JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 7, December 2010

Multi-Agent Maze Exploration

Elad H. Kivelevitch* and Kelly Cohen†
University of Cincinnati, Cincinnati, Ohio, 45221

DOI: 10.2514/1.46304

Mazes have intrigued the human mind for thousands of years, and have been used to measure cognitive abilities of laboratory animals. In recent years, mazes have been used to examine the artificial intelligence of robots by observing their ability to traverse mazes using algorithm for maze exploration and exploitation. A simulation of a multi-agent system is used to demonstrate the benefits of utilizing a group of several robots in maze exploration. Using a behavioral algorithm based on Tarry's algorithm, it is shown that the group performance improves and becomes more robust as the number of robots increases. In addition, the amount of data transfer required for group coordination can be minimized to a small set of data items, which is independent of either the number of robots in the group or the maze size. As a result, the above multi-agent approach can be scaled up to mazes or groups of any size, as indicated by the results of the MATLAB-based simulation.

I. Introduction

MAZES have intrigued the human mind for thousands of years [1]. One-well known example is based on the interaction between Theseus and the Minotaur, which dwelt in the center of a maze. Although the story is mythological, a maze from the ancient Greek era can be found at the historical site of Knossos. Mazes have also been used for many years in scientific research where the capabilities of small laboratory animals were tested [2–4], and in recent years mazes have been used to research the artificial intelligence of robots by examining their ability to traverse unfamiliar mazes [5–7]. Maze exploration algorithms have been studied since in the second half of the nineteenth century [8] and they are closely related to Graph Theory, used both in mathematics and in computer science [9,10]. Usually these algorithms pertain to a single agent traversing a maze. However, multi-agent systems, especially systems of autonomous robots, are the focus of contemporary research in robots for various applications [11–13]. Moreover, as computers and robots technologies evolve, it is evident that robots, multi-robot systems in particular, will be developed and used autonomously in various situations: (a) searching and working in hazardous or poisonous environments, such as nuclear plants and waste sites, or burning buildings, (b) monotonous and tedious tasks, such as carrying loads around a plant or warehouse, and (c) operations where the risk and cost of sending humans are unacceptable, e.g., voyages to other planets in the solar system [8]. Thus, maze exploration algorithms need to be expanded to the multi-agent case.

The rest of this article is organized in the following way: first, the maze problem and types of mazes are introduced, followed by known maze exploration algorithms. Then, the notion of multi-agent systems is discussed along with some examples of such systems. In the next section, the model, which includes a problem statement, and the solution we used are presented. Next, the figures of merit used to measure the performance of our solution are defined, followed by results. Finally, conclusions and directions for future work are summarized.

Received 12 July 2009; accepted for publication 11 November 2010.

* Visiting Research Associate, Department of Aerospace Engineering and Engineering Mechanics. elad.kivelevitch.uc@gmail.com
† Associate Professor, Department of Aerospace Engineering and Engineering Mechanics, and AIAA Associate Fellow.

A. Mazes

Before we describe the problem at hand, let us first define the terms which will be used in this paper. A *maze* is a grid-like n-dimensional (usually two-dimensional area) space of any size, usually rectangular [14]. A *cell* is the elementary maze item. A cell is defined as a bounded n-dimensional space, usually a two-dimensional area, and is interpreted as a single location. The term *agent* is defined as an entity that is able to move within the maze. An agent can move in any arbitrary direction, provided that the space is not occupied by an *obstacle* or that the motion does not violate the obstacle. Thus, we can define two types of obstacles: an obstacle that occupies a cell completely, called a *wall*, and an obstacle that separates two adjacent cells, called a *partition* [14]. It is assumed that any case the obstacle prevents not only the agents motion, but also the agents sensing or the existence of other agents in the obstructed space. In the next sections, the classification of mazes according to their properties and some known algorithms for maze exploration are discussed.

1. Mazes Classification

In typical maze problems, an agent is initialized at a certain *start* location and is then required to achieve a certain *goal* location. The goal location can be inside the boundaries of the maze, for example, the location of a virtual food or a mythological Minotaur to be slain, or it can be an exit point from the maze. Maze problems can be classified according to the properties of the maze and the properties of the agents traversing it [14]. The maze *dimension* defines the number of dimensions the maze uses, for example, a planar maze uses two dimensions and a cubic maze uses three [15]. In general, the maze can use any number of dimensions, although it will be difficult for a human being to conceptualize more than three dimensions. The maze *size* is the number of cells that comprise the maze. The size is usually strongly related to the length of the path that connects the start and the goal positions, or *distance*. The distance is also determined by the obstacles in the maze, which can be described by either the absolute *number* of obstacles or, more significantly, the *density* of the obstacles, i.e. the ratio of the number of occupied cells to the maze size. A dense maze may increase the distance, but not necessarily the complexity of the maze. Specifically, a *labyrinth* is a maze in which only one tortuous path is possible, thus traversing the labyrinth from start to end may be lengthy, but not complicated.

The obstacles also provide more sensor information to the agents in the form of distinguishable shapes. For example, an empty maze will not provide any mapping information to the agent, unless the agent is capable of sensing and processing the entire space of the maze. A more complicated case is where the agent is unable to distinguish between repetitive obstacle patterns in the maze, a phenomenon called *aliasing* [14].

Obstacles also determine the maze topology [16]. A maze can be represented by a graph G, comprised of n nodes, which are the cells, and e edges, where each edge represents a passable path connecting a pair of two cells. In the general case, the graph may comprise s sub-graphs, where each sub-graph is not connected to the other graphs, thus making parts of the maze unreachable for the agent. If the graph is connected, there are two possible cases for a graph leading from the start point to the goal: only a single path connects the two points, or more than one path connects them. The former case can be represented by a *tree*, while the latter is simply a graph, see Fig. 1.

The a-priori knowledge that the agent has regarding the maze is also an important factor. It can range from no a-priori knowledge to complete a-priori knowledge. In the former case, the agent has to use its own sensory capabilities to study and map the environment. Once mapping is done, the agent can plan a path that will lead it from the start to the goal. If the agent has complete a-priori knowledge, then it can simply plan its path to the goal. However, a-priori knowledge is a necessary but insufficient condition to successful path planning, because not all mazes are *static*. Typically, there are three different causes to changes in the maze [14]:

1) Changes to the maze structure, i.e. partition or walls changing their location.
2) Changes in the location of the goal, e.g. the food is moved, or the location of the enemy, e.g. the location of the Minotaur.
3) Changes caused by the motion of other agents in the maze, in the case of a multi-agent group. When an agent occupies a cell we assume that another agent will not be able to occupy the same cell, thus the cell is temporarily blocked until it is vacated from the agent currently occupying it.

An agent ability to learn and interpret the entire state of the maze is a significant classification of the maze as well [17]. If the agent is capable of grasping the entire state of the maze, and for each sensory stimulus there is at least one reinforcement action the maze is considered as *Class 0*. On the other hand, if the agent is capable of obtaining

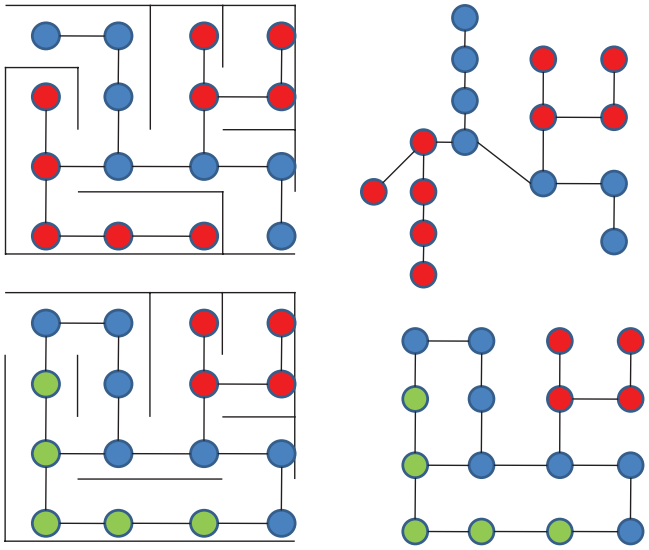

Fig. 1 Mazes represented as a tree (top) and as a graph (bottom).

the entire state of the maze, but only some sensory stimulus have at least one reinforcement action the maze is considered as *Class 1*. Finally, if the agent is incapable of obtaining the entire state of the maze, the maze is classified as *Class 2*. Obviously, Class 2 mazes are considered the most complicated mazes; however, if the agent uses history, it can eliminate at least some of the uncertainty of Class 2 mazes [17].

2. Algorithms for Maze Exploration

In a general maze exploration, there are two aims that the algorithm has to satisfy: finding a goal and then leading the agent out of the maze. Maze exploration algorithms have been studied since in the second half of the nineteenth century [8]. Maze exploration algorithms differ by the amount of a-priori knowledge given as their input. When the maze is assumed to be known in full prior to the start of the exploration, algorithm for path planning are usually based on geometry or graph theory. If the maze environment is not known, the agents have to explore it by themselves, sense the environment, and apply the newly acquired information for further path planning. Therefore, these algorithms are referred to as "online" algorithms [8], and are usually designed for an agent with vision or tactile sensors. In many cases, the algorithms depend on the agent ability to sense its environment. In the 1940s, Claude Shannon designed an algorithm that allowed his mechanical mouse to traverse a maze of 25 cells separated by thin walls in order to find "cheese" hidden in one of the cells [18]. The algorithm was relatively simple:

1) Enter a cell and mark the direction in which you entered.
2) Try to leave the cell in a direction of 90° (either to the left or to the right) to the direction you entered.
3) If the chosen direction is blocked by a wall, return to the center of the cell, mark the direction you tried to use, and return to the previous step.
4) Repeat until you get to the next cell.

Shannon's algorithm was demonstrated in a laboratory and proved to work in any maze, in which there are four directions of motion separated by 90° from each other [19]. However, the algorithm is not efficient, as it allows the blind mechanical mouse to bump into walls and retreat from them. Moreover, a mouse will enter a dead-end cell just to learn that it is a dead-end and leaving it after four moves. Another well-known algorithm that uses the similarity between mazes and graphs is Tarry's algorithm [20]. In this algorithm, the agent starts at an arbitrary initial vertex, and then follows a set of three simple rules:

1) The agent marks the direction in which it moved along an edge.
2) When the agent enters a vertex, it mark the direction from which it arrived to the vertex.
3) The agent will leave the vertex in the direction of an edge, which has not been traveled before. If no such edge can be found, the agent will travel an edge, which has already been traveled, but in the other direction.

Tarry's algorithm will cause the agent to travel the entire maze, or graph, in a depth first fashion, where each edge is traveled twice, and the agent finishes the motion at the same initial vertex it started from. Tremaux's algorithm is similar to Tarry's algorithm in the fact that it is based on graphs search and rules, as described by Ball [15]. According to this algorithm, the agent follows these rules:

1) Whenever the agent comes to a new junction or crossroads, it should take any path from it.
2) Whenever the agent comes from a new path to an old junction, or to a dead-end, it should retreat down the path leading it to this junction.
3) Whenever the agent comes to an old junction via a path already traversed, it should take a new path if there is one, otherwise take an old path.
4) A path you have traversed twice should not be entered.

Tremaux's algorithm has also been used in graph theory as a breadth-first search in a graph. Fraenkel [21] proposed an algorithm which improves both Tarry's and Tremaux's algorithms, by minimizing the times each edge is traversed to not more than once in each direction. The algorithm follows generally Tarry's algorithm with precedence to untraveled edges, and it counts the number of new vertices encountered and subtracts from it the number of untraveled edges leading from it. By doing so, if the number becomes zero at some vertex, the agent should follow an untraveled edge leading from it, and if none exists it should retreat its way until reaching the initial position.

It is worth noting that all three algorithms described above assume that an agent is capable of following corridors (or edges in the graph theory notation), identifying dead-ends and intersections (i.e. vertices), and knowing which vertex has been visited, and the direction of travel of each edge. While these assumptions are valid in modern robotics, with modern sensing and computational capabilities, this is not necessarily the case in simple robots. We assume that our robot is capable of knowing its position by usage of a GPS-like sensor, and sensing its surroundings using various vision and proximity sensors (see Fig. 2).

Another simple algorithm, which requires much less of the agent's sensing capabilities is the Pledge algorithm [22]. This algorithm can be used only by an agent trapped inside a maze, and assuming that the direction towards the exit is known. In this case the agent selects the direction to the exit as a reference direction. Then, it will follow this direction until it bumps into an obstacle. Then, the agent should turn to the left, and follow the obstacle, where in each step the azimuth is measured relative to the reference direction, by adding its rotation to the previous azimuth. Once the agent returns to the original reference direction, it should continue moving in this direction until the next obstacle.

All the algorithms presented above are designed for a single agent, or robot, working its way in to or out of a maze. However, none of these algorithms has allowed a group of agents to work a maze in cooperation. The Pledge algorithm, for example, has no provision for such expansion. In this effort, we decided to expand a rule based

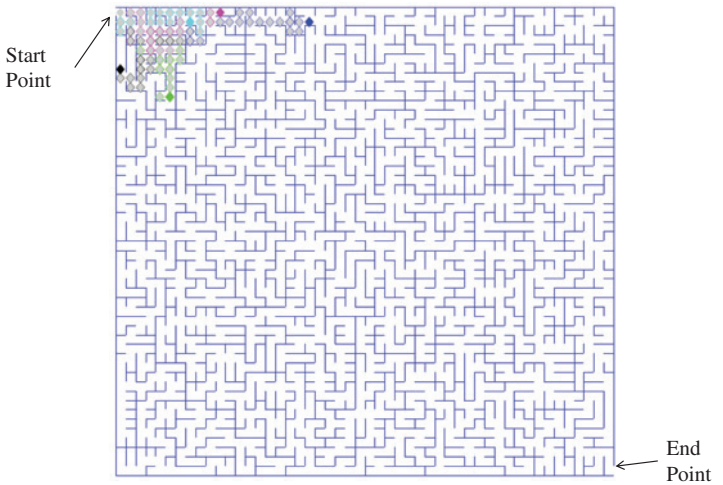

Fig. 2 Two-dimensional maze investigated (based on http://webscripts.softpedia.com/script/Games/Matlab-Maze-33561.html).

algorithm, e.g. Tarry's algorithm, to allow the agents to work together. The next subsection explains the concepts of multi-agent systems, and specifically the nature of rule-based coordination of multi-agent systems.

B. Multi-Agent Systems

Before discussing the topic of cooperative multi-agent system, one must define the terminology used. In general, multi-agent systems are inherently interdisciplinary, and different researchers use different terminology that is specific to their area of expertise. For example, Mataric [23] uses the term multi-agent systems to systems of multiple robots, and the behavior that result from their interactions is referred to as group behavior. Reif and Wang [24] refer to a system with multiple robots as a very large scale robotics system (VLSR). Beni and Wang [11] used the term swarm to define a multi-robot system. In this document, since it deals with simulated UAVs, the term for each individual within the system is *agent*, and the whole group is called *multi-agent system* or *group*.

In multi-agent systems, the term *behavior* is used to describe two different phenomena: the actions and movement of one agent in respect to its environment and states, and the global actions and movement of all the agents. In this research, the term *agent behavior* is used in reference to the individual agent behavior and the term *emergent behavior* describes the behavior of the whole system of agents.

Other researchers used slightly different terminology. Mataric [23] used *basic behavior* to describe behavior that "either achieves, or helps achieve, a relevant goal". In a later work, Mataric [25] used Steels' [26] term *emergent properties* to describe the resulting overall behavior of a multi-agent system. Reynolds [27], on the other hand, referred to the actions of each agent simply as *behaviors*, and does not use any terms to describe the resultant systemic behavior. Other terms used to describe the resultant systemic behavior are: *swarm intelligence* [11] and *behavior* [24].

In this work, the agent's behavior is defined by a system of heuristic decisions. The basis for inference which in turn leads to decisions are referred to as *rules*. In the literature, two major types of rules such as reactive and deliberative, or planner-based [28] are found. A reactive set of rules makes decisions based upon its current state and knowledge of its environment (local or global). A deliberative set of rules, on the other hand, makes decisions several steps forward into the future, based on estimation of the current situation and on a model of the environment in the near future. The different types of rules yield different characteristics: reactive models usually yield robust behavior to a noisy and dynamic environment, but cannot be classified as optimal. On the other hand, deliberative models can be designed using optimal control methods, but is usually sensitive to changes in the environment and uncertainties. Many of the systems discussed in this section incorporate a hybrid deliberative/reactive model, and utilize partial knowledge of the environment to provide better decision-making capabilities to the agents. In the hybrid model used in this work, each agent will use self-generated or other agent's partial data of the environment as a basis for decision-making. Table 1 gives a list of the terms used in the study.

1. Taxonomy of Multi-Robotic Systems

In order to discuss the qualities of existing multi-agent systems models in a meaningful manner, it is important to classify each system architecture according to the system assumptions and capabilities. Dudek et al. [13] provides a taxonomy of multi-agent systems that classifies them by use of seven categories: size, communications range, communications bandwidth, communications topology, collective reconfigurability, processing capabilities and group composition. It has been shown in previous works that this taxonomy can also be used to analyze the performance of the group [29–31]. Of significant importance in this work is the group size. The size category refers to the total number of agents acting in a system within an environment. Dudek classifies all agents within an environment to be a part of the same collective, but, as Lotspeich [32] offered in his work, it is possible in a real-world application to have

Table 1 Terms and definitions

Term	Definition
Agent	An individual in a group, representing a single robot
Multi-agent system	A group of agents or robots
Emergent behavior	The behavior of the whole system of agents
Rules	The criteria used by the heuristic algorithm in decision making

different groups of agents acting within the same environment with or without any form of collaboration. This work will use the term size to denote the number of agents in a particular group of agents. Dudek et al. [13] define in their taxonomy the following categories: a group of a single agent (SIZE-ALONE), a group of two agents (SIZE-PAIR), a group of a finite number of agents (SIZE-LIM), and a group of infinite number of agents (SIZE-INF). Although this is not possible in real-world, Dudek offers a notion of a group of agents in search and rescue mission, in which the environment is filled with more agents until either the goal is met or until every space of the environment is filled with an agent. In our case, this can be easily understood as the entire maze being filled by agents. We will show that for practical purposes, even smaller amounts of agents are enough in order to deem the maze as entirely "filled" by the agents.

Another important category in Dudek et al.'s [13] taxonomy is the communication range. Communication range is used to describe the distance that the agents are capable of sending their messages to other agents in the group (note: this does not include other communication capabilities such as communication with a human operator). Dudek uses COM-NONE to refer to a situation where there exists no *explicit* communication between the agents. The agents can, of-course, use *implicit* communication via manipulating the targets in the environment as offered by Franklin [33], or as insects use pheromones (chemical scent markers) left in the environment (also known as stygmergy [34,35]). When explicit communication exists, COM-NEAR applies to systems that assume some limit to the communication range, be it arbitrarily large whereas COM-INF assumes that there exists no such a limit, or in other words all the agents are able to communicate with one another regardless of range. This is, of course, unrealistic, but can be used to classify a system in which the communication range is far greater than the environment size, and because all the agents are acting within the environment, each agent acts within the communication range of the other agents.

In this work, we assume COM-INF range, although the algorithm itself is based on the stygmergy of ants, meaning that the same algorithm can, in theory, be achieved by using markers left in the environment. In the other categories defined by Dudek, we use the following: broadcast communications topology (TOP-BROAD), limited communication bandwidth (BAND-LOW), motion coordinated by communications (ARR-COM) for reconfigurability, turing machine equivalent (PROC-TME) as the basic computational model, and a group composed of identical agents (CMP-IDENT).

2. Coordination of Multi-Robotic Systems

Advances in technology have made it possible to field autonomous groups of robots that can be deployed in teams to accomplish various missions [36]. For example, one such direction is the US Department of Defence (DOD) UAV roadmap which has called for development of multiple UAV systems [37].

Coordinating multi-robotic systems depends on the amount of knowledge existing before the mission. If the information is complete, and no changes are made to the environment during the time of execution, optimal algorithms can be used for mission coordination [12,36,38]. However, such cases are not too common for various reasons, and indeed the true strength of an autonomous group rises from its ability to work in an unknown territory, with many changes happening while the mission is executed. Solving this problem using optimization can be extremely complicated, and most algorithms dealing with this problem have been based on heuristic solutions [35,39–43]. These algorithms usually use simple rules, which are based on behavioral models, to yield the desired, often complex [44,45], *emergent behavior*.

Of specific interest to our research is the way ants are capable of coordinating their mission. Since the 1990s, several researchers has proposed to solve complex problems, e.g. The Traveling Salesman Problem or network routing, by using algorithms inspired by the emergent behavior of ants foraging[46–48]. It has been found that foraging of ants can be explained by a set of simple rules:

1) If the ant does not carry food, it should choose its next move with higher probability to the direction with higher scent of the "food" pheromone.
2) When the ant moves towards the food, it will release the "home" scented pheromone.
3) When the ant reaches a food source, it will start carrying food and releasing "food" scented pheromone.
4) If the ant carries food, it should follow the path with strongest scent of "home" pheromone.
5) Once the ant reaches the nest, it will leave the food at the nest, and repeat from 1.

This simple mechanism generates the emergent behavior of ants foraging and bringing food from the source to the nest on the shortest path. The use of pheromones and other signs in the environment was proposed to solve various

problems, using a mechanism called stygmergy. We show that maze exploration algorithms can be expanded by similar simple means of communications, either implicit through the environment or limited explicit communications.

II. Model

A. Problem Statement

We solve the following problem: given a maze of $n - by - m$ cells, a group of agents must start at a starting point and find its way to the goal, which is the exit at the other end of the maze. Without losing generality, we define the entry point to the maze as the top left corner, and the exit from the maze (the goal) as the bottom-right corner. The maze is considered as complete when the entire group finds its way to the goal. In the next sections, we describe the maze and the capabilities of the agent we use in our simulation.

1. The Maze

The maze we solve is generated by a maze generation program written in MATLAB® based on http://webscripts.softpedia.com/script/Games/Matlab-Maze-33561.html. The program generates a maze, which is of the tree type, i.e. only one solution exists from the start point to the goal location. The program allows to control various parameters of the maze: the number of rows, the number of columns, and the pattern of the partitions dividing the cells. In addition, all the corridors between partitions are a cell wide, and we assume that the cell dimensions allow only a single agent to be in it at a certain time instance, see Fig. 2.

The maze pattern allows the maze to have random, vertical, horizontal, checkerboard, spiral, and burst options. We always choose the random pattern, which randomly chooses one of the other patterns, in order to test the ability of the algorithm to traverse a maze of any shape, without preference to a certain pattern. The maze generation algorithm is initialized with all the partitions intact, which leaves $n - by - m$ unconnected regions. Then, based on the selected pattern, partitions are gradually deleted, and the regions separated by the partition are unified. When the number of regions reaches 1, i.e. the entire maze is connected, the maze generation algorithm rests.

The program allows to generate a rectangular maze of any size and pattern, which will allow us to test the scalability of our solution to any maze size.

2. The Agent

We base the agent properties on the specification of K-TEAM Corporation's Khepera III robot, as described in http://www.k-team.com/kteam/index.php (Fig. 3). This robot, which is the third generation of the Khepera robots, is a robot that has been designed and developed to be used in a multi-robot swarm. It features a powerful 400 MHz processor, 64 MB RAM, nine infrared proximity sensors with a range of up to 25 cm, two infrared ground proximity

Fig. 3 The Khepera III robot, whose specifications we use for the agent's definition. Photo taken from http://www.k-team.com/kteam/index.php.

sensors for line following, and five ultrasound sensors with range of 20 cm to 4 m. For wireless communications, it has a Bluetooth adapter and wireless ethernet card. The robot has a diameter of 130 mm, and height of 70 mm, and it moves at speeds of up to 0.5 m/s. Although wireless ethernet communications between any two robots are performed via a wireless router, the router is merely a communications channel, and we will assume that the communication is direct between the robots.

The Khepera III is not equipped with navigation system; however, we assume that our agents know their accurate cell position using a device that transmits their location to them. This device comes to simulate a local GPS-like navigation system. In a laboratory setting, we intend to implement this system by using video cameras that will be able to see the entire maze from above. The video will be processed in order to calculate the location of each robot, and then this location will be transmitted to the robots using the wireless network. Naturally, such a process may result in estimation errors, but we assume that since a cell size is about 20 cm in each dimension (has to be a bit larger than the robot itself), the estimation errors will be much smaller than cell resolution.

B. Multi-Agent Solution

In this section, we define our solution to the given problem. We first discuss the processing scheme of each agent, followed by an explanation of the algorithm, and we conclude with detailing the communications required to be transmitted by each agent.

1. Agent Processing Scheme

The simulation runs in discrete steps. During each step, all the agents follow the entire processing scheme described below. The processing is done one agent at a time, which is a limitation on the real case where each robot is able to process its motion in parallel to the other robots.

Each agent follows the following processing scheme: it first receives communications from the other agents, and the communications, combined with sensors inputs, are processed to generate a mapping of the environment. Using the mapping, the agent invokes its maze exploration algorithm, and determines its move to the next cell. This move is executed by the agent, which then updates its position, based on the input from the external navigation system, and its mapping of traveled and untraveled cells. Finally, the agent transmits its findings of the environment to the other agents.

We defined the problem as getting a group of agents into a maze from one point, finding the exit on the other side of the maze, and then getting the entire group out of the maze through that exit. Therefore, the maze exploration algorithm has to deal with the two phases: finding the exit as quickly as possible and leading the agents out of the found exit.

During the search for the exit, each agent follows a generalization of Tarry's algorithm, but the difference is that the knowledge which cell has been visited is shared by all the agents. Each agent follows the following rules:

1) The agent should move to cells that have not been traveled by any agent.
2) If there are several such cells, the agent should choose one arbitrarily.
3) If there is no cell that has not been traveled by an agent, the agent should prefer to move to a cell that has not been traveled by it.
4) If all the possible directions have already been traveled by the agent, or if the agent has reached a dead-end, the agent should retreat until a cell that meets one of the previous conditions.
5) All the steps should be logged by the agent in its history.
6) When retreating, mark the cells retreated from as "dead end".

To fulfill the above conditions, each agent holds a map of the maze, where each cell is represented by its column, row, and four directions (up, down, left, and right). For each combination of column, row, and direction, the agent maintains a score in the range of [0,1], where 0 is "avoid moving in this direction" and any value above it represents the inclination of the agent to move in this direction. Initially, all the locations and directions are given the score of 1. From this situation, an agent gives a score of 0 to a direction, which is blocked by an obstacle, as seen by the agent's sensors, or to a direction marked as dead end. Other values are reduced based on the agent's motion: the direction in which the agent moved, and in the new cell the direction leading to the previous cell.

When the data are communicated to the other agents, each agent reduces the value of a triple column, row, direction based on the value communicated by the other agents. Thus, the agents are repelled from each other by the choices

of the other agent. Of course, when there are no other options, the agents will converge to each other, because a value greater than zero is still better than the value of exactly zero.

When one agent chooses to move to a direction of a cell already occupied by the other agent, it will wait for the cell to be vacated. This situation may result in a dead-lock if one agent is trying to enter a dead-end while the other agent has already reached the dead-end and is now retreating from it. To solve this problem, each agent will reduce the value of the selected direction as if it has already moved in this direction. This will allow the agent further away from the dead-end to choose a new direction if the cell is not vacated, because the agent occupying it comes from a dead-end. Thus, all possible "dead-locks" are resolved by this procedure.

Remembering that the maze in question is of the tree type, we take advantage of the fact that only one solution exists that connects the entry point and the exit point, and since all the agents start from the same entry point, they all share at least a single location in their path. Thus, the algorithm of the second phase is very simple: given the path of the agent that was the first to find the exit, the *pioneer*, each agent has to find the last location from its own-logged history that matches a location on the path of the pioneer. This location is denoted *Last Common Location* (LCL). Then, the agent plans the shortest path from its current location to the LCL by skipping paths that start and end at the same location, in other words dead-ends. Then, using the logged history of the pioneer, the same process is followed to plan a path from the LCL to the exit point. The two paths are combined at the LCL and from now the agent is simply required to follow this path.

While the agent is on its way to the exit, it will wait for other agents if they occupy the position to which it needs to move. As at this point all the agents are on their way to the exit, all of them will move to any vacant position along the planned path toward the end. The exit is assumed to be unoccupied when they get there, so that all of them will be able to exit the maze.

2. Inter-Agent Communication

To facilitate the coordination between the agents, each agent is required to communicate its mapping of the environment to the other agents. Each agent maintains its own copy of the mapping, and the maximum amount of communication, if bandwidth and processing are unlimited, is to transmit the entire map. However, most of the information is both redundant, i.e. each agent has it already, and not changing from one step to the other. The only changes are caused by the motion of the agents, the immediate local information around the agent, and the agent's own decision. Thus, the data required for coordination can be minimized to the following: the agent location, the agent knowledge about the directions of motion from its current location, and the agent's next location. Therefore, only eight pieces of information are required: current column and row (position), a score for each direction (up, down, left, and right), and the next column and row (location).

III. Results

We set the maze size to be constant, a 10-by-10 maze. Different groups of agents, with size ranging from a single agent to 40 agents, were simulated. Each group size was tested 250 times using a random maze pattern, which allowed testing the solution vs all the maze patterns. The results of the runs were analyzed using the figures of merit (FOMs) defined in the following subsection.

We also try to find an optimal group size to explore a certain maze and to see whether this optimal group size can be correlated to the maze size. Finally, we show that our solution can be scaled up to any maze size and any group size.

A. Figures of Merit

The number of steps it takes the group to enter the maze, find the exit, and get all the agents out is counted. The first FOM is the algebraic average of the number of steps. This FOM represents the time it will take the group to explore the maze.

The second FOM is the average number of steps measured from the time-step the pioneer finds the exit and until the last agent reaches the exit. Using this FOM, we differentiate the exploration phases, in which larger groups perform better, and the pioneer following phase, in which larger groups perform worse. Thus, this FOM is expected to be rising monotonically with the size of the group.

The third FOM is the standard deviation of the number of steps. This FOM comes to express how robust the groups performances are. Smaller groups can find the exit quickly or slowly, but the result depends on the chance. Larger groups explore the entire maze before finding the exit. This is the strength of the group: its performance should be relatively predictable.

The last FOM is the percentage of maze which was actually explored before the pioneer finds the exit. This FOM should be in correlation with the third FOM.

B. Performance

Figure 4 shows the results of simulations runs with a maze size of 10-by-10 cells and the number of agents in the group varies from a single agent to 40 agents. Each group was run 250 times in random pattern maze generation. The top line is the total number of steps required for the entire group to enter the maze, find the goal, and get to the goal. As can be seen, the total number of steps sharply decreases with an increase in the group size for small groups, reaches a minimum and then increases linearly with the size of the group. The minimum is achieved for eight agents in the group, and is around 70 steps.

The results show that adding agents to the group improves the group's ability to explore the maze and find the goal quickly. This can also be seen in Fig. 4b. However, adding agents to the group requires more steps to get them all into the maze, because only a single agent can occupy the entry point at a certain time. However, this is not the only reason for the increase in the number of steps: it takes also more steps since the pioneer agent finds the goal until the last agent gets there. This can be seen in the middle line of Fig. 4a, which grows linearly from a group size of two agents and on.

There is a positive side to adding more agents to the group: the group's performance, in terms of average time to get all the agents out of the maze, is less sensitive to the maze. This can be seen by the standard deviation, the bottom line of Fig. 4a, which decreases for small groups and then becomes relatively constant with group size. The reason is that more agents in the group means that these agents explore the maze more comprehensively (Fig. 4b) and this decreases the sensitivity to the maze itself. The good thing about it is that the number of agents required to reduce the group's sensitivity is small. Indeed, it is even smaller than the number of agents required to minimize the average number of steps.

It is interesting to see whether the aforementioned results are representative for various sizes of mazes. Figure 5 shows the results of simulation runs, for group sizes ranging from a single agent to 40 agents, for different sizes of mazes.

As can be seen, the phenomena described above repeat itself for all sizes of mazes. There is a group size that minimizes the number of steps required for the group, the group performance becomes less sensitive to the maze,

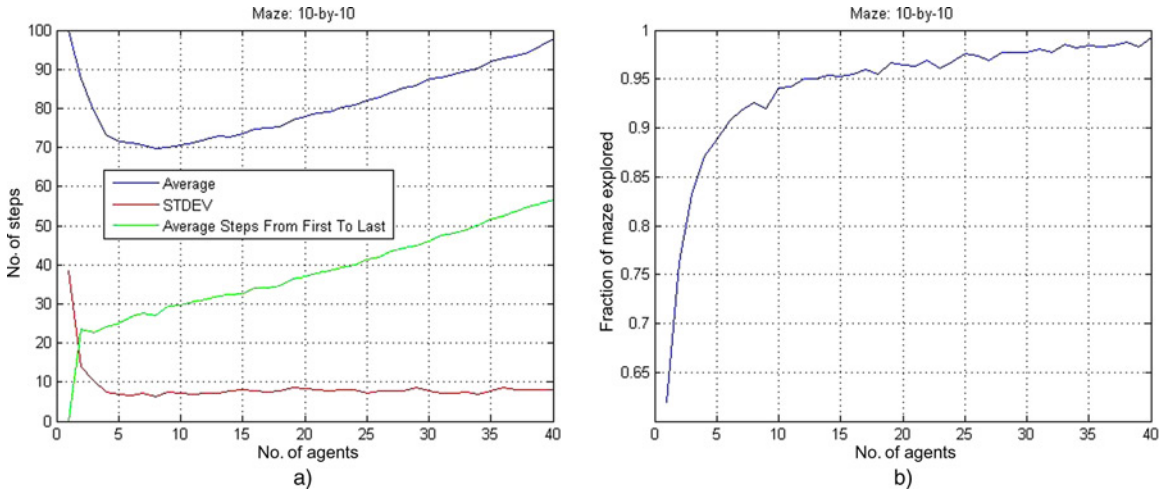

Fig. 4 Results of a 10-by-10 maze simulation: a) Results of simulation runs with 10-by-10 maze: number of steps vs the group size. b) Results of simulation runs with 10-by-10 maze: fraction of maze explored by the group.

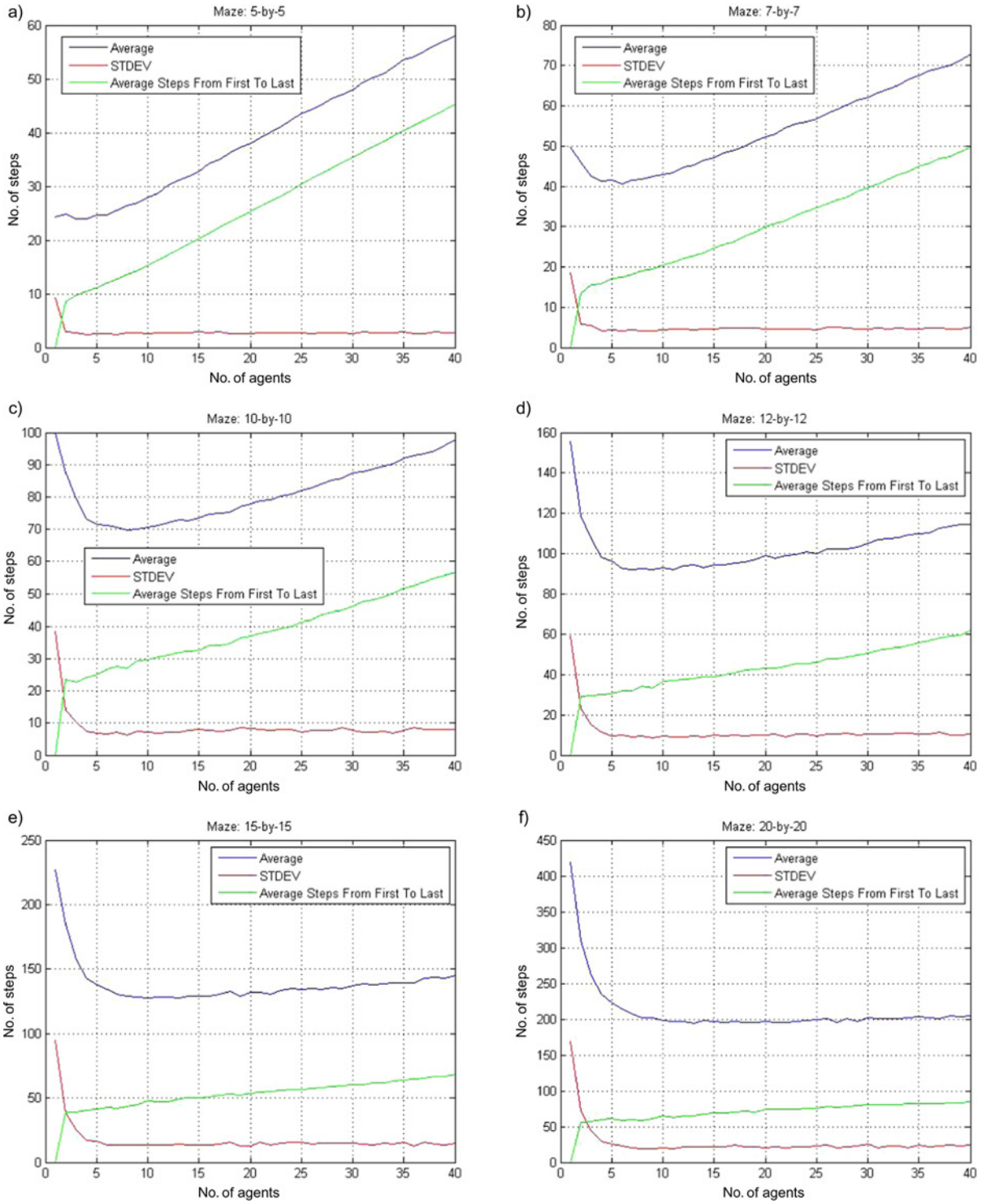

Fig. 5 Results for various maze sizes.

and the number of steps required from the pioneer to the last agent increases linearly. The difference is in the effect that the maze size has on the phenomena: for smaller mazes, they are more significant than for larger mazes. For example, for a 20-by-20 maze, the average number of steps increases very slightly after the minimum is reached for a group of about 15 agents (the accurate minimum is almost indiscernible).

C. Scalability

Next we would like to know whether the solution we propose is scalable to any size of maze and group. Figure 6 shows the results of a large maze, comprised of 100-by-100 cells. It can be seen that the group can find its goal

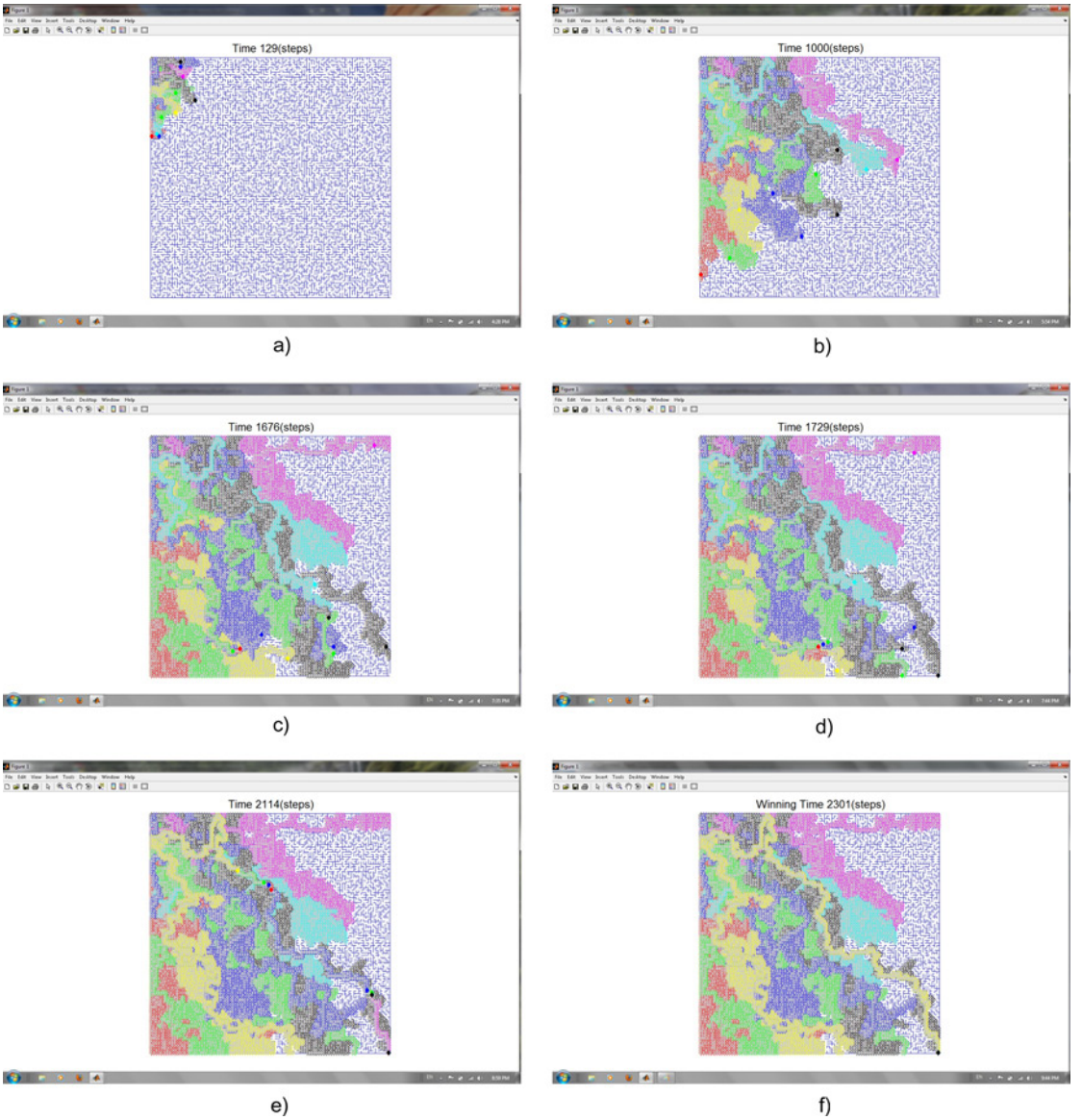

Fig. 6 A time series showing a 100 × 100 cells maze solved by a group of agents: a) The maze traversed after 129 steps. b) The agents at the middle of solution, after 1000 steps. c) The first agent almost reaches the goal, after 1676 steps. d) The pioneer agent reaches the end, after 1729 steps. e) The agents converge to the solution, taken after 2114 steps. f) All the agents reach the end of the maze.

and it takes 2301 steps to get the last agent out of the maze. The group size in this case is 10 agents. Remembering that the data each agent communicates are only the local information around the agent, it is not surprising that the agents processing and the communications are almost not affected by the maze size. The only requirement is that the agents will be capable of storing a map of the entire maze and log their path history, which is longer as the maze size increases. Thus, only the agent memory should be of any concern.

The scalability to group size was shown in the previous subsection, where we use a solution for group sizes of 1–40 agents. Here, the scalability should be divided into two parts: the amount of processing each agent is required to perform and the amount of processing required by the hosting machine. At the level of the single agent, as the agent is required to process the communications of the other agents in order to generate a single map of the maze, the complexity of the processing increases linearly with the number of agents. The level of the host machine depends on the architecture of the processing. In our case, the PC that was used to run the simulation in MATLAB® was running all the agents in a sequence, without any parallel processing. Thus, the complexity, and the time required for the simulation, increases quadratically with the number of agents in the group. However, in real application this is not the case, because each agent resides on a different host: the robot that executes the algorithm. Thus, we assume that using real robots will keep the complexity linear with the number of agents in the group.

IV. Conclusion

We present the problem of a simple maze, where a group of coordinating agents is required to move from an entry position to the goal position without a-priori knowledge of the maze itself. We introduce a unique multi-agent approach, which enhances Tarry's algorithm to the multi-agent case. The developed strategy enables dispersion of the agents in the maze and finds an efficient solution to the maze, in terms of the average amount of steps required for the solution. It is important to note that while Tarry's algorithm, which is used by the single agent, is a depth-first search algorithm, the behavior of the entire group is breadth-first search, because each agent is repelled by the other agents. Thus, we combine the two major search modes in graph theory. This shows the difference between agent behavior and emergent behavior of a group. Dispersing the agents in the maze allows the group to cover larger fractions of the maze, thus the group's performance becomes more robust to the different shapes of the maze.

We are currently expanding this effort to show that the solution we propose can work with a group of agents that do not start at the same location, therefore nothing guarantees that they have a common location to converge to when the exploration phase is over. Further directions for future work include extending the algorithm to solve nontree mazes, validating it with real robots in a laboratory setting, and incorporating human intervention into the group's logic.

References

[1] Kern, H., *Through the Labyrinth: Designs and Meanings over 5,000 Years*, edited by F. Robert, Prestel Publishing, Munich, London, New York, September 2000.

[2] Barnes, C., "Memory Deficits Associated with Senescence: A Neurophysiological and Behavioral Study in the Rat," *Comparative and Physiological Psychology*, Vol. 93, No. 1, February 1979, pp. 74–104. http://dx.doi.org/10.1037/h0077579

[3] Olton, D. S., and Samuelson, R. J., "Remembrance of Places Passed: Spatial Memory in Rats," *Journal of Experimental Psychology: Animal Behavior Processes*, Vol. 2, No. 2, April 1976, pp. 97–116. http://dx.doi.org/10.1037/0097-7403.2.2.97

[4] Morris, R., "Developments of a Water-Maze Procedure for Studying Spatial Learning in the Rat," *Journal of Neuroscience Methods*, Vol. 11, No. 1, 1984, pp. 47–60. http://dx.doi.org/10.1016/0165-0270(84)90007-4

[5] Dracopoulos, D., "Robot Path Planning for Maze Navigation," *The 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence*, Vol. 3, May 1998, pp. 2081–2085.

[6] Lumelsky, V., "A Comparative Study on the Path Length Performance of Maze-searching and Robot Motion Planning Algorithms," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, February 1991, pp. 57–66. http://dx.doi.org/10.1109/70.68070

[7] Werbos, P., and Pang, X., "Generalized Maze Navigation: SRN Critics Solve What Feedforward or Hebbian Nets Cannot," Vol. 3, October 1996, pp. 1764–1769.

[8] Rao, N., and Iyengar, S., "Autonomous Robot Navigation in Unknown Terrains: Incidental Learning and Environmental Exploration," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 6, Nov/Dec 1990, pp. 1443–1449. http://dx.doi.org/10.1109/21.61213

[9] Milkova', E., and Slaby', A., "Graph Algorithms in Mutual Contexts," *7th WSEAS International Conferences on Applied Computer and Applied Computational Science (ACACOS 08)*, Hangzhou, China, 6–8 April 2008, WSEAS Press, pp. 721–726.

[10] Biggs, N., Lloyd, E. K., and Wilson, R. J. *Graph Theory, 1736–1936*, Clarendon Press, New York, NY, December 1986.

[11] Beni, G., and Wang, J., "Swarm Intelligence in Cellular Robotic Systems," *Proceedings of the NATO Advanced Workshop on Robotics and Biological Systems*, 1989.

[12] Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M., and Baum, M., "Cooperative Control for Autonomous Air Vehicles," *Proceedings of the Cooperative Control Workshop*, Florida, December 2000.

[13] Dudek, G., Jenkin, M. R. M., Milios, E., and Wilkes, D., "A Taxonomy for Multi-agent Robotics," *Autonomous Robots*, Vol. 3, No. 4, 1996, pp. 375–397.

[14] Bagnall, A. J., and Zatuchna, Z. V., "On the Classification of Maze Problems," *Applications of Learning Classifier Systems, Studies in Fuzziness and Soft Computing*, edited by L. Bull and T. Kovacs, Vol. 183, Springer, Berlin, Heidelberg, April 2005, pp. 307–316.

[15] Ball, W. W. R. *Mathematical Recreations & Essays*, 11th ed., McMillan, New York, NY, January 1939.

[16] Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H., "Learning View Graphs for Robot Navigation," *Autonomous Robots*, Vol. 5, No. 1, 1998, pp. 111–125. http://dx.doi.org/10.1023/A:1008821210922

[17] Wilson, S. W., "The animat path to AI," *Proceedings of the First International Conference on Simulation of Adaptive Behavior on from Animals to Animats*, MIT Press, Cambridge, MA, 1990, pp. 15–21.

[18] Shannon, C. E., "Presentation of a Maze Solving Machine," *Cybernetics: Circular, Causal and Feedback Mechanisms in Biological and Social Systems, Transactions Eighth Conference, 1951*, edited by M. M., von Foerster and H. L. Teuber, Vol. included in Part B, Josiah Macy Jr. Foundation, New York, NY, 15–16 March 1952, pp. 169–181.

[19] Sutherland, I., "A Method for Solving Arbitrary-Wall Mazes by Computer," *IEEE Transactions on Computers*, Vol. C-18, No. 12, December 1969, pp. 1092–1097. http://dx.doi.org/10.1109/T-C.1969.222592

[20] Ore, O. *Theory of Graphs*, 3rd ed., American Mathematical Society, Providence, RI, 1962.

[21] Fraenkel, A. S., "Economic Traversal of Labyrinths," *Mathematics Magazine*, Vol. 43, No. 3, 1970, pp. 125–130. http://dx.doi.org/10.2307/2688386

[22] Abelson, B. H., and DiSessa, A. A., *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, MIT Press, Cambridge, MA, reprint, illustrated ed., 1986.

[23] Mataric, M. J., "Designing and Understanding Adaptive Group Behaviors," *Adaptive Behavior*, Vol. 4, 1995, pp. 51–80. http://dx.doi.org/10.1177/105971239500400104

[24] Reif, J., and Wang, H., "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots," *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR94')*, 1994.

[25] Mataric, M. J., "Behavior Based Control: Examples from Navigation, Learning, and Group Vehavior," *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents*, Vol. 9, 1997, edited by H. Hexmoor et al.

[26] Steels, L., "The Artificial Life Roots of Artificial Intelligence," *Artificial Life*, Vol. 1, 1994, pp. 75–110. http://dx.doi.org/10.1162/artl.1993.1.1_2.75

[27] Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics*, Vol. 21, 1987, pp. 25–34. http://dx.doi.org/10.1145/37402.37406

[28] Usher, J. M., and Wang, Y.-C., "Intelligent Agents Architectures for Manufacturing Control," *Proceedings of the Industrial Engineering Research Conference 2000*, 2000.

[29] Gurfil, P., and Kivelevitch, E., "Flock Properties Effect on Task Assignment and Formation Flying of Cooperating Unmanned Aerial Vehicles," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 221, No. 3, June 2007, pp. 401–416. http://dx.doi.org/10.1243/09544100JAERO120

[30] Kivelevitch, E., and Gurfil, P., "Taxonomy of Mission Performance for Diverse and Homogenous UAV Flocks," *Proceedings of the 2005 AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, San Francisco, CA, 15–18 August 2005, AIAA-2005-5828.

[31] Gurfil, P., "Evaluating UAV Flock Mission Performance using Dudek's Taxonomy," *Proceedings of the 2005 American Control Conference*, IEEE American Control, IEEE, Portland, OR, 2005, pp. 4679–4684.
[32] Lotspiech, J. T., "Distributed Control of a Swarm of Autonomous Unmanned Aerial Vehicles," Master's Thesis, 2003, Air Force Institute of Technology.
[33] Franklin, S., "Coordination without Communication," http://www.msci.memphis.edu/franklin/coord.html
[34] Parunak, H. V. D., and VanderBok, R. S., "Managing Emergent Behavior in Distributed Control Systems," *Proceedings of the ISA-Tech '97*, 1997.
[35] Parunak, H. V. D., Purcell, M., and O'Connell, R., "Digital Pheromones for Autonomous Coordination of Swarming UAVs," *AIAA's 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles*, AIAA, Portsmouth, Virginia, 20–23 May 2002, AIAA no. 2002-3446.
[36] Schumacher, R. S. P. C. J. W. M. C., and Sparks, A., "Optimal vs. Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, 11–14 August 2003, AIAA no. 2003-5586.
[37] Of the Secretary of Defence (OSD), O., *UAV Roadmap 2002–2027*, 2002.
[38] Schumacher, C., Chandler, P., and Pachter, M., "UAV Task Assignment with Timing Constraints," AFRL-VA-WP-TP-2003-315, 2003, United States Air Force Research Laboratory.
[39] Lua, C. A., Altenburg, K., and Nygard, K. E., "Synchronized Multi-Point Attack by Autonomous Reactive Vehicles with Simple Local Communication," *Proceedings of the IEEE Swarm Intelligence Symposium*, April 2003, http://www.cs.ndsu.nodak.edu/lua/folder/synAttack.pdf.
[40] Chandler, P. R., Pachter, M., Swaroop, D., Fowler, J. M., Howlett, J. K., Rasmussen, S., Schumacher, C., and Nygard, K., "Complexity in UAV Cooperative Control," *Proceedings of the American Control Conference*, May 2002, pp. 1831–1836.
[41] Guo, W., and Nygard, K., "Combinatorial Trading Mechanism for Task Allocation," *Proceedings of the 13th International Conference on Computer Applications in Industry and Engineering*, June 2001.
[42] Schlecht, J. W., "Mission Planning for Unmanned Air Vehicles using Emergent Behavior Techniques," McNair Scholars Day, April 2001.
[43] Cunningham, C. T., and Roberts, R. S., "An Adaptive Path Planning Algorithm for Cooperating Unmanned Air Vehicles," *IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001.
[44] Gaudiano, P., "Swarming of Unmanned Air (and Ground) Vehicles," Slides, Icosystem Corporation.
[45] Guadiano, P., Shargel, B., Bonabeau, E., and Clough, B. T., "Control of UAV Swarms: What the Bugs Can Teach Us," *2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations-Aerospace*, AIAA, San Diego, California, 15–18 September 2003, AIAA No. 2003-6624.
[46] Dorigo, M., Maniezzo, V., and Colorni, A., "The Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, Vol. 26, 1996, pp. 29–41. http://dx.doi.org/10.1109/3477.484436
[47] Dorigo, M., and Di Caro, G., "Ant Colony Optimization: a New Meta-heuristic," Vol. 2, 1999, pp. 1470–1477
[48] Schoonderwoerd, R., Holl, O., Bruten, J., and Rothkrantz, L., "Ant-based Load Balancing in Telecommunications Networks," *Adaptive Behavior*, Vol. 5, 1996, pp. 169–207. http://dx.doi.org/10.1177/105971239700500203

Mike Hinchey
Associate Editor