# An Integrated Approach for Achieving Multirobot Task Formations

Antidio Viguria and Ayanna M. Howard, *Senior Member, IEEE*

***Abstract*—In this paper, a problem, called the initial formation problem, within the multirobot task allocation domain is addressed. This problem consists in deciding which robot should go to each of the positions of the formation in order to minimize an objective. Two different distributed algorithms that solve this problem are explained. The second algorithm presents a novel approach that uses cost means to model the cost distribution and improves the performance of the task allocation algorithm. Also, we present an approach that integrates distributed task allocation algorithms with a behavior-based architecture to control formations of robot teams. Finally, simulations and real experiments are used to analyze the formation behavior and provide performance metrics associated with implementation in realistic scenarios.**

***Index Terms*—Auctions, market-based coordination, multirobot teams, task allocation.**

## I. Introduction

In RECENT studies, it has been shown that mobile sensor networks can enable effective achievement of a variety of Earth-monitoring applications, especially those that require spatially distributed recording of environmental parameters [21]. In most of these applications, individual sensors are tasked to collect information about their neighboring sensors using peer-to-peer communication. Unfortunately, as the size of the network increases, bandwidth limitations and the absence of feasible communication channels severely limits the possibility of conveying and using global information. As such, the utilization of decentralized techniques for forming new sensor topologies and configurations is a highly desired quality of mobile sensor networks. Establishment of these sensor configurations involves determining how to allocate sensor positions to mobile sensor agents in order to achieve a desired topology—a similar research objective that is found when focusing on the task allocation problem with teams of robots. This problem can also be applied to intelligent environments [15], where distributed intelligent sensors have to be coordinated in order to perform a common objective.

Manuscript received July 14, 2008; revised October 7, 2008 and December 16, 2008. First published March 24, 2009; current version published April 15, 2009. Recommended by Guest Editor M.-Y. Chow. This work supports research in Reconfigurable Sensor Networks for the National Aeronautics and Space Administration, Earth Science Technology Office. The work of A. Viguria was supported by a Fullbright Scholarship from the Fullbright Commission in Spain.

A. Viguria is with the Advanced Center for Aerospace Technologies (CATEC), 41309 Seville, Spain (e-mail: aviguria@catec.aero).

A. M. Howard is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ayanna.howard@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2009.2014056

In the last few years, different approaches have been used to solve the task allocation problem: centralized [2], [3], [30], hybrid [4], [10], and distributed. The distributed approach, considered ideal for teams of robots or mobile sensors, has characteristics that fit most applications: high fault tolerance, fast response for dynamic changes in the environment, and low computational complexity. Basically, two main distributed approaches have been studied in depth in order to solve the problem for independent loosely coupled tasks: behavior-based [20], [29] and market-based [4], [6], [23]. However, only does not require direct communication between robots is not required only in [20]. The rest of the algorithms fall within the domain of what we call *time-distributed* algorithms, which means that robots make decisions based on interagent communications transmitted at different time instances. This type of algorithm is more fault-tolerant than a centralized approach, and can obtain more efficient solutions than a completely distributed approach. So far, the approach that receives more attention has been the market-based approach, which uses negotiations to allocate the different positions, since it offers a good compromise between communication requirements and the quality of the solution.

A special case of task allocation is the initial formation problem [7] in which each robot can only be allocated to one task. In order to illustrate the differences between both problems, we can think about the general task allocation problem as a multiple traveling salesman problem (TSP) [13] and the initial formation problem can be viewed as a classical job assignment problem [11] where robots are the workers and tasks are the jobs to be executed by those workers.

Different approaches can be used to solve the initial formation problem. This problem has typically been solved optimally using centralized solutions such as the Hungarian method [11]. However, this kind of solution assumes that all the information is available and has all the disadvantages related to centralized systems: low fault tolerance and slow response to dynamic changes in the environment. Other approaches, such as [1], use a parallel algorithm based on auctions to obtain the optimal solution for the assignment problem. Nevertheless, a parallel algorithm needs to have updated information transmitted between all the nodes. From a robotics perspective, this approach does not lead to any advantages in fault tolerance or response from changes in the environment, since it only improves the time complexity of the algorithm. Different distributed approaches have also been developed recently, but the tasks have to be communicated to all the robots at the beginning. A distributed heuristic with local communication is explained in [31], while in [32], the agents are controlled by hybrid models using distributed potential fields. Both approaches fail to return a highly efficient solution since more than one robot can execute the same task. In [24], a solver of the TSP is used to decide which robot should execute which task. However, this approach first solves a much more difficult problem to obtain a solution to the assignment problem.

In this research, we have decided to use a market-based approach for addressing the initial formation problem [27]. The main reason is that we are interested in not only obtaining a feasible solution, but also an efficient one. As was said before, this approach offers a good compromise between communication requirements and the quality of the solution. It is an intermediate solution between centralized, where all the information is available, and distributed, where only local information is accessible, while obtaining solutions close to the optimum and offering a good level of fault tolerance and reaction to changes in the environment.

The main contribution of this study is the utilization of cost means in a market-based approach to improve the efficiency of the task allocation algorithm. Furthermore, we compare the performance of two different task allocation algorithms integrated in a complete robot system and study the effects of the other modules, such as the path planner module, on the task allocation efficiency.

This paper is organized as follows. In the next section, a basic market-based algorithm (BS) that solves the initial formation problem will be explained. Moreover, different modifications of the BS algorithm that improve its results will be addressed. In Section III, the integration of the task allocation algorithms within a complete robot architecture is explained. This architecture is used for both simulations and real experiments. Next, simulation results will be presented and discussed in Section IV, showing the advantages and disadvantages of each algorithm. Also, the task allocation algorithms have been integrated in a complete robotic architecture and we will study how this integration affects the algorithm performances. These algorithms have been implemented on a team of physical robots and the results from several experiments will be explained in Section V. Finally, conclusions are provided in Section VI.

## II. Market-Based Algorithms for the Initial Formation Problem

The initial formation problem becomes really important within the field of formation control [8], [14], where using local information and control laws, the distributed algorithm is able to drive a given formation error to zero. As it is stated in [9], these algorithms require a first step that assigns the robots to the formation positions while taking into account their initial positions, i.e., answer the question who goes where?

We have decided to use a market-based approach to solve this problem (see Fig. 1). In this context, a more formal definition of the initial formation problem can be stated as follows:

*Given a number of tasks, $\{T_1, T_2, \ldots, T_N\}$, a team of robots $\{R_1, R_2, \ldots, R_M\}$, a function $C(T_i, R_j)$ that specifies the cost of executing task $T_i$ by robot $R_j$, find the assignment that allocates one task per robot and minimizes the global cost defined as $\sum_{j=1}^{M} C(T_i, R_j)$, where task $i$ is assigned to robot $j$.*

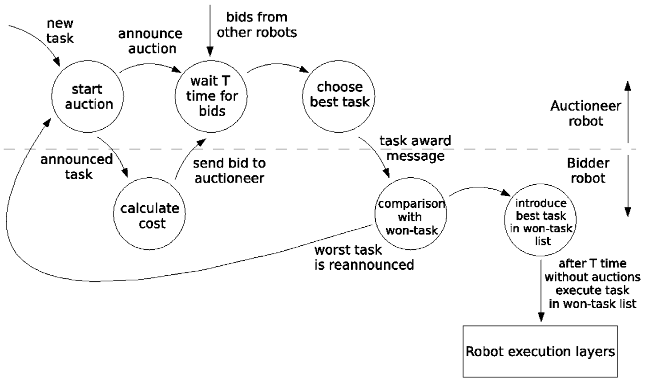

Fig. 1. Task allocation process using a market-based approach that solves the initial formation problem. Two roles are played dynamically by robots: auctioneer and bidders. The auctioneer is the agent in charge of announcing the tasks and selecting the best bid from all the received bids.

Next, two different distributed task allocation algorithms that solve the initial formation problem will be explained.

### A. BS: Basic Market-Based Algorithm

In the basic algorithm, positions associated with an initial robot formation are recast as biddable tasks in a formation auction. To determine position assignment, a robot agent (auctioneer) dynamically plays the role of announcing the tasks and selecting the lowest cost bid from all the received bids during the auction. All other robots dynamically play the role of bidder in which each robot bids for tasks and, subsequently, keeps the task with the lowest cost. If during the auction process, a robot bidder wins a new task that has a lower cost than the one already won, it would sell the old task to the robot with the best bid but worse than its own bid. The best bid worse than the robot's bid is selected in order to avoid infinite loops in the negotiation. This scenario could happen when two robots have the best bids for at least three tasks as shown in Fig. 2. Upon termination of the auction, each robot will have an assigned position in the formation.

Since robots minimize their own costs instead of the global cost, there are situations when this algorithm does not obtain satisfactory results. This usually happens when a robot has to take a task that is the worst one for its own interest, as can be seen in Fig. 3. In this example, the global cost obtained with the market-based algorithm is 66.67% greater than the optimal allocation.

### B. RTMA: Robot and Task Mean Allocation Algorithm

In order to solve the initial formation problem, the task allocation algorithm has to solve two main problems.

1) If I won more than one task, how do I determine which one to keep?
2) How do I calculate the bid for a certain task?

In the BS algorithm, bids are the distance between the robot position and the tasks (we are only considering waypoint tasks) and if one robot wins more than one task, it keeps the one that is closest to itself, i.e., the one with the lowest cost or best bid.

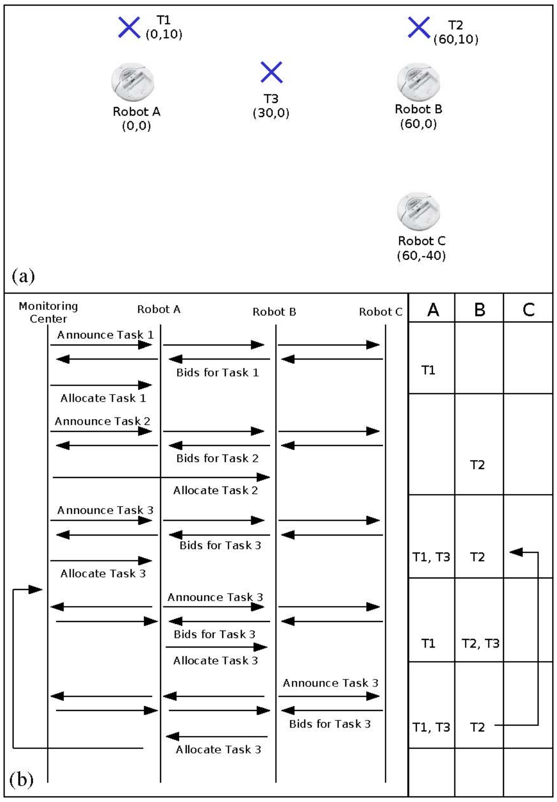

Fig. 2. (a) Initial position of the robots and the tasks. (b) Messages exchanged among the different agents and how an infinite loop appears in the negotiation protocol when auctioneers always allocate the task to the robot with the best bid without considering their own bid.

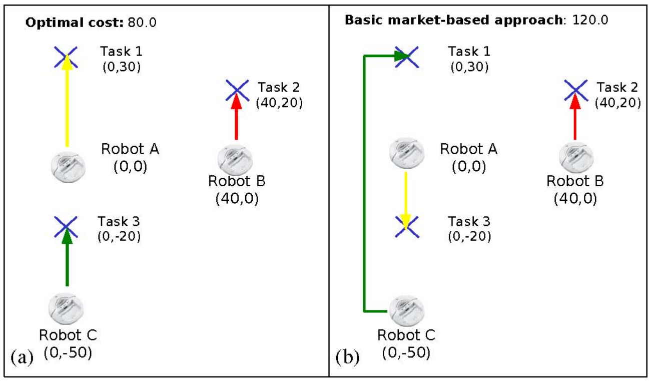

Fig. 3. Difference in cost between the optimal allocation and the one obtained with the basic market-based algorithm.

Therefore, if our objective is to improve the BS algorithm, one or both of these aspects must change. Moreover, the new algorithm must keep the advantages of the market-based approach: fault tolerance, linear increment in number of messages with respect to number of robots, and high adaptation to changes in the environment using reallocations.

First, we try to choose in a more clever way the task that must be kept when a robot wins more than one task. This is accomplished using additional knowledge available to the system. Instead of keeping the task with the smallest distance to the robot, the task with highest difference between the distance to the robot and the mean of its distance to all the robots will be selected. In other words, suppose that there are a finite number of robots $N_R$ and robot $R_k$ has won tasks $T_i$ and $T_j$. In this case, robot $R_k$ will keep task $T_i$ if and only if

$$\sum_{r=1}^{N_R} \frac{D(R_r, T_i)}{N_R} - D(R_k, T_i) > \sum_{r=1}^{N_R} \frac{D(R_r, T_j)}{N_R} - D(R_k, T_j)$$

where $D(R_k, T_i)$ is the distance between robot $R_k$ and task $T_i$. The reason behind this idea is to make the robot willing to choose the task that is best for the team, not just for itself. So, robots will more probably win tasks that have a high cost for the rest of the robots and a low one for themselves. In this way, we make the algorithm less selfish and make robots try to minimize the global cost and not just their own costs.

The question that arises now is how to calculate the mean of the distances for a certain task. During the normal operation of the algorithm, the auctioneer receives bids from all functioning robots in order to allocate the task to the best robot. At this moment, the auctioneer knows all the distances between every robot and the current task. Thus, the mean is calculated by the auctioneer and transmitted to the robot within the message that informs the robot that has won the task. The major difference with the BS algorithm is that the robot must remember the mean associated with the won task. Furthermore, the robot is able to compare their means to different tasks because it remembers the mean of the task already won and the mean of the new allocated task is sent by the auctioneer as it was explained previously.

Also, we must change the way that costs are calculated. In the original algorithm the cost function used to calculate the bid for a certain task is the distance between the robot and the task. However, in this improved algorithm, the cost function will be the difference between the distance of the robot and the task minus the mean of the distances between that robot and all the tasks, i.e.,

$$C(R_i, T_j) = D(R_i, T_j) - \sum_{t=1}^{N_T} \frac{D(R_i, T_t)}{N_T} \quad (1)$$

where $C(R_i, T_j)$ is the cost function for robot $R_i$ and task $T_j$ and the total number of tasks is $N_T$. The idea is to decrease substantially the cost of a task when it is close to a robot and the rest of the tasks are far away from the same robot. But if all the tasks have similar costs, the cost reduction will be smaller. Therefore when bids are received by the auctioneer, the tasks from robots that are in the first situation will be favored with respect to the tasks from robots that are in the second situation. Also, if a task is far away from a robot and the rest are close to the robot, the cost of the task will be kept almost the same.

The only drawback to this improvement is that robots must know the different tasks at the beginning in order to calculate the mean of the distances. However, the extra resources needed for the algorithm are almost the same as that for the BS algorithm since robots only have to memorize the mean calculated at the beginning and implement one basic cost function operation. The RTMA algorithm implements both improvements as can be seen in Algorithm 1. Also, in Fig. 4, it can be observed that the messages are exchanged between the robots in the negotiation

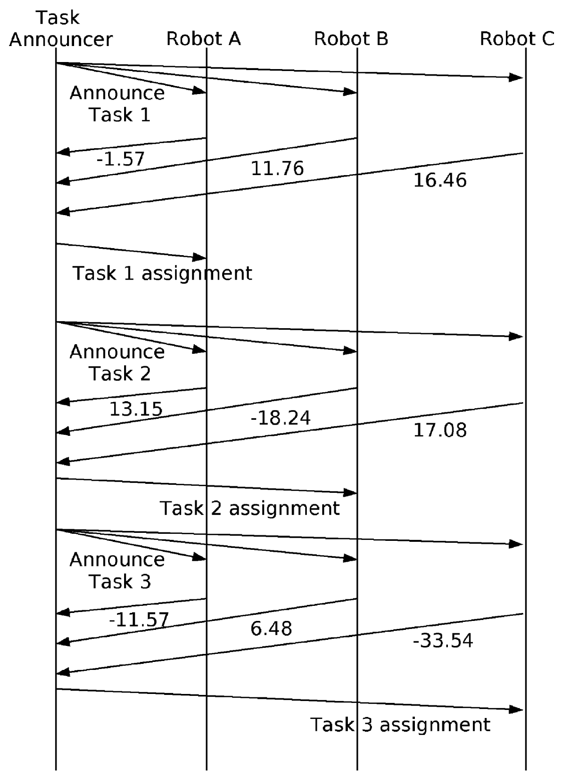

Fig. 4. Messages interchanged in the negotiation process using the RTMA algorithm for the scenario showed in Fig. 3. The final allocation is better than the one obtained with the BS algorithm.

process using the RTMA algorithm for the initial positions and tasks represented in Fig. 3.

**Algorithm 1** Bidder algorithm

```
a new message is received
if new message is a task notification then
  mean = mean + $D(R_k, T_i)/N_T$
else if new message is a task announcement then
  calculate bid (distance to the task minus mean)
  send bid to the auctioneer
else if new message is a task award then
  if the robot has already won a task then
    if (cost of the new task - mean of the new task costs)
    $>$ (cost of the won one - mean of the won task costs)
    then
      introduce old task in announcement-task list and
      delete it from won-tasks list
      introduce won task in the won-tasks list
    else
      introduce won task in the announcement-task list
    end if
  else
    introduce won task in the won-tasks list
  end if
end if
```

Finally, in our algorithms, there is no requirement to have only one auction active at a single instance of time. These auctions can be started in parallel, and therefore, there is no need to have any synchronization method between robots to decide who is the auctioneer (such as a token algorithm). This type of mechanism is usually needed in task allocation algorithms that use local plans to calculate marginal costs as bids for solving the general task allocation algorithm [4]. In these algorithms, task bids depend on multiple auctions (not just the current auction). In the initial formation problem, only one task is allocated per robot. Thus, task costs only depend on one auction.

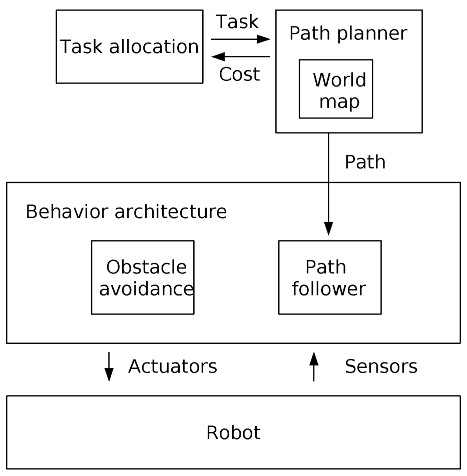

Fig. 5. Scheme that shows the integration of a task allocation algorithm in a complete system ready to be used in a real-world application. The path planning algorithm is used to calculate the cost of the tasks and as an input for the path follower algorithm that is combined with obstacle avoidance using the DAMN architecture.

## III. Implementation of Multirobot Systems

A multirobot architecture has been used to test the decentralized algorithms presented in this paper. This architecture is designed for heterogeneous robots [28] and divided into three layers. The highest layer is independent from the type of robot and is the one aware of the existence of other robots. Thus, the task allocation algorithm is implemented in this layer and can be used, without modification, in both simulations and real robots. Moreover, the communication among robots is based on IP, so it can also be used as an interprocess communication method for simulations. The other two layers are used to execute the different tasks allocated to the robot and make easier the creation of new algorithms by using a modular and component-based architecture.

We have integrated our task allocation algorithms in a complete system ready to be used in real-world applications. As can be seen in Fig. 5, in each robot, the task allocation algorithm has been integrated with a path planner algorithm and the execution of the tasks are within a behavior architecture that combines the path following algorithm with obstacle avoidance.

Two of the most popular path planning algorithms have been implemented: $A^*$ algorithm [17] and Rapidly exploring random trees (RRTs) [12]. These allow the system to integrate map-based information in the task allocation scenarios. The first algorithm is based on a heuristic estimator to find the optimal solution faster than general search algorithms such as breadth-first or depth-first search. Even so, for robotic applications, the $A^*$ algorithm still requires a significant amount of processing power, specially for large state spaces with constraints. RRTs is also a search algorithm that has a random nature and the quality of the solution cannot be determined a priori, but it is faster than $A^*$. This algorithm works like a search tree that starts from an initial state and is expanded by performing incremental motions toward the direction of random points. The main difference between this algorithm and a random walk is that the latter suffers from a bias toward places already visited, while RRTs works in the opposite manner by being biased toward places not yet visited. Specifically, we have used the biased version of RRTs with a probability equal to 0.05.

For navigation, we have selected the distributed architecture for mobile navigation (DAMN) architecture [22] to combine the obstacle avoidance and path follower algorithms. This architecture was designed to combine different behaviors, specially, for mobile robots in unknown and dynamic environments, which fits our demonstration scenario. Each of the behaviors votes for a set of possible actuator values satisfying its objectives. Then, an arbiter combines those votes and generates actions that reflect the behaviors objectives and priorities. Regarding the behaviors, a laser scanner was used as the sensor for the obstacle avoidance and the Pure Pursuit algorithm [18] has been used as the path follower. The Pure Pursuit algorithm geometrically determines the curvature that will drive the vehicle to a chosen path point defined as one lookahead distance from the current position of the robot.

## IV. Simulated Experiments

First, we simulated our distributed task allocation algorithms in scenarios without obstacles where the task cost is calculated as the Euclidean distance. Next, we integrated the task allocation algorithms within a robot architecture that couples the task allocation algorithm with navigation modules. The effect that the different modules, specially the path planner module, causes to the task allocation efficiency was studied.

### A. Ideal Simulations

Both algorithms have been tested using initial positions of the robots and formations calculated at random in a virtual world of $1000 \times 1000$ m$^2$ without obstacles. The simulations have been accomplished using a variety of scenarios in which the number of robots and tasks ranged from 2 up to 20, and for every case, 100 simulations were run. These results are shown in Fig. 6, where errors in percentage in comparison with the optimal solution are presented. The optimal solution has been calculated using the Hungarian method [11]. It can be observed that the RTMA algorithm performs much better than the BS algorithm. Both algorithms obtain efficient results up to eight robots and tasks, where the largest error is less than 10%. For more than eight robots, only the RTMA algorithm obtained efficient results, with a maximum error of 5.98% in the case of 20 robots. As can be seen in Fig. 6, the error with respect to the optimal solution seems to be bounded by a linear function based on the number of robots and tasks for all algorithms. However,

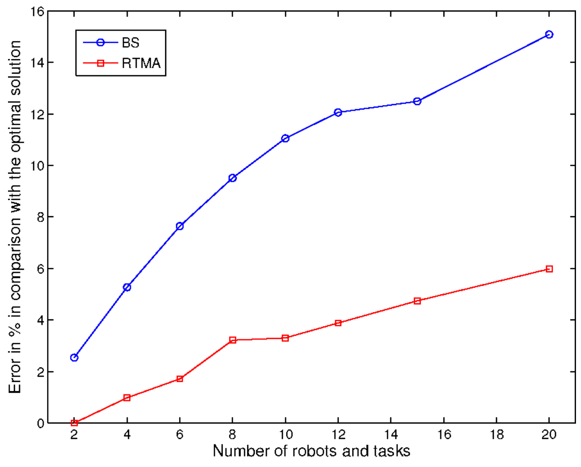

Fig. 6. Mean error in percentage in comparison with the optimal solution for both types of algorithms and calculating the initial positions of the robots and the points of the formations at random over 100 simulations.

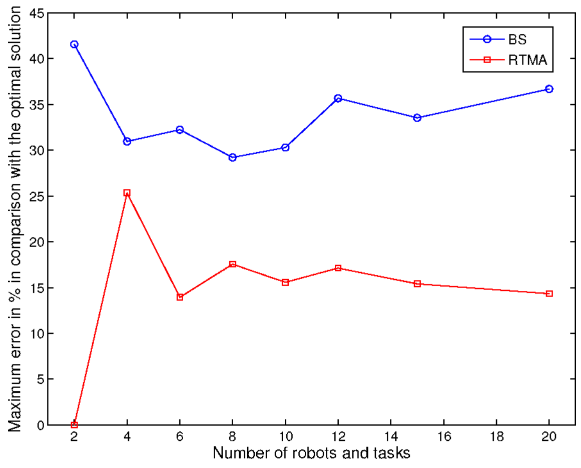

Fig. 7. Maximum errors in percentage in comparison with the optimal solution in 100 simulations for both types of algorithms and calculating the initial positions of the robots and the points of the formations at random.

the RTMA algorithm is the one with lowest slope. It is also important to point out that for two robots and tasks the RTMA algorithm always obtains the optimal solution.

The results of Fig. 6 only show statistically how good the algorithm is based on the mean. However, it could be the case that an algorithm could have good results on average but there are some situations where its results have large errors. Therefore, another important parameter to consider is the maximum error with respect to the optimal solution over all the simulations. In Fig. 7, the maximal errors in percentage are shown. The BS algorithm is still worse than the RTMA algorithm. It can be observed that the mean of the maximum errors considering all the cases is 14.91% for the RTMA algorithm and 33.77% for the BS algorithm, which is greater than the mean error observed in Fig. 6. This means that these algorithms do not have a constant behavior and, for a specific situation, results could be worse than the average.

All the results presented have been calculated using random position of the robots and random points of the formations

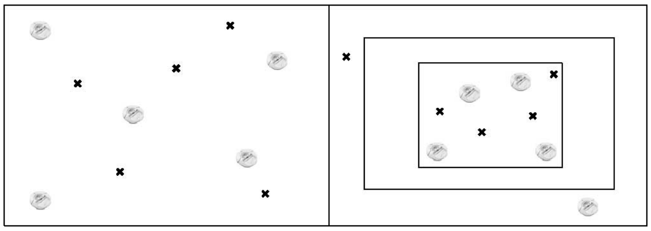

Fig. 8. Types of formations used in the simulations. (Left) Initial positions of the robots and the formations calculated at random. (Right) Most of the formation points and the initial positions of the robots calculated at random in the small box and the others calculated outside the big box and calculated also at random.

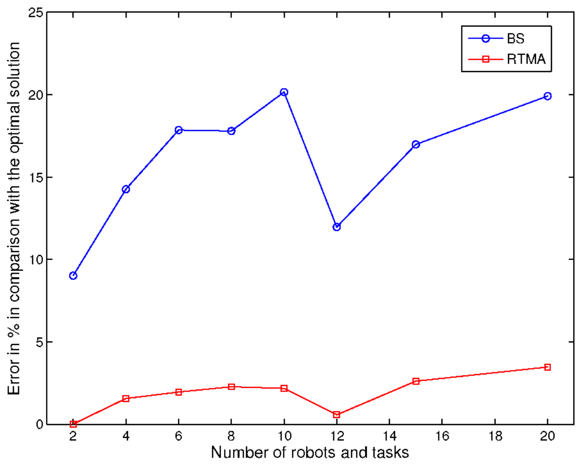

Fig. 9. Mean error in percentage in comparison with the optimal solution for the different types of algorithms and calculating the inital positions of the robots and the points of the formations as it is described in the right part of the Fig. 8 over 100 simulations.

uniformly distributed. However, the quality of the solution for some of the algorithms depends on the type of formations. In Fig. 8, there are two types of formations: the one on the left is calculated entirely at random and is the one used so far, the other formation on the right has a structure formed by two boxes. Most of the points and robots of the formation are in the small box, and the others outside the big box. As can be seen in Fig. 9, the RTMA algorithm obtains again the best results and they are similar to the results showed in Fig. 6. However, the BS algorithm behaves differently and obtains worse results than the ones obtained with the other type of formation, specially for low number of robots and tasks. Therefore, it could be said that the BS algorithm is less robust to changes in the nature of the formation, i.e., the results in comparison with the optimal solution are different and this difference only depends on the type of random distribution used to calculate the points and positions of the robots.

## B. Realistic Simulations

After the integration of the task allocation algorithm within a robot architecture, the effect that individual robot errors cause to the task allocation efficiency is studied. There are different sources of errors: localization, path planning, etc. In this study, we concentrate our efforts to study the effect of the path planner module on the bidding process. Other sources of errors and their influences on the task allocation will be considered for future work.

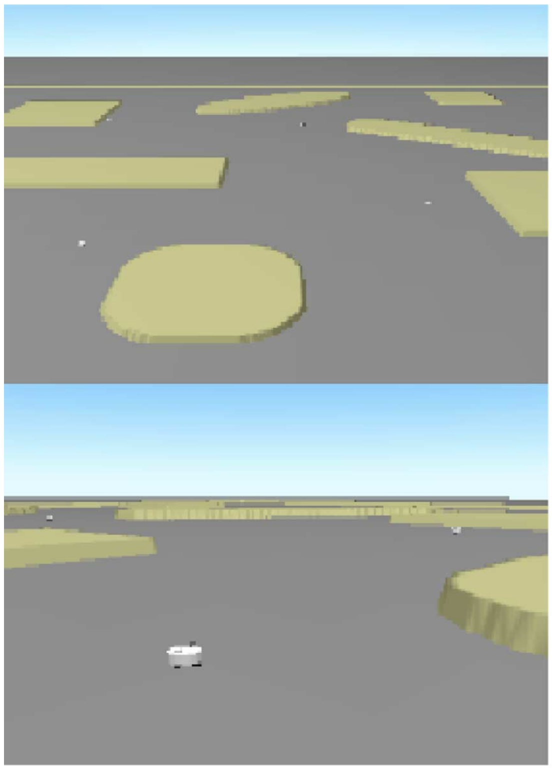

Fig. 10. Snapshot of the simulator Player/Gazebo. (Top) Aerial view of the environment with obstacles. (Bottom) Close view of the 3-D model of our test platform.

First, global costs obtained with the two path planners ($A^*$ and RRTs) are compared. Second, it is studied for each of the task allocation algorithms (BS and RTMA), whether the difference between global costs using both path planners is equivalent. Finally, we are interested in the effect that obstacle density has on the performance of our task allocation algorithms and whether differences depend on the path planner algorithm.

The Player/Gazebo [5] software has been used to simulate the environment and the robots. We focus on a monitoring application, where robots have to navigate toward some specific locations and take environmental measurements. As will be seen in Section V, we used the iRobot Create platform to test our algorithms. For that reason, we created a 3-D model of those robots to be used in the simulator (see Fig. 10).

We made a classification based on the percentage of nonnavigable terrain (in this case obstacles): high navigable terrains (less than 15% of nonnavigable terrrain), medium navigable terrain (between 15% and 30% of nonnavigable terrain), and low navigable terrain (more than 30% of nonnavigable terrain). For our simulations, we have used three different scenarios, all of them with a $75 \times 75$ m$^2$ area, to test our complete robotic system for this type of application. The first scenario (see Fig. 11) has 5% of nonnavigable terrain. The second scenario has 20% of nonnavigable terrain (see Fig. 12), and the last scenario has 40% of nonnavigable terrain (see Fig. 13). Also, Figs. 12 and

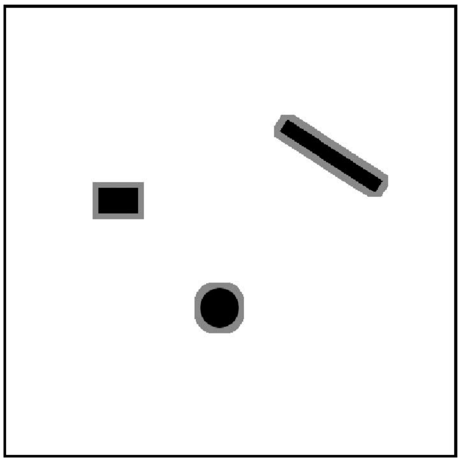

Fig. 11. Scenario with 5% of nonnavigable terrain. The obstacles are increased virtually to the size of the robot, so they do not navigate too close to them. This reduce the probability of a collision due to noises and inaccuracies in the sensors and the map.

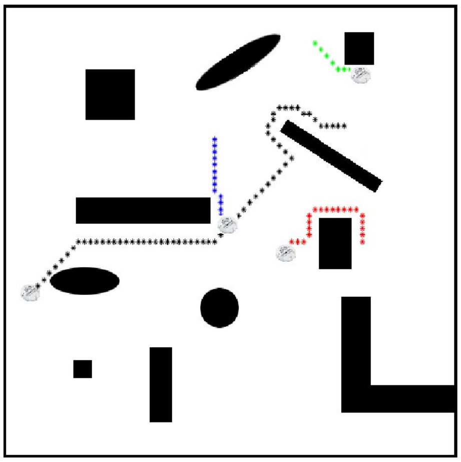

Fig. 12. Scenario with 20% of nonnavigable terrain. The paths show the solution of one of the random missions obtained using the BS task allocation with the $A^*$ path planner.

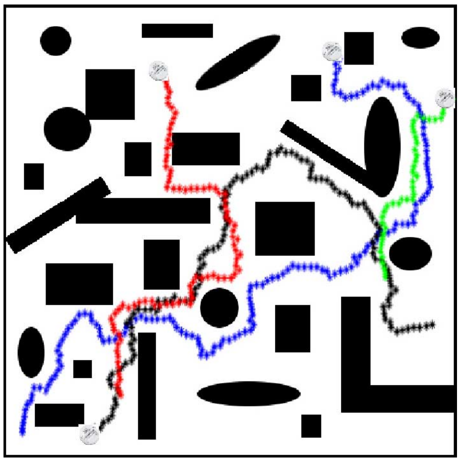

Fig. 13. Scenario with 40% of nonnavigable terrain. The paths show the solution of one of the random missions obtained using the RTMA task allocation with the RRTs path planner.

TABLE I
RESULTS COMPUTED FOR FORMATIONS WITH DIFFERENT NUMBER OF ROBOTS, TASKS, AND OBSTACLES OVER 20 SIMULATIONS PER CASE USING THE $A^*$ ALGORITHM

| *Tasks & Robots* | *Scenario* | *BS* | | *RTMA* | | *Optimum* |
|---|---|---|---|---|---|---|
| | | Mean | Error | Mean | Error | |
| 4 | 5% | 124.74 | 4.58% | 119.99 | 0.60% | 119.27 |
| 4 | 20% | 127.38 | 8.70% | 119.62 | 2.07% | 117.19 |
| 4 | 40% | 161.94 | 6.85% | 153.61 | 1.35% | 151.55 |
| 6 | 5% | 144.02 | 6.54% | 139.88 | 3.47% | 135.18 |
| 6 | 20% | 187.76 | 7.53% | 177.25 | 1.51% | 174.60 |
| 6 | 40% | 216.88 | 7.08% | 204.92 | 1.17% | 202.54 |
| 8 | 5% | 197.52 | 8.94% | 186.72 | 2.98% | 181.31 |
| 8 | 20% | 208.08 | 9.16% | 195.86 | 2.75% | 190.61 |
| 8 | 40% | 261.62 | 12.63% | 235.74 | 1.48% | 232.29 |

Each cell represents the mean of the global cost and the mean error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Fig. 11 for the 5% scenario, in Fig. 12 for the 20% scenario, and in Fig. 13 for the 40% scenario.

13 show the solution obtained using our algorithms and the path followed by the robots using the $A^*$ and RRTs algorithms, respectively. It can be observed directly how the $A^*$ obtains the optimal path while the RRTs has a lower rate of finding a path close to the optimal one. This fact will have a large impact on the performance of the task allocation algorithms, as will be commented next. Also, it is interesting to observe that in Fig. 12 one robot is forced to navigate to the furthest task for itself due to the outcome from the BS algorithm.

Due to the complexity of these simulations, only 20 simulations have been run per case, where the position of the robots and tasks has been calculated at random (avoiding the areas considered obstacles in the world). We suppose that robots have access to an occupancy map of the environment. We first ran our simulations using the $A^*$ for path planning. The results obtained from these simulations are showed in Table I, where each cell represents the mean of the global cost over 20 missions, i.e., the sum of the distance traveled by all the robots. It can be seen that the RTMA algorithm still obtains better results than the BS algorithm when it is integrated in a complete robotic system. Also, the results obtained with the complete system are equivalent, in comparison with the optimal solution, to the results obtained in the previous section (see Fig. 6). The improvements obtained with RTMA, in comparison with the BS algorithm, are of the same order of magnitude and both algorithms obtain similar results in all the scenarios. Therefore, the integration of our task allocation algorithms in a complete robotic system, with the $A^*$ planner, does not affect the task allocation algorithms performance.

The same random missions, for each scenario, have been used for the optimal solution and the two task allocation algorithms. The optimal solution has been calculated using the $A^*$ algorithm with the Hungarian method [11], i.e., all the different optimal paths between every robot and task have been calculated using the $A^*$ algorithm, then the distances of all these paths have been used as the values of the cost matrix that represents the task allocation problem as a job assignment problem. Finally, the Hungarian method has been applied to that cost matrix to calculate the optimal assignment.

TABLE II
RESULTS COMPUTED FOR FORMATIONS WITH DIFFERENT NUMBER OF ROBOTS, TASKS AND OBSTACLES OVER 20 SIMULATIONS PER CASE USING THE RRTS ALGORITHM

| *Tasks & Robots* | *Scenario* | *BS* | | *RTMA* | | *Optimum* |
|---|---|---|---|---|---|---|
| | | Mean | Error | Mean | Error | |
| 4 | 5% | 159.68 | 33.88% | 155.91 | 30.72% | 119.27 |
| 4 | 20% | 155.49 | 32.68% | 149.92 | 27.92% | 117.19 |
| 4 | 40% | 190.78 | 31.98% | 186.34 | 29.00% | 144.55 |
| 6 | 5% | 172.02 | 27.25% | 171.33 | 26.74% | 135.18 |
| 6 | 20% | 235.70 | 34.99% | 223.96 | 28.27% | 174.60 |
| 6 | 40% | 290.44 | 43.39% | 291.62 | 43.98% | 202.54 |
| 8 | 5% | 242.09 | 33.52% | 229.44 | 26.54% | 181.31 |
| 8 | 20% | 258.40 | 35.56% | 252.77 | 32.61% | 190.61 |
| 8 | 40% | 354.56 | 52.66% | 345.69 | 48.82% | 232.29 |

Each cell represents the mean of the global cost and the mean error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Fig. 11 for the 5% scenario, in Fig. 12 for the 20% scenario, and in Fig. 13 for the 40% scenario.

Next, we tested our task allocation algorithms with the RRTs instead of the $A^*$ algorithm. The results are shown in Table II. First, it can be observed that these results are worse than using the $A^*$ algorithm. This makes sense since the RRTs algorithm does not ensure any kind of efficiency of the solution. Also, when RRTs are used, the differences between both algorithms decrease and there is even one case, where the BS algorithm performs a little bit better than the RTMA algorithm.

In summary, it has been shown that the performance of the task allocation algorithms is better with the $A^*$ algorithm rather than RRTs. The percentage on nonnavigable terrain in the scenario seems to not affect the performance of the system since similar results have been obtained for the three different scenarios. Also, the use of RRTs reduces the advantages obtained with a more complex algorithm, such as the RTMA algorithm, and make the results of both algorithms very similar.

We propose the use of the RTMA algorithm only when the application permits the use of the $A^*$ path planner. However, in applications where the $A^*$ algorithm is too slow, such as scenarios with high-dimensional state spaces with constraints, we propose the use of the BS algorithm since the use of a nonoptimal path planner (for example RRTs) is needed.

## V. REAL-WORLD EXPERIMENTS

In this section, the distributed task allocation algorithms, integrated within the explained robot architecture, are implemented in real robots. Numerous experiments have been run using the team of robots. Results from experiments with and without obstacles are commented.

### A. *Description of the Testbed*

We used a team of six mobile robots in order to test our task allocation algorithms. Each of these robots is based on the iRobot Create platform (see Fig. 14). This platform is ideal for the robotics research community due to its low price and open communication protocol with access to all its sensors and actuators. However, this platform is only suitable for indoor or flat terrain outdoor experiments.

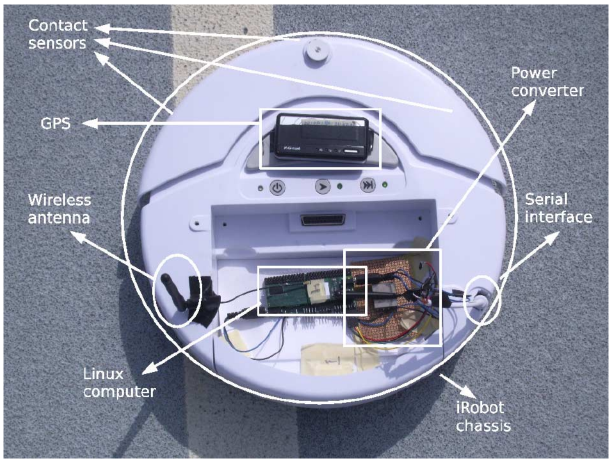

Fig. 14. Robot used in the experiments. iRobot Create upgraded with micro linux computer, wireless communication and GPS.

We have added to the platform a micro linux computer, the Connex 400XM processor from Gumstix. This motherboard contains a 400 MHz ARM processor, two serial ports compatible with TTL levels, and bluetooth capabilities. Additionally, two more boards were used: a Robostix board was added to have easy access to the serial ports and power supply pins, and a Wifistix board that provides wireless 802.11b/g capabilities to the linux computer. The Robostix board also includes an Atmel ATMega 128 RISC microcontroller, providing both SPI and I2C serial ports, general purpose IO pins, PWM outputs, and an ADC unit. These characteristics enable the Robostix to function as an ideal board to implement low-level controllers or to be used as an interface between the robot sensors and the linux computer. In this particular case, all these interfaces were not used since a unique serial port was used for the communication with the robot. Nevertheless, these characteristics will be used in the future with the integration of new sensors to the platform. The three boards create a compact, cheap, and small embedded computer suitable for applications that involve small robots. The iRobot battery was used to power the embedded computer. A dc/dc converter was used to stabilize the voltage of the iRobot battery and bring it down to the required level.

With respect to the sensors, a global positioning system (GPS) with a bluetooth interface was added to each robot as the main localization sensor. The GPS is an off-the-shelf product designed for cell phones and personal digital assistants (PDAs) with meter precision. The only problem with this GPS was that it gave us some trouble dealing with bluetooth interference between the different GPS. Since the quality of the GPS is not good enough [19], we used a Kalman filter [25] to combine the local odometry and the GPS measurements to obtain a decent global localization. The local odometry was obtained from the wheel encoders using the open communication protocol. The last sensors are the three front contact sensors that come with the iRobot platform (see Fig. 14). The actuators are the electric motors that move the wheels in a differential drive configuration. Information between the robots is exchanged over the bidirectional wifi link using standard user datagram protocol (UDP) sockets.

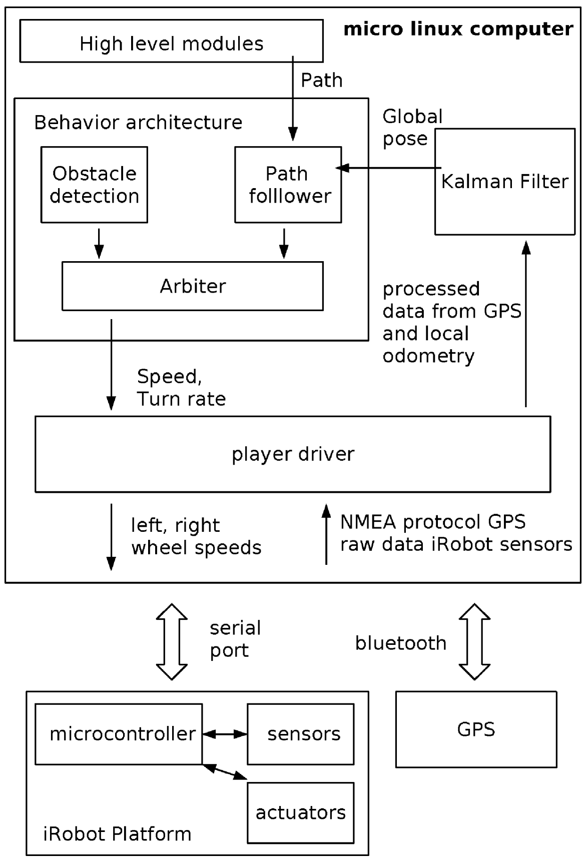

Fig. 15. Scheme that shows the integration of the micro linux computer with the robot platform and its sensors. It also depicts the relation between the different modules that implement the robot controller.

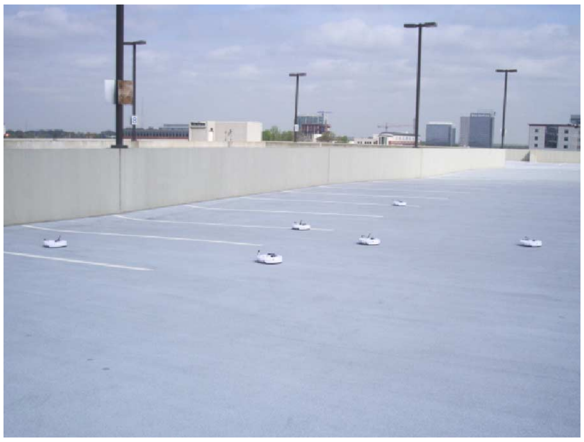

Fig. 16. Team of robots running one of the experiments in an arena of $10 \times 10$ m$^2$.

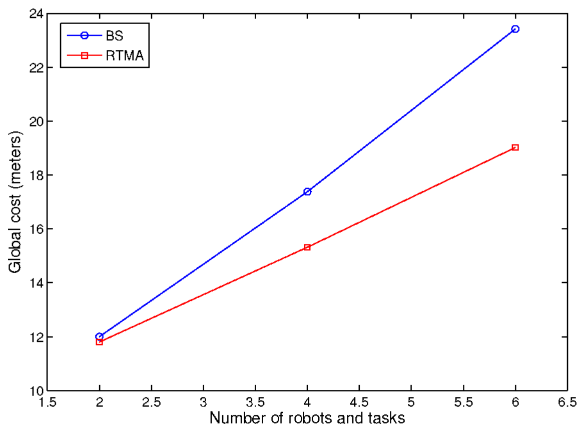

Fig. 17. Results from the experiments in an arena of $10 \times 10$ m$^2$ (mean of the global cost). The BS and RTMA algorithms have been tested with two, four, and six real robots, obtaining results similar to the simulated ones.

We have used the same multirobot architecture [16] that we implemented in the simulations to easily integrate robot modules with high-level algorithms such as our task allocation algorithms. This fact combined with the implementation of a player driver [5] to control the iRobot platform enabled us to use the same software both in simulation (using Player/Gazebo) and real experiments. After the allocation process has concluded, the high-level algorithm passes a path to the robot controller. The relation between the task allocation and path planner modules is the same as shown in Fig. 5. In this case, the robot controller is composed of two behaviors: obstacle detection and path follower. Both behaviors are combined using a DAMN architecture as explained in Section IV. The first behavior is used to move the robot backward when one of the contact sensors is activated. The second behavior uses the Pure Pursuit algorithm [18] to follow the received path. The votes from both behaviors are combined by an arbiter that communicates the desired speed and turn rate to the player driver (see Fig. 15). Then, the player driver communicates these values (as left and right wheel speeds) to the iRobot platform by means of the serial port and using the open communication interface. This player driver also reads the NMEA data units from the GPS and the sensor values transmitted by the iRobot platform, and transforms them to the data structures used in the Player project. These data structures are combined in the Kalman filter module to obtain an estimated global position that is used by the Path Follower module. One of the main advantages in using software from the Player project with commercial robots was that the development time was reduced significantly since the low-level communication with sensors and actuators was already implemented in the open-source project.

## B. *Results From Experiments*

First, experiments with different number of robots but without obstacles have been performed. Specifically, four experiments have been run with two robots, six with four robots, and eight with six robots. In total, 18 experiments have been run with each of the two algorithms. All these experiments have been performed in a $10 \times 10$ m$^2$ arena (see Fig. 16) where the positions of the robots and tasks have been calculated at random.

The main objective of these experiments is to show that our simulation results are still valid even when we use real robots, with all the noise and imperfections. The results from the experiments are shown in Fig. 17. It can be seen that these results follow the same trend as the simulation results where the RTMA algorithm obtains better results than the BS algorithm and the

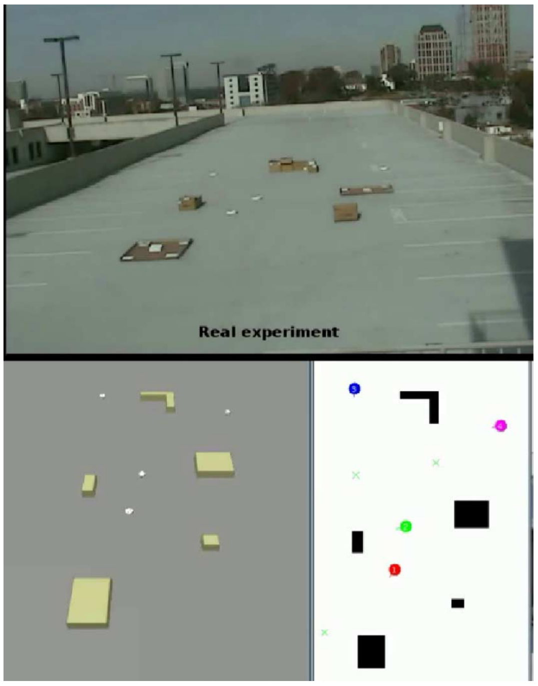

Fig. 18. Team of robots running one of the experiments in an arena of $15 \times 23$ m$^2$ with obstacles. Visual interface used to follow the experiments.

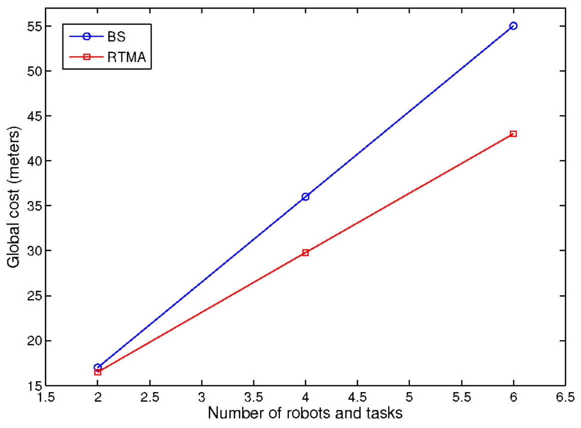

Fig. 19. Results from the experiments in an arena of $15 \times 23$ m$^2$ with obstacles (mean of the global cost). The BS and RTMA algorithms have been tested with two, four, and six real robots integrated in a complete robotic system including the $A^*$ algorithm as path planner.

difference between both of them increases with the number of robots and tasks.

On the other hand, experiments with obstacles have been performed. The same testbed was used but with an area of $15 \times 23$ m$^2$ (see Fig. 18). Since the best results are obtained with the $A^*$ algorithm, only experiments with this path planner were performed. Different number of robots were considered. Again, four experiments have been run with two robots, six with four robots, and eight with six robots. In total, 18 experiments have been run with each of the two algorithms. All these experiments have been performed in the described arena, where the positions of the robots and tasks have been calculated at random avoiding the areas with obstacles.

The results from the experiments are shown in Fig. 19. It can be seen that these results obtain similar results to the ones obtained in simulation, where obstacles were considered. It can be observed how the difference between both algorithms increases with the number of robots, and again, the RTMA algorithm obtains better results than the BS algorithm. Finally, some videos of the experiments can be viewed from [26].

## VI. Conclusion

The initial formation problem has been stated and two different algorithms that solve this problem in a distributed way have been explained. Two different types of formation have been used. In the first one, the error in comparison with the optimal solution seems to increase in a linear way with the number of robots and tasks. In the second one, the error is kept slightly constant. Also, in this second type of formation, the BS algorithm obtains much worse results than with the first type of formation. Therefore, the RTMA algorithm is considered more robust to the type of formation since its performance remains similar. This is possible due to the novel use of cost means to improve the efficiency of the task allocation algorithm.

Simulations in a realistic environment have been presented where the task allocation algorithm has been integrated with a path planning algorithm and the execution of the tasks is within a behavior architecture that combines a path following algorithm with obstacle avoidance. When the path planner algorithm $A^*$ is used, it has been proven that the improvements applied in the RTMA algorithm still obtain better results than the BS algorithm in a complete robotic system that considers most of the problems that are faced in a real application. However, when the path planner algorithm used is RRTs, the differences between both algorithms decreases. Therefore, we have shown that it is important to evaluate the performance of the task allocation algorithms considering the whole robotic system since a part of it (the path planner in our case) can affect the performance of the task allocation algorithm.

Finally, we have run experiments that show our results are still valid even when we use real robots, with all the noise and imperfections typical of real experiments.

## References

[1] D. P. Bertsekas, "A new algorithm for the assignment problem," *Math. Program.*, vol. 21, no. 1, pp. 152–171, Dec. 1981.

[2] B. Brumitt and A. Stenz, "Grammps: A generalized mission planner for multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Leuven, Belgium, 1998, vol. 2, pp. 1564–1571.

[3] P. Caloud, W. Choi, J. Latombe, C. Le Pape, and M. Yim, "Indoor automation with many mobile robots," in *Proc. IEEE Int. Workshop Intell. Robot. Syst. (IROS)*, Ibaraki, Japan, 1990, vol. 1, pp. 67–72.

[4] M. B. Dias and A. Stenz, "Opportunistic optimization for market-based multirobot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Lausanne, Switzerland, 2002, pp. 2714–2720.

[5] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. 11th Int. Conf. Adv. Robot. (ICAR)*, Coimbra, Portugal, 2003, pp. 317–323.

[6] B. P. Gerkey and M. J. Matarić, "Murdoch: Publish/subscribe task allocation for heterogeneous agents," in *Proc. 4th Int. Conf. Auton. Agents*, Barcelona, Spain, 2000, pp. 203–204.

[7] A. Howard and A. Viguria, "Controlled reconfiguration of robotic mobile sensor networks using distributed allocation formalisms," presented at the NASA Sci. Technol. Conf. (NSTC), College Park, MD, 2007.

[8] X. Hu and M. Egerstedt, “Formation constrained multi-agent control,” *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 947–951, Dec. 2001.

[9] M. Ji and M. Egerstedt, “Role-assignment in multi-agent coordination,” *Int. J. Assist. Robot. Mechatronics*, vol. 7, no. 1, pp. 32–40, 2006.

[10] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, “A practical decision-theoretic approach to multi-robot mapping and exploration,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, 2003, vol. 3, pp. 3232–3238.

[11] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Res. Logistics Quart.*, vol. 2, pp. 83–97, 1955.

[12] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[13] E. L. Lawler, *The Traveling Salesman Problem*. New York: Wiley, 1985.

[14] J. Lawton, R. Beard, and B. Young, “A decentralized approach to formation maneuvers,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 6, pp. 933–941, Dec. 2003.

[15] J.-H. Lee, K. Morioka, N. Ando, and H. Hashimoto, “Cooperation of distributed intelligent sensors in intelligent environment,” *IEEE/ASME Trans. Mechatronics*, vol. 9, no. 3, pp. 535–543, Sep. 2004.

[16] I. Maza, A. Viguria, and A. Ollero, “Networked aerial-ground robot system with distributed task allocation for disaster management,” in *Proc. IEEE Int. Workshop Saf., Secur. Rescue Robot*, Gaithersburg, MD, 2006.

[17] N. Nilson, *Problem-Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.

[18] A. Ollero and O. Amidi, “Predictive path tracking of mobile robots,” in *Proc. 5th Int. Conf. Adv. Robot., Robots Unstruct. Environ. (ICAR)*, 1991, vol. 2, pp. 1081–1086.

[19] S. Panzieri, F. Pascucci, and G. Ulivi, “An outdoor navigation system using GPS and inertial platform,” *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 2, pp. 134–142, Jun. 2002.

[20] L. E. Parker, “Alliance: An architecture for fault-tolerant multi-robot cooperation,” *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 220–240, Apr. 1998.

[21] “National workshop on future sensing systems—Living, nonliving, and energy systems,” Tech. Rep., Nat. Sci. Foundation, Lake Tahoe, NV, 2002.

[22] J. K. Rosenblatt, “DAMN: A distributed architecture for mobile navigation,” *J. Exp. Theor. Artif. Intell.*, vol. 9, no. 2, pp. 339–360, 1997.

[23] S. Sariel and T. Balch, “A distributed multi-robot cooperation framework for real time task achievement,” in *Proc. 8th Int. Symp. Distrib. Auton. Robot. Syst. (DARS)*, 2006, pp. 187–196.

[24] S. L. Smith and F. Bullo, “Target assignment for robotic networks: Asymptotic performance under limited communication,” in *Proc. Amer. Control Conf.*, New York, 2007, pp. 3585–3590.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambdrige, MA: MIT Press, 2005.

[26] A. Viguria and A. Howard. (2009). Videos of the experiments [Online]. Available: http://grvc.us.es/∼antidio/mechatronics_2009.html

[27] A. Viguria and A. Howard, “Upper-bound cost analysis of a market-based algorithm applied to the initial formation problem,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS 2007)*, San Diego, CA, pp. 2326–2331.

[28] A. Viguria, I. Maza, and A. Ollero, “SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Rome, Italy, 2007, pp. 3339–3344.

[29] B. B. Werger and M. J. Matarić, “Broadcast of local eligibility for multi-target observation,” in *Proc. Distrib. Auton. Robot. Syst. 4*, 2000, pp. 347–356.

[30] X. Yuan and S. X. Yang, “Multirobot-based nanoassembly planning with automated path generation,” *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 3, pp. 352–356, Jun. 2007.

[31] S. Yun and D. Rus, “Optimal distributed planning of multi-robot placement on a 3d truss,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, San Diego, CA, 2007, pp. 1365–1370.

[32] M. M. Zavlanos and G. J. Pappas, “Dynamic assignment in distributed motion planning with limited information,” in *Proc. Amer. Control Conf.*, New York, 2007, pp. 1173–1178.

**Antidio Viguria** received the Engineering degree in telecommunication from the University of Seville, Seville, Spain, in 2004, and the M.S. degree from Georgia Institute of Technology, Atlanta, in 2008. He is currently working toward the Ph.D. degree at the University of Seville.

From 2004 to 2006, he was a Researcher at the University of Seville. Beginning in 2006, he was a Fulbright Scholar and a Graduate Student at Georgia Institute of Technology. He is currently working at the Advanced Center for Aerospace Technologies (CATEC), Seville. His current research interests include multirobot coordination and unmanned aerial vehicles.

**Ayanna M. Howard** (SM’99) received the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 1994 and 1999, respectively, and the B.S. degree in engineering from Brown University, Providence, RI, in 2003.

She is an Associate Professor at Georgia Institute of Technology, Atlanta. Her current research interests include the concept of humanized intelligence, the process of embedding human cognitive capability into the control path of autonomous systems. Her accomplishments have been documented in over 12 featured articles.

Dr. Howard was named as one of the world’s top young innovators of 2003 by the prestigious *MIT Technology Review* journal and in the “Rise of the Machines” article published in *TIME Magazine* in 2004. She received the IEEE Early Career Award in Robotics and Automation in 2005.