i.e. the one with the highest Q-value, with probability $(1-\varepsilon) \approx 1$, according to the ε–greedy approach [37]. This policy addresses the fact that in a graph with multiple paths to the goal it may happen that a suboptimal path is found prior to the complete exploration of the graph and thus, sufficient exploration is required to fully explore the unknown graph. The following mechanism is thus proposed for an agent to make a routing decision in Step 2 of Algorithm 1.

The edge $\{i(t), j\}$ to be taken by agents leaving node i is selected with the following probability distribution.

$$\begin{aligned} P(\{i(t), j^*\}) &= 1-\varepsilon \\ P(\{i(t), j\}, j \neq j^*) &= \tfrac{\varepsilon}{d_i - 1} \end{aligned} \quad (20)$$

where $\{i(t), j^*\}$ is the edge suggested by the RL based system.

B. Comparison with ACO

Ant Colony Optimization is a graph search algorithm initially proposed by M. Dorigo [1]. This is a randomized swarm steering algorithm to find a path within a graph. The algorithm is inspired by the mechanism of optimization used by ants in their colonies to find a minimum path to their food [41], [42]. The ant colony algorithm proposed by Dorigo, explores an unknown graph by multiple agents in a distributed manner by using a centralized database [43]. The algorithm implements two local decision policies: Trail and attractiveness [1]. Further detailed about ACO can be seen at [46].

IV. Simulation

This section presents the simulation setup and results for agents exploring a maze using the methods proposed in this paper. The RL-based search algorithm of Sections II and III is simulated. The results are compared to those for agents searching the maze with random search, with ACO and with a hybrid ACO and random search system. It is shown that the proposed RL approach significantly speeds up the search.

To measure effectiveness, an additional performance measure is defined and used to evaluate the approaches.

Definition (Performance Measure): The size of a 2D maze is defined as $M = LW$, where L is the length of the maze and W its width. The Performance Measure (PM) of a search is defined as the ratio of the mean time T taken by the first agent to reach the goal to the maze size M. That is to say.

$$PM = \tfrac{T}{M} \quad (21)\blacksquare$$

It is desired for this performance measure to be small.

A. Simulation Results

For the purpose of simulation several maze structures were generated using the Matlab Maze Toolbox [45]. Outcomes of typical simulations are shown in Fig. 2. These results are for mazes of size 5×5 with 10 agents exploring them. All agents start at location $(1,1)$ and their unknown goal is present at location $(5,5)$. These two locations are numbered as maze cell 1 and 25 respectively along the bottom left axes of Fig. 2. Fig. 2 (a) shows the case when the agents explore the maze through random search and without any intelligent decision-making. It is observed that the first agent reaches the goal after 45 time steps. This is noted by observing when the first point appears in the upper left plane of the 3D graph. At the end of the simulation spanning 100 time steps, only 3 agents have reached the goal using random search.

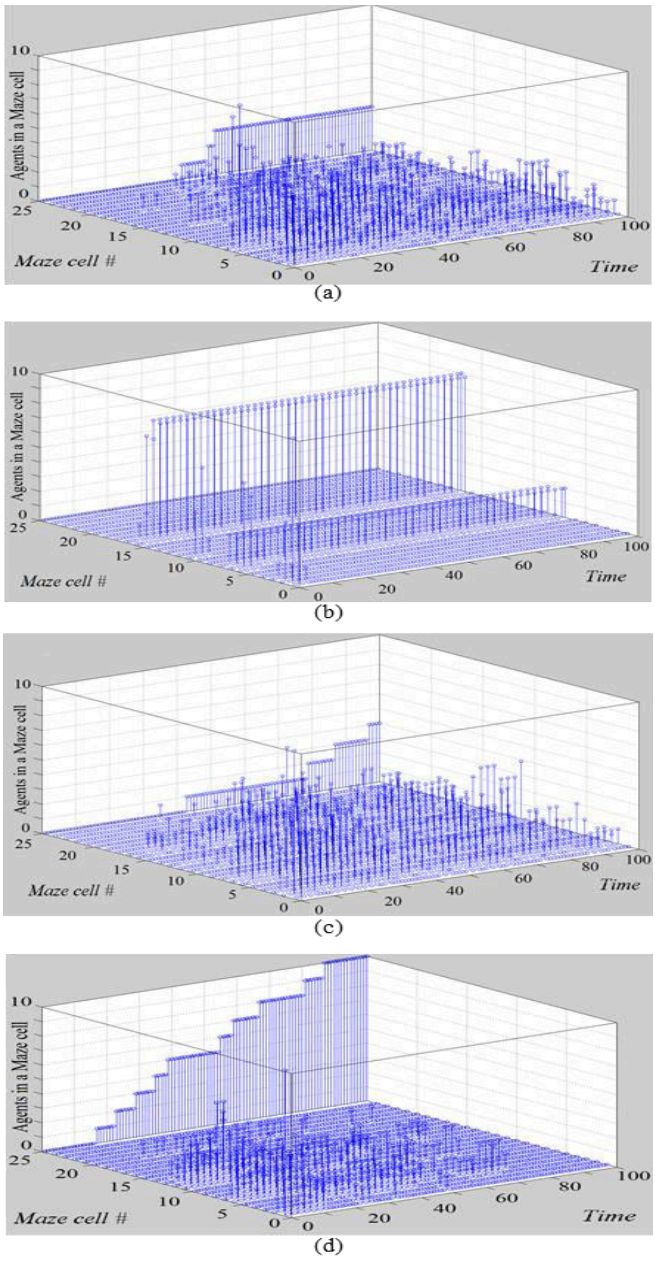

Fig. 2: Simulation results of Maze Exploration, (a) Random Search (b) ACO (c) ACO and Random search (d) RL Based System

In [31], ACO is implemented to explore a penalty maze. In this paper ACO is implemented for the maze as mentioned above. Fig. 2 (b) shows the case when the agents explore the maze through a local version of ACO without its global part of Daemon Action so that the comparison between two approaches can be made fairly. It is observed that the agents mostly do not reach the goal. This happens since ACO takes into account the number of agents that went through a path and not the number of agents who visit the same node again. In this way the agents get into stagnation of the search by visiting the same set of nodes again and again. The ACO can also be implemented along with random search. In this hybrid system there are two types of agents, one following ACO and others following the random search. The agents following random search also deposit pheromones as the other agents do. Fig. 2 (c) shows a typical case of maze search by the hybrid system. In case of the hybrid system there is some improvement as compared to simple random search and ACO since now agents have pheromone deposition from random agents to help other agents to get out of stagnation. This is noted by observing when the first point appears in the upper left plane of the 3D graph. After 100 time steps, 4 agents have reached the goal using hybrid search.

Fig. 2 (d) shows a typical case when the agents are equipped with the RL system based on Theorems 1 and 2 as developed in Sections II and III. It is observed that the first agent reaches the goal within 10 time steps, the minimum time for this size of maze, while the next two reach the goal within 16 time steps. At the end of the simulation, all agents have reached the goal. Note that, as time passes, more and more information is stored in the maze in the form of memory nodes dropped by the agents, thus routing successively improves.

In the second part of the simulation, results were obtained for five different cases with different maze sizes M. Each of these simulations was run M times and the mean time required for the first agent to reach the goal, T is calculated in each case. The performance measures PM for these maze sizes and for various numbers of agents are calculated. Here, it is observed that the performance measure PM for the RL-equipped agents, is almost twice as small as that for agents using the random search and hybrid search. The PM of a system purely based on ACO is even higher than random search-based and hybrid system. It is observed that the best performance measure using random search and hybrid search is approximately 1, while the performance measure using the RL-based routing system is approximately 0.5. Moreover, the graphs show that the RP-based system scales much better as the maze size increases. Figures are excluded due to page limit, and are available at [46].

V. Conclusion

This paper establishes a strategy for steering a swarm of autonomous agents out of an unknown maze to some goal located at an unknown location. The strategy is based on principles of reinforcement learning. Two theorems show that simple rules of learning from past experience make the maze exploration significantly faster than standard random search. Based on these results, a ε–greedy RL routing algorithm, that uses only local information exchanges, is developed in Section III to balance exploitation and exploration of the unknown maze. Simulation results show that maze exploration using minimal information and based on RL is far superior to exploration using random search, search based on ACO and search based on hybrid ACO and random search.

References

[1] H. Abelson and A. A. diSessa, Turtle Geometry, MIT Press. Cambridge, 1980.

[2] M. Jansen, M. Oelinger, K. Hoeksema, and U. Hoppe, "An Interactive Maze Scenario with Physical Robots and Other Smart Devices," in Proceedings of the 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education, 2004.

[3] Y. Kobayashi, Y. Wada, and T. Kiguchi, "Knowledge Representation and Utilization for Optimal Route Search," IEEE Transactions on Systems, Man, and Cybernetics-Part B, Cybernetics Vol. SMC-16, No. 3, May/June 1986.

[4] S. X. Yang, and M. Meng, "Neural Network Approaches to Dynamic Collision-Free Trajectory Generation," IEEE Transactions on Systems, Man, and Cybernetics-Part B, Cybernetics Vol. 31, No. 3, June 2001.

[5] E. P. Silva Jr., M. A. P. Idiart, M. Trevisan, and P. M. Engel, "Autonomous Learning Architecture for Environmental Mapping," Journal of Intelligent and Robotic Systems 39: 243-263, 2004.

[6] N. Roy, and G. Dudek, "Collaborative Robot Exploration and Rendezvous: Algorithms, Performance Bounds and Observations," Autonomous Robots 11, 117-136, 2001.

[7] L. Moremo, J. M. Armingol, S. Garrido, A. de La Escalera, and M. A. Salichs, "A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors," Journal of Intelligent and Robotic Systems 34: 135- 154, 2002.

[8] Z. Cai and Z. Peng, "Cooperative Co-evolutionary Adaptive Genetic Algorithm in Path Planning of Cooperative Multi-Mobile Robot Systems," Journal of Intelligent and Robotic Systems 33: 61-71, 2002.

[9] B. F. Goldiez, A. M. Ahmad and P. A. Hancock, "Effects of Augmented Reality Display Settings on Human Wayfinding Performance," IEEE Transactions on Systems, Man, and Cybernetics-Part C, Reviews Vol. 37, No. 5, September 2007.

[10] M. A. Wiering and H. van Hasselt, "Ensemble Algorithms in Reinforcement Learning," IEEE Transactions on Systems, Man, and Cybernetics-Part B, Cybernetics Vol. 38, No. 4, August 2008.

[11] A. M. Whitbrook, U. Aickelin, and J. M. Garibaldi, "Idiotypic Immune Networks in Mobile-Robot Control," IEEE Transactions on Systems, Man, and Cybernetics-Part B, Cybernetics Vol. 37, No. 6, Dec. 2007.

[12] J. Suzuki and Y. Yamamoto, "Building an artificial immune network for decentralized policy negotiation in a communication end system," in Proc. 4th World Conf. SCI, Orlando, FL, July 2000.

[13] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm intelligence from natural to artificial systems, Oxford University Press, Inc. New York 1999

[14] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," Artificial Intelligence Review Vol. 31, Numbers 1-4, 61-85, DOI: 10.1007/s10462-009-9127-4, 2009.

[15] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992.

[16] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[17] D. Dasgupta, Z. Ji, and F. Gonzalez, "Artificial immune system (AIS) research in the last five years," in Proc Congress on Evolutionary Computation, CEC '03, 2003.

[18] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," Springer Acta Mechanica, 2010.

[19] X.-S. Yang; and S. Deb, "Cuckoo search via Lévy flights," World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214, 2009.

[20] X.-S. Yang and S. Deb, "Engineering optimization by cuckoo search," Int. J. Mathematical Modeling and Numerical Optimizations Vol. 1, No. 4, 330-343, 2010.

[21] X. F. Yang, "Firefly algorithms for multimodal optimization. Stochastic Algorithms," Foundations and Applications, SAGA 2009. Lecture Notes in Computer Sciences. 5792. pp. 169–178, 2009.

[22] S. M. Farahani, A. A. Abshouri, B. Nasiri, and M. R. Meybodi, "Some hybrid models to improve firefly algorithm performance," Int. J. Artificial Intelligence, Vol. 8 S(12), 97-117, 2012

[23] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "A Gravitational Search Algorithm," Journal of Information Science Vol 179, Issue 13, pp 2232-2248, 2009.

[24] .H. S. Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," International Journal of Bio-Inspired Computation 1 (1/2): 71–79, 2009.

[25] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization," Proceedings of IEEE International Conference on Neural Networks, IV. pp. 1942–1948, 1995.

[26] C. Li and S. Yang, "Fast Multi-Swarm Optimization for Dynamic Optimization Problems," in Proc. Fourth International Conference on Natural Computation, ICNC '08, 2008.

[27] P. Rabanal, I. Rodríguez and F. Rubio, Using River Formation Dynamics to Design Heuristic Algorithms, Lecture Notes in Computer Science, 2007, Volume 4618, 2007.

[28] P. Degond, and S. Motsch, "Continuum Limit of Self-driven Particles with Orientation Interaction," SIAM Journal of Applied Math, arXiv: 0710.0293, 2007.

[29] J.M. Bishop, "Stochastic Searching Networks," in Proc. 1st IEE Conf. on Artificial Neural Networks, pp 329–331, London, 1989.

[30] P.D. Beattie, and J.M. Bishop, "Self-Localization in the 'Scenario' Autonomous Wheelchair,". Journal of Intelligent and Robotic Systems 22, pp 255–267, Kluwer Academic Publishers, 1998.

[31] D. Jones, D.A. Harrison, and A.J. Davies, "Experience Outweighs Intelligence: an investigation into the use of Ant Colony Systems for Maze Solving," in Proceedings of the ACIS Fourth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'03), 2003.

[32] "Maze. available online at http://en.wikipedia.org/wiki/Maze

[33] W. D. Pullen. "Maze classification" Internet: http://www.astrolog.org/labyrnth/algrithm.htm#solve, January 24, 2011 [February 06, 2013]

[34] "Maze solving algorithm" Internet: http://en.wikipedia.org/wiki/Maze_solving_algorithm [February 06, 2013]

[35] N. S. V. Rao, S. Kareti, and W. Shi, Robot Navigation in Unknown Terrains: Introductory Survey of Non-Heuristic Algorithms, Report prepared by Oak Ridge National Laboratory, July 1993.

[36] R. Diestel, Graph Theory, Fourth Edition. Springer Heidelberg Dordrecht London New York, 2010.

[37] R. S. Sutton and A.G. Barto, Reinforcement Learning– An Introduction, MIT Press, Cambridge, Massachusetts, 1998.

[38] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, Vol. 4 pp. 237-285 1996.

[39] S.D. Poisson, "Research on the Probability of Judgments in Criminal and Civil Matters," Elibron Classics, 1838.

[40] IEEE: Guidelines for 64 bits global Identifier. March, 1997.

[41] S. Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels, "Self-organized shortcuts in the Argentine ant," Nature wissenschaften, Vol. 76, pages 579-581, 1989

[42] J.-L. Deneubourg, S. Aron, S. Goss et J.-M. Pasteels, "The self-organizing exploratory pattern of the Argentine ant," Journal of Insect Behavior, Vol. 3, page 159, 1990

[43] T. Stutzle, and H. Hoos, "Improvements on the Ant System: Introducing the MAX-MIN Ant System," I R.F. Albrecht G.D. Smith, N.C. Steele (ed), Artificial Neural Networks and Genetic Algorithms. Springer Verlag, Wien New York, Pages 245-249, 1998.

[44] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Trans. Systems Man Cybernet. B, 26 29–42, 1996.

[45] J. Kubica, "Toolbox to create a maze in MatLab" Internet: http://www.ri.cmu.edu/, 2003, [November 2012].

[46] Aurangzeb, M., "Internal structure and Dynamic Decisions for Coalitions on Graphs," Doctoral Thesis, Department of Electrical Engineering, University of Texas at Arlington. (In Preparation)