Multi-Agent Swarm Spreading Approach in Unknown Environments

Shadi Alian[1], Nazeeh Ghatasheh[2] and Mua'ad Abu-Faraj[1]

**[1] Department of Computer Information Systems, The University of Jordan
Aqaba, 77110, Jordan**

**[2] Department of Business Information Technology, The University of Jordan
Aqaba, 77110, Jordan**

Abstract
Swarm spreading has been gaining more focus since the more reliance on artificial intelligence in solving complex problems. Such importance comes from the various possible applications of the swarm robotics physically and virtually. This work has more focus on proposing an enhanced approach for spreading virtual autonomous swarms over an unknown environment. The main considerations included having less direct communication among the agents, covering the whole environment with as few steps as possible, keeping the agents on the move with less waiting time and the possibility of applying the simulation environment to various problems. The preliminary results are promising as they show interesting outcomes over the setup environment.

Keywords: *Multi-Agent, Swarm Spreading, Pheromone-based Communication, Simulation.*

1. Introduction

Swarm Intelligence has attracted recently much attention of researchers around the world. Swarm Intelligence which is considered as a branch of Artificial Intelligence has applications to solve different kind of problems. Swarm behavior characterizes many species in nature, as an example flocks of birds, social insects (bees, wasps, ants, and termites), fish schools, and herds of land animals. These species tend to adopt swarm behavior due to its biological needs to stay together [1].

A swarm can be defined as a group of agents cooperating with each other in a certain behavioral pattern, the goal of agents cooperation is to achieve some goal that is beyond the capabilities of a single agent [1]. There are different models of swarm intelligence that have been studied, these models include ant colony optimization [2], particle swarm optimization [3],[4], honey bee swarming [5], and bacterial foraging [6].

Swarm intelligence models from computational point of view, are considered as computing algorithms that are useful for undertaking special cases of distributed optimization problems. Swarm based algorithms is inspired from nature, and considered as population-based algorithms. These algorithms are capable of producing low cost, fast, and robust solutions to several complex problems [1],[3]-[6]. Swarm intelligence is motivated by its potential future applications. One of its applications is the inspection autonomously of complex engineered structures by swarm agents, such as inspecting turbines from the inside [2]. Another application is using robots in searching/exploration behavior, it also can be used in aggregation for dispersal, gathering, and clustering [3],[7]. Swarm systems have a high degree of robustness, that is the system's effectiveness is ensured, even if a high percentage of the swarm is lost. This robustness results in a distribution of risk which is an appropriate approach to safety critical scenarios [8].

This research aims to provide an enhanced approach for spreading virtual autonomous swarm agents over an unknown environment. In this paper, we present a modified version of the model presented in [9]. The agents used in this model has no direct communication between them. The simulation environment can be controlled using different variables. Three different approaches were proposed to enable agents to decide the movement direction.

The remainder of this paper is organized as follows; The next section presents some of swarm spreading algorithms related work. It further provides the research methodology. Then comes a description of the model with its simulation environment and implementation. Followed by the results. While the last section summarizes the advantages and limitations of the presented model, it also provides some concluding remarks and open questions of the field.

2. Related Work

Since ages the natural swarms have been studied and analyzed in order to understand the way they behave. Since few decades they have been replicated in virtual environments [10]. Such replications were for the sake of many reasons serving different fields as military, medicine, transportation, and education. Related to that, various researchers have been developing, deploying, evaluating, enhancing and proposing various approaches for spreading over unknown environments efficiently and effectively; either by physical or virtual robots [7],[10]-[13].

The authors in [11] validated spreading algorithms targeting real physical robots using a simulation environment. In which the idea was to spread teams of small autonomous robots having no major communication channels and few sensors. Where the main objective was to cover a large span of the unknown environment, keep a minimal cross-team communication range, deal with various obstacles and have a certain level of intelligence. Their approach proposed different robot behaviors that are the "Random Walk, Find Opening, Avoid Robots, Random Range, and Backtrack Range". The behaviors define how the robots interact with the environment and with each other as how the robots behave when they are outside the communication range. The authors supposed that their developed algorithms and approach make it efficient to deploy a relatively large number of robots while having a decent economic feasibility. Though the robots relying on their approach seem to be useful, they are unaware of their exact location, the locations of other robots, and may face technical communication issues.

In order to assess the spreading algorithms over unknown environments, several movement techniques were closely analyzed in [12]. The main aim was to virtually evaluate the physical robots spanning percentage in different environments as two-dimensional plane, a building or a terrain. Coverage percentage was the primary evaluation parameter for autonomous non-communicating robots, that are characterized by limited observation sensors, low computation power and assumed to have a limited energy resources. As the authors based their work on a real physical robot that has limited awareness and resources. However the robots were unable to directly communicate with each other, the authors implemented an indirect communication scheme that let the robots seeks traces of other robots. The robots can travel in the environment by moving forward, in reverse and change direction according to local independent decisions. These basic movement options enabled four basic spreading techniques that are random steps, following an edge, moving towards an open space or move away from close robots. The main findings show that the best results were achieved when the robots spot the locations of other robots, namely the "Fiducial" algorithm. While reaching small areas was an issue, hence the robots consider it as an obstacle in the environment and tend to move away.

Brick & Mortar [13] is a proposed spreading algorithm in unknown environments for which the authors claim achieving better performance than similar algorithms. Its basic idea is to quickly spread a group of autonomous robots over an unknown environment in emergency and critical situations. Despite the difficulty of direct cross-team communication, the robots are able to work in coordination by leaving messages that other robots can sense and interpret. Brick & Mortar model divides the environment into cells that can have one of four labels that are unexplored, explored, visited and walls. As illustrated, their approach has many features as robots adaptation to sudden moves to new locations, monitoring the progress, ability to optimize resources and avoiding repetitive tasks. However Brick & Mortar was proven theoretically efficient, it lacks actual testing results of real physical robots in practice and needs to consider signaling urgent messages needed in emergency situations.

3. Methodology

This research adopts an experimental approach to reach and validate the outcomes of a multi-agent simulation model. In which the main components are the simulation environment (an unknown space), the autonomous multi-agent teams (the swarms) and the spreading algorithm (the logic). Where the basic setup of the components is inspired from the literature, taking the most suitable configurations and characteristics. Further adaptations and additions are tested to reach a relatively improved performance and overcome major issues. The actual implementation of the component was sought to be modular, interoperable, flexible and scalable, in order to establish a multi-purpose customizable solution. Figure 1 illustrates the conceptual block diagram of the desired implementation. Swarms are supposed to consist of autonomous configurable agents independently from the environment. Similarly the environment configurations are upon demand and interoperable with various external components. In other words, the whole setup is not case specific and can be used for further experiments or applications.

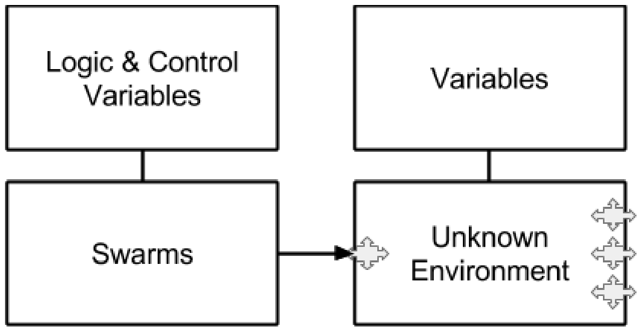

Figure 1: Block diagram of the components

A number of simulation software platforms exist for multi-agent modeling as Mason, NetLogo, Java Swarm, Objective-C Swarm and Repast. The evaluation of these free platforms in [14],[15] indicated that selecting the most suitable platform is a trade-off between different characteristics. Table 1 compares the main characteristics of each simulation platform. However Mason platform has some drawbacks, it has been found to be the most suitable platform to the multi-agent simulation of this work.

Table 1: Concise comparison of simulation software platforms (Adapted from [14]-[16])

	Mason	NetLogo	Java Swarm	Obj. C Swarm	Repast
Maturity	***P***	***F***	***G***	***G***	***F***
Programming Experience	***P***	***G***	***F***	***F***	***P***
Properties Modification	***P***	***F***	***P***	***P***	***G***
User Interface	***P***	***G***	***P***	***P***	***G***
Simulation Speed	***G***	***F***	***F***	***F***	***G***
Documentation	***G***	***G***	***F***	***F***	***F***
Modularity	***G***	***F***	***P***	***P***	***F***
Intensive/Complex Computations	***G***	***P***	***P***	***P***	***P***
Control/Debug	***G***	***P***	***P***	***P***	***F***
Suitability	***G***	***F***	***P***	***P***	***G***

Criteria: **G**ood, **F**air, **P**oor

MASON [16] is a multi-agent Java-based toolkit for a variety of simulation tasks as swarm robotics, machine learning [17] and social complexity. Among the different alternatives, Mason is characterized by its higher speed, modularity, extensibility and high ability to serve intensive simulation tasks. Other interesting objectives of Mason are the separation between the visualization area and the simulation model, platform independence, two and three dimensional visualization, execution checkpoints and portability. Such features make it easier to switch between different simulation models and an increased separability. Above all, Mason is an open source environment freely available as a result of the collaboration between the Computer Science Department and the Center for Social Complexity at George Mason University. Mason's was built over three architectural levels that are the utility level, the model level and the visualization level. In which the general operations reside in the utility classes, the basic simulation classes are in the "SimState" and the basic visualization classes are in the "GUIState". Two of the levels (layers), SimState and GUIState, are shown in Figure 2 which illustrates how the simulation model could be detached easily.

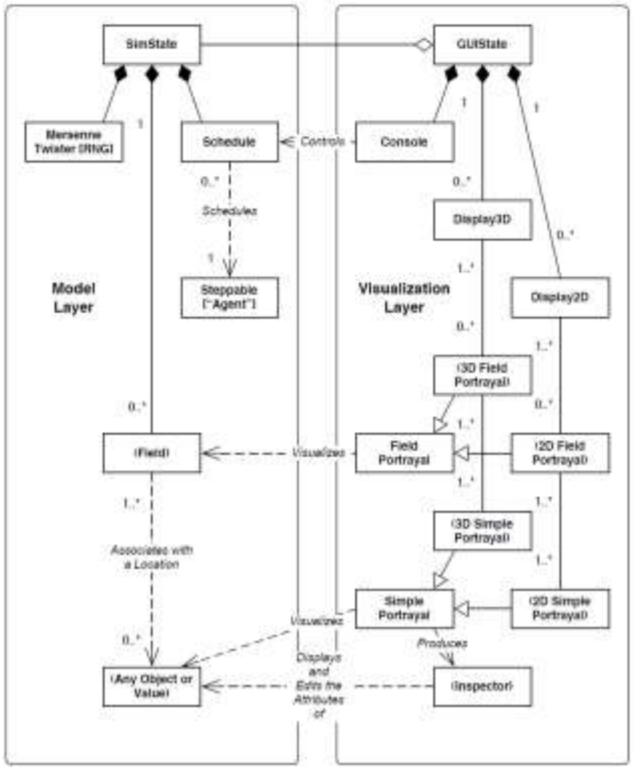

Figure 2: Architecture of Mason simulation environment [16]

4. The Model

The idea behind the model presented in this work relies heavily on [9] with many enhancements and additions. While Mason [16] is used as the simulation environment to run the model. The model agents have no direct communications and the rules of the agents are adapted from "The Game of Life" [18]. Furthermore, different variables to control the simulation environment were added that include the diagonal movement of the agents. In terms of steering the agents to span the environment, there are three different approaches that enable the agents to advance to a new movement direction.

The model consists of four main components as shown in Figure 3. First, schedualable objects which are the Agents and pheromones that extends the Steppable entity from Mason [15]. A collection of Agents form swarms in the model, while the Pheromones represent the indirect communication mean between agents within the simulation environment. Second, model elements that contain DispersSimState entity that implements SimState entity from Mason [15]. DespersSimState represents the module that steps the simulation in time and space. Third, GUI which contains DispersSimStateWithUI entity that implements GUIState entity from Mason[15]. DispersSimStateWithUI represents the interface of the model that wraps DispereSimState model. Fourth, utility entities which contain Directions, AgentStatus, LocationInfo and LocationStatus.

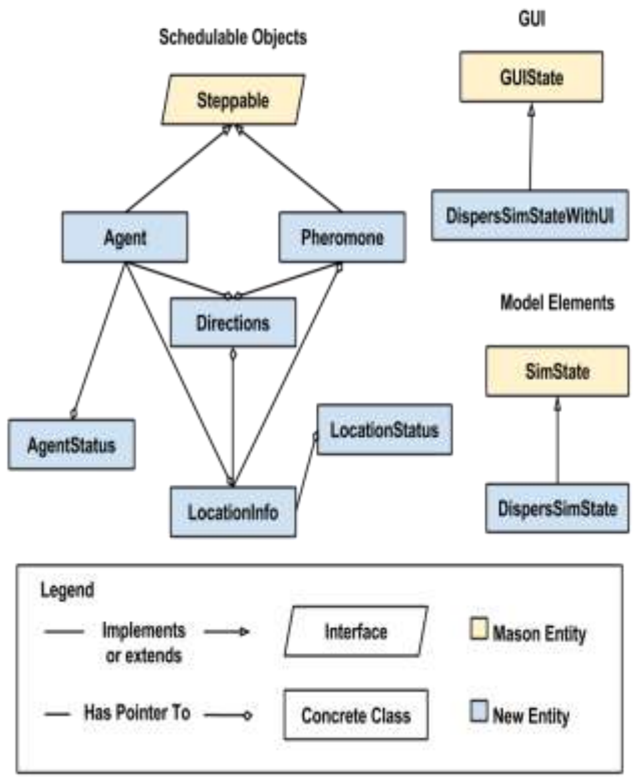

Figure 3: Main components of the model

Figure 4 shows the general processes the simulation will go throw. The initialization process sets up the environment variables including the width, height, door size, door position and other control variables that the agents use, as they are briefly explained after . At each time step many things happen. First filling the door with agents if empty. Second, stepping each live agent in the environment in sequence starting with the first-in and ending with the ones on the door, or alternatively randomly stepping the agents. The simulation stops when all the agents are dead.

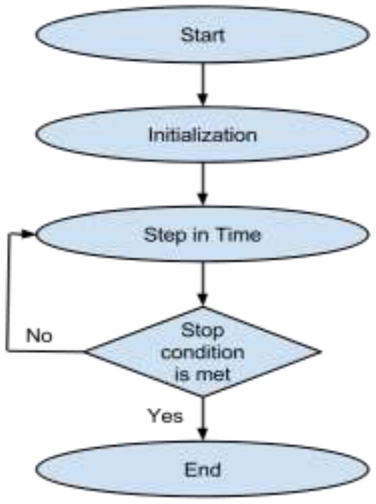

Figure 4: General processes of the simulation

The main model is concentrated in the logic of the agent. The main principles used to design the logic of the agent are zero direct communication among them, depending only on local information collected from the environment, and making a greedy decision depending on collected information. To get the agent inside the environment, it gets positioned at one of the door gates then it starts its journey. There is alot of information the agent will look at before making the decision of moving, waiting or dying. Such information include Directions, Pheromones, Neighbors and Teams.

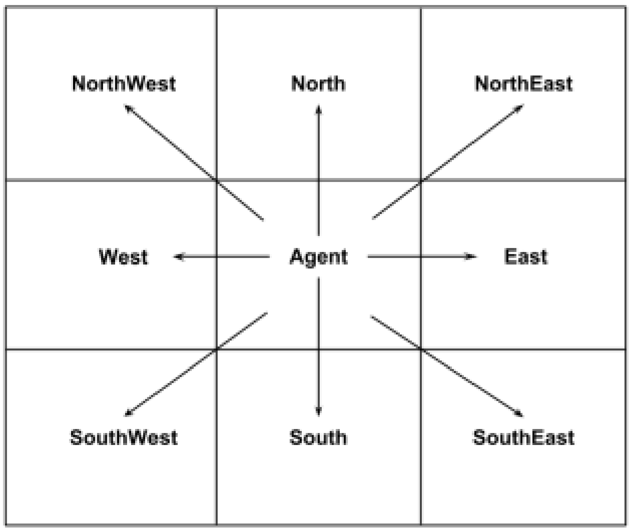

Figure 5: Directions cover all eight positions one step around the agent in the environment

Teams is a way to group a swarm of agents together in which they follow each other and use the same pheromones. Teams are determined by the door gate from which the agent is entering the environment, for instance three gates will result in having three teams. Having a team will help in guiding a swarm to follow a leader for the sake of exploring the environment in all possible directions as shown in Figure 5. Having multiple teams will help exploring the environment more efficiently. On the other hand Pheromones formulate an indirect way of communication between the agents. Each team has its own pheromone that will be used to keep the agents of the same team as close as possible to each other, consequently leading to more efficient coverage of the environment.

Agent Algorithm

```
update current location information
update neighbors location info
if agent is leader
  direction = getNewDirection()
  if direction exists
    move to direction
  else // agent is a follower
    if agent can follow current pheromone
      move to location current pheromone is pointing to
if agent didn't move
  if agent can wait
    put agent in waiting mode
  else
    put agent in dead mode
```

Agent stepping algorithm is straight forward. First it will collect information about current and surrounding locations. Relying on the collected information the agent will be identified as leader or follower. The leader will try to find the next location to move to based on the steering method, on the other hand the follower will only try follow its team using the pheromones left in the environment. If the agent could not move in either situations the stepping will enter in the wait-die algorithm which uses the collected information to decide if the agent should wait, die or switch teams. The wait-die algorithm uses the rules of "The Game of Life"[18]. The agent would switch teams only if killing the agent at that location will block another team, if so switch that agent to the blocked team.

The proposed spreading algorithm in this work has many variables that control it accordingly, as shown in Table 2. Variables max wait time, dead count, and wait count are used by the wait-die algorithm, they are used as limits to make decisions on the status of the agent if moving is not possible. Pheromone fading variable is used to fade the pheromone in decreasing manner if no agent from the same team passes throw the location where that pheromone is located at, which releases that location for other teams to move to that location and place their own pheromone, because it is not allowed more than four pheromones at each location. The rest of the variables are self explanatory.

Table 2: Control variables, descriptions and default values

Variable	Description	Value
# of teams	The number of teams will enter the environment	3
max wait time	The maximum number of steps the agent will stay in waiting status without moving	100
dead count	The maximum number of dead agents around an agent at a specified step	4
wait count	The maximum number of waiting agents around an agent at a specified step	4
pheromone fading	The percentage a pheromone fades at each step if no agent passes throw that location	0.1
door position	The position of the door according to the environment	right bottom
environment size	The width and height of the environment in pixels	50 x 50
obstacles	Objects that occupy locations in the environment with different shapes and sizes	none
agent size	The size of the agent in pixels 1, 4, 9, 16 ...	1
agent step	Number of pixels the agent will move at each step	1

5. Results and Discussion

This section shows major preliminary results of the proposed algorithm using the default values of the control variables mentioned in Table 2. Three runs of the simulation were performed each with a different steering algorithm. Figures 6, 7 and 8 show the results of the Hug-the-wall steering algorithm which has the main focus in this work. While Figure 9 and 10 represent the results of using random and smart steering algorithms respectively.

The results highlight five main readings that are visited, covered, number of agents, average waiting time and number of branches. Visited indicates the percentage of visited locations in the environment by any agent at some step. Covered indicates the percentage of locations that have a dead agent. Number of agents indicates the number of the agents used to fill the environment. Average waiting time indicates the average waiting time of all agents at each step. Number of branches indicates the number of branches made by all teams in all the steps executed for each run.

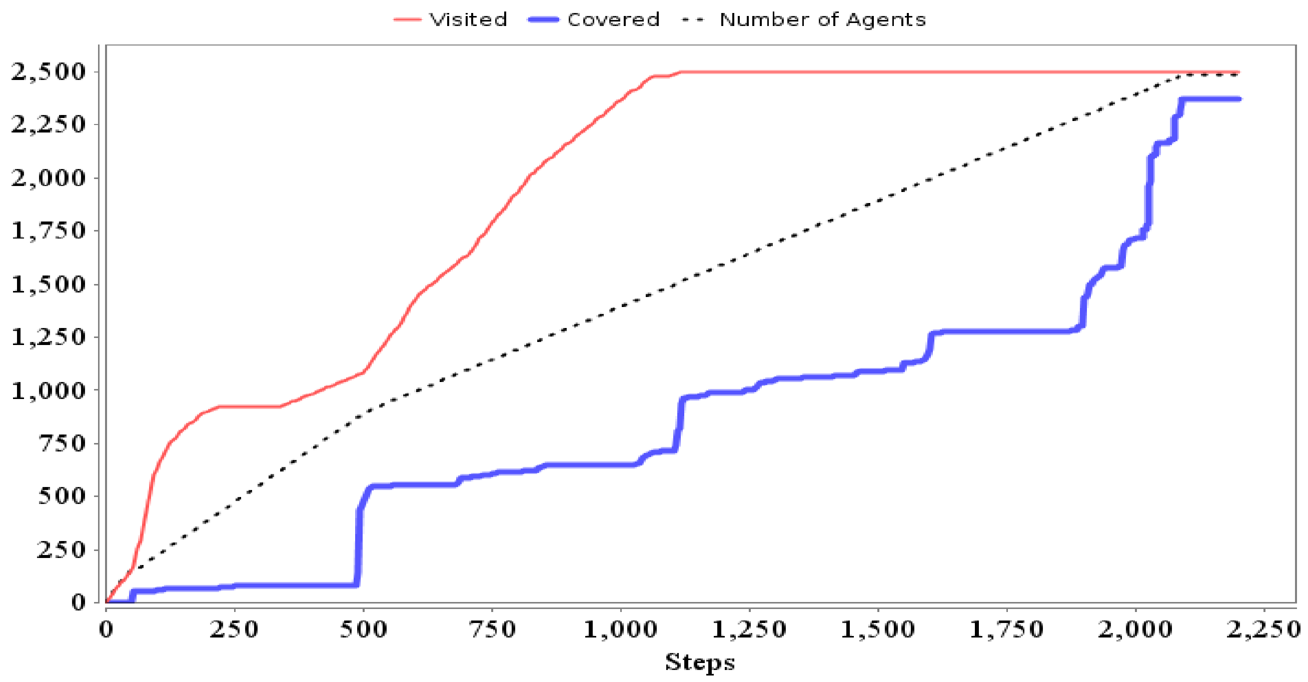

Figure 6: Hug-the-wall steering algorithm results with the relevance number of agents in dotted black, coverage in blue and visited locations in red all in an empty environment of (50 x 50) locations.

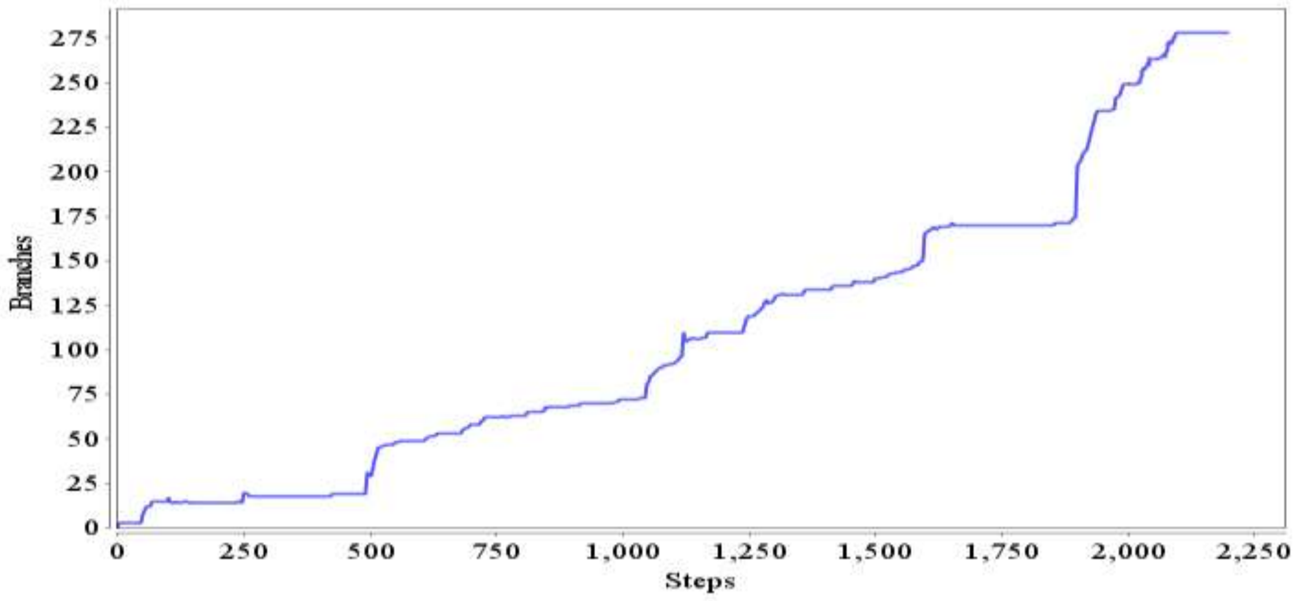

Figure 7: Hug-the-wall steering algorithm results with the number of branches related to the number of steps in an empty environment of (50 x 50) locations.

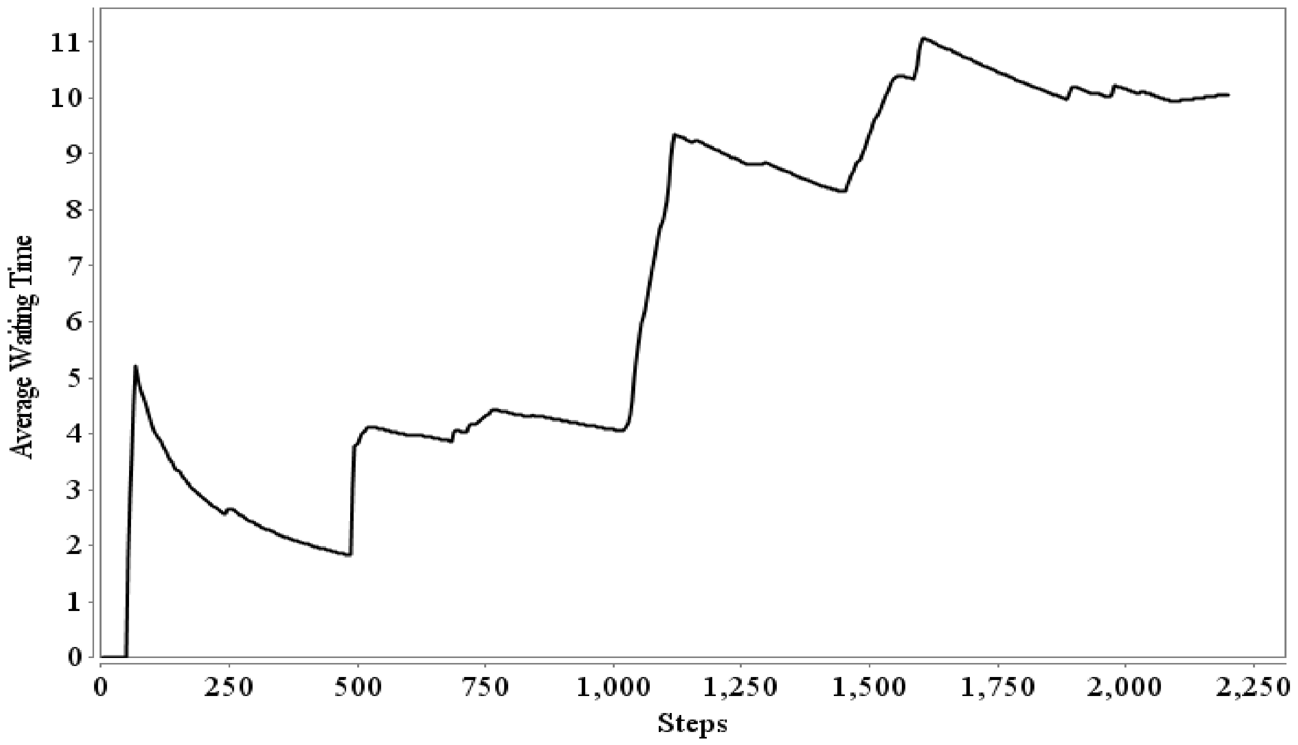

Figure 8: Hug-the-wall steering algorithm results with the Average waiting time of all agents at each step related to the number of steps in an empty environment of (50 x 50) locations.

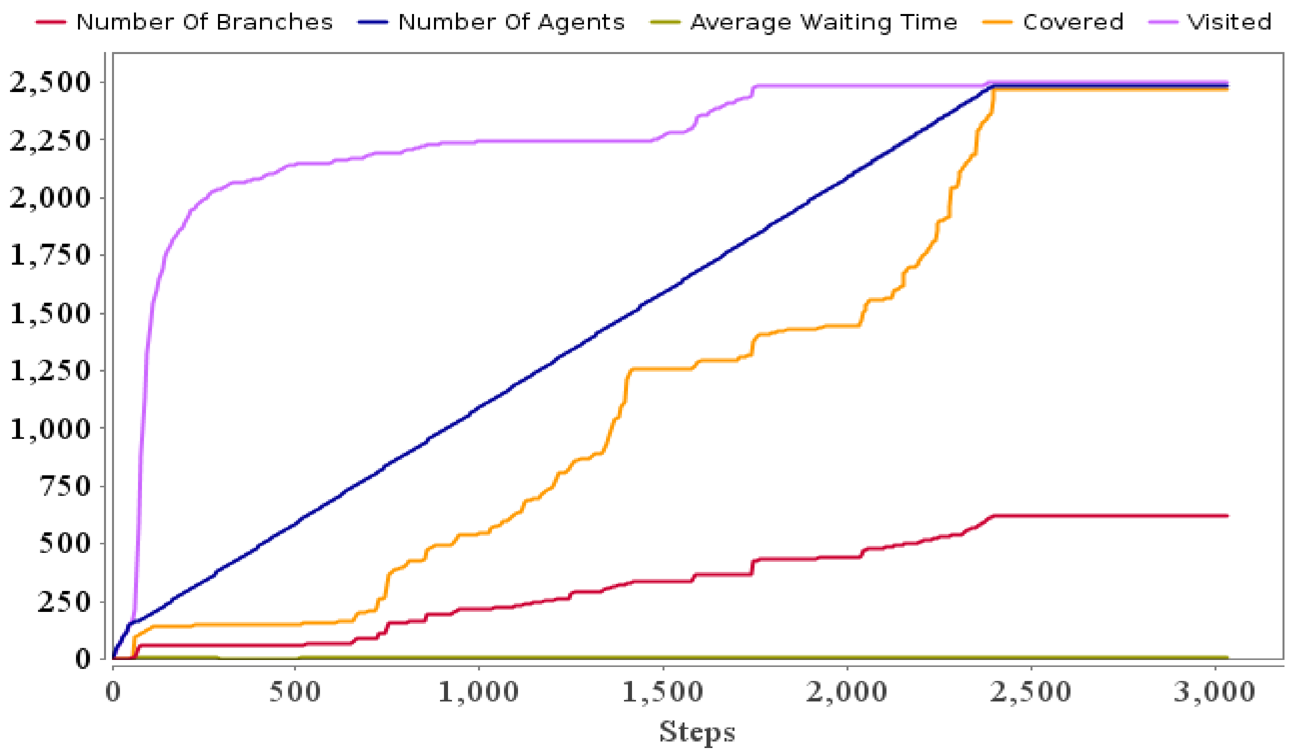

Figure 9: Random steering algorithm results with the relevance number of agents in blue, coverage in orange, Average waiting time of all agents at each step in green, number of branches in blue and visited locations in purple all in an empty environment of (50 x 50) locations.

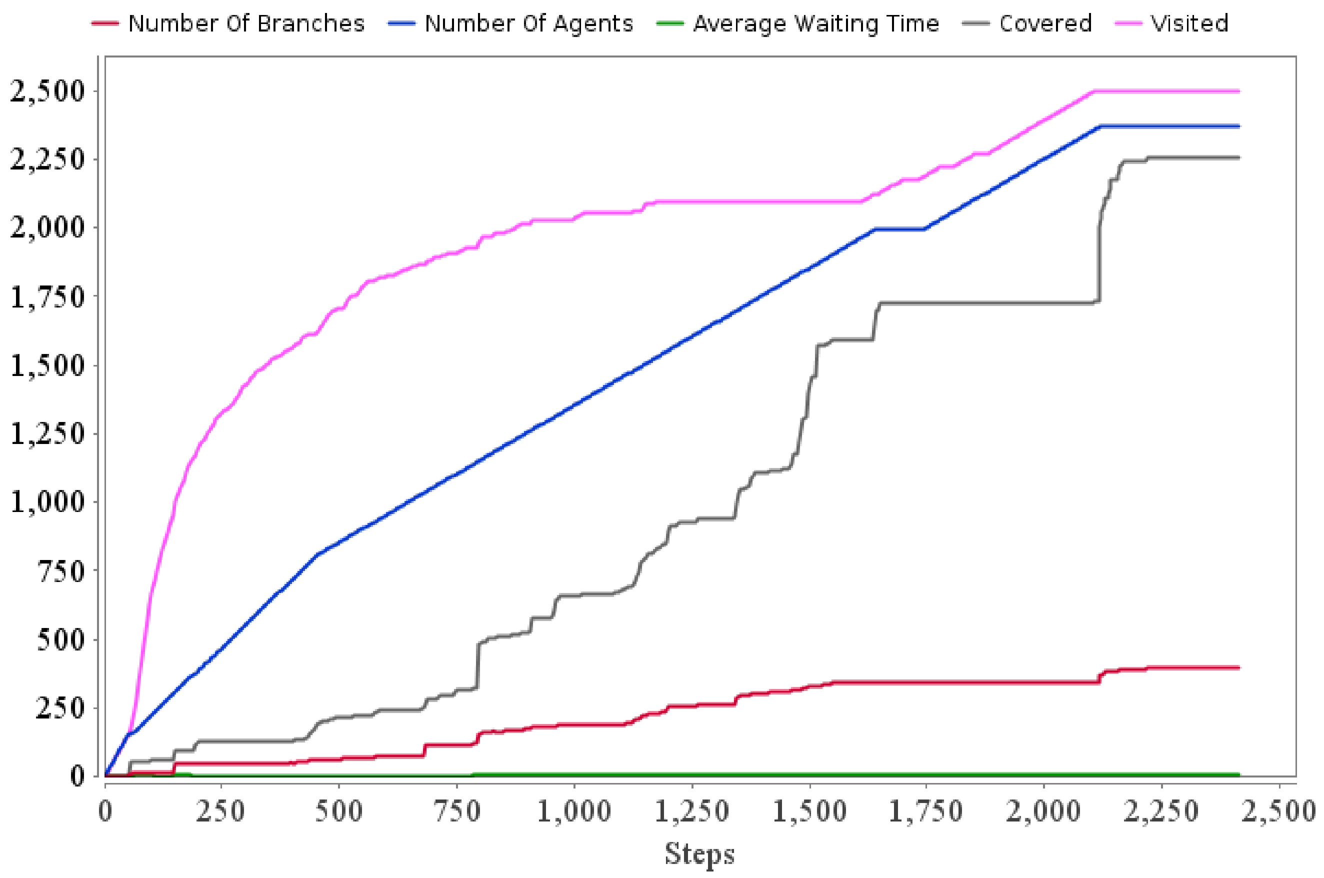

Figure 10: Smart steering algorithm results with the relevance number of agents in blue, coverage in gray, Average waiting time of all agents at each step in green, number of branches in red and visited locations in pink all in an empty environment of (50 x 50) locations.

Using the main model which is hug-the-wall steering algorithm we achieved complete coverage of the environment in time less than A (A is the area of the environment in pixels) and it was completely visited in time less than half of A, with a relatively very good average waiting time of the agents at each step. We can notice the constant increase in the number of agents throughout the simulation which indicates no blocking of the door until the environment is fully covered.

6. Conclusions and Future Work

Due to the importance of the autonomous virtual swarm spreading in unknown environments and its various applications, this work proposes a promising approach that utilizes a multi-agent spreading algorithm. With the preliminary results we achieved a full coverage and visiting of the environment in a very competitive time relative to the area of the environment. The proposed model has many enhancements over others in terms of directions, control variables and zero direct communication among the agents. The proposed approach emphasizes covering the whole environment efficiently, reducing the waiting time and high level of applicability to different simulation problems.

Starting from the results of testing the proposed approach, possible future work aims to study the significance ranking of the control variable, improving the way the agents look ahead for possible directions, adding obstacles to the simulation environment, and analyzing the applicability of the approach over real problems as spatial maps or computer networks. Furthermore, an interesting application would be considering a distributed environment as the various important issues related to big data.

References

[1] C. Blum and D. Merkle, Eds., Swarm Intelligence: Introduction and Applications, Natural Computing Series. Springer, 2008.

[2] A. Colorni, M. Dorigo, and V. Maniezzo, “Distributed Optimization by Ant Colonies,” Varela, F. Bourgine, P. (eds.), in Proceedings of the First European Conference on Artificial Life, pp. 134–142. MIT Press, Cambridge, 1992.

[3] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” in Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995.

[4] Y. Wang, A. Liang, and H. Guan, “Frontier-based multi-robot map exploration using particle swarm optimization,” in Swarm Intelligence (SIS), 2011 IEEE Symposium on, 2011, pp. 1-6.

[5] D. Teodorovic, and M. Dell’orco, “Bee Colony Optimization-A Cooperative Learning Aproach to Complex Transportation Problems,” Advanced OR and AI Methods in Transportation, pp. 51–60, 2005.

[6] K. Passino, “Distributed Optimization and Control Using Only a Germ of Intelligence,” in: Proceedings of the 2000 IEEE International Symposium on Intelligent Control, pp. 5–13, 2000.

[7] T. Hsiang, E. Arkin, M. Bender, S. Fekete, and J. Mitchell, “Online dispersion algorithms for swarms of robots,” in Proceedings of the nineteenth annual symposium on Computational geometry. ACM, 2003, pp. 382-383.

[8] E. Sahin, “Swarm robotics: From sources of inspiration to domains of application," in Swarm Robotics, ser. Lecture Notes in Computer Science, E. Sahin and W. Spears, Eds. Springer Berlin Heidelberg, 2005, vol. 3342, pp. 10-20. Available: http://dx.doi.org/10.1007/978-3-540-30552-1_2

[9] T. Hsiang, and M. Sztainberg, “Pheromone-guided dispersion for swarms of robots,” in Nineteenth European Workshop in Computational Geometry, 2003.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems. New York, NY, USA: Oxford University Press, Inc., 1999.

[11] S. Damer, L. Ludwig, M. A. LaPoint, M. Gini, N. Papanikolopoulos, and J. Budenske, “Dispersion and exploration algorithms for robots in unknown environments,” pp. 62300Q-62300Q-10, 2006. Available: http://dx.doi.org/10.1117/12.668915

[12] R. Morlok, and M. Gini, “Dispersing robots in an unknown environment,” in Distributed Autonomous Robotic Systems 6, R. Alami, R. Chatila, and H. Asama, Eds. Springer Japan, 2007, pp. 253-262. Available: http://dx.doi.org/10.1007/978-4-431-35873-2_25

[13] E. Ferranti, N. Trigoni, and M. Levene, “Brick mortar: an on-line multi-agent exploration algorithm,” in Robotics and Automation, 2007 IEEE International Conference on, 2007, pp. 761-767.

[14] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, “Agent-based simulation platforms: Review and development recommendations,”SIMULATION, vol. 82, no. 9, pp. 609-623, 2006. Available: http://sim.sagepub.com/content/82/9/609.abstract

[15] J. Barbosa, and P. Leitao, “Simulation of multi-agent manufacturing systems using agent-based modelling platforms,” in Industrial Informatics (INDIN), 2011 9th IEEE International Conference on, 2011, pp. 477-482.

[16] S. Luke, C. Revilla, L. Panait, K. Sullivan, and G. Balan, “Mason: A multiagent simulation environment,” Simulation, vol. 81, no. 7, pp. 517-527, Jul. 2005. Available: http://dx.doi.org/10.1177/0037549705058073

[17] S. Luke, Essentials of Metaheuristics, 2nd ed. Lulu, 2013.

[18] M. Gardner, “Mathematical Games: The Fantastic Combinations of John Conway’s New Solitaire Game Life,” Scientific American, vol. 223, pp. 120–123, 1970.

Shadi Alian received the B.Sc. degree in Computer Science from Yarmouk University, Irbid, Jordan, in 2004, then he was awarded a merit-based scholarship to continue his M.Sc. degree in Computer Science from Northeastern Illinois University, Chicago, Illinois, USA, in 2007. He is, at present, lecturer at The University of Jordan, Aqaba, Jordan, since 2010. His research interests include Multi-agent algorithms, Mutation testing, automatic test data generation and natural language processing.

Nazeeh Ghatasheh received the B.Sc. degree in computer information systems from The University of Jordan, Amman, Jordan, in 2004, then he was awarded a merit-based full scholarship to continue his M.sc. Degree in Electronic Business Management and the Ph.D. Degree in Electronic Business from the University of Salento, Lecce, Italy, in 2008 and 2011 respectively. He conducted research activities in the aerospace field related applications, information and communication technologies in the telecommunication industry, and corporate knowledge management. His research interests include image processing and its applications, knowledge representation and management, corporate learning, and e-Business. Dr. Ghatasheh, at present, is an assistant professor at The University of Jordan, Aqaba, Jordan, since 2011.

Mua’ad Abu-Faraj received the B.Eng. degree in Computer Engineering from Mu’tah University, Mu’tah, Jordan, in 2004, the M.Sc. degree in Computer and Network Engineering from Sheffield Hallam University, Sheffield, UK, in 2005, and the M.Sc. and Ph.D. degrees in Computer Science and Engineering from the University of Connecticut, Storrs, Connecticut, USA, in 2012. He is, at present, assistant professor at The University of Jordan, Aqaba, Jordan. He is currently serving as reviewer for the IEEE Micro, IEEE Transactions on Computers, Journal of Supercomputing, and International Journal of Computers and Their Applications (IJCA). His research interests include computer architecture, reconfigurable hardware, image processing, cryptography, and wireless networking. Dr. Abu-Faraj is a member of the IEEE, ISCA (International Society of Computers and their Applications), and JEA (Jordan Engineers Association).