# A Game of Drones: Game Theoretic Approaches for Multi-robot Task Allocation in Security Missions

**Conference Paper** *in* Advances in Intelligent Systems and Computing · November 2018

DOI: 10.1007/978-3-319-70833-1_69

CITATIONS
2

READS
299

**5 authors**, including:

Juan Jesús Roldán
Universidad Politécnica de Madrid
**35** PUBLICATIONS **197** CITATIONS
SEE PROFILE

Mario Garzon
National Institute for Research in Computer Science and Control
**36** PUBLICATIONS **178** CITATIONS
SEE PROFILE

Jaime Del Cerro
Universidad Politécnica de Madrid
**102** PUBLICATIONS **875** CITATIONS
SEE PROFILE

Antonio Barrientos
Universidad Politécnica de Madrid
**224** PUBLICATIONS **1,615** CITATIONS
SEE PROFILE

# A game of drones: game theoretic approaches for multi-robot task allocation in security missions

Kala Garapati[1], Juan Jesús Roldán[1], Mario Garzón[1], Jaime del Cerro[1], and Antonio Barrientos[1]

Centro de Automática y Robótica (UPM-CSIC), Universidad Politécnica de Madrid, José Gutiérrez Abascal, 2, 28006, Madrid, Spain

**Abstract.** This work explores the potential of game theory to solve the task allocation problem in multi-robot missions. The problem considers a swarm with dozens of drones that only know their neighbors, as well as a mission that consists of visiting a series of locations and performing certain activities. Two algorithms have been developed and validated in simulation: one competitive and another cooperative. The first one searches the best Nash equilibrium for each conflict where multiple UAVs compete for multiple tasks. The second one establishes a voting system to translate the individual preferences into a task allocation with social welfare. The results of the simulations show both algorithms work under the limitation of communications and the partial information, but the competitive algorithm generates better allocations than the cooperative one.

**Keywords:** Multi-robot mission, Swarm, Task allocation, Game theory, Security

## 1 Introduction

In the last years, multi-robot systems have been applied to multiple areas, such as security [1], search and rescue [2], environmental monitoring [3] and agriculture [4]. The advantages of these systems are diverse: they are more effective - since they have more resources to reach the same objectives - and efficient - since each task can be solved by the best agent, as well as they provide flexibility and fault tolerance.

Task allocation is a relevant problem in multi-robot systems, especially with large fleets and swarms. This problem can be solved by using multiple algorithms: centralized or distributed, deterministic or stochastic, market-based or society-based... The solution also depends on the criteria used for optimization: distance, time, consumption, coordination...

This work takes into account a scenario where a fleet of drones has to visit a set of locations to perform a set of activities. For instance, take pictures to locate victims and create a network to allow communications in a disaster area. The fleet cannot be connected to a station due to the large distance, presence of shadow zones and problems of security. The communications between the drones

are limited because the size of fleet makes it inapplicable and they could cause problems of security. The mission can be completed without solving all the tasks, but the performance depends on the number of solved tasks and the time to solve them.

The conditions of this scenario suggest the application of distributed algorithms. These algorithms should be able to work under limited communications between the agents and partial information of the mission. The proposal of this work is to use game theoretic algorithms, since they can be distributed in the fleet and solve the problem locally. Specifically, two algorithms have been developed and validated in simulation: one competitive and another cooperative. The first one poses conflicts between the drones and searches the best Nash equilibrium. The second one obtains the social allocation from the individual preferences by means of a voting system.

The rest of paper is organized as follows: Section 2 collects the state of art about task allocation algorithms. Section 3 contains the main notions of game theory that are applied to develop the algorithms. Section 4 describes the competitive algorithm, whereas section 5 describes the cooperative one. Section 6 addresses the simulations performed to validate both algorithms and discusses their results. Finally, section 7 summarizes the main conclusions of the work.

## 2 State of art

Multi-robot mission planning is a well-known problem that involves multiple issues, such as the design of fleets, the communications between robots, the control and coordination architectures, and the task and motion planning problems [5]. In turn, task planning can be split into task decomposition, which is the division of the mission into tasks that can be performed by single robots, and task allocation, which is the assignation of these tasks to the robots looking for optimizing performance. This latter problem is common in literature (several surveys can be found in the references [6], [7] and [8]).

The basic problem of task allocation consists of a set of tasks (T), a set of robots (R) and a set of utilities of the robots to execute the tasks (U). The objective of this problem is to find the allocation of the tasks to the robots that maximizes the utility of the fleet [9].

Nevertheless, this problem can reach more complexity. On the one hand, various variables can be used as utilities: mission time, distance covered by fleet, robot consumption or a set of them. On the other hand, several restrictions can be added to the planning, generating a constraint satisfaction problem [10]. Some of these restrictions are the features of the tasks (their geographic areas and time intervals), the dependencies between the tasks, which can be represented by Allen's Interval Algebra [11] (relationships of precedence, parallelism, meeting, overlapping, simultaneous starting or finishing), and the features of the robots (their dynamics and kinematics models, fuel or battery consumptions, initial positions and types of sensors and actuators).

As shown in figure 1, there are two main paradigms of task allocation algorithms: centralized and distributed. Centralized algorithms involve a central agent that allocates the tasks to the other agents. The main advantage of these algorithms is they can provide optimal solutions in terms of costs, time and resources, whereas the main drawbacks are they may not be robust (if the central agent fails, the task allocation is not possible) and scalable (they could not deal with huge amounts of tasks and robots). In contrast, distributed algorithms are based on the communication of information and allocation of tasks between equal agents. These algorithms have more flexibility, robustness and fault tolerance, but the quality of the solutions depends on the information managed by the agents.

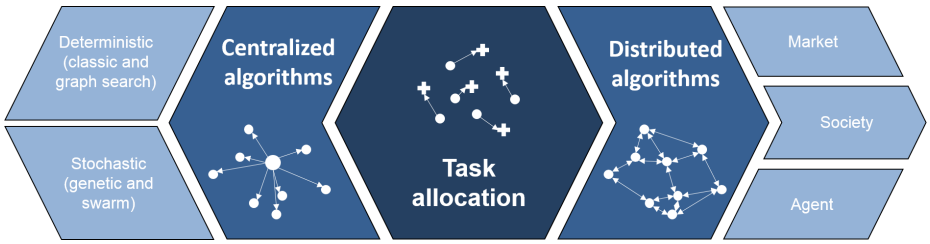



**Fig. 1.** Classification of task allocation algorithms.

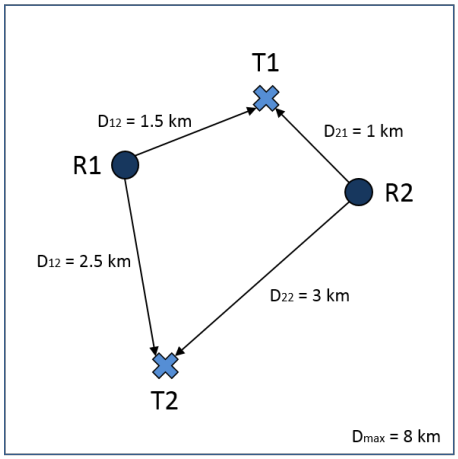
The centralized algorithms apply deterministic or stochastic techniques. Deterministic techniques follow rigid procedures, which means if they start from a certain initial conditions, they will always follow the same paths and reach the same results. A good example is the formulation of task allocation as a stable marriage problem and the application of Gale-Shapley algorithm to solve it [12]. Stochastic techniques integrate random factors, so the solutions are generated combining heuristic and randomness and the ones with the best performance are selected. Two of these techniques are the genetic algorithms [13] and particle swarm optimization [14].

The distributed algorithms include market, society and agent-based methods. Market-based approaches consider the robots follow their own interests and establish a mechanism of market for distributing the tasks. Two good examples are game theoretical algorithms [15] and auction mechanisms [16]. Society-based approaches propose a collective decision making in the fleet by using strategy diffusion [17] or coalition formation [18]. Agent-based approaches distribute the robots in the scenario and provide them with the capability to autonomously take nearby tasks [19].

## 3 Game theory and task allocation

Game theory applies mathematical models to study situations where some players have to make decisions to get some outcomes [20]. In our case, the players are

a set of R robots, the decisions are a set of T tasks and the outcomes represent the potential of the robots to perform the tasks. Figure 2 shows the normal form of the game with two robots and two tasks.




Game

| R1 / R2 | R2 → T1 | R2 → T2 |
|---|---|---|
| R1 → T1 | (0, 0) | (7.5, 6) |
| R1 → T2 | (6.5, 8) | (0, 0) |

Preferences

| R / T | T1 | T2 |
|---|---|---|
| R1 | 1 | 2 |
| R2 | 1 | 2 |

**Fig. 2.** A simple game defined by multi-robot task allocation.

The outcomes for the players are obtained through the utility defined in equation 1: its value is 0 when two robots perform the same task, and varies with the distance between the robot and task from 1 (maximum distance) to a $d_{max} + 1$ (minimum distance). This formula looks for punishing the assignation of the same task to multiple robots over any other consideration. Additionally, it establishes a direct proportionality instead of a inverse one in order to avoid the distortion caused by exponential variables.

$$U_{R_i,T_j} = \begin{cases} 0, & \text{if } \exists R_k : R_k \to T_j \\ d_{max} - d_{R_i,T_j} + 1, & \text{otherwise} \end{cases} \quad (1)$$

The solution of a game must be a Nash Equilibrium (NE), which is a profile of strategies where each player gets the highest available payoff and, therefore, he/she has not got incentives in changing his/her decisions.

Some games can be solved by searching dominant strategies and discarding dominated ones. A dominant strategy for a player is a decision that produces the highest payoff considering all the possible decisions of the rest of players. Similarly, a dominated strategy for a player is a decision that produces less payoff than other one independently of the decisions of the rest of players.

Nevertheless, we have found some particularities in the game defined by multi-robot task allocation. The first one is that there are not dominant strategies, since the utilities of the main diagonal are zero. This fact implies a robot cannot choose a task that is better than the rest, because any other robot could

choose the same task and the payoffs for both robots would be zero. The second one is that when the number of tasks is equal to the number of robots, all the elements out of main diagonal are Nash equilibria. In this situation, if a robot changes its task, it will choose the task of other robot and their payoffs will decrease to zero. These features have been taking into account at the time of developing the algorithms.

## 4 Competitive algorithm

The competitive algorithm poses conflicts between each robot and their neighbors and search the task allocation that provides the best Nash equilibrium. This method is shown in algorithm 1 and will be explained below.

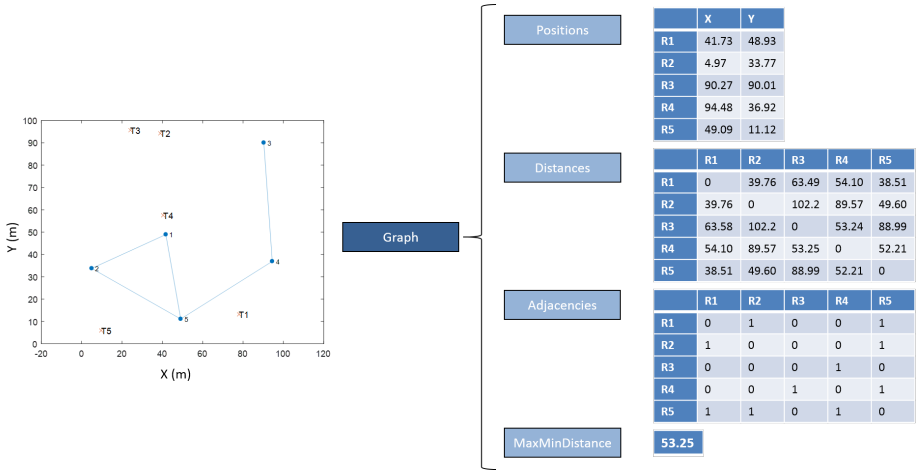
**Algorithm 1** Competitive algorithm.

```
function COMPETITIVE(R, T)
    graph = createGraph(R)
    tasks = createTasks(T)
    utility = calculateUtility(graph, tasks)
    for each robot ∈ graph do
        robot = robotAwareness(graph, utility)
        conflict = taskConflict(robot.preferences)
        if conflict = 0 then
            robot.allocation = robot.preferences
        else
            robot.allocation = searchBestNashEquilibrium(robot.preferences)
        end if
        taskAllocation ← robot.taskAllocation
    end for
    return taskAllocation
```

The first step of the algorithm is to generate a graph with the information of the fleet. As shown in figure 3, this information includes the positions of robots, the distances between them, the adjacencies among them, and the distance required to connect all the robots to the graph. This graph determines the communications among the robots, as well as the information that they can manage.

Then, the algorithm uses the robot graph and the task list to calculate the utility matrix, which contains the payoffs that the robots can obtain by performing the tasks. Although the robot graph, task list and utility matrix contain complete information, the robots only manage the partial information that they know according to their locations and connections.

The structure with the information managed by each robot, which can be seen in figure 4, is generated by an awareness function. This information includes the connections of the robot with its neighbors, their utilities and preferences




| | X | Y |
|---|---|---|
| R1 | 41.73 | 48.93 |
| R2 | 4.97 | 33.77 |
| R3 | 90.27 | 90.01 |
| R4 | 94.48 | 36.92 |
| R5 | 49.09 | 11.12 |

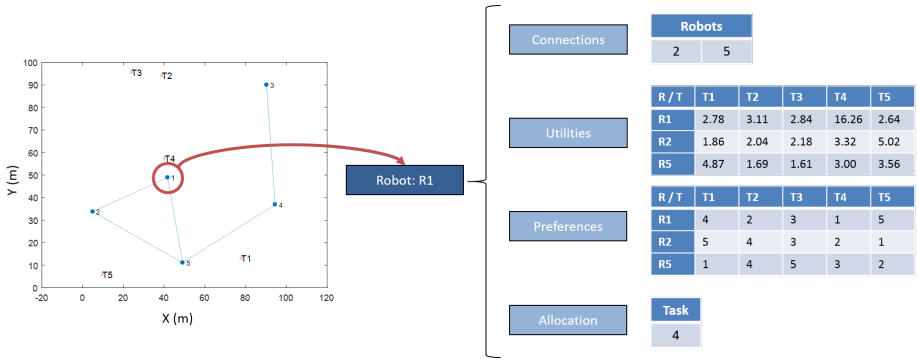
| | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| R1 | 0 | 39.76 | 63.49 | 54.10 | 38.51 |
| R2 | 39.76 | 0 | 102.2 | 89.57 | 49.60 |
| R3 | 63.58 | 102.2 | 0 | 53.24 | 88.99 |
| R4 | 54.10 | 89.57 | 53.25 | 0 | 52.21 |
| R5 | 38.51 | 49.60 | 88.99 | 52.21 | 0 |

| | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| R1 | 0 | 1 | 0 | 0 | 1 |
| R2 | 1 | 0 | 0 | 0 | 1 |
| R3 | 0 | 0 | 0 | 1 | 0 |
| R4 | 0 | 0 | 1 | 0 | 1 |
| R5 | 1 | 1 | 0 | 1 | 0 |

| MaxMinDistance |
|---|
| 53.25 |

**Fig. 3.** An example of graph class and how it is managed by algorithm.

regarding the tasks and, when the algorithm ends, the task that the robot must perform.




| Robots | |
|---|---|
| 2 | 5 |

| R / T | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|
| R1 | 2.78 | 3.11 | 2.84 | 16.26 | 2.64 |
| R2 | 1.86 | 2.04 | 2.18 | 3.32 | 5.02 |
| R5 | 4.87 | 1.69 | 1.61 | 3.00 | 3.56 |

| R / T | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|
| R1 | 4 | 2 | 3 | 1 | 5 |
| R2 | 5 | 4 | 3 | 2 | 1 |
| R5 | 1 | 4 | 5 | 3 | 2 |

| Task |
|---|
| 4 |

**Fig. 4.** An example of robot class and how it is managed by algorithm.

The next step is to determine if there are conflicts between the robot and their neighbors for the tasks. If there is not any conflict, the robot can perform its preferred task. Otherwise, the robot has to compete against its neighbors to achieve its preferred task.

This competition is split into two functions. In the first one, all the Nash equilibria are searched in the preference matrix and stored in an equilibria matrix. It must be remarked that here the information is partial and the numbers of robots and tasks can be different, so the permutations can be or not be Nash equilibria. In the second one, the best Nash equilibrium in terms of social util-

ity is searched in the equilibria matrix. Three criteria can be used: the sum of utilities (which is related to the total distance and the fleet consumption), the minimum utility (which is related to the maximum robot distance and the mission time), and the difference between maximum and minimum utilities (which allows to synchronize the movements of the robots).

The functions to solve the conflicts are complete (i.e. in a conflict of $R_c$ robots for $T_c$ tasks, all the robots perform different tasks, so if $R_c$ and $T_c$ are equal, all the tasks are performed). However, the limitation of information may introduce errors in the allocation of tasks. For instance, if two drones are close to the same task but they do not know each other, both drones may compete with other drones and win the right to perform this task. This fact may cause some tasks are unassigned while others are performed by multiple robots, but it is the price to be paid to use a distributed algorithm that is able to work with partial information.

## 5 Cooperative algorithm

The cooperative algorithm establishes a hybrid scheme with leaders, citizens and dependents, as well as a voting system to allocate the tasks to the robots. This method is shown in algorithm 2 and will be explained below.

**Algorithm 2** Cooperative algorithm.

```
function COOPERATIVE(R, T)
    graph = createGraph(R)
    tasks = createTasks(T)
    utility = calculateUtility(graph, tasks)
    [leaders, citizens, dependents] = clasifyRobots(graph)
    for each leader ∈ leaders do
        leader = robotAwareness(graph, utility)
        for each citizen ∈ citizens do
            citizen = robotAwareness(graph, utility)
            citizen.vote = votePreferences()
        end for
        leader.vote = votePreferences()
        leader.result = countPreferences()
        leader.allocation = searchBestNashEquilibrium(leader.result)
        taskAllocation ← leader.taskAllocation
    end for
    return taskAllocation
```

The implementation of a voting system requires giving up one of the assumptions of the work: to allow the communications between the robots until all of them know the votes of the rest, or to pass from the distributed to a hybrid scheme where a group of robots count the votes. In this work, we have chosen

the second option, establishing three types of robots: leaders (they vote and count), citizens (they only vote) and dependents (their preferences are voted by citizens and leaders).

The first steps of cooperative algorithm are similar to competitive one. The graph of robots is built, the list of tasks is created and the utilities of the robots to perform the tasks are computed.

After these steps, the algorithm selects the leaders according to the concept of centrality. We have studied various types of centrality (betweenness, closeness, degree...) and we have chosen the one based on betweenness [21]. This centrality quantifies the number of times that a node take part of the shortest path between other two nodes. The algorithm orders the robots by decreasing centrality and chooses leaders until every robot is connected to a leader. Then, it breaks the required connections to ensure that every citizen is only connected to one leader. The next steps are voting and counting, which are shown in figure 5 and explained below.

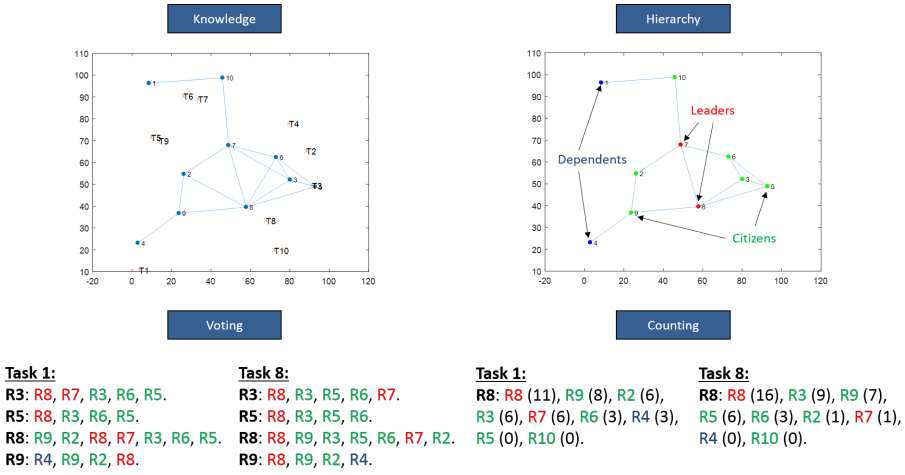



**Fig. 5.** An example with the procedures of voting and counting.

During the voting step, each leader or citizen makes a list with the robots that perceives as the bests for performing each task. The results are a series of matrices with $T$ columns that correspond to all the tasks and $r_i \leq R$ rows that contain the known robots ordered by decreasing utility. Then, the citizens transfer their votes to the leaders, whereas the leaders gather all the votes.

During the counting step, the leaders take the preferences of the citizens and themselves and transform them into task allocations. For this purpose, we have studied multiple voting systems (plurality with elimination, Borda count, successive elimination...) and we have selected the Borda count [22], because it considers all the preferences and it is not affected by the order. In this system,

the $r_i$ robots voted for $t_j$ task by $i$ robot receive scores from $r_i - 1$ (the first) to 0 (the last).

Finally, the leaders sum the scores for all the robots and search the best Nash equilibria, obtaining the assignations of tasks for all the robots (leaders, citizens and dependents). For this purpose, they have to create virtual robots to ensure the number of robots is greater or equal than the number of tasks. This virtual robots have null utility for all the tasks, so they are not chosen for performing them. Last, the leaders transfer the citizens their assignations and all the robots are able to perform their tasks.

## 6 Experiments and results

The experiments consisted of 50 simulations performed in *Matlab* that took into account 50 robots and tasks. The results of these simulations are shown in table 6, where "completed tasks" represents the percentage of tasks that are assigned to any robot and "social utility" means the deviation from the optimality in the sum of utilities of all the robots. The optimal task allocation is computed from the complete information of the game by a centralized greedy algorithm. As shown in this table, the competitive algorithm obtains better results than the cooperative one regarding to both variables.

**Table 1.** Results of the simulations.

| Variable | Competitive | Cooperative |
|---|---|---|
| Completed tasks (mean) | 82.80% | 79.40% |
| Completed tasks (std. dev.) | 7.57% | 10.38% |
| Social utility (mean) | 86.74% | 65.03% |
| Social utility (std. dev.) | 6.33% | 7.70% |

A one-way analysis of variance (ANOVA) has been performed in order to check the significance of results. This analysis shows that the difference between competitive and cooperative algorithms in terms of "social utility" is significant with $\alpha = 0.05$ ($F = 237.21$, $p = 0.00$), whereas the difference in terms of "completed tasks" is significant considering $\alpha = 0.10$ ($F = 3.5$, $p = 0.06$). Figure 6 shows the box and whisker diagrams for "completed tasks" and "social utility" of competitive and cooperative algorithms.

## 7 Conclusions

This work explores the application of game theoretic algorithms in task allocation problems, taking into account distributed fleets, limited communications and partial information. Specifically, we have developed, tested and compared a distributed competitive algorithm and a hybrid cooperative one.

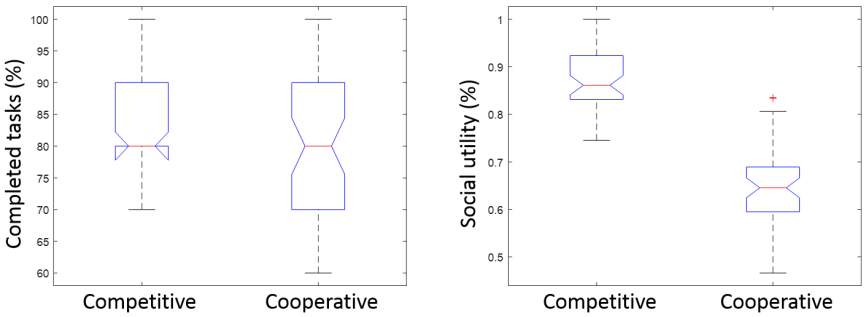



**Fig. 6.** Box and whisker diagrams for "completed tasks" and "social utility" of competitive and cooperative algorithms.

The developed algorithms are able to allocate more than the 80% of the tasks. The competitive one reaches a performance of 80% related to the optimal centralized algorithm, whereas the cooperative one reaches the 65% of this algorithm. The overall results can be considered as positive, taking into account the restrictions due to the algorithm distribution and partial information.

The error of competitive algorithm can be explained because the agents have partial information. Therefore, if two robots do not know each other, they may decide to do the same task. In the case of cooperative algorithm there is another relevant factor: the robots with more neighbors will receive more votes, which gives priority to leaders and popular citizens over the rest of citizens and the dependents.

Two algorithms will be developed in future works: a hybrid competitive one and a distributed cooperative one. The first one will combine the competitive algorithm with the use of leaders. This strategy could lead to an approximation to the results of centralized algorithm, since there are less robots involved in decision making. The second one will apply the voting system by taking advantage of the communications instead of establishing a hierarchy. In this case, the objective is to avoid the problems caused by partial information.

**Acknowledgments.** This work is framed on SAVIER (Situational Awareness Virtual EnviRonment) Project, which is both supported and funded by Airbus Defence & Space. The research leading to these results has received funding from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. Fase III; S2013/MIT-2748), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU, and from the DPI2014-56985-R project (Protección robotizada de infraestructuras críticas) funded by the Ministerio de Economía y Competitividad of Gobierno de España.

## References


1. Garzón, M., Valente, J., Roldán, J. J., Cancar, L., Barrientos, A., Del Cerro, J. A multirobot system for distributed area coverage and signal searching in large outdoor scenarios. Journal of Field Robotics (2015).
2. Garzón, M., Valente, J., Zapata, D., Barrientos, A. An aerial-ground robotic system for navigation and obstacle mapping in large outdoor areas. Sensors, 13(1), 1247-1267 (2013).
3. Alvarado, M., Gonzalez, F., Fletcher, A., Doshi, A. Towards the development of a low cost airborne sensing system to monitor dust particles after blasting at open-pit mine sites. Sensors, 15(8), 19667-19687 (2015).
4. Roldán, J. J., Garcia-Aunon, P., Garzón, M., de León, J., del Cerro, J., Barrientos, A. Heterogeneous Multi-Robot System for Mapping Environmental Variables of Greenhouses. Sensors, 16(7), 1018 (2016).
5. Yan, Z., Jouandeau, N., Cherif, A. A. A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems, 10(12), 399 (2013).
6. Mosteo, A. R., Montano, L. A survey of multi-robot task allocation. Instituto de Investigación en Ingeniería de Aragón, University of Zaragoza, Zaragoza, Spain, Technical Report No. AMI-009-10-TEC (2010).
7. Jiang, Y. A survey of task allocation and load balancing in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 27(2), 585-599 (2016).
8. Jia, X., Meng, M. Q. H. A survey and analysis of task allocation algorithms in multi-robot systems. In Robotics and biomimetics (ROBIO), 2013 IEEE international conference on (pp. 2280-2285). IEEE (2013).
9. Khamis, A., Hussein, A., Elmogy A. Multi-robot task allocation: A review of the state-of-the-art, in: Cooperative Robots and Sensor Networks 2015, Springer, pp. 31-51 (2015).
10. Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M. D., Camacho, D. Performance Evaluation of Multi-UAV Cooperative Mission Planning Models. In Computational Collective Intelligence (pp. 203-212). Springer International Publishing (2015).
11. Allen, J. F. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11), 832-843 (1983).
12. Roldán, J. J., Lansac, B., del Cerro, J., Barrientos, A. A Proposal of Multi-UAV Mission Coordination and Control Architecture. In Robot 2015: Second Iberian Robotics Conference (pp. 597-608). Springer International Publishing (2016).
13. Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M. D., Camacho, D. Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. Soft Computing, 1-18 (2016).
14. Yang, J., Zhang, H., Ling, Y., Pan, C., Sun, W. Task allocation for wireless sensor network using modified binary particle swarm optimization. IEEE Sensors Journal, 14(3), 882-892 (2014).
15. Hernández, E., Barrientos, A., Del Cerro, J. Selective Smooth Fictitious Play: An approach based on game theory for patrolling infrastructures with a multi-robot system. Expert Systems with Applications, 41(6), 2897-2913 (2014).
16. Schneider, E., Sklar, E. I., Parsons, S., Ozgelen, A. T. Auction-based task allocation for multi-robot teams in dynamic environments. In Conference Towards Autonomous Robotic Systems (pp. 246-257). Springer International Publishing (2015).
17. Jiang, Y. Concurrent collective strategy diffusion of multiagents: the spatial model and case study. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 39(4), 448-458 (2009).

18. Padmanabhan, M., Suresh, G. R. Coalition formation and Task Allocation of multiple autonomous robots. In Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on (pp. 1-5). IEEE (2015).
19. Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., Dorigo, M. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. Autonomous agents and multi-agent systems, 28(1), 101-125 (2014).
20. Jackson, M. O. A brief introduction to the basics of game theory. Stanford University (2011).
21. Freeman, L. C. A set of measures of centrality based on betweenness. Sociometry, 35-41 (1977).
22. Chamberlin, J. R., Courant, P. N. Representative deliberations and representative decisions: Proportional representation and the Borda rule. American Political Science Review, 77(03), 718-733 (1983).