applied sciences

MDPI

Article

Intelligent Exploration Approaches Based on Utility Functions Optimization for Multi-Agent Environment Applications

José Oñate-López, Loraine Navarro, Christian G. Quintero M. * and Mauricio Pardo

Department of Electrical and Electronics Engineering, Universidad del Norte, Barranquilla 081007, Colombia; jaonate@uninorte.edu.co (J.O.-L.); lorainen@uninorte.edu.co (L.N.); mpardo@uninorte.edu.co (M.P.)
* Correspondence: christianq@uninorte.edu.co

Abstract: In this work, the problem of exploring an unknown environment with a team of agents and search different targets on it is considered. The key problem to be solved in multiple agents is choosing appropriate target points for the individual agents to simultaneously explore different regions of the environment. An intelligent approach is presented to coordinate several agents using a market-based model to identify the appropriate task for each agent. It is proposed to compare the fitting of the market utility function using neural networks and optimize this function using genetic algorithms to avoid heavy computation in the Non-Polynomial (NP: nondeterministic polynomial time) path-planning problem. An indoor environment inspires the proposed approach with homogeneous physical agents, and its performance is tested in simulations. The results show that the proposed approach allocates agents effectively to the environment and enables them to carry out their mission quickly.

Keywords: exploration algorithms; multi-agent systems; coordination mechanism; search-target problem; intelligent task allocation

Citation: Oñate-López, J.; Navarro, L.; Quintero M., C.G.; Pardo, M. Intelligent Exploration Approaches Based on Utility Functions Optimization for Multi-Agent Environment Applications. *Appl. Sci.* **2021**, *11*, 2408. https://doi.org/10.3390/app11052408

Academic Editor: Manuel Armada

Received: 18 January 2021
Accepted: 22 February 2021
Published: 9 March 2021

1. Introduction

Research on autonomous agents can produce a revolutionary effect in exploration schemes like the ones developed by search/rescue teams in accidents, natural disasters, or military tasks. This kind of activity can be automated or considered a mobile-robotic problem where it is needed to search an unknown number of targets in a static environment [1]. There are several applications in this field, such as planetary exploration [2], surveillance [3], rescue [4], or cleaning [5], in which the complete coverage of terrain is part of the inherent goal of a mission.

The goal to tackle the environment exploration problem lies in determining how an agent should move to obtain as much new information as possible [6]. A proper exploration algorithm should have two properties, completeness and effectiveness. Completeness requires that the agents cover most of the environment, while effectiveness means that the agents achieve completeness by minimal efforts (i.e., exploration time, power consumption, etc.). In particular, a good exploration algorithm for the so-called graph-like environment corresponds to finding the shortest round trip through all nodes of the graph, which is the well-known traveling salesman problem (NP-hard for known graph-like environments) [7].

In static environments; since the targets are not moving, the target-searching problem is comparable to the exploration problem. Here, a target represents an unknown situation in the environment that the agents require to find complete the exploration tasks. As the agents cover more areas of the environment, they can find the targets. Hence, if the agents can achieve complete coverage; then, they can find all targets.

Typically, there are several benefits of using multiple agents over single-agent systems [8]. First, cooperating agents can complete a single task more quickly than a single agent [9]. In addition, multiple agents may be able to locate targets more efficiently if they exchange certain information such as positions and regions previously explored; and

therefore, avoiding re-exploration [10]. Finally, it is expected that redundancy and greater fault-tolerance will be gained by using multiple agents, while keeping the system simpler than a single powerful and advanced agent.

The basic idea in exploration algorithms is to identify the boundary of a covered area in a map or frontier; and then, select the appropriate frontier for the agent to move. By crossing frontiers, the known area increases accordingly. However, there exist two main related challenges. First, selecting the proper frontier to maximize the information gain in a multiple frontier problem; second, how an agent can efficiently and safely approach a frontier.

The present work looks for the proper handling of the first challenge. Thus, an intelligent approach is presented for coordinating several agents using a market-based model considering the three tasks to accomplish the agent job: knowing the environment, making decisions, and interacting socially in a multi-agent system. For executing the three tasks, an agent is limited by available information, time, and hardware-processing specifications. Moreover, three cognitive styles can be identified for an agent. First, the Satisfier who simply tries to find a solution that is "good enough". A satisfier selects the first option that fulfills the problems at hand without analyzing if such a solution is optimal. Second, the Maximizer who tries to make an optimal decision. A maximizer tends to take more time to decide due to the need to carefully maximize the performance for all variables and make the tradeoffs. The utility function approach is optimized with genetic algorithms for maximizers. Finally, the Intelligent who may also learn or use knowledge to achieve its task. An intelligent seeks to use little information to make good decisions based on the agent introspection and social requirements. The utility function model using artificial neural networks with low computational cost inputs is proposed for the intelligence.

This work compares different exploration algorithms for agents in a simulated indoor environment to challenge selecting the proper crossing frontier to maximize the information gain. Random algorithm and nearest frontier-based exploration are implemented for satisfiers. The utility function approach is optimized with genetic algorithms for maximizers. The utility function model using artificial neural networks with low computational cost inputs is proposed for the intelligence. This paper presents several simulations to explore the proposed approach properties, and comparisons with related approaches are provided. It can be observed that the intelligent cognitive style reduces significantly the time required to cover an unknown environment with a team of agents. The present paper is structured as follows: Section 2 presents the related work; Section 3 deals with the proposed approach; Section 4 focuses on the implementation; Section 5 describes system test and experimental validation and finally, Section 6 addresses conclusions and future work.

2. Related Work

The design of intelligent exploration algorithms and coordination mechanisms for multi-agent systems applied to target searching includes different multi-agent coordination approaches for environment exploration.

According to the occupancy maps, the most common deliberative technique is the frontier-based exploration algorithm [11]. The technique is based on identifying the boundary or frontier of the map-covered area; and then, select the proper frontier where the next agent should move. When the target frontier is assigned to an agent, the goal is to approach the frontier using the least number of moves while avoiding obstacles. To effectively plan the path to reach the frontier, probability-based algorithms, such as Probabilistic Road Map and Rapid Exploring Random Trees, can be used [12]. This path planning is a Non-Polynomial (NP) hard problem, and it can require heavy computations in cluttered environments. In addition, the development of an exploration algorithm based on a utility function to make effective decisions by a single robot has been explored. For example, the utility function can be designed to achieve new navigation goals by the agent, considering short distances and higher chances to increase the knowledge about the map. Some param-

eters (factors) used to enhance the implementation of these algorithms, and therefore, the agent decision-making, are cluster, distance, clearance, and unreachable-points [13].

Single robot exploration has been expanded towards the implementation of cooperative exploration. To enable such exploration using frontier-based algorithms, multiple agents must share their local maps to find together the global frontiers. If the agents can be located within the environment, they can share and merge their findings by summing or multiplying the state values in each local map [14]. However, if the location is not accurate enough, the agents must use probabilistic algorithms to merge the local map information. For instance, particle filters can support a group of agents to merge local maps under the agents uncertainties [15]. Furthermore, the agents can negotiate to allocate frontiers to more suitable agents when the local maps are merged into a global map. In [16], a potential field-based algorithm makes it possible for each agent to select the closest frontier to approach. In addition, to avoid an agent from attempting to move to a nearby but inaccessible frontier, a higher priority is given to a visible frontier. An optimal frontier assignment algorithm allows the agents to select frontiers sequentially. Once an agent has selected its destination frontier, this frontier relative weight will decrease so that the next agent will no longer select this frontier [17].

In the literature, the random search is most frequently applied for target searching [18]; however, it is key to coordinate agents to search efficiently for multi-agent systems. For instance, by creating a special pattern following it, the agents have better sense/sensitive coverage and can avoid re-exploring the area that has already been covered [19].

Several projects involve multi-agent cooperation architectures. For instance, Grabowski et al. consider teams of miniature agents who overcome the limits imposed by their small scale by exchanging mapping and sensor data [20,21]. As a part of this plan, a team leader incorporates the information collected by the other agents. In addition, it orders the other agents to bypass obstacles or direct them to unknown areas. Mataric and Sukhatme review different task assignment strategies in agent teams and analyze the team performance through extensive experimentations [22]. Burgard et al. consider the problem of a collaborative exploration of an unknown environment by multiple agents [23] but restricting the agents movement so that two agents do not approach the same target position or visit a position in the visibility area of another. The algorithm considers at the same time the utility of frontier cells and the cost for achieving them. Coordination is achieved by trading off the utilities and the cost by reducing the utilities depending on the number of agents already going to a specific zone. Simmons et al. extend the approach presented in [24] by distributing the computation and by using a more sophisticated notion of expected information gain by considering current map knowledge and the individual agent capabilities. Berhault et al. consider coordinating a team of mobile agents visiting several targets in a partially unknown area [25]. Their approach is based on combinatorial auctions, where agents bid on target bundles instead of single targets, as is often the case in auction strategies for exploring unknown environments. The idea is to consider synergies among targets to optimize exploration. Billard et al. study the influence of communication, learning, and the number of agents in the task of mapping the target locations in a dynamic environment [26]. The research is based on a theoretical framework based on probabilistic modeling and analyzed via simulation and physical implementation. The results of multiple experiments are compared with those anticipated by the probabilistic model, and they agree that the probabilistic model is a good approximation of a multi-agent system. These results show an effective approach for learning target locations that often change.

Cooperative multi-agent systems are those in which multiple agents interact jointly to solve tasks or optimize utility [27]. For example, in [28] Shi. H et al. researched on a distributed cooperative strategy using pedestrian behavior. As a result of interactions among agents, the complexity of multi-agent problems can increase rapidly with the number of agents or their behavioral sophistication.

Recently, some authors have worked on new strategies to apply frontier-based exploration methods. For example, a frontier-based exploration concept and a multi-agent flood

algorithm have been used to obtain a better exploration in an unknown area [29]. In the case of [30], Gomez et al. have worked on combining frontier-based exploration concepts and map-building using semantic information. They propose a semantic frontier classification and selection considering a cost-utility function. Here, the semantic information is used to classify the frontier as a free area or transit area considering the map geometric characteristics. On the other hand, N. Mandoui et al. work on a frontier-based approach in which robots share their local frontier points instead of sharing the whole grip map to save communication bandwidth [31]. In this work, the robots perform dynamic cooperation in which they have some specific information about their mates, such as positions.

Some authors have worked on efficiency improvement of frontier-based exploration methods. For example, Fand and Ding combine a frontier point evaluation with random frontier points optimization (RFPO) and SLAM algorithms. The idea is to work with a multistep exploration, where the algorithms do not plan a global path from the current position, but instead establish a local exploration path size. Once an agent reaches the local exploration path size, the current optimal frontier point is reselected to avoid the robot from making repetitive decisions [32].

Other authors have focused on the integration of deep reinforcement learning techniques with existing frontier integration methods [33–37]. Some authors, such as H. Li, have proposed a deep learning-based decision algorithm that uses a neural network for learning [34]. In [35], Z. Chen et al. propose a deep learning approach for multi-agent exploration. In that work, they use agent multi-channel maps, known regions, unexplored areas, and obstacles as inputs of a convolutional neural network. Other works such as Hu, J. et al. [36] propose an approach that incorporates a technique called dynamic Voronoi partitions. This method reduces duplicated exploration areas by assigning different targets location to individual robots. In [37] Yu C. et al. worked on a distributed multi-agent deep reinforcement learning approach to cooperative strategies for multi-robot pursuit. In other cases, such as the work done by Y. Hou et al., a radial basis neural network is used to build a continuous occupancy grid map [38]. Moreover, the research conducted by Shrestha et al. implements a deep generative neural network model to predict unknown regions [39].

On the contrary to all approaches discussed above, this work aims to contribute to the performance evaluation of different cognitive levels in the decision-making process, emphasizing the need to develop intelligent algorithms that can take effective decisions with little information based on the modeling of optimal decision-makers.

3. Proposed Approach

Exploration can be defined as the process of selection and execution of actions to maximize environmental awareness; therefore, models of the physical environment can be acquired. For a proper model generation, the agent has to interpret its sensors outcomes to make correct inferences about the surroundings, which corresponds to the map-building problem. The accuracy of the built map depends on the agent location when mapping and acquiring suitable models of the environment. This fundamental problem in mobile robotics is called Simultaneous Localization and Mapping (SLAM). SLAM is defined as a problem with an iterative solving strategy; while an agent navigates in an unknown environment, it must incrementally build a map of its surroundings; and localize itself within the built map. Additionally, as exploration is carried out, the agent must choose its viewpoints to ensure that the sensory measurements contain new and useful data. Thus, the accuracy of the map depends on the choice of viewpoints during exploration.

In exploration applications, there is a trade-off between the amount of knowledge acquired and its acquisition cost. An explorer agent aims to obtain the maximum environment knowledge at a minimum cost (i.e., minimum time and/or power). This work proposes the design of a hybrid coordination mechanism based on the interaction of three actions for intelligent agents: communication, intelligent task allocation, and negotiation in a multi-agent system.

Communication is important in any multi-agent system. The purpose is to integrate a market-based task allocation algorithm with cooperative negotiation for agent task completion. Figure 1 shows a diagram based on an individual agent analysis to carry out an overall mission considering the environment information provided by itself and its teammates. There are different degrees of cooperation in a multi-agent system. The highest is "global cooperation," and corresponds to when an agent while making its own decision, keeps trying to maximize the global utility function that considers all agents actions in the system.

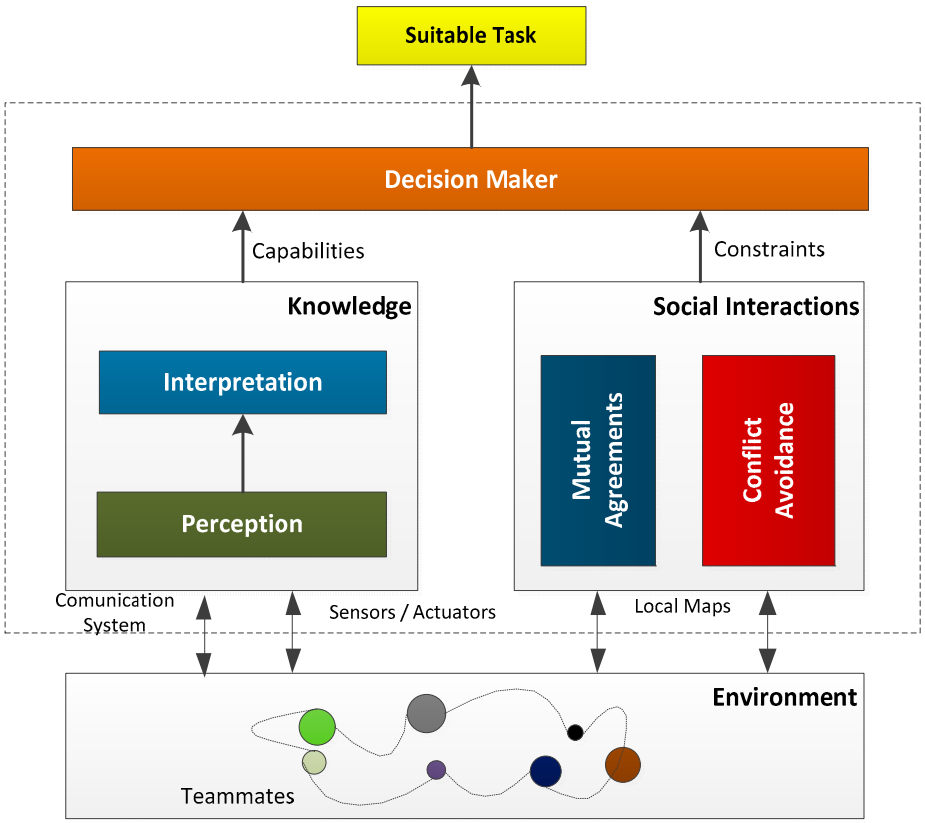

Figure 1. Proposed approach for agent decision model.

The process of finding an appropriate task for an agent depends on three subprocesses in this approach. For any agent, the suitability of an agent task (S_{Ti}) is proposed to be defined in terms of the Knowledge acquired from the teammates (K_{TMi}) and the Environment (K_{Ei}), the accuracy of the communications system (a_c) and the sensors (a_s), the results obtained with the Decision-making Algorithm (DM_{Ri}), and the constraints given by negotiation within the Social Interactions (SI_{Ni}). Each variable may change in time. This behavior determines the possibility of dynamic task allocation in the decision-making algorithm.

$$S_{Ti} = f(a_c K_{TMi}, a_s K_{Ei},\ DM_{Ri},\ SI_{Ni}) \text{ for each task i} \quad (1)$$

3.1. Knowledge

This process involves both perception and interpretation processes by the agent, allowing it to know its teammates and the environment. These processes consider location, map building, tasks in progress, and agent performance, and capabilities. Here, hardware developments (sensors and communications systems) are necessary to acquire accurate information. In this sense, the more accurate information is obtained in this process, the more computational-cost reduction is reached for the next level.

In deliberative exploration, a map of the environment is needed to help the agents finding the optimum path to fully and efficiently cover a region. A grid map can be

applied directly using sensor readings and location information. Hence, communication among agents is required to construct covered regions in the global map, know teammates locations, and memorize the grid cells already visited to avoid re-explorations.

3.2. Social Interactions

This process involves two or more agents making a joint decision. First, the agents verbalize the requests and then agree to a concession process or searching for new alternatives. The agreement can be reached through cooperative negotiation, where agents attempt to achieve the maximum global utility considering all their activities worth. It is possible to develop effective negotiation rules by exploiting the work with multiple agents.

When a task is assigned to an agent, conflicts or collisions among teammates arise; therefore, a negotiation algorithm is needed to reach an agreement that enables the negotiation process among agents. In this sense, the proposed negotiation algorithm depends on selecting the minimum cost decision from four approaches: waiting, teammate waiting, task switching, or new path assignment.

3.3. Decision Maker

This process is individual for each agent, and it is the intelligent method to obtain an optimal decision with minimal information. High intelligent performance at this level reduces the communication cost. Three cognitive styles characterize a decision-making algorithm: satisfiers, who try to find a "good enough" solution; maximizers, who try to make an optimal decision; and intelligent, who may also learn or use knowledge to achieve their goals. The core point of this process is how to coordinate agents to cover the environment efficiently. The primary purpose of the exploration is to cover the whole environment in a minimum of time. As a result, the agents must be aware of which areas of the environment have been explored previously. The task allocation implicates a new grid cell assignment where each agent should go and the path to get there.

The proposed intelligent approach uses a utility function to calculate the reaching suitability of each point. For this reason, a utility function U_i based on the trade-off between the costs of reaching the grid cell and its gain in coverage is proposed and shown in Equation (2). The proportional cost $P(C_{Ri})$, as in Equation (3), defines agent suitability as a function of the time required to move to a particular grid cell t_{Rj} compared to other agents in terms of maximum time to get there. This guarantees the best solution for the team and allows it to be implemented into a multi-agent system. The unexplored area size gives the proportional gain $P(G_{Ci})$ of grid cells that an agent can cover with its sensors when reaching a grid cell.

$$U_i = P(G_{Ci}) - P(C_{Ri}) \; for \; i = 1, 2, \ldots, \; n \tag{2}$$

where n is the number of grid cells in the environment.

$$P(C_{Ri}) = \frac{t_{Rj}}{\max(t_{R1}, t_{R2}, \ldots, \; t_{Rm})} \; for \; j = 1, \; 2, \; \ldots, \; m \tag{3}$$

where m is the number of agents in the environment.

4. Implementation

The proposed approach is implemented in a simulation platform for multi-agent systems for indoor environments. Figure 2 shows the simulation tool for the indoor environment. The indoor platform has a total area of 180 cm × 150 cm, divided into 30 grid cells of 30 cm × 30 cm. Each cell is identified with a number from 1 to 30. Obstacles and targets are positioned randomly with a maximum of three homogeneous agents.

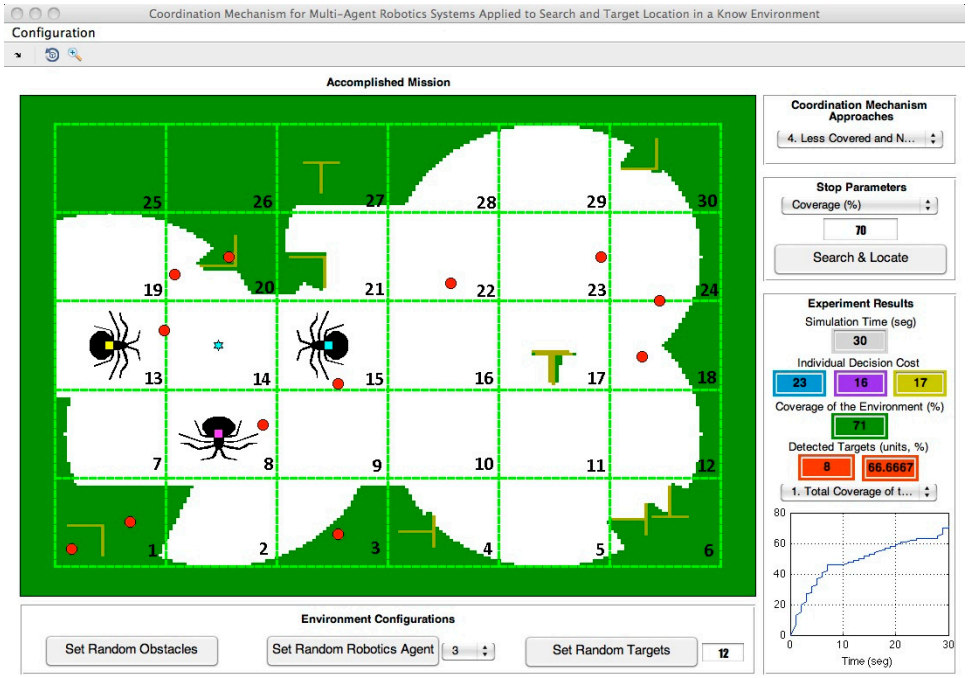

Figure 2. Simulation tool for indoor application.

Here, the modeling approach for the utility function defined in Equation (2) to allocate a new grid cell where an agent should go through the path to reach it is presented. In the proposed approach, the shortest path and cost to reach a particular grid cell are achieved by implementing a change to the Dijkstra's algorithm, where the cost of moving from one cell to another is dynamic due to agent orientation.

The Dijkstra's algorithm is a graph-based searching algorithm that solves the single-source shortest-path problem for a graph with nonnegative edge path costs, producing a shortest-path tree. A basic description of this algorithm is as follows.

1. Assign to every node a tentative distance value: set it to zero for the initial node and infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes consisting of all nodes except the initial node.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. Even though a neighbor has been examined, it is not marked as visited at this time, and it remains in the unvisited set.
4. When all of the current node neighbors are considered, the current node is marked as visited and remove from the unvisited set. A visited node will never be rechecked; its distance recorded now is final and minimal.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal), then the process stops. The algorithm has finished.
6. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to Step 3.

The following is a brief explanation of the most important exploration algorithms implemented with an example situation and the evaluation function values. In the graphs presented below (e.g., Figure 2) each agent is identified with a square of a color and the grid cell where has been assigned is indicated with a star of the same color. The next graphs (Figures 3–13) represent the blue agent decision function evaluation indicating the chosen cell.

Based on the knowledge acquired from teammates and the environment that each exploration algorithm needs to run, the Random Exploration algorithm does not use any information about the environment and teammates. This is the less intelligent algorithm. It looks for short response times, as discussed below. Nearest Cell Exploration and Farthest Cell Exploration need to run the shortest path algorithm for obtaining the cost to reach each cell. Frontier-based Exploration and Less Covered and Farthest Cell Exploration add to the decision-making process to construct the global environment coverage map to make decisions. For the proposed exploration algorithm, the Correlation Filter Exploration needs the global environment coverage image, and the Task Allocation based on Utility Function Approach (a neural-network-based approach) adds the teammates position to calculate the Euclidian distance. Besides, a genetic algorithm optimizes the utility function with heavy computation because it calculates cell by cell the actual coverage area expected and the proportional cost. It is necessary that each agent knows (and saves in memory) the teammates positions and paths, the global environment coverage, the travel cost to each cell, and the actual coverage gain obtained by visiting each cell. Thus, optimal decisions and multiple-agent coordination is ensured.

4.1. Exploration Algorithms

4.1.1. Random Exploration

A cell is selected randomly from all the possible ones that a robotic agent can reach and has not been visited. In random exploration, actions are randomly generated with a uniform probability distribution, independent of exploration costs or expected rewards. Strategies from the previous group, such as random exploration, distributed Boltzman exploration, and semi-uniform distributed exploration, do not use any exploration-specific knowledge and ensure exploration by merging randomness into action selection. The assignments of this approach are uniformly distributed in pseudorandom cells with the Mersenne Twister algorithm [40].

4.1.2. Nearest Cell Exploration

As the environment has 30 grid cells, the input to this algorithm is given by the cost to reach each unvisited cell but accessible from the agent initial cell. This cost is determined by the number of moves that the robot requires to reach a particular cell. Here the unit cost is 1 for 90-degree rotation, and four units for walking from one cell to the next one. This cost can represent the power consumption of the robot to produce these movements. Thus, this algorithm finds the cell that has a minimum cost to reach it (see Figure 3).

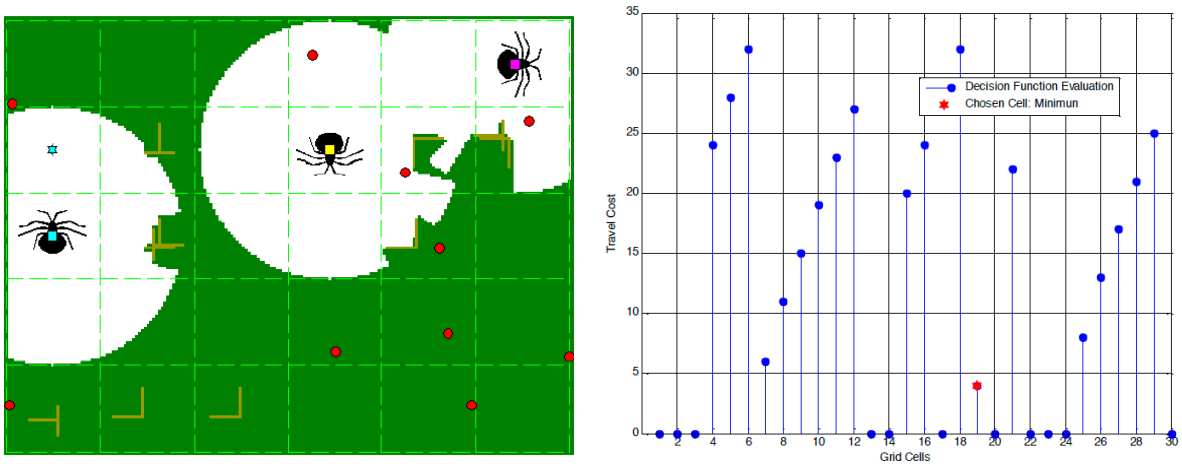

Figure 3. Example case and decision function evaluation for nearest cell exploration.

4.1.3. Farthest Cell Exploration

Similarly, this algorithm input is given by the cost to reach each unvisited cell but accessible from the agent initial cell; however, this algorithm finds the cell with the maximum cost. Even though it seems contradictory to use the maximum cost, the farthest cell use actually improves area coverage. It is important to note that this technique is only valid in known environments (see Figure 4).

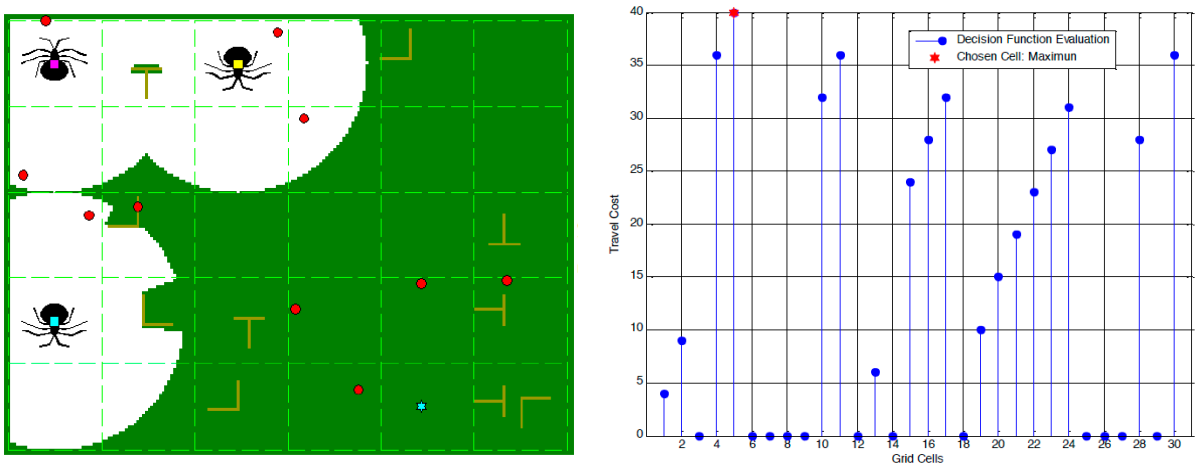

Figure 4. Example case and decision function evaluation for farthest cell exploration.

4.1.4. Frontier-Based Exploration

The idea of frontier-based exploration strategy is to detect borders between already explored environment regions and those regions where the agent has not acquired information yet. Hence, the agent looks for traversable regions in the on-going map construction and those adjacent to unexplored regions and holes in the map. This algorithm input is the sum of the proportional costs to reach from the origin cell to each cell where the robotic agent can reach and has not been visited, with the percentage of how much has been covered each cell. Thus, this algorithm finds the less covered and nearest cell that minimizes the cost function of Equation (4) (see Figure 5).

$$TaskAssignment_i = \min\left\{Coverage_{Cell_j} + Cost_{Ri}\left(pos_{ini}, angle_{ini}, Cell_j\right)\right\} \quad (4)$$

for each agent i and for $j = 1, 2 \ldots n$, where n is the number of unvisited cells with no obstacles in them.

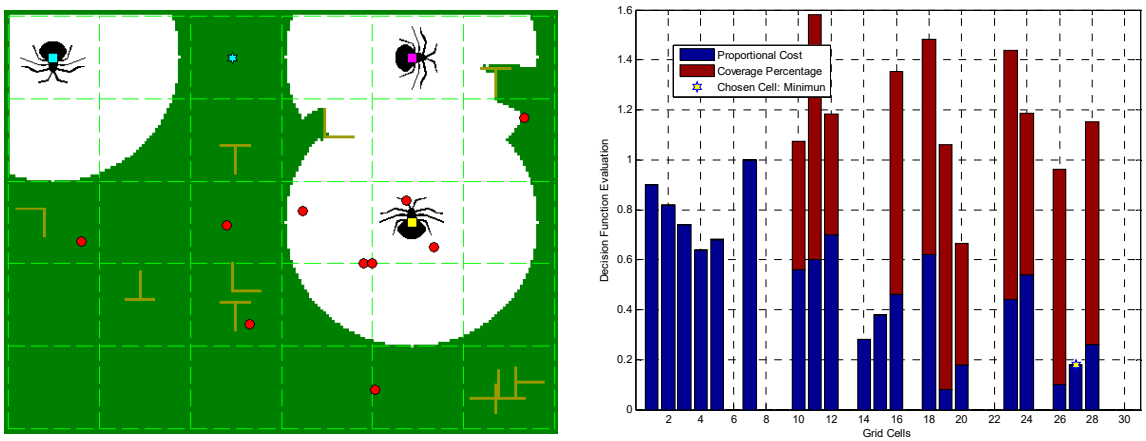

Figure 5. Example case and decision for frontier-based exploration.

4.1.5. Less Covered and Farthest Cell Exploration

This algorithm input is the sum of the proportional cost to reach each unvisited cell but accessible from the agent initial cell. The cost assignment goes from 0% to 100% as needed so that each cell is covered in its entirety. This algorithm finds the cell that maximizes the cost function of Equation (5) (see Figure 6).

$$TaskAssignment_i = \max\left\{Uncoverage_{Cell_j} + Cost_{Ri}\left(pos_{ini}, angle_{ini}, Cell_j\right)\right\} \quad (5)$$

for each agent i and for $j = 1, 2 \ldots n$, where n is the number of unvisited cells with no obstacles in them.

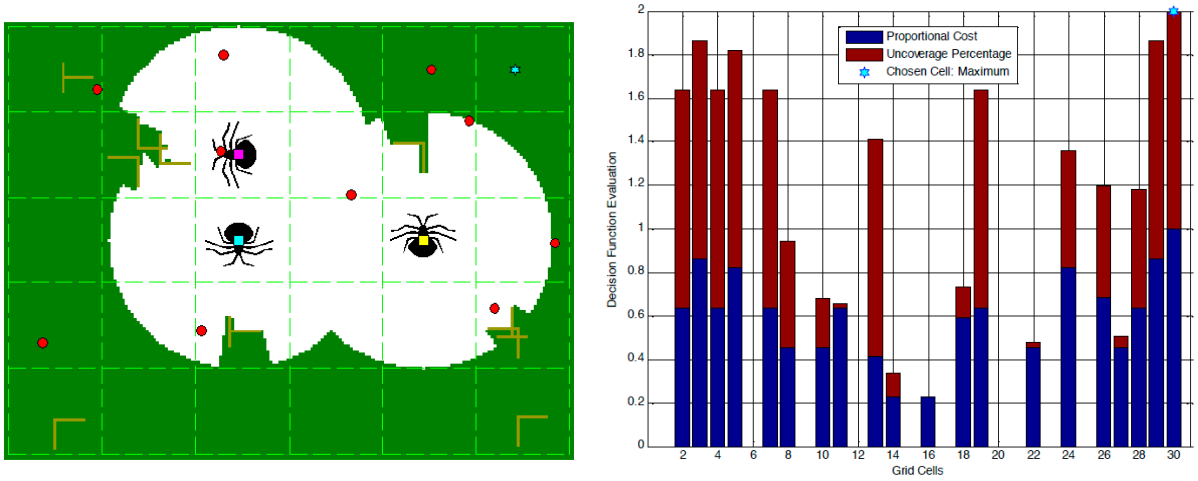

Figure 6. Example case and decision for less covered and farthest cell exploration.

4.1.6. Correlation Filter Exploration

This algorithm maximizes the use of the global coverage area construction by each agent. It is based on a binary image of the environment, where the pixels marked with black are positions that have been covered or where there are obstacles (see Figure 7). Using the region observed by the agent sensor as a correlation filter, it is possible to find different points where the maximum coverage is achieved, and that position is chosen since it has a lower cost to reach.

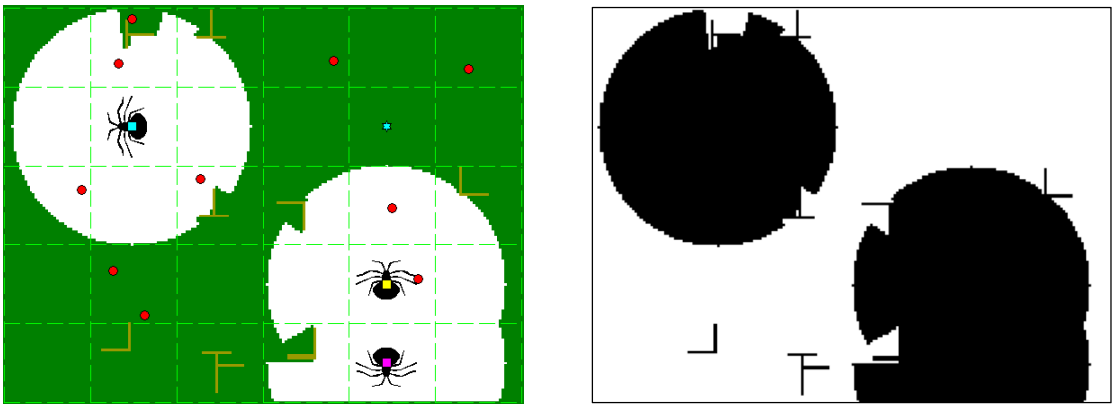

Figure 7. Example case and environment binary image for correlations filter exploration.

The basic idea in this approach is the correlation implementation in 2D. Given a square filter, the results of correlation can be computed by aligning the center of the filter with a

pixel. Then, all overlapping values are multiplied together and add up the result, as shown in Equation (6) as

$$Correlation\ Image(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} (CF(i,j) * EBI(x+i, y+j)), \quad (6)$$

where *CF* is the Correlation Filter of size 2 N × 2 N and *EBI* is the Environment Binary Image. Figure 8 presents an example in the simulation environment and the correlation levels between the environment and the sensor area.

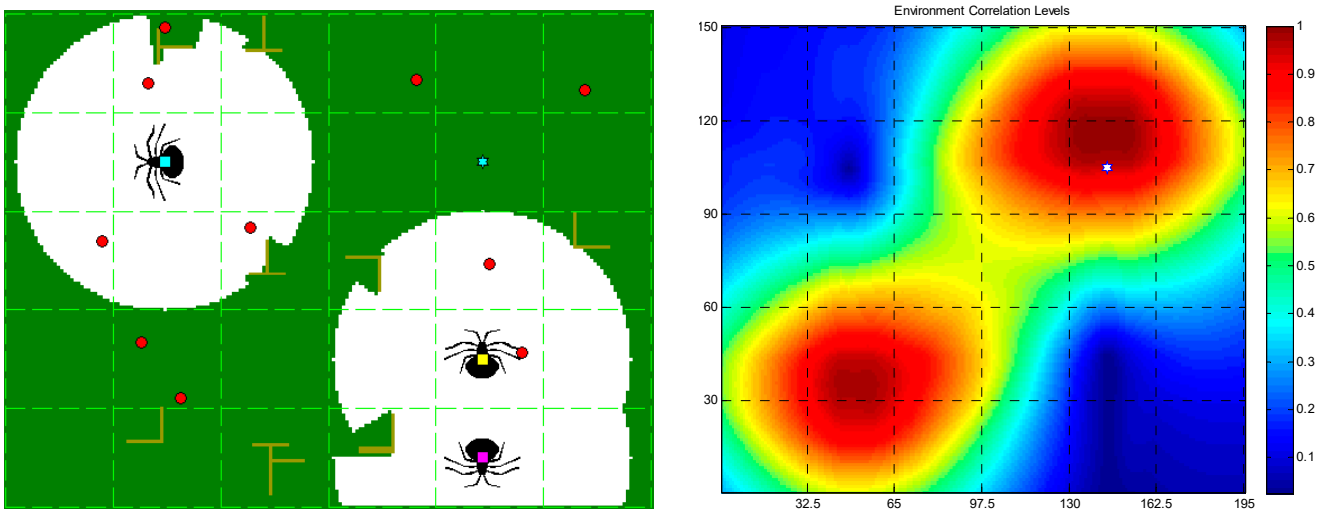

Figure 8. Example case and environment correlation levels and decision for correlations filter exploration.

4.1.7. Task Allocation based on Utility Function Approach

Two methods are used as input for obtaining the cost and gain without requiring high computational resources due to neural networks use to model the utility function. The Euclidian distance from each agent to the specific cell represents the cost. As a new approach, the gain is obtained with the correlation level with the image of the environment in a specific cell by applying the correlation filter described above. High correlation levels point out positions in the environment where the agent can obtain maximum coverage.

The utility function with extensive computation is used to train the neural network, obtaining good performance with feed-forward back-propagation network architecture, three hidden layers, and sigmoid tangential transfer functions. On the other hand, it is proposed to optimize the running of the utility function exploiting the parallel computing nature of genetic algorithms (GAs).

The implemented GAs have:

1. A genetic representation of the solution domain: populations or chromosomes are the grid cells not visited by the agents.
2. The fitness function to evaluate the solution domain is the utility function with extensive computation.
3. A standard representation of the solution is an array of bits with the size of the number of grid cells in the environment. The position set to 1, is the chosen cell. The main property that makes these genetic representations practical is that their components are easily aligned due to their fixed size, simplifying crossover operations.
4. The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions (usually randomly); and then, improves it through repetitive application of the selection, crossover, and mutation operators.

5. For this, the objective function to maximize is the utility function proposed in (2). Next, an example is presented for a task assignment process. Figures 9–12 show the proportional costs, gains, and utility values for the decision in the example case.

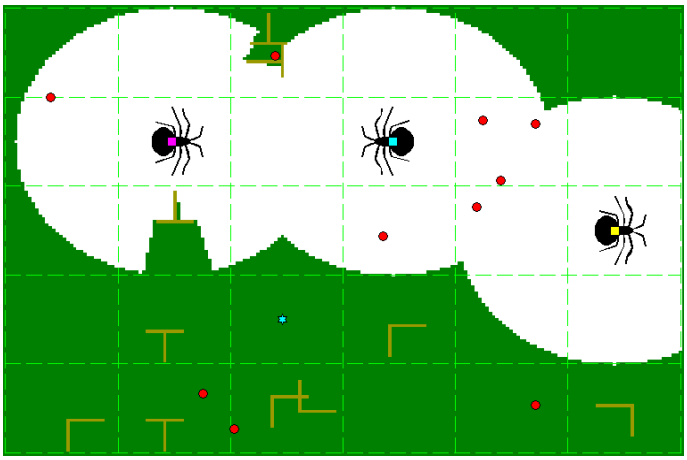

Figure 9. Example case for utility function approach.

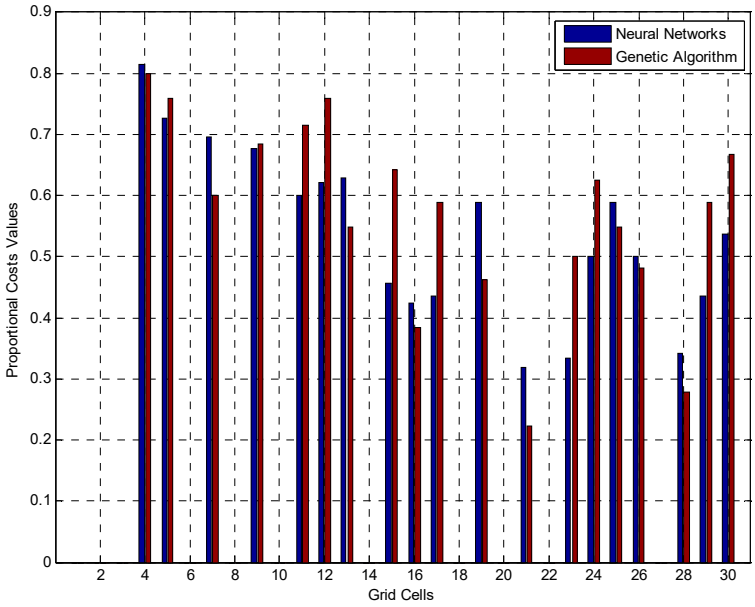

Figure 10. The Euclidian distance for neural networks and the proportional cost in (3) for GAs.

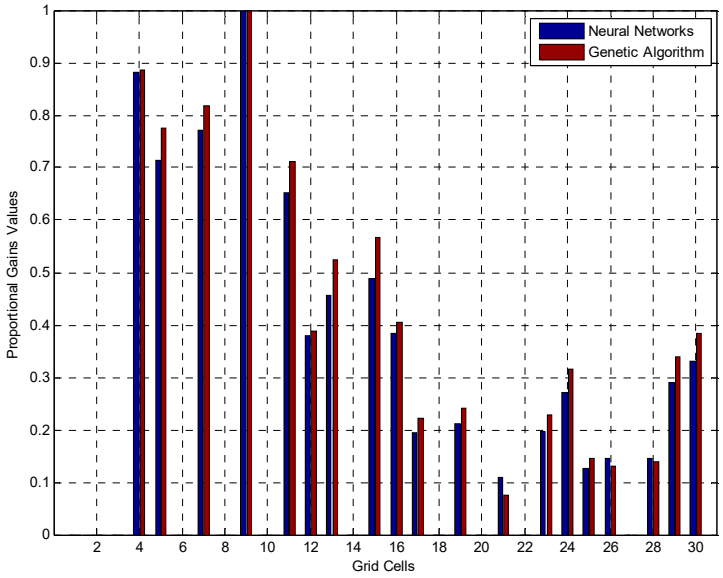

Figure 11. Correlations levels for neural networks and the proportional gain in (2) for GAs.

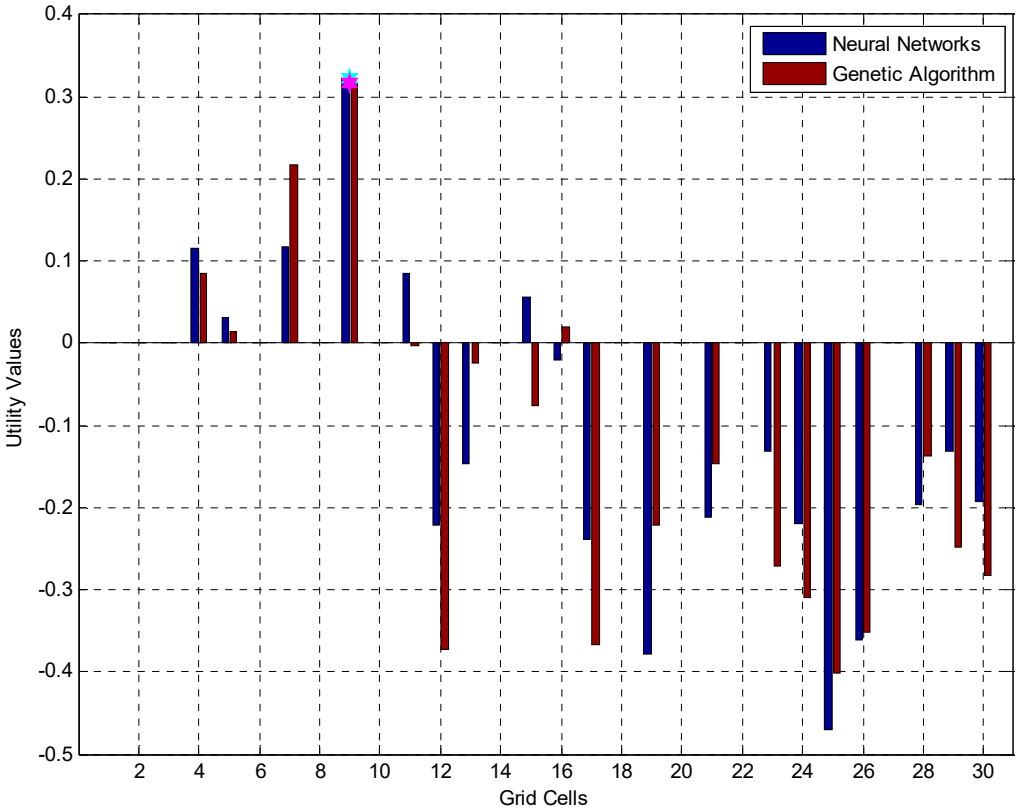

Figure 12. Utility values for each approach, indicating the chosen cell with the star.

Figure 12 shows how the modeled function with neural networks does not represent exactly the utility function with extensive computation optimized with GA, but the decision obtained is the same. Figure 13 shows the need to model and optimize the utility function given the response times of each algorithm. To illustrate this, 100 tests are performed of the decision in different environments for each algorithm obtaining the response time of each decision.

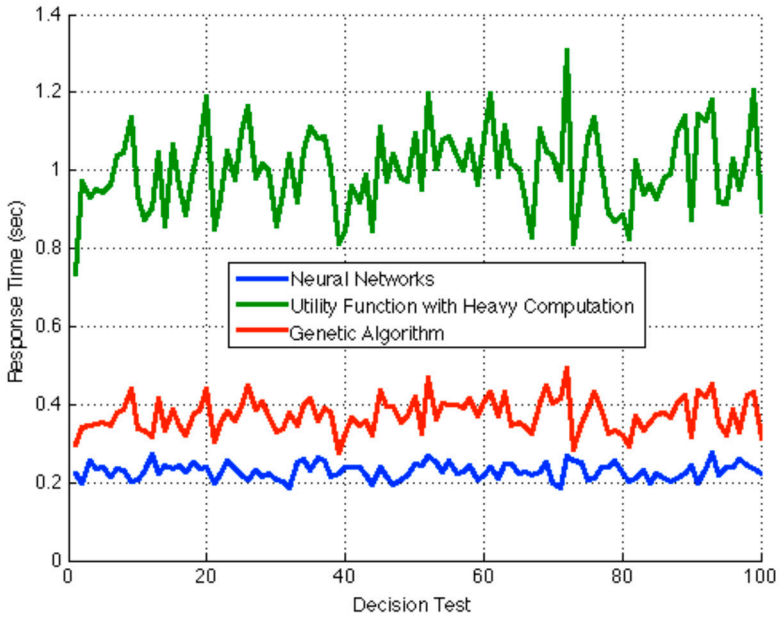

Figure 13. Response time of decisions for the utility function approach.

4.2. Knowledge Comparison Among Algorithms

To complement the decision-making algorithm implementation stage, based on (1), Table 1 presents the knowledge acquired from teammates and the environment that each algorithm needs to run. Previously, all algorithms know the map of the environment and the obstacle grid cells.

Table 1. Knowledge Comparison.

Decision-Making Algorithm	Knowledge Acquired from Teammates and the Environment
Random	Grid cells already visited.
Nearest Cell	Grid cells already visited. Travel cost to each cell.
Farthest Cell	Grid cells already visited. Travel cost to each cell.
Frontier Based Exploration	Grid cells already visited. Travel cost to each cell. Global coverage map.
Less Covered and Farthest Cell	Grid cells already visited. Travel cost to each cell. Global coverage map.
Correlation Filter	Grid cells already visited. Global coverage map.
Utility Function Modeled with Neural Networks	Grid cells already visited. Teammates positions. Global coverage map.
Utility Function Optimized with Genetic Algorithms	Grid cells already visited. Travel cost to each cell. Global coverage map.

4.3. Negotiation Algorithms

As explained in the proposed hybrid coordination mechanism, the negotiation algorithm is based on choosing the minimum cost decision (time to complete the task) from four approaches: waiting, teammate waiting, task switching, or new path assignment. Next, four example cases are presented to show the operation of each negotiation algorithm,

present the collision scene (the collision cell is marked with a yellow star), and report the obtained solution.

4.3.1. Waiting

The Waiting algorithm corresponds to keep the robotic agent still for a specific time until no collision are detected (see Figure 14).

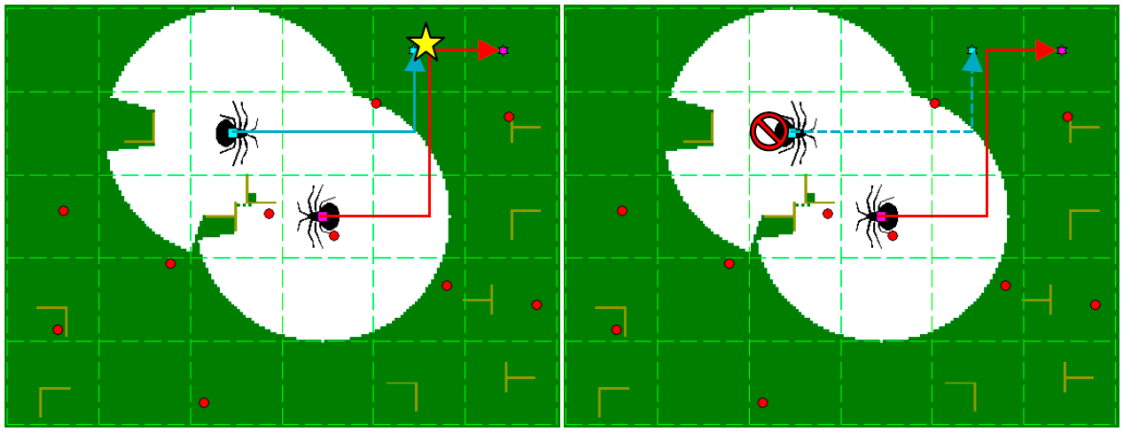

Figure 14. Collision scene and the best solution implementing waiting approach to minimize the time to complete the task for the cyan robot.

4.3.2. Teammate Waiting

The Teammate Waiting algorithm corresponds to keep the collision partner still for a specific time until no further collisions are detected (see Figure 15).

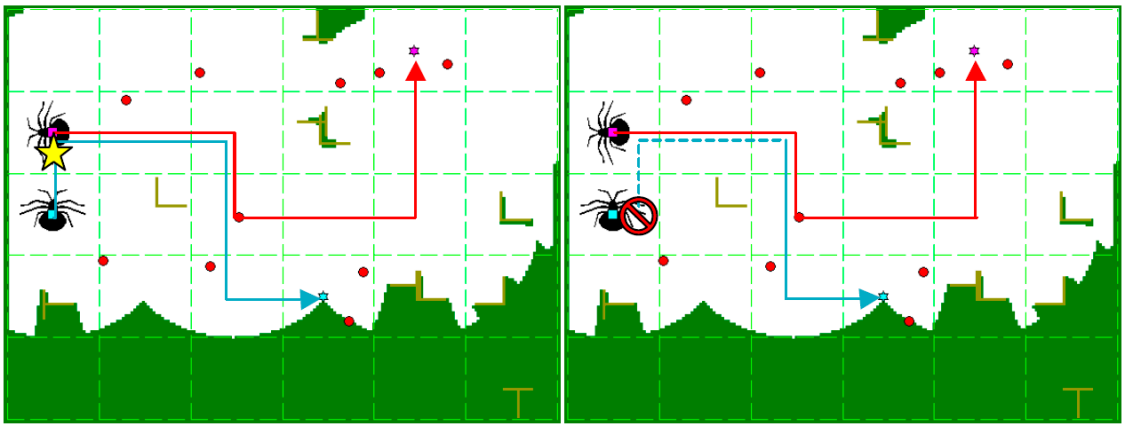

Figure 15. Collision scene and the best solution implementing the teammate waiting approach to minimize the time to complete the task by the magenta robot.

4.3.3. Task Switching

The Task Switching algorithm corresponds to exchange the cell to reach between colliding robotic agents. This approach does not permit to stop the exploration of any robotic agent increasing the possibility of gain coverage (see Figure 16).

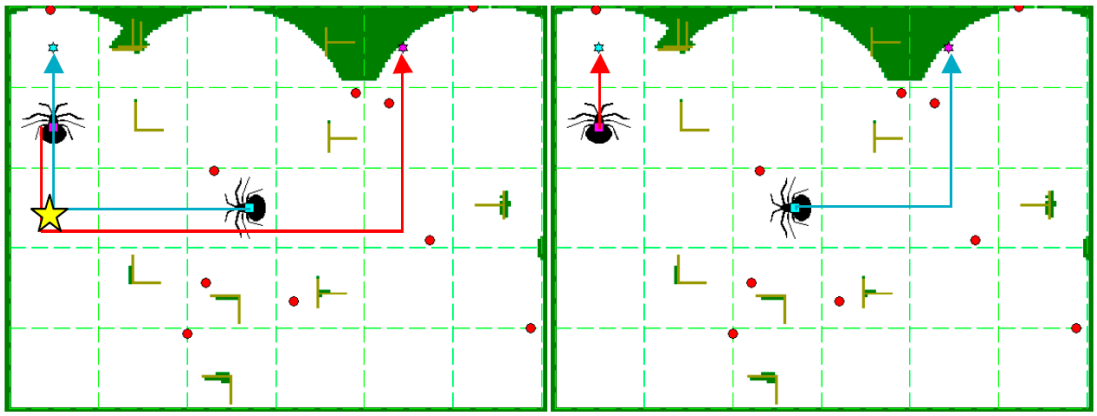

Figure 16. Collision scene and the best solution implementing the task switching approach to minimize the time to complete the task by the magenta robot.

4.3.4. New Path Assignment

This approach finds a new route to reach the assigned target cell in the task allocation process putting the collision cell as an obstacle. Since finding a new route cannot be assured, this approach not always provides a solution for a collision event (see Figure 17).

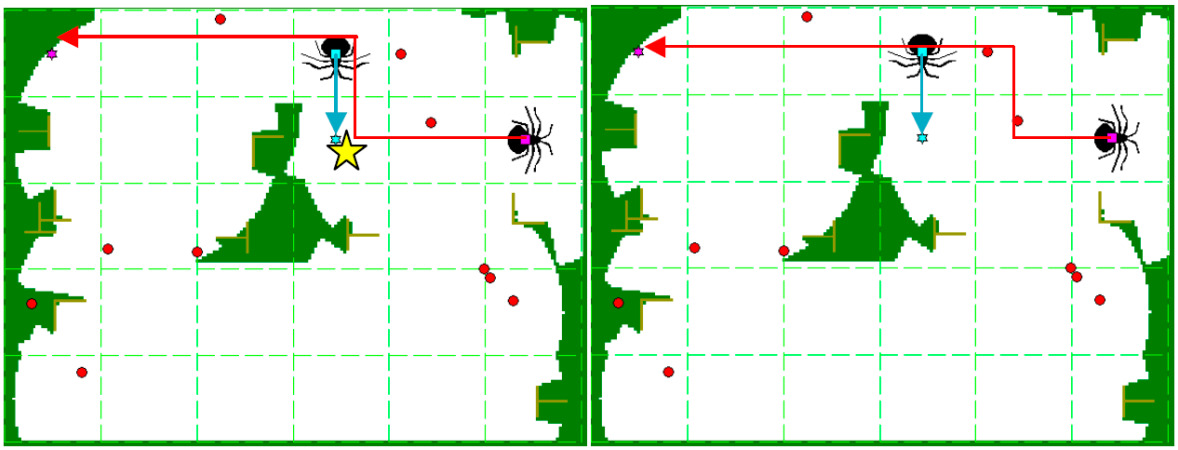

Figure 17. Collision scene and the best solution implementing the new path assignment approach to minimize the time to complete the task by the magenta robot.

5. Experimental Results

The approach is implemented in the simulation environment presented in Section 4. Each experiment involves exploring the environment by three agents to achieve a 97% environment coverage (29 out of 30 cells). Moreover, each experiment has been performed with a different and random configuration of the environment to make data collection independent. This includes different positions of the obstacles, targets, and agents.

Additionally, a series of experiments is performed to get a quantitative assessment of the proposed approach improvements over non-intelligence.

The experimental design is defined as follows:

Hypothesis: To which grid cell or node in the map should an agent move to minimize the time it takes to fully explore the environment and find the maximum number of possible targets given its teammates positions and the areas already explored? What is the exploration algorithm for achieving that objective?

Response Variables:

1. Response Time is the machine time in seconds for each algorithm to do the allocation process for the task.
2. Decision Cost represents the time required to complete the task.
3. Decision Gain is the percentage-gain coverage that is expected when the task is completed.
4. Social Interaction Level determines if a task assignment produces collision or not.
5. Coordination Level measures the capability of the exploration algorithm to use multiple agents. It is performed using the standard deviation of the individual cost for each agent.
6. Re-Exploration Level is the percentage of the re-explored environment when the coverage is 90%.
7. Exploration Time is the time when the agents reach 97% coverage of the environment, and it is measured in seconds.
8. Coverage Effectiveness is the percentage of the environment covered in 60 s.
9. Search Effectiveness is the percentage of targets found in 60 s.
10. Collision Avoidance Algorithm determines which is the algorithm selected when a collision occurs.

Factors: exploration algorithms. The proposed approach vs. previous non-intelligent techniques. For the non-intelligent techniques; there are considered (1) Random exploration, (2) Nearest cell exploration, (3) Farthest cell exploration, (4) Frontier based exploration, and (5) Less covered and farthest cell; and for the proposed approach, there are (6) Exploration based on correlation filters, (7) Utility function modeled using neural networks, and (8) Utility function optimized using GAs. Figure 18 shows the legend for each of the following figures that summarize the tested algorithms results.

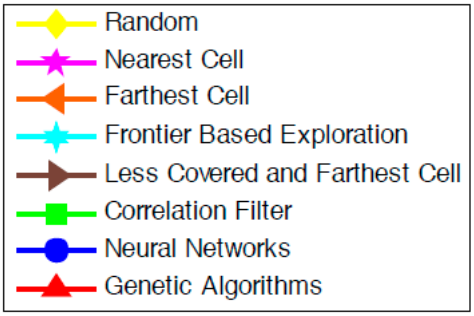

Figure 18. Legend employed for the results of each algorithm testing.

Three graphs are developed for some response variables: cumulative average, box plots, and multiple comparison test. In some cases, the cumulative average and the box plots show differences between the exploration algorithms, but it is necessary to implement a comparison method since some confidence intervals overlap. In particular, it is necessary to know which pairs of means are significantly different and not. A test that can give such information is known as a multiple comparison procedure. If applied a t-test, the alpha value would apply to each comparison; therefore, the chance of incorrectly finding a significant difference would increase with the number of comparisons. Multiple comparison procedures are designed to provide an upper bound on the probability that any comparison being incorrectly considered significant. To do this, Tukey's honestly significant difference criterion (HSD or Tukey-Kramer) is implemented together with an analysis of variance (ANOVA) to identify which means are significantly different from each other. In these graphs, two means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap.

About the response time variable, the three proposed exploration algorithms take a longer time to perform the decision-making process compared to the non-intelligent approaches, as seen in Figure 19. This proves the stated regarding the information required

for the execution of each algorithm. Based on Figure 19, all the non-intelligent approaches are not different statistically; but, according to the time response; correlation filter, neural networks, and genetic algorithm are statistically different.

The decision cost measures the average time needed to complete the task assigned by the exploration algorithm. Based on Figure 20, it is interesting to notice how the utility function optimized with a genetic algorithm is similar to nearest cell exploration. Furthermore, it is proved that the farthest cell exploration, and less covered and farthest cell exploration are the costliest decisions in terms of time and power consumption.

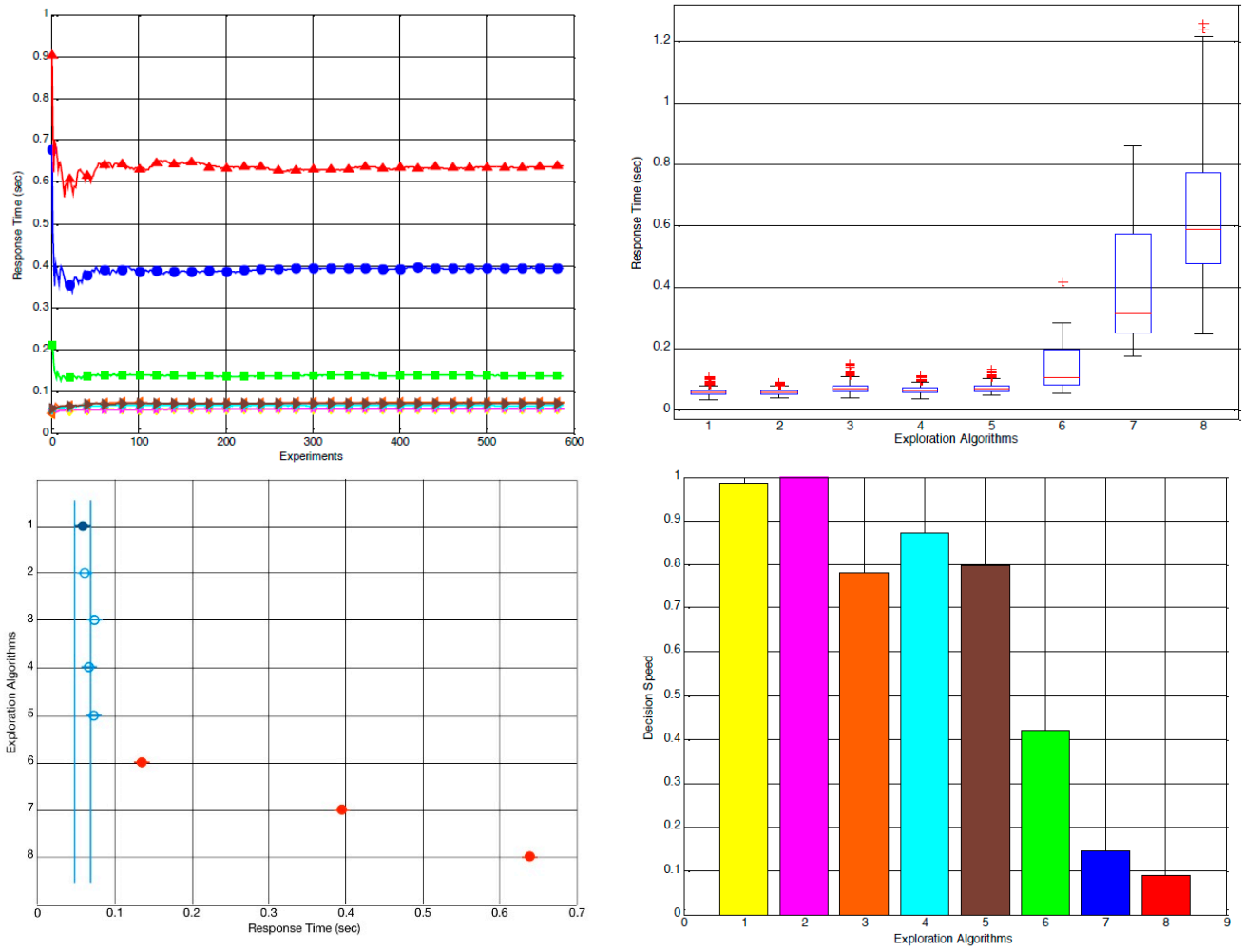

Figure 19. Cumulative average for response-time experiments. Box plot for response time experiments. Multiple comparison test for response time experiments. Response time ranking.

The decision gain measures the average coverage area obtained when the robotic agent completes the task assigned by the exploration algorithm. Based on Figure 21, it is clear to see how the decisions taken using the correlation filter algorithm obtain the most coverage gain compared to the other algorithms, proving its capacity to maximize the use of global coverage area construction by each robotic agent.

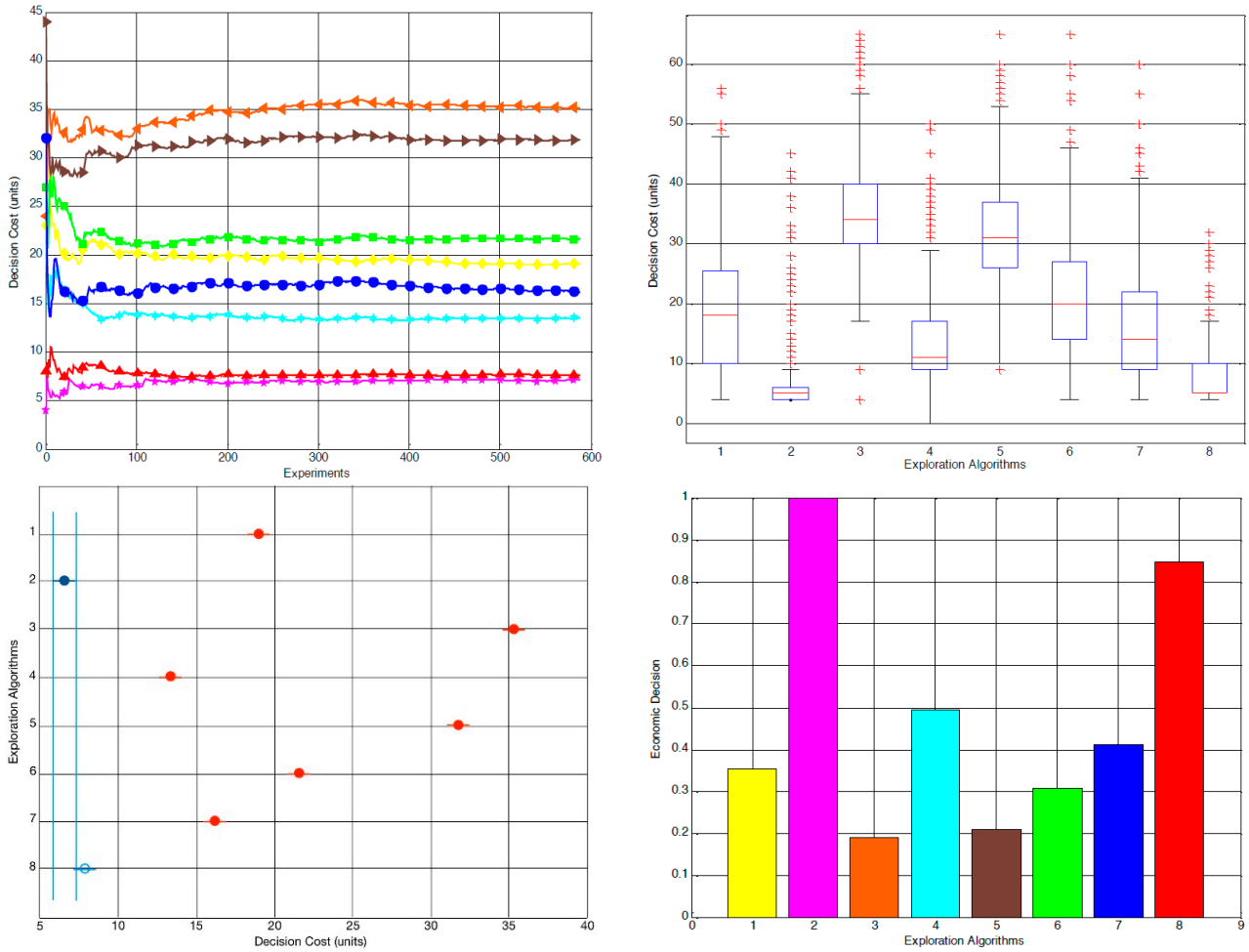

Figure 20. Cumulative average for decision cost experiments. Box plot for decision cost experiments. Multiple comparison test for decision cost experiments. Decision cost ranking.

The results obtained in the social interactions section define that an algorithm with a high rate of collision is not very sociable because their decisions affect their teammates. The utility function optimized with the genetic algorithm presents the lowest collision frequency, demonstrating that is it a sociable algorithm. The results presented in Figure 22 allow classifying the exploration algorithms by social levels.

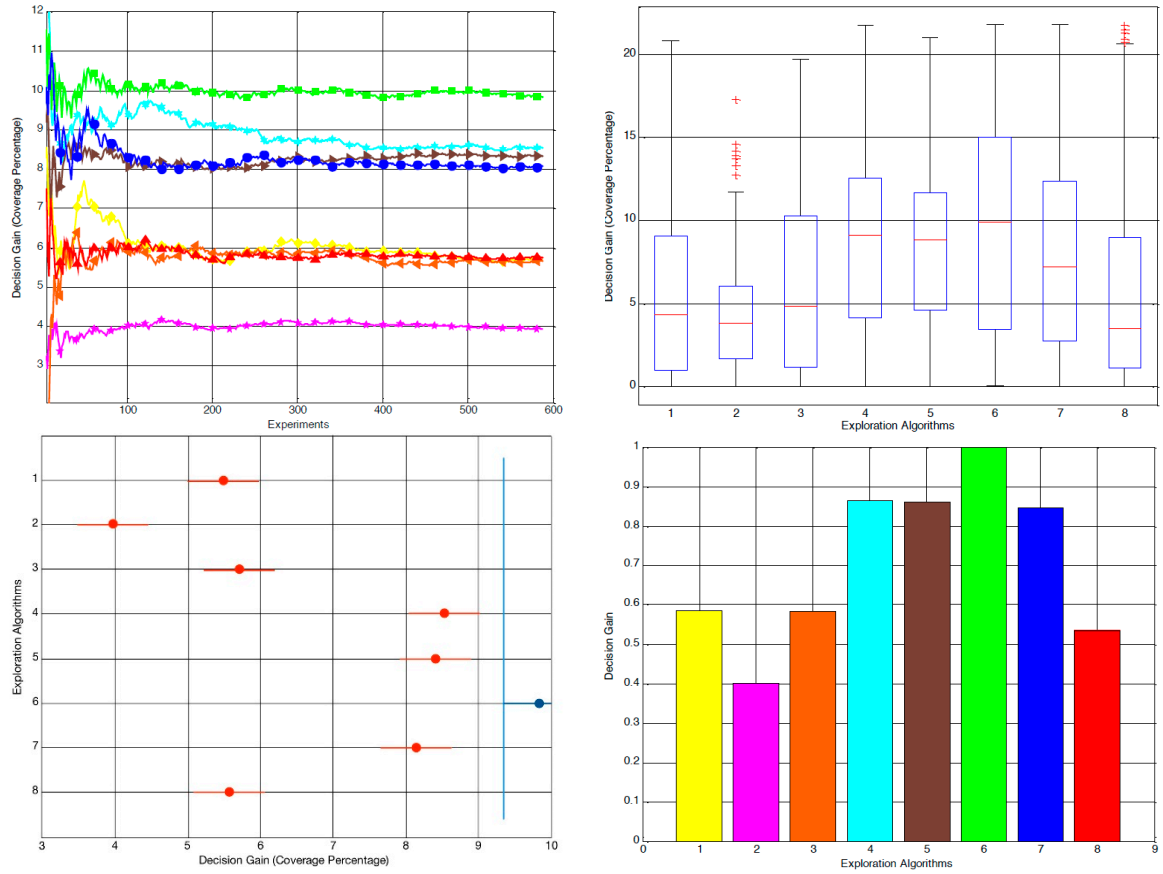

Figure 21. Cumulative average for decision gain experiments. Box plot for decision gain experiments. Multiple comparison test for decision gain experiments. Decision gain ranking.

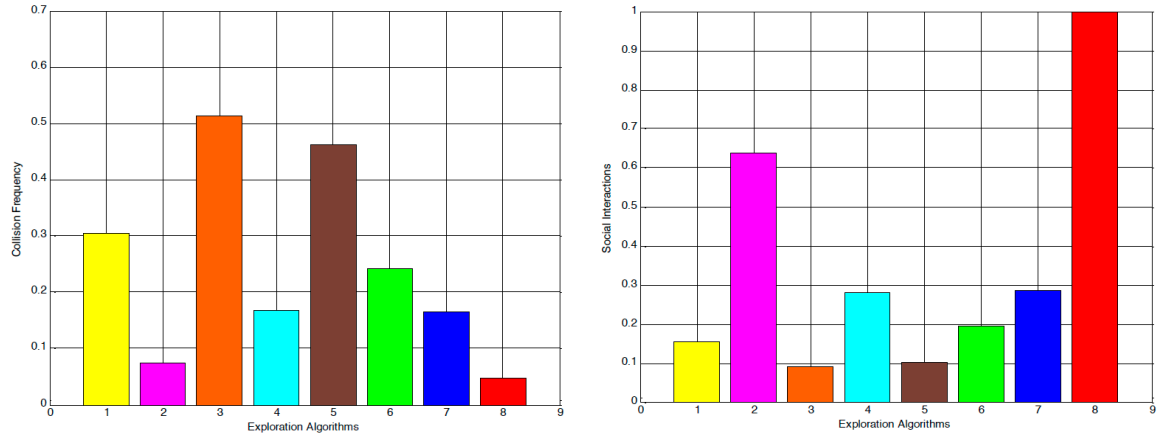

Figure 22. Collision frequency for each exploration algorithm. Social interaction ranking.

The coordination level (individual cost-time to complete the tasks) measures an algorithm ability to distribute the work evenly among robotic agents. Figure 23 shows that the algorithms that reach the farthest cell are the ones that present lower coordination. Again, the utility function optimized with a genetic algorithm and the nearest cell and frontier-based exploration presents the best coordination levels.

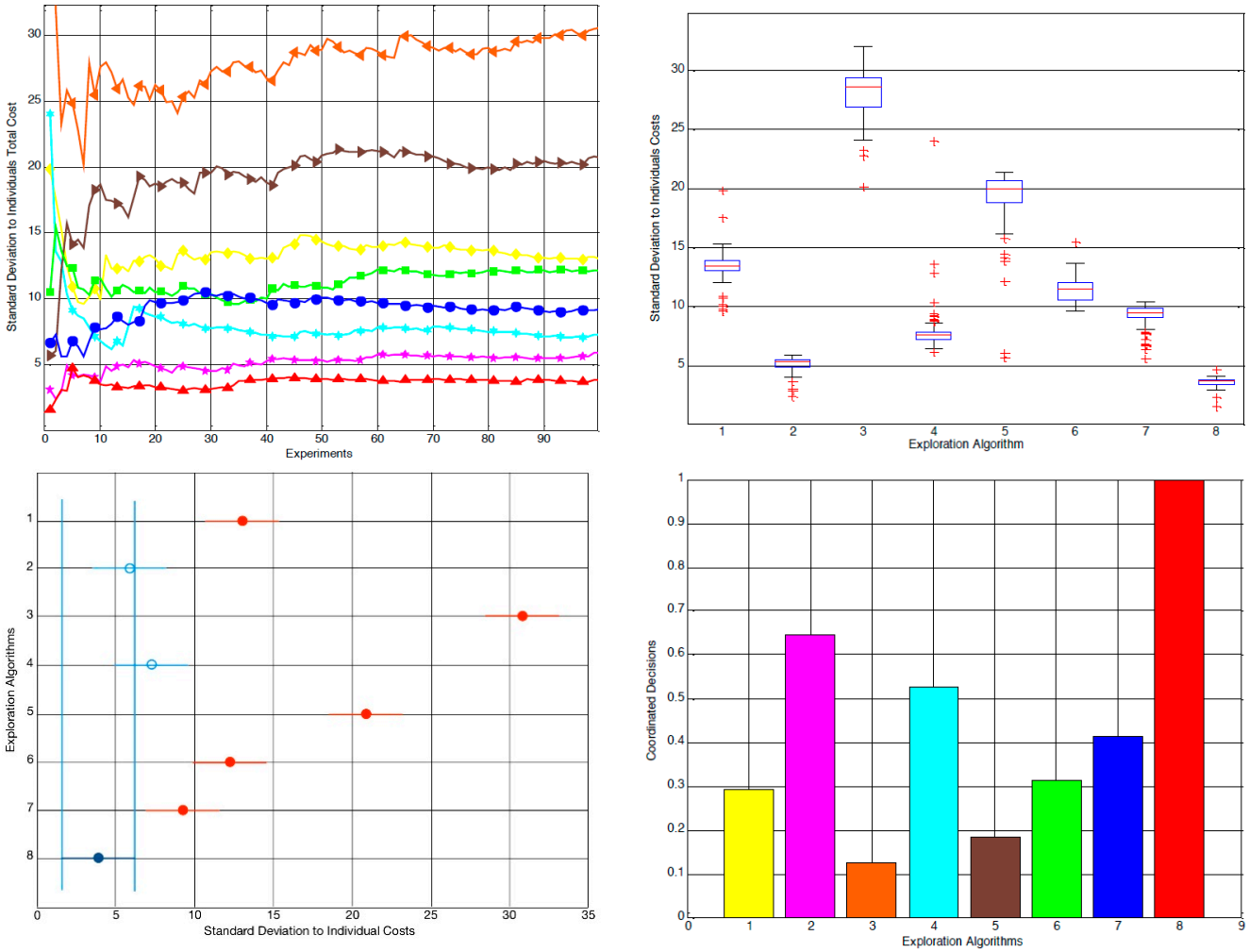

Figure 23. Cumulative average for coordination level experiments. Box plot for coordination level experiments. Multiple comparison test for coordination level experiments. Coordination ranking.

One of the most important features of exploration algorithms is avoiding re-exploration, because this is considered a loss of power and time by the robot team. Based on Figure 24, the supremacy of the utility function optimized with a genetic algorithm over the other algorithms is clear.

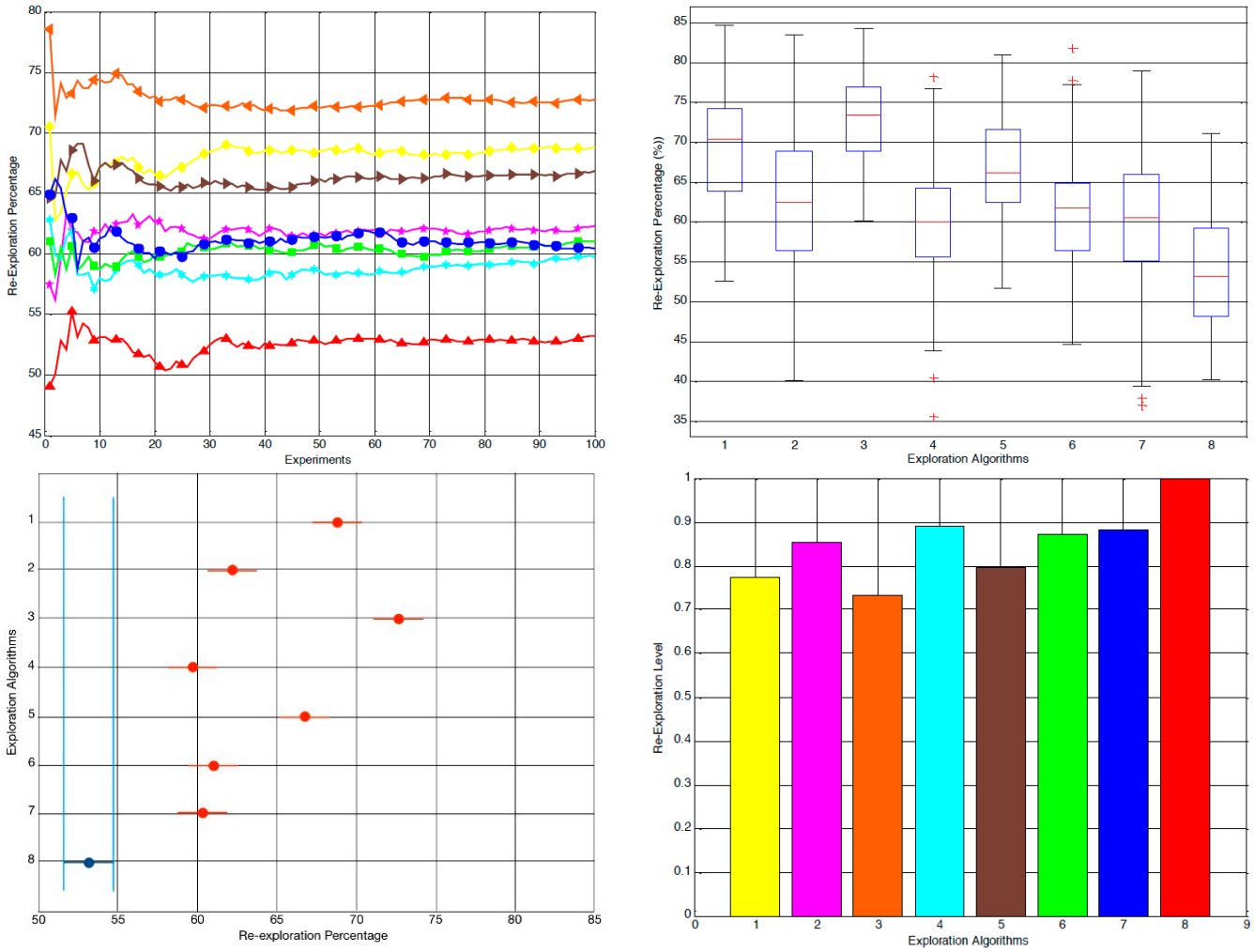

Figure 24. Cumulative average for re-exploration level experiments. Box plot for re-exploration level experiments. Multiple comparison test for re-exploration level experiments. Re-exploration ranking.

The most important goal in an exploration problem is time. If high coordination with three agents in the environment is required, Figure 25 shows that exploration algorithms that trade-off coverage gain and time cost to reach a specific grid cell present better performance. Moreover, farthest and random cell algorithms make re-exploration of the environment increasing the exploration time.

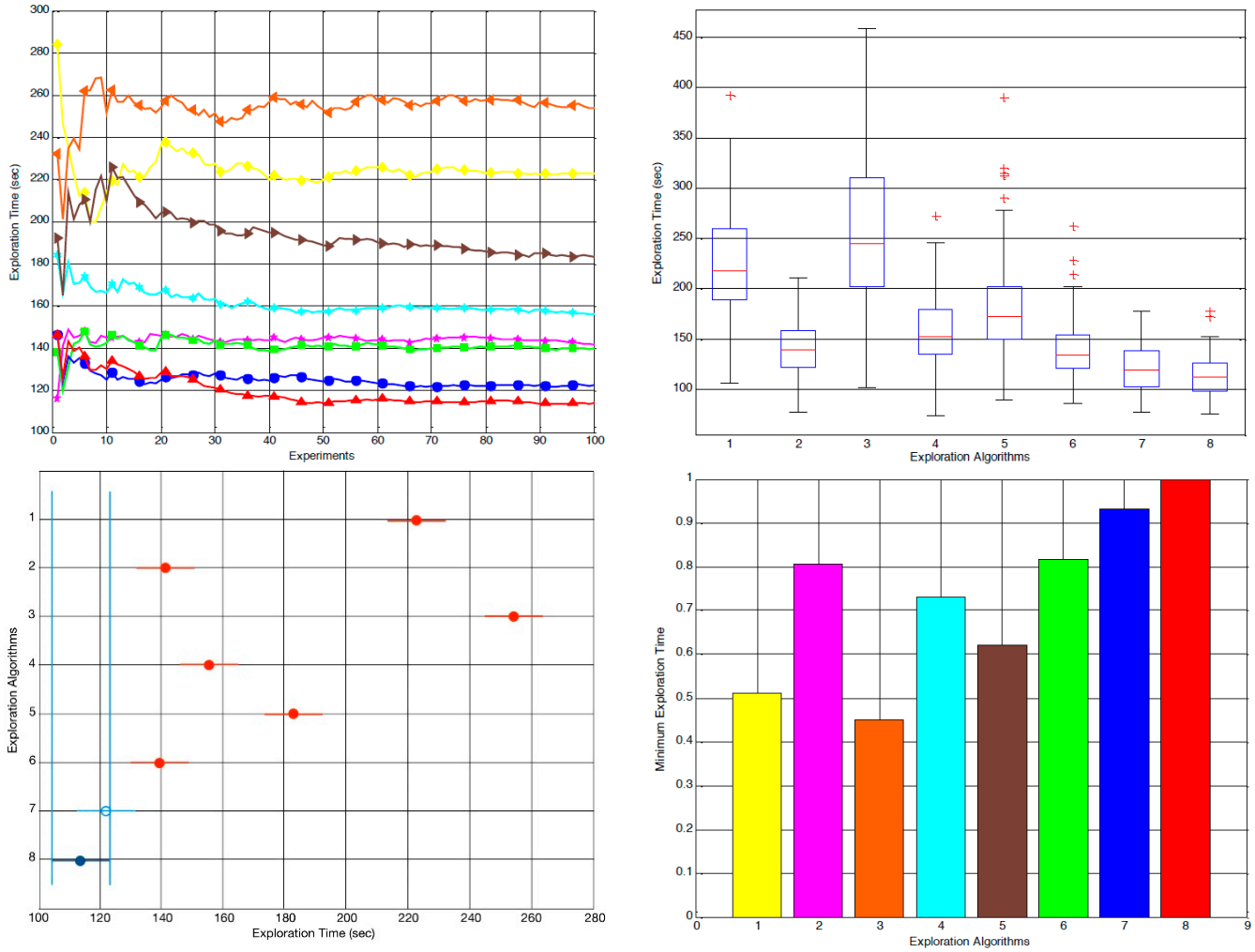

Figure 25. Cumulative average for exploration time experiments. Box plot for exploration time experiments. Multiple comparison test for exploration time experiments. Exploration time ranking.

Given the physical characteristics of the environment used to obtain these results, the analysis of the next variables aims to determine the coverage and searching effectiveness of each algorithm regardless of the environment. As a first step, both variables increase their rate to increase the number of robotic agents in the environment as shown in Figures 26 and 27.

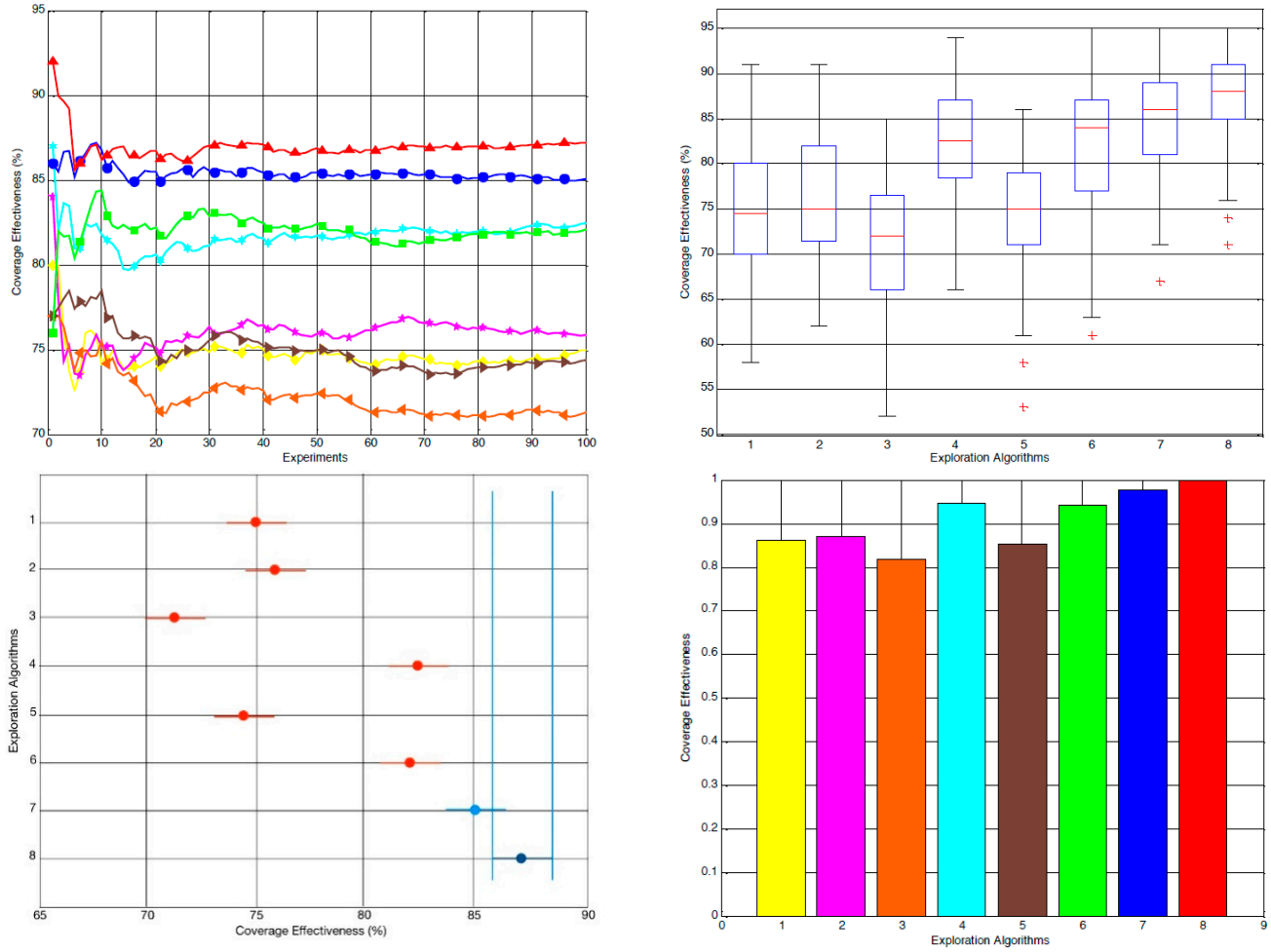

Figure 26. Cumulative average for coverage effectiveness experiments. Box plot for coverage effectiveness experiments. Multiple comparison test for coverage effectiveness experiments. Coverage effectiveness ranking.

Additionally, Figure 28 shows the frequency of collision avoidance performance for each negotiation algorithm in the experiments. The results show how the algorithms actually select the most appropriate according to the collision scene.

Figure 29 shows a radar chart or spider plot that permits selecting of the appropriate exploration algorithm for any restriction or mission goal represented with the response variables analyzed for the indoor environment. This presents normalized values for each variable response and results for three-agent exploration.

Before starting the analysis of results, it is essential to define an exploration algorithm intelligence, such as its ability to generate important decisions (in terms of environment coverage) in minimum time with the least amount and quality of information from teammates and the environment.

About the response time, the three proposed exploration algorithms take a longer time to perform the decision-making process compared to the non-intelligent approaches. This proves the stated above regarding the information necessary for the execution of each algorithm.

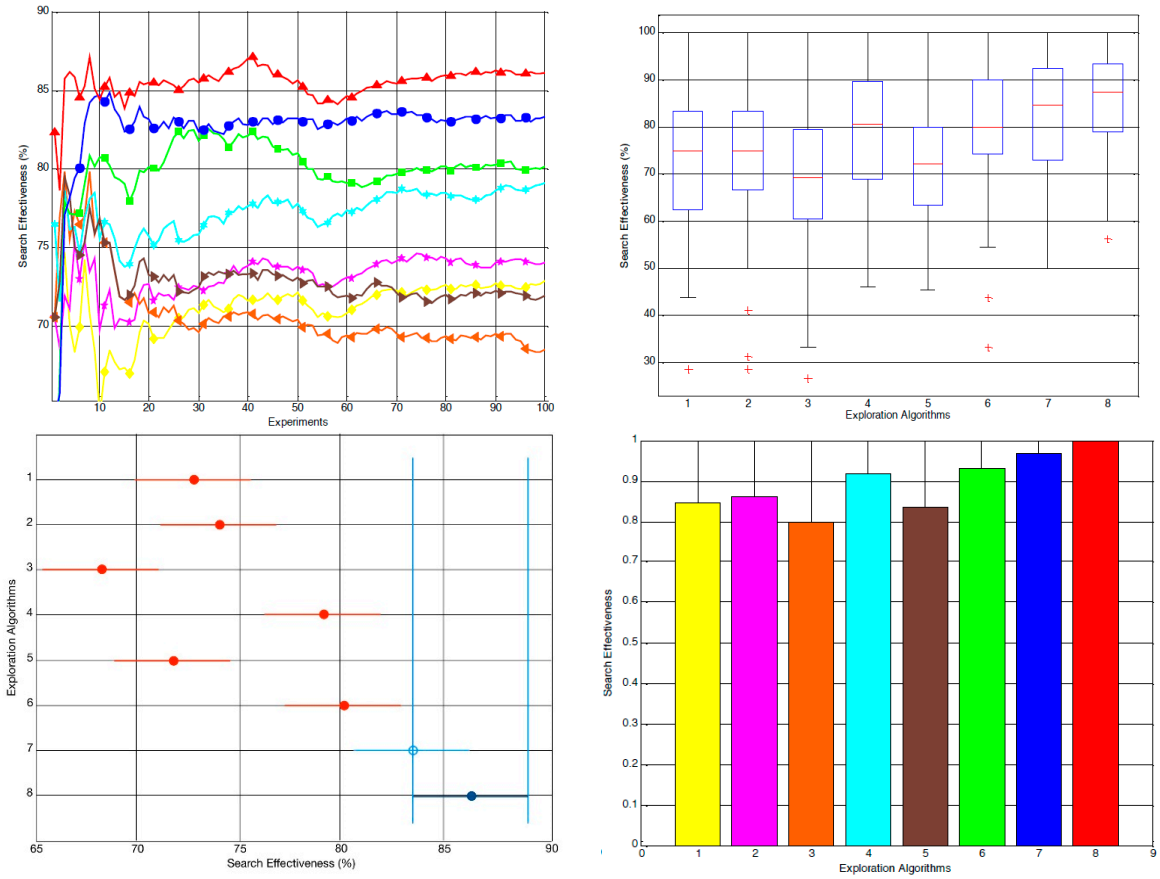

Figure 27. Cumulative average for search effectiveness experiments. Box plot for search effectiveness experiments. Multiple comparison test for search effectiveness experiments. Search effectiveness ranking.

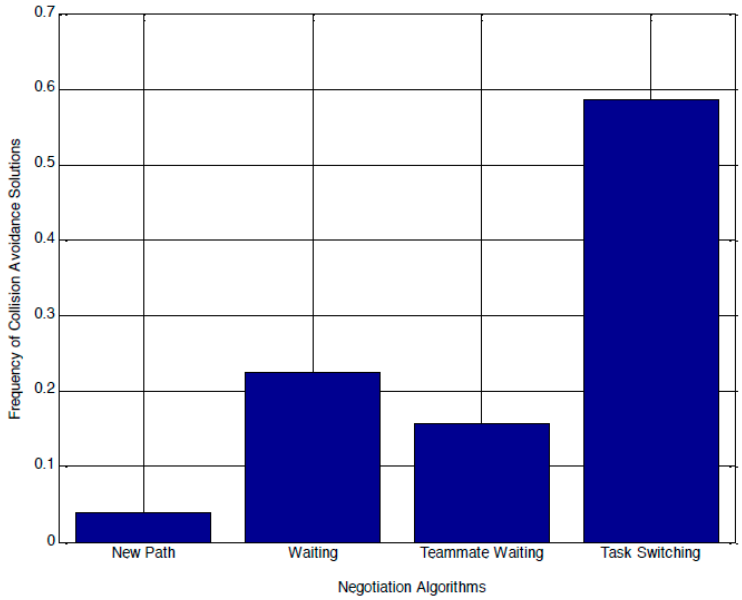

Figure 28. Frequency of collision avoidance solutions for each negotiation algorithm.

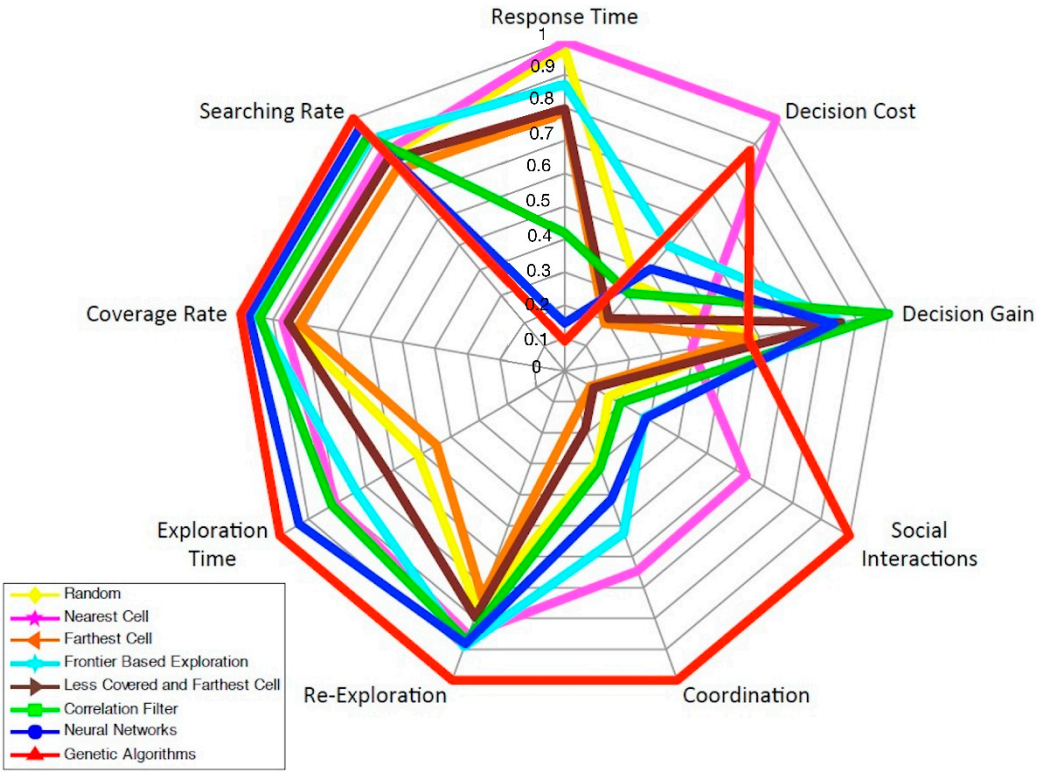

Figure 29. Evaluation of exploration algorithms with the response variables in a radar chart.

The decision cost measures the average time needed to complete the task assigned by the exploration algorithm. Based on Figure 29, it is interesting to notice how the utility function optimized with GA is similar to the nearest cell exploration. Furthermore, it is proved that the farthest cell exploration and less covered and farthest cell exploration take the most expensive decisions (in time and power consumption).

The decision gain measures the average coverage area obtained when the agent completes the task assigned by the exploration algorithm. Based on Figure 29, it is clear to note how the correlation filter algorithm decisions obtain more coverage gain than the other algorithms, proving its capacity to maximize the use of global coverage area construction by each agent.

The results obtained in the social interactions section conclude that an algorithm with a high rate of collision is not very sociable, since agents' decisions affect their teammates. The utility function optimized with a GA had the lowest collision frequency, demonstrating a good sociable algorithm.

The coordination level measures the ability to distribute the work evenly among agents. Clearly, the algorithms that reach to farthest cell present the lower coordination. Again, the utility function optimized with a GA together with the nearest cell and frontier-based exploration shows the best coordination levels.

One of the most important exploration algorithms features is re-exploration avoidance because this is considered a loss of power consumption and time by the agent team. Figure 29 shows the supremacy of the utility function optimized with a GA over the other algorithms.

The most important goal in the exploration problem is the exploration time. In mono-agent systems, the nearest cell exploration presents the best performance. The exploration algorithms that consider the trade-off between the coverage gain and cost in time to reach a particular grid cell present a better performance for a high coordination requirement. Moreover, farthest and random cell algorithms make re-exploration of the environment increasing the exploration time.

6. Conclusions

The proposed approach has been inspired in an indoor environment with physical agents, and its performance tested in simulations. The results show that our proposal effectively allocates the agents over the environment and enables them to carry out their mission quickly. They also demonstrate that our coordination method outperforms other methods developed so far.

The goal then is not to conclude which exploration algorithm is better, but to highlight each one taking into account the response variables discussed above. By analyzing the results presented above, the explorations algorithms that consider the cost in time of the mission show better performance. This is reflected in the fact that farthest and random exploration algorithms make re-exploration of the environment by increasing the exploration time. On the other hand, algorithms that trade-off coverage gain and time cost to reach a particular grid cell have improved performance regarding coverage efficiency. In a high coordination requirement, with more agents in the environment, the proposed intelligent task allocation algorithm based on optimizing a utility function using genetic algorithms and neural networks is significantly better than other approaches.

The following future works can be considered: make dynamic task allocation in the decision-making algorithm, implement intelligent techniques in the negotiation process and implement these algorithms in dynamic environments.

Author Contributions: Formal analysis, J.O.-L.; investigation, J.O.-L. and C.G.Q.M.; methodology, J.O.-L., C.G.Q.M. and M.P.; software, J.O.-L. and C.G.Q.M.; supervision, C.G.Q.M. and M.P.; validation, J.O.-L.; L.N.; C.G.Q.M. and M.P.; writing—original draft, J.O.-L.; L.N.; C.G.Q.M. and M.P.; writing—review and editing, L.N.; C.G.Q.M. and M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by Universidad del Norte, Barranquilla—Colombia.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. Albers, S.; Kursawe, K.; Schuierer, S. Exploring unknown environments with obstacles. *Algorithmica* **2002**, *32*, 123–143. [CrossRef]
2. Apostolopoulos, D.S.; Pedersen, L.; Shamah, B.N.; Shillcutt, K.; Wagner, M.D.; Whittaker, W.L. Robotic Antarctic meteorite search: Outcomes. In Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea, 21–26 May 2001; pp. 4174–4179.
3. Hougen, D.F.; Benjaafar, S.; Bonney, J.C.; Budenske, J.R.; Dvorak, M.; Gini, M.; French, H.; Krantz, D.G.; Li, P.Y.; Malver, F.; et al. Miniature robotic system for reconnaissance and surveillance. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 501–507.
4. Sugiyama, H.; Tsujioka, T.; Murata, M. Collaborative movement of rescue robots for reliable and effective networking in disaster area. In Proceedings of the 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, San Jose, CA, USA, 19–22 December 2005; p. 7.
5. Simoncelli, M.; Zunino, G.; Christensen, H.I.; Lange, K. Autonomous pool cleaning: Self localization and autonomous navigation for cleaning. *Auton. Robot.* **2000**, *9*, 261–270. [CrossRef]
6. Yamauchi, B.; Schultz, A.; Adams, W. Integrating exploration and localization for mobile robots. *Adapt. Behav.* **1999**, *7*, 217–229. [CrossRef]
7. Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S. Collaborative multi-robot exploration. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 476–481.
8. Cao, Y.U.; Fukunaga, A.S.; Kahng, A.B.; Meng, F. Cooperative mobile robotics: Antecedents and directions. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Pittsburgh, PA, USA, 5–9 August 1995; pp. 226–234.
9. Guzzoni, D.; Cheyer, A.; Julia, L.; Konolige, K. Many robots make short work: Report of the sri international mobile robot team. *AI Mag.* **1997**, *18*, 55–64.

10. Fox, D.; Burgard, W.; Kruppa, H.; Thrun, S. Probabilistic approach to collaborative multi-robot localization. *Auton. Robot.* **2000**, *8*, 325–344. [CrossRef]
11. Yamauchi, B. Frontier-based approach for autonomous exploration. In Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA, Monterey, CA, USA, 10–11 July 1997; pp. 146–151.
12. Calisi, D.; Farinelli, A.; Locchi, L.; Nardi, D. Autonomous navigation and exploration in a rescue environment. In Proceedings of the 2005 IEEE International Workshop on Safety, Security and Rescue Robotics, Kobe, Japan, 6–9 June 2005; pp. 110–115. [CrossRef]
13. Da Silva Lubanco, D.L.; Pichler-Scheder, M.; Schlechter, T. A Novel Frontier-Based Exploration Algorithm for Mobile Robots. In Proceedings of the 2020 6th International Conference on Mechatronics and Robotics Engineering, ICMRE 2020, Barcelona, Spain, 12–15 February 2020; pp. 1–5. [CrossRef]
14. Yamauchi, B. Frontier-based exploration using multiple robots. In Proceedings of the International Conference on Autonomous Agents, Minneapolis, MN, USA, May 1998; pp. 47–53. [CrossRef]
15. Ko, J.; Stewart, B.; Fox, D.; Konolige, K.; Limketkai, B. A Practical, Decision-theoretic Approach to Multi-robot Mapping and Exploration. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27–31 October 2003; pp. 3232–3238. [CrossRef]
16. Fang, G.; Dissanayake, G.; Lau, H. A behaviour-based optimisation strategy for multi-robot exploration. In Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, Singapore, 1–4 December 2004; pp. 875–879. [CrossRef]
17. Burgard, W.; Moors, M.; Stachniss, C.; Schneider, F.E. Coordinated multi-robot exploration. *IEEE Trans. Robot.* **2005**, *21*, 376–386. [CrossRef]
18. Cheng, C.; Leng, G. Cooperative Search Algorithms for Distributed Autonomous Agents. In Proceedings of the IEEE/RSJ International Conference on Intelligent Agents and Systems, Sendai, Japan, 28 September–2 October 2004; pp. 394–399.
19. Ogras, U.Y.; Dagci, O.H.; Ozguner, U. Cooperative control of mobile robots for target search. In Proceedings of the IEEE International Conference on Mechatronics 2004, ICM'04, Istanbul, Turkey, 5 June 2004; pp. 123–128. [CrossRef]
20. Grabowski, R.; Navarro-Serment, L.E.; Paredis, C.J.J.; Khosla, P.K. Heterogeneous teams of modular robots for mapping and exploration. *Auton. Robot.* **2000**, *8*, 293–308. [CrossRef]
21. Grabowski, R.; Khosla, P.; Choset, H. Autonomous Exploration via Regions of Interest. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27–31 October 2003; pp. 1691–1696. [CrossRef]
22. Mataric, M.; Sukhatme, G. Task-allocation and coordination of multiple robots for planetary exploration. In Proceedings of the 10th International Conference on Advanced Robotics, Buda, Hungary, 22–25 August 2001; pp. 61–70.
23. Burgard, W.; Moors, M.; Schneider, F. Collaborative Exploration of Unknown Environments with Teams of Mobile Robots. In *Advances in Plan-Based Control of Robotic Agents. Lecture Notes in Computer Science*; Beetz, M., Hertzberg, J., Ghallab, M., Pollack, M.E., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2466. [CrossRef]
24. Simmons, R.; Apfelbaum, D.; Burgard, W.; Fox, D. Coordination for multi-robot exploration and mapping. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, Austin, TX, USA, 30 July–3 August 2000; pp. 852–858.
25. Berhault, M.; Huang, H.; Keskinocak, P.; Koenig, S.; Elmaghraby, W.; Griffin, P.; Kleywegt, A. Robot Exploration with Combinatorial Auctions. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27–31 October 2003; pp. 1957–1962. [CrossRef]
26. Billard, A.; Ijspeert, A.J.; Martinoli, A. A multi-robot system for adaptive exploration of a fast-changing environment: Probabilistic modeling and experimental study. *Conn. Sci.* **1999**, *11*, 357–377. [CrossRef]
27. Wooldridge, M. *An Introduction to Multi-Agent Systems*, 2nd ed.; John Wiley and Sons Ltd.: Hoboken, NJ, USA, 2009.
28. Shi, H.; Li, J.; Li, Z. A distributed strategy for cooperative autonomous robots using pedestrian behavior for multi-target search in the unknown environment. *Sensors* **2020**, *20*, 1606. [CrossRef] [PubMed]
29. Blatt, F.; Szczerbicka, H. Combining the Multi-Agent Flood Algorithm with Frontier-Based Exploration In Search & Rescue Applications. In *Simulation Series, Proceedings of the 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Seattle, WA, Canada, 9–12 July 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7. [CrossRef]
30. Gomez, C.; Hernandez, A.C.; Barber, R. Topological frontier-based exploration and map-building using semantic information. *Sensors* **2019**, *19*, 4595. [CrossRef]
31. Mahdoui, N.; Frémont, V.; Natalizio, E. Communicating Multi-UAV System for Cooperative SLAM-based Exploration. *J. Intell. Robot. Syst. Theory Appl.* **2020**, *98*, 325–343. [CrossRef]
32. Fang, B.; Ding, J.; Wang, Z. Autonomous Robotic Exploration Based on Frontier Point Optimization and Multistep Path Planning. *IEEE Access* **2019**, *7*, 46104–46113. [CrossRef]
33. Niroui, F.; Zhang, K.; Kashino, Z.; Nejat, G. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 610–617. [CrossRef]
34. Li, H.; Zhang, Q.; Zhao, D. Deep Reinforcement Learning-Based Automatic Exploration for Navigation in Unknown Environment. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 2064–2076. [CrossRef] [PubMed]
35. Chen, Z.; Subagdja, B.; Tan, A.H. End-to-end Deep Reinforcement Learning for Multi-agent Collaborative Exploration. In Proceedings of the 2019 IEEE International Conference on Agents, ICA 2019, Jinan, China, 18–21 October 2019; pp. 99–102. [CrossRef]

36. Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14413–14423. [CrossRef]
37. Yu, C.; Dong, Y.; Li, Y.; Chen, Y. Distributed multi-agent deep reinforcement learning for cooperative multi-robot pursuit. *J. Eng.* **2020**, *2020*, 499–504. [CrossRef]
38. Hou, Y.; Ruan, X.; Zhu, X.; Li, C. Frontier-based exploration on continuous radial basis function neural network map. In Proceedings of the Chinese Control Conference, CCC, Wuhan, China, 25–27 July 2018; pp. 5534–5538. [CrossRef]
39. Shrestha, R.; Tian, F.P.; Feng, W.; Tan, P.; Vaughan, R. Learned map prediction for enhanced mobile robot exploration. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 1197–1204. [CrossRef]
40. Matsumoto, M.; Nishimura, T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Modeling Comput. Simul.* **1998**, *8*, 3–30. [CrossRef]