Intelligent Service Robotics (2021) 14:707–727
https://doi.org/10.1007/s11370-021-00393-4

ORIGINAL RESEARCH PAPER

Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms

J. G. Martin[1] · J. R. D. Frejo[1] · R. A. García[1] · E. F. Camacho[1]

Received: 16 January 2021 / Accepted: 13 October 2021 / Published online: 13 November 2021

Abstract
The paper proposes the formulation of a single-task robot (ST), single-robot task (SR), time-extended assignment (TA), multi-robot task allocation (MRTA) problem with multiple, nonlinear criteria using discrete variables that drastically reduce the computation burden. Obtaining an allocation is addressed by a Branch and Bound (B&B) algorithm in low scale problems and by a genetic algorithm (GA) specifically developed for the proposed formulation in larger scale problems. The GA crossover and mutation strategies design ensure that the descendant allocations of each generation will maintain a certain level of feasibility, reducing greatly the range of possible descendants, and accelerating their convergence to a sub-optimal allocation. The proposed MRTA algorithms are simulated and analyzed in the context of a thermosolar power plant, for which the spatially distributed Direct Normal Irradiance (DNI) is estimated using a heterogeneous fleet composed of both aerial and ground unmanned vehicles. Three optimization criteria are simultaneously considered: distance traveled, time required to complete the task and energetic feasibility. Even though this paper uses a thermosolar power plant as a case study, the proposed algorithms can be applied to any MRTA problem that uses a multi-criteria and nonlinear cost function in an equivalent way. The performance and response of the proposed algorithms are compared for four different scenarios. The results show that the B&B algorithm can find the global optimal solution in a reasonable time for a case with four robots and six tasks. For larger problems, the genetic algorithm approaches the global optimal solution in much less computation time. Moreover, the trade-off between computation time and accuracy can be easily carried out by tuning the parameters of the genetic algorithm according to the available computational power.

Keywords Multi-robot system · Task planning · Multi-robot task allocation (MRTA) · Robotic sensor network · UAV · UGV · Branch and bound · Genetic algorithm · Thermosolar plant

1 Introduction

In recent decades, there have been great advances in the field of robotics with several works with humanoid robots [1], robotic arms [2] and in mobile robots or *unmanned vehicles*. Particularly, these unmanned vehicles have been used for plenty of different applications such as agriculture [3,4], mapping [5,6], surveillance [7,8] or exploration [9].

Multi-robot systems (MRS) are groups of robots which aim to collaborate in order to fulfill a set of tasks in an efficient manner [10,11]. MRS can be used to tackle a set of tasks, which entails multiple advantages in comparison with the use of a single robot, i.e., complex tasks that are difficult to address by using a single robot might be easily solved by combining two or more robots, thus increasing global efficiency and decreasing task completion times.

1.1 Multi-robot task allocation

MRS typically have a multi-robot task allocation (MRTA) problem associated. MRTA problems have been widely studied in the last two decades [12–15]. This problem addresses the question of allocating tasks to robots in the best pos-

✉ J. G. Martin
jgarmar@us.es

J. R. D. Frejo
jdominguez3@us.es

R. A. García
ramongr@us.es

E. F. Camacho
efcamacho@us.es

1 Department of Systems Engineering and Automation, University of Seville, Seville, Spain

sible way to optimize one or more performance indexes, such as energy consumption or total time spent among others. According to the taxonomy proposed in [16], the MRTA problem can be classified as:

- Single task robot (ST) or multi-task robot (MT) problem, depending on the number of tasks each robot can perform at a time: just one or more than one.
- Single-robot task (SR) or multi-robot task (MR) problem, depending on the number of robots needed to perform a task: only one or more than one.
- Instantaneous assignment (IA) or time-extended assignment (TA), depending whether future allocations are taken into account or not.

A further classification typology can be added according to the different types of robots used:

- Homogeneous MRTA problem if all the robots have the same characteristics.
- Heterogeneous MRTA problem if there are different types of robots with their own characteristics.

Within MRTA solution approaches, two clear types can be distinguished: *centralized* and *decentralized*. In many cases, the use of a centralized approach in which we focus in this paper makes it possible to improve the performance substantially but in return it has the negative counterpart of increasing the computational load. The performance also depends on the algorithm used. The most commonly used are either:

- Market-based approaches such as auction algorithms [17,18] are inspired in economy and provide a simple way of allocating resources but imply explicit communication among the agents, which sometimes may cause the communication cost be too high. The main advantage of these approaches is the scalability and robustness they exhibit, although optimality is never ensured since any agent of the system has complete information about it. Auctions have been widely used throughout human history in order to allocate resources and in the case of robots, which are not programmed to be selfish or to lie, the auction-based algorithm [18–20] is appropriate.
- Optimization-based algorithms [21] focus on finding the optimal solution of a problem mathematically, given a set of constraints (usually a NP-Hard problem with the corresponding computational burden) like in the Optimal Assignment Problem (OAP) [22]. These approaches often depend on a central agent and as a consequence, are highly susceptible to attacks or communication failures. Within these algorithms, there are two main subtypes:
  - Deterministic, such as gradient and Hessian-based methods [23,24].
  - Stochastic, such as meta-heuristic algorithms [25] like the Ant Colony Algorithm [26], Cuckoo Search [27–29], Particle Swarm Optimization (PSO) [30–32] or genetic algorithm (GA) [33] among others.

1.2 MRTA in the context of thermosolar plants

Thermosolar plants cover large extensions of land situated in areas of high solar irradiance, in which mirrors are used to concentrate solar thermal energy to generate electric power. The most common type of thermosolar plant nowadays is the Concentrated Solar Plant with Parabolic Trough Collector (CSP PTC).

This type of plant uses parabolic mirrors to focus the solar energy toward some piping situated in the focal point of the parabola. Thermal oil is used as a heat transfer fluid (HTF) circulating through these pipes and absorbing radiation. Then, the HTF is transported to a power generation plant through a manifold that collects the hot oil from the collectors. Interested readers can find more information about these plants in [34].

CSP PTC plants are usually operated by controlling the total flow that goes through the manifolds, generally with the aim of maintaining the oil temperature close to a given setpoint. However, this control has an important drawback whenever the irradiance is not uniform in the whole plant. A localized decrease on the irradiance, produced by a cloud, might trigger a decrease in the flow, but since the decrease would be localized in a certain area of the plant there would be other collectors where this decrease in flow might induce an increase in the oil temperature. Increases in the oil temperature above the maximum admissible can be harmful for the process and even dangerous to the plant equipment and therefore, in some cases, collectors must be defocused, as a security measure, although that means throwing away energy.

The problem of defocusing has been addressed from different approaches by proposing the use of predictive control strategies as model predictive control (MPC), [35,36]. In other works, as [37], controlling the flow in each collector by using valves is proposed. For these types of controllers, having an estimation of the distribution of the irradiance throughout the plant is needed.

MRS are frequently used in mobile robot sensor networks (RSN) [38], a particular case of wireless sensor networks (WSN) [39], where assembling a plant-wide network with sensors is considered an extremely expensive option and an alternative solution is sought by using robots that transport the sensors to different places in the plant. This is the case of thermosolar plants, where on the one hand, direct normal irradiance (DNI) sensors, pyrheliometers, are quite expensive and, on the other hand, there is a familiar and structured environment where all the possible paths for the robots can be pre-determined. Therefore, in this scenario, unmanned vehicles equipped with DNI sensors would be sent to different areas of the plant to take measurements [40]. Since a robot cannot take measurements at two different places at the same time, although going to a spot first and then to another is possible, we are dealing with a ST-TA problem according to the classification previously introduced. If we also take into account that we only need one vehicle to take a measurement and that vehicles can be of different types, we can complete the classification of the problem as a heterogeneous, ST , SR,TA, MRTA one. Another work where a MRS has been proposed in the solar power plants context is [41], where mobile robots are proposed for the inspection and maintenance of the plants.

In this paper, a genetic algorithm and a branch and bound algorithm are applied to a specific, heterogeneous, ST-SR-TA MRTA problem, particularized in a case study of a RSN that collects data all along a thermosolar power plant. For this purpose, two types of robots with different possible paths and speeds are proposed: UGVs and UAVs.

1.3 Contributions and outline

The main contribution of this work is the MRTA problem formulation of an MRS in the context of inspection and measurement tasks in a solar thermal plant. The problem has been formulated using a cost function with multiple objectives designed taking into account that our MRS is of the ST-SR-TA type as well as the fact that there may be tasks that are more urgent than others or that it may be more desirable to use some robots instead of others, e.g., it is better to use ground vehicles whenever possible since that way, the battery of aerial vehicles will be preserved in case some urgent missions arise.

This formulation includes the use of discrete variables that code the information of the allocations and allows to greatly increase the speed of obtaining a solution since the number of variables involved in coding an allocation falls drastically. Finally, both a B&B algorithm and a genetic algorithm have been specially designed for this formulation and a comparison of the two approaches has been made. The genetic algorithm designed makes use of a set of solutions associated with the problem that are used by the algorithm as initial population, which speeds up its performance and ensures a good solution for the allocation.

This paper is organized as follows. The problem statement and the mathematical formulation are presented in Sect. 2. In Sect. 3, the proposed algorithms to approach the problem are presented. In Sect. 4, the problem is particularized for a case of study based on a thermosolar power plant and a fleet of robots composed of UAVs and UGVs. The results of proposing several scenarios for the optimization of the problem are analyzed in Sect. 5. Finally, the conclusions and future works are drawn in Sect. 6.

2 Problem statement

In this section, the MRTA problem considered is formulated in the most generic possible way, assuming that there is a fleet of robots receiving orders from an upper layer, which also decides which tasks are to be done. It should be taken into account that, although in this paper we apply this formulation in the context of CSP PTC, an equivalent formulation consisting of using a MRS as a RSN can be efficiently used for other MRTA problems.

The MRTA problem considered is expressed mathematically for a set of N robots $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ and a set of M tasks $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. According also to the taxonomy mentioned in Sect. 1, the problem can be defined as follows:

- Heterogeneous, as there can be different types of robots.
- ST, as each robot can only fulfill one task at a time.
- SR, as only one robot is needed to fulfill one task.
- TA, since obtaining the optimal allocation for all tasks requires having into account that:
  - A robot can fulfill a new task once it has completed the previous task.
  - Each task requires a certain amount of time to be completed.
  - The best allocation may require that one or more robots stay still while other robots perform more than one tasks.

2.1 Discrete variables

A set of discrete variables $u_i(n)$ has been considered to model the problem mathematically. These variables, $u_i(n)$, represent the nth allocated task of robot i, i.e., n provides information on the order in which a robot is performing the allocated tasks. Thus, $u_i(n) \in \mathcal{S} \cup \{0\}$, since robot i is either performing a task as its nth mission or not doing anything ($u_i(n) = 0$). Note that if this variable takes value 0 for a robot i at any n_k, $u_i(n_k) = 0$, it makes no sense that any $u_i(n_{k+l}) \neq 0$ with $l > 0$. As n must be as large as the maximum number of tasks that a single robot can perform and there are no limitations on this respect. The most extreme case is that one single robot performs all the tasks and so, it is considered that $n \in [1, \ldots, M]$ and as a consequence, there are $N \times M$ variables. An example of these variables can be seen in Table 1

One of the main advantages of using discrete variables is that the constraints associated with the SR problem (a

Table 1 In the allocation presented, robot 1 does task 1 first and, once, it is finished disregarding the time its takes to do so, it performs task 3 while robot 2 performs task 2

Robot	First robot			Second robot		
	$u_1(1)$	$u_1(2)$	$u_1(3)$	$u_2(1)$	$u_2(2)$	$u_2(3)$
U	1	3	0	2	0	0

Note that since $u_1(3) = 0$, robot 1 is not doing anything after fulfilling task 3 and the same happens with robot 2 after completing task 2

robot cannot be assigned to more than one task at a time) are satisfied automatically with no need of soft or hard constraints. A further advantage is that the number of discrete variables used is smaller than when formulating the problem with binary constraints, therefore making the computational cost lower and making addressing nonlinear cost functions easier.

2.2 Cost function

In order to carry out the optimization, the proposed cost function must take into account multiple criteria:

- Time employed to complete each task, i.e., the amount of time elapsed between the allocation and the completion of a task.
- Total distance traveled by each robot.
- Feasibility of the allocation from an energetic point of view.

However, the approach proposed allows changing these criteria just by changing the cost function.

In this work, we have also considered that:

- Each task may have a different weight as not all the tasks are equally urgent. Moreover, it may be more desirable to use a certain robot than another, which makes the distance traveled by robots have different weights as well.
- It is also necessary to take into account that allocations must be feasible, i.e., an allocation cannot use more energy than the energy available. Notice as well that a robot must not only have enough energy to fulfill its allocated tasks but also to go to a charge station after completing them.

A cost function based on different criteria such as the delay on performing the tasks or the distance traveled by robots can be formulated as follows:

$$\min_{U} \quad J = \sum_{j=1}^{M} \delta_j t_j(U) + \sum_{i=1}^{N} \lambda_i d_i(U) + \gamma(U)$$
$$\gamma(U) = \alpha_1 \gamma_1(U) + \alpha_2 \gamma_2(U)$$
$$\text{s.t.} \quad u_i(n) \in \mathcal{S} \cup \{0\} \; \forall \, i, n \qquad (1)$$

where $U = [u_1(1), u_1(2), .., u_1(M), u_2(1), \ldots, u_N(M)]$, δ_j corresponds to the priority given to a certain task j, $t_j(U)$ is the time that it takes to complete task j in a given allocation, λ_i corresponds to the penalty of using robot i and $d_i(U)$ corresponds to the distance traveled by robot i. Note that these parameters may vary depending on the problem and they depend on the application (there may be applications in which performing the tasks as fast as possible is desired, applications where limiting the movement of the robots is the best option, or mixed applications where we can have both, robots mobility limited differently depending on the robot and urgent and minor tasks). Function $\gamma(U)$ is used to model the following soft constraints:

- Function $\gamma_1(U)$ is a function which takes value 0 when the allocation is feasible from an energetic point of view, i.e., when there is enough battery for the allocation to be fulfilled and to reach the battery stations from the final points. In case the allocation is not feasible, it takes the same value as the number of robots with not enough energy to fulfill the allocation. This way, the penalization is higher when the allocation implies more robots out of energy.
- Function $\gamma_2(U)$ is used to ensure that all the tasks are performed and that each task is allocated only once. Thus, it models the ST formulation of the problem. It takes value 0 when no task is repeated and every task is performed and otherwise, it takes a value that equals the number of tasks allocated to more than one robot plus the number of tasks not allocated to any robot.

The weights α_1 and α_2 are much larger than the rest of the weights of the cost function since it is better to obtain an allocation that complies with the constraints than one that does not.

Note that the constraints can be modeled as soft constraints since non-compliance is not critical. The fact that an allocation makes a robot perform a task that has already been performed by a different robot is not desirable, but there is no physical problem with it. On the other hand, the fact that a robot does not have enough energy to carry out the tasks allocated to it will only imply that some of those tasks will not be performed in time, since the robot will have to recharge its battery. Anyway, in the case that an unfeasible allocation has to be avoided, the value of α_1 and α_2 can be increased. Finally, it should be noted that in the case that there is no feasible solution, posing the constraints as *soft* constraints will make the algorithms return those allocations that are closer to the feasible zone.

The problem stated in this paper (1), and the algorithms proposed in Sect. 3, do not depend on how functions t_j, d_i

Table 2 Note that U_1 and U_2 represent the same assignment under this formulation since both represent that robot 1 performs first task 1 and then task 3 and robot 2 performs task 2

Robot	First robot			Second robot		
U_1	1	3	0	0	2	0
U_2	1	0	3	2	0	0

and γ are obtained, i.e., these can be either linear or non-linear functions. In fact, other performance indexes, such as energy consumption or carbon emissions, could be added to the current ones, or even replace them, changing neither the problem formulation nor the proposed algorithm. However, it is important to remark that the faster these functions are computed, the larger the problem solved optimally could be.

The specific functions used in this paper are detailed in Sect. 4.3.

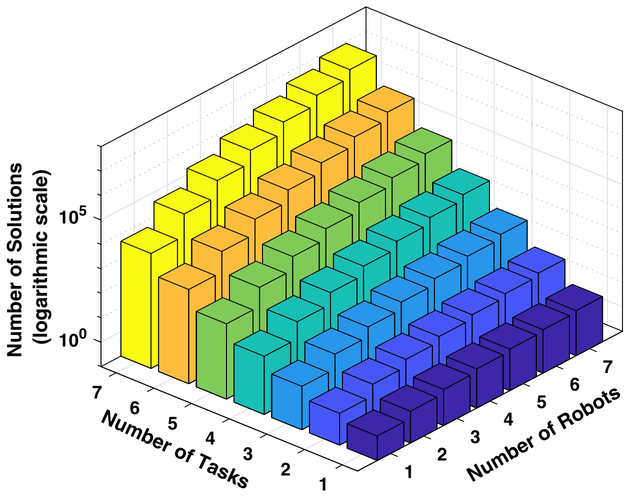

Fig. 1 Number of solutions (in logarithmic scale) depending on the number of tasks and the number of robots (from 1 to 7 robots and from 1 to 7 tasks)

3 Proposed control algorithms

The considered MRTA problem (1) entails a discrete optimization with $N \cdot M$ variables, which can take $M+1$ discrete values from 0 to M. As a consequence, computation times will increase exponentially with the size of the problem making impossible to solve the problem exhaustively, even for relatively small problems. More specifically, the number of feasible discrete solutions can be computed with:

$$N_{\text{sol}} = (N \cdot M)^{M+1} \tag{2}$$

where it can be seen that with 3 robots and 3 tasks the number of feasible solutions is 6561, with 5 robots and 5 tasks, the number of feasible solutions goes up to 244,140,625, with 8 robots and 8 tasks, the number of feasible solutions reaches $1.8014 \cdot 10^{16}$. However, the number of solutions to be evaluated could be further bounded according to the soft constraint α_2 introduced in the previous section, i.e., focusing on solutions that perform all tasks once. Also, those solutions that represent the same assignments as one already considered (produced due to changes in the zeroes in U as can bee seen in Table 2) can be removed from the count.

To this end, it is possible to make use of combinatorial analysis to find a formula for the exact number of plausible (potentially optimal) solutions.

$$N_{\text{sol}} = \begin{cases} P_M \cdot \sum_{r=0}^{N-1} \text{PR}_{M-1}^{N-1-r,M-N+r} \cdot C_N^{N-r} & \text{if } N \leq M \\ P_M \cdot \sum_{r=0}^{M-1} \text{PR}_{M-1}^{M-1-r,r} \cdot C_N^{M-r} & \text{if } N > M \end{cases} \tag{3}$$

where $P_M = M!$ is the number of permutations without repetition of $\mathcal{S}$, $\text{PR}_{M-1}^{g-1,M-1-(g-1)} = \frac{(M-1)!}{(g-1)!((M-1)-(g-1))!}$ is the number of permutations with repetitions and it is used to compute the number of ways in which $\mathcal{S}$ can be divided into a given number of groups, g, and $C_N^g = \frac{N!}{g!(N-g)!}$ is the combinatorial number that counts the number of ways of creating groups of size g out of $\mathcal{R}$.

The number of solutions obtained for some problems with from 1 to 7 tasks and from 1 to 7 robots and the increasing trends with both the number of tasks and the number of robots can be better observed in Fig. 1.

As a consequence, in order to carry out the optimization, this work considers two different approaches: for small-sized problems, the optimal solution can be found by using a Branch and Bound (B&B) algorithm (see Sect. 3.2). However, for medium- and large-scale problems finding the optimal solution using B&B also becomes unfeasible. In this case, the use of a genetic algorithm (GA) is proposed (see Sect. 3.3).

Both algorithms (B&B and GA) make use of a set of initial solutions that provide an upper bound on the value of the optimal cost function. The computation of these initial solutions is detailed in the following Sect. 3.1.

3.1 Initial solutions

This section presents the set of suboptimal initial solutions that has been used in this work for the application of the B&B and GA algorithms. More specifically, the set of initial solutions for the problem presented in Sect. 2, is composed of the following subsets:

1. The N initial solutions resulting from solving the shortest path problem for every robot i:

$$\begin{aligned} \min_{U_i} \quad & d_i(U_i) \\ \text{s.t.} \quad & \varphi(U_i) = 0 \\ & u_i(n) \in \mathcal{S} \cup \{0\}\ \forall\, n \end{aligned} \tag{4}$$

where $U_i = [u_i(1), u_i(2), .., u_i(M)]$ and $\varphi(U_i) = 0$ is the set of constraints, ensuring that all the tasks are carried out once. Notice that this problem is the well-known traveling salesman problem (TSP). TSP is a NP-Hard problem that can be solved in an exact way using Held–Karp algorithm [42] in a time $\mathcal{O}(n^2 2^n)$. However, the nearest neighbor (NN) algorithm [43], a heuristic greedy algorithm consisting of choosing the nearest non-visited node in each step, is used to generate feasible solutions in a very reasonable time.
2. The N initial solutions resulting from solving the shortest time problem for every robot i. The distances considered in these problems are the ones resulting from adding the estimated traveling times between robots and tasks and the operation times required in the corresponding tasks:

$$\begin{aligned} \min_{U_i} \quad & t_i^*(U_i) \\ \text{s.t.} \quad & \varphi(U_i) = 0 \\ & u_i(n) \in \mathcal{S} \cup \{0\}\ \forall\, n \end{aligned} \tag{5}$$

where t_i^* is the working time of each robot considering both the time needed to reach a task and the time needed to perform the task (note that $t_i^*(U_i) = t_i(U_i)$ if in the allocation U_i only robot i is used), $U_i = [u_i(1), u_i(2), .., u_i(M)]$ and $\varphi(U_i) = 0$ is the set of constraints, ensuring that all the tasks are carried out once. This problem is addressed in a similar way to the shortest path problem. However, since the criteria are different, the solution of this problem is both feasible and different from the problem mentioned in (4).
3. The initial solution resulting from solving the assignment problem using only distances:

$$\begin{aligned} \min_{U(1)} \quad & \sum_{i=1}^{N} \lambda_i d_i(U) \\ \text{s.t.} \quad & \varsigma(U) = 0 \\ & u_i(1) \in \mathcal{S} \cup \{0\}\ \forall\, i \end{aligned} \tag{6}$$

where $U(1) = [u_1(1), u_2(1), .., u_N(1)]$ and $\varsigma(U_N) = 0$ is the set of constraints, ensuring that all the tasks are carried out (if $M <= N$) or that all the robots are assigned to one task (if $M > N$). In case there are more tasks than robots ($M > N$), the problem is iterated by considering the allocation in the subsequent nth orders of the robots. Each iteration takes into account the previous allocation of the robots, the remaining tasks and the distance among them, and it is solved using Kuhn–Munkres algorithm [44] in a time $\mathcal{O}(n^3)$.
4. The initial solution resulting from solving the assignment problem using only times:

$$\begin{aligned} \min_{U(1)} \quad & \sum_{j=1}^{M} \delta_j t_j(U) \\ \text{s.t.} \quad & \varsigma(U) = 0 \\ & u_i(1) \in \mathcal{S} \cup \{0\}\ \forall\, i \end{aligned} \tag{7}$$

Note that this problem can be hard to solve directly since the time in which tasks are performed will depend on the assignment in the previous iteration. Thus, in this set of initial solutions, we address the problem similarly to the distance assignment problem but considering the time of reaching the task and performing it instead of the distance.

All the considered initial solutions can be fast and easily computed by using well-known algorithms. It has to be pointed out that the used set of initial solutions is independent of the subsequently applied control algorithms (B&B and GA). Therefore, the set used can be adapted for each problem. The only necessary requirement is that the initial solutions have to be computed fast enough to allow subsequent computations within the B&B or GA algorithm. In order to have a good performance, it is also important that the initial solutions correspond to the different criteria of the cost function. For example, if a cost function minimizes time and distance simultaneously the algorithms should be fed with initial solutions reflecting an opposing behavior (i.e., minimizing only time or space).

3.2 Branch and bound algorithm

For small sized problems, the optimal solution for the optimization problem presented in Sect. 2 can be computed by using the Branch and Bound (B&B) algorithm [45]. This algorithm, used for discrete and combinatorial optimization, consists of developing the tree of possible solutions sequentially, discarding the partial solutions that have a cost function higher than a set boundary, until the optimal solution is found. B&B has been used in some works to address the MRTA problem, as in [46], where it is used in combination with a Monte Carlo search tree [47]. However, to the best of our knowledge, it has always been applied to the MRTA problem using binary variables for the task assignments while in this work it is applied using discrete variables (which allows to substantially decrease the number of leaves on the search tree).

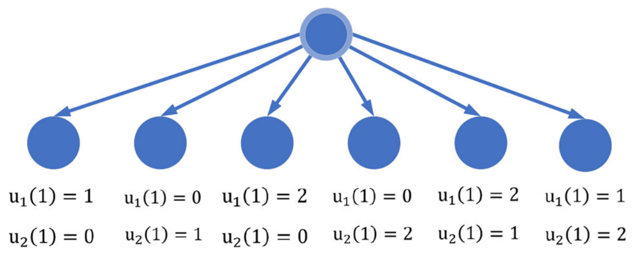

Fig. 2 Initial allocation Tree with 2 robots and 2 tasks. The "branch" is done by adding a single task to the task queue of each robot (in every possible way). The entire set of possible allocations for the firstly allocated tasks (composed of 6 leaves in this case) is generated

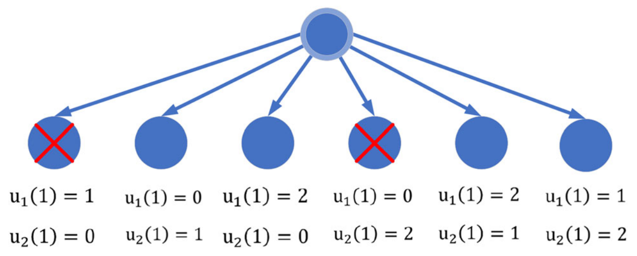

Fig. 3 Bound with J^*: Discards for the tree shown in Fig. 2 assuming that the partial cost function of the first and fourth leaves are higher than $J^*(U)$ and are therefore discarded

More specifically, in the present work, the use of the B&B algorithm is proposed for solving small-scale [less than 10^5 solutions according to (3)] MRTA problems formulated as in (1). The algorithm generates all the possible feasible combinations for each nth allocated task removing the ones that show a cost function higher than the current upper bound for the optimal cost function. For the remaining task allocations, the corresponding possible combinations for the following $(n+1)$th allocated tasks are generated comparing their new cost function with an upper bound. The algorithm continues until all the possible optimal task allocations (for any order) have been explored.

More specifically, the algorithm applied in this paper is composed of the following steps:

1. The best (lowest) cost function obtained within the set of previously computed initial solutions (see Sect. 3.1) is taken as the initial estimation of the B&B threshold (J^*).
2. Then, all the possible task allocations for the firstly allocated tasks for each robot are generated, i.e., the new "branch" is done by adding a single task to the task queue of each robot (in every possible way). A simple example with 2 robots and 2 tasks can be seen in Fig. 2.
3. The partial cost function of the generated allocations is computed. This function corresponds to the distances and times of the allocated tasks taking into account that many tasks can be unallocated to any robot at this step. The partial cost function used is similar to (1) but redefining $\gamma_2(U)$ in order to eliminate the penalization for unallocated tasks:

$$\begin{aligned} \min_U \; & J^*(U) = \sum_{j=1}^{M} \delta_j t_j(U) + \sum_{i=1}^{M} \lambda_i d_i(U) + \gamma(U) \\ & \gamma(U) = \alpha_1 \gamma_1(U) \\ \text{s.t.} \; & u_i(n) \in \mathcal{S} \cup \{0\} \; \forall \, i, n \end{aligned} \tag{8}$$

In contrast with the original cost function defined in (1), $\gamma_2(U)$ is not used since, in this case, also partial assignments which do not fulfill all the tasks are evaluated.

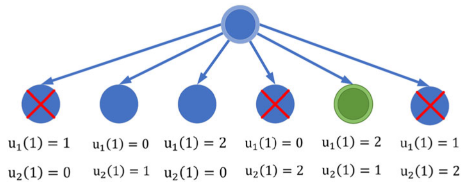

Fig. 4 Cost function updates for the tree shown in Fig. 3. It is assumed that the cost function of the allocation on the fifth leaf is lower than $J^*(U)$ and, as a consequence, the value of $J^*(U)$ is updated. Subsequently, the sixth leaf (with a cost function higher than the updated value of $J^*(U)$) is removed accordingly

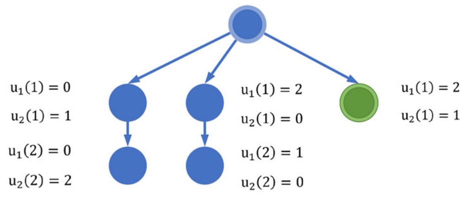

Fig. 5 Allocation tree resulting from considering the task assigned in the second place for the search tree in Fig. 4

Therefore, this partial cost function $J^*(U)$ will be lower than an original cost function $J(U)$ if some tasks are not allocated or equal to the original cost function if all the tasks are allocated ($J^*(U) \leq J(U)$).

4. The task allocations with a cost function higher than the B&B threshold are discarded as it can be seen in Fig. 3.
5. If any allocation is complete (i.e., all the tasks are allocated) and the corresponding cost function is lower than the current value of the B&B threshold, this value is updated (also discarding the allocations lower than the new cost function) as it can be seen in Fig. 4.
6. The allocations which have not been discarded are used to generate a new set of possible allocations including the nth allocated task for those solutions which have still tasks to allocate, as it can be seen in Fig. 5.
7. The algorithm ends if the remaining leaves cannot be increased any more (i.e., if all the tasks are allocated). Otherwise, it returns to step 3 (Fig. 6).

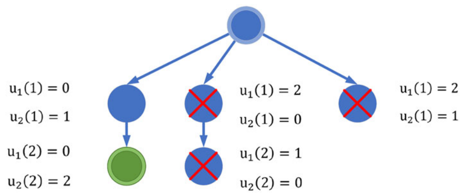

Fig. 6 Cost function update for the tree shown in Fig. 5. It is assumed that the cost function of the complete allocation on the first leaf is lower than the current value of $J^*(U)$ (the one corresponding to the last leaf) and, as a consequence, the second and third leaves (with a cost function higher than the updated value of $J^*(U)$) are also removed. Since only one candidate leaf remains ($U = [0, 1, 0, 2]$) and this leaf cannot be further increased, the remaining allocation is the optimal one

3.3 Genetic algorithm

The previously presented B&B algorithm computationally explodes for medium and large-sized problems. As a result, another faster but suboptimal algorithm has to be used for these cases.

The genetic algorithm (GA) is a meta-heuristic algorithm inspired by the process of natural selection and evolution which relies on crossover, mutations and selection [48]. The GA has been applied to similar MRTA problems in works such as [49], where it is used in combination with intra-path constraints in order to solve the allocation of a heterogeneous fleet of robots in a disaster scenario; [50], where a multi-objective cost function including both energy and time criteria is used; [51], where a multi-objective cost function is also used in a multi-robot task context including time constraints and discrete variables are used to model the problem; and [33,52], where similar structured environments with discrete positions of the robots and discrete variables are used.

In this work, we have designed a GA including the use of specially developed crossover and mutation strategies that generate ordered solutions for the next generations. These strategies generate new allocations that do not introduce unfeasibility from the viewpoint of tasks (allocations that do not fulfill all the tasks or that fulfill a task more than once cannot be fathered by allocations that do) and thus, reduce drastically the number of possible allocations and the computation load. With the same criterion, it has been taken into account that any allocation for a robot u_i including intermediate zeroes is equivalent to the same allocation removing these zeroes and adding them at the end. The crossover and mutation methods presented in this section are novel to the best of our knowledge.

An example of the individuals or chromosomes which are used in this paper can be seen in Table 3, where the nth allocated tasks for each robot R_i are concatenated. Note that each individual or chromosome corresponds to a full candidate solution to the MRTA problem (i.e., all the tasks assigned to all of the robots).

Notice that, even though the crossover and mutation strategies developed do not generate unfeasible allocations from the viewpoint of tasks, they can still produce unfeasible allocations from an energetic point of view.

The GA applied for the considered MRTA problem is composed of the following steps:

1. The set of feasible initial solutions presented in Sect. 2 is used to generate the initial population (i.e., the first individuals) of the GA. In the case that the population size is larger than the number of initial solutions previously computed, the rest of the population is filled with randomly generated potentially optimal solutions. To do this, we permute the set $\mathcal{S}$ randomly and then go through all the tasks, selecting a random robot from $\mathcal{R}$ and allocating the task to the robot. This way, the resulting individual will not activate γ_2 in (1).
2. The cost function is computed for the entire set of individuals of the current generation. The fittest individuals are selected as the elite population, which survive automatically into the next generation.
3. The algorithm generates a new set of individuals, which are distributed as follows:
   (a) The elite population fraction, $\mathcal{G}_{\text{E}}$, defines the percentage of the new population that is considered to be elite individuals.
   (b) The Crossover fraction, $\mathcal{G}_{\text{C}}$, defines the percentage of the non-elite new population which is composed of individuals generated by crossover. An example can be seen in Fig. 7. Each crossover strategy is composed of the following steps:
      i. Two elite individuals are randomly selected.
      ii. One task is randomly selected.
      iii. The selected task is removed for both individuals.
      iv. The task allocations are shifted in order to correct the individual obtained in the previous step.

Table 3 Gen example

Robot	First robot						···	ith robot						···	Nth robot					
Gen example	1	3	···	5	···	0	···	2	0	···	0	···	0	···	4	0	···	0	···	0
Order	O_1	O_2	···	O_n	···	O_M	···	O_1	O_2	···	O_n	···	O_M	···	O_1	O_2	···	O_n	···	O_M

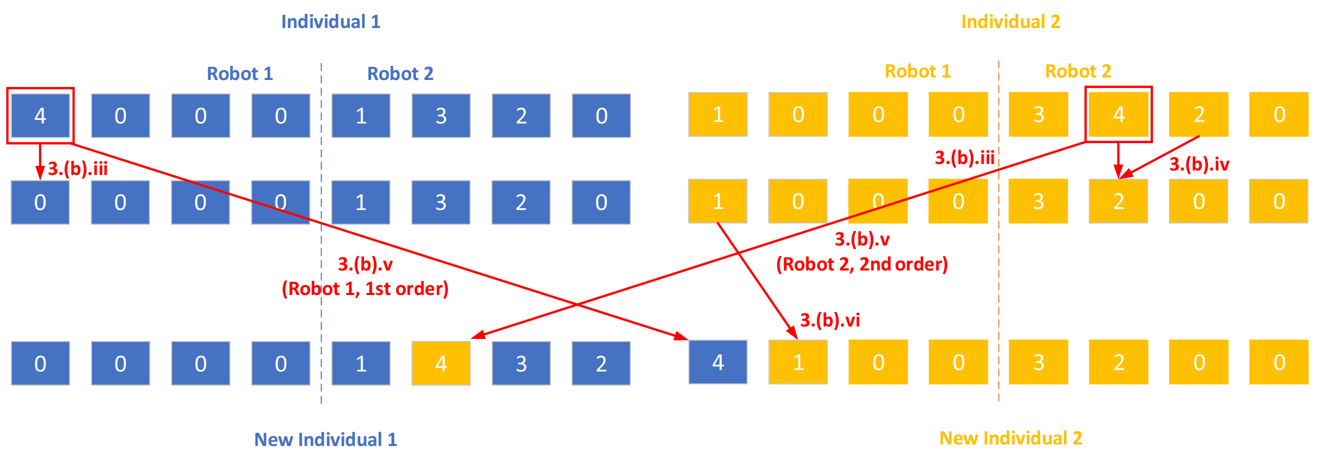

Fig. 7 Example of the crossover strategy with 2 robots and 4 tasks. The crossover is composed of three logical operations (elimination 3.(b).iii, shifting 3.(b).iv and 3.(b).vi, and recombination 3.(b).v)

v. The selected task is reassigned to each individual but using the allocation considered by the other individual (i.e., for the first individual, the selected task is now allocated for the position originally given by the second individual).

vi. The task allocations are shifted again in order to correct the individual obtained in the previous step.

(c) $\mathcal{G}_{\mathrm{M1}}$ is the fraction of individuals generated by mutations, which are generated by using the first kind of mutation proposed. An example can be seen in Fig. 8. Each mutation for the first strategy is composed of the following steps:

i. One elite individual is randomly selected.
ii. One task is randomly selected.
iii. The selected task is removed.
iv. The task allocation is shifted in order to correct the individual obtained in the previous step.
v. A new robot and position is randomly selected for the selected task.
vi. The selected task is included in the randomly selected position.
vii. The task allocation is shifted again in order to correct the individual obtained in the previous step.

(d) $\mathcal{G}_{\mathrm{M2}}$ is the fraction of individuals generated by mutations, which are generated using the second kind of mutation proposed. An example can be seen in Fig. 9. Note that $\mathcal{G}_{\mathrm{M2}} = 1 - \mathcal{G}_{\mathrm{M1}}$ by definition, since there are only 2 types of mutation. Each mutation for the second strategy is composed of the following steps:

i. One elite individual is randomly selected.
ii. Two tasks are randomly selected.
iii. The selected tasks are exchanged.

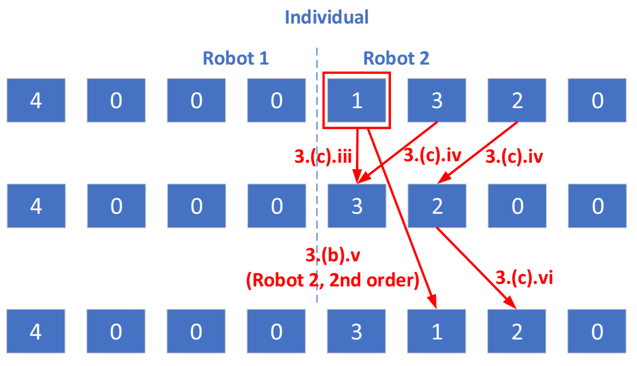

Fig. 8 Example of the first kind of mutation strategy with 2 robots and 4 tasks

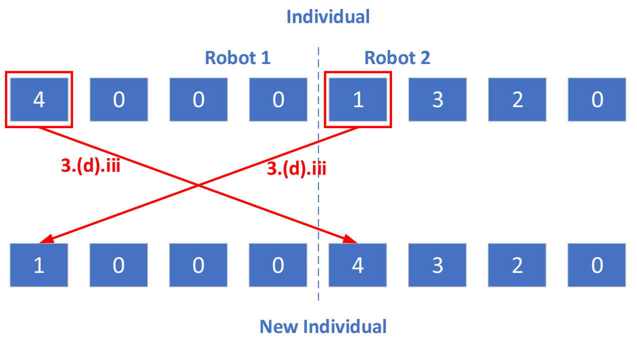

Fig. 9 Example of the second kind of mutation strategy with 2 robots and 4 tasks

(e) The algorithms end if the number of generations is equal to the maximum number of generations or if the solution has not been improved during a maximum number of stall generations. Otherwise, the algorithm goes back to Step (2).

Example 1 Let us take a case in which we have a population size of 100 individuals. To create the next generation, we will copy the 10 best individuals from the current generation. Then, considering $\mathcal{G}_C = 0.8$, 80% of the 90 remaining individuals, i.e., 72 individuals, must be created by crossover. To do this, we take random selections of elite individuals and do crossovers between them until we reach the 72 individuals we need. The remaining individuals $100 - 10 - 72 = 18$ are created by mutations. Assuming $\mathcal{G}_{M1} = 0.5$ and $\mathcal{G}_{M2} = 0.5$ half of them, 9, are going to be generated by selecting a random elite individual and applying mutation type 1 and the other half by doing the same but applying mutation type 2.

Note that, since we start from a potentially optimal initial population (the initial solutions plus the potentially optimal random solutions) and since the proposed genetic algorithm always maintains this property between generations, the middle zeroes in the gen do not affect the performance and speed of this approach as the algorithm will not produce not potentially optimal solutions.

4 Case study

In this section, the models used to simulate the robots are described. Likewise, the solar power plant layout considered is detailed. Finally, the problem formulated in Sect. 2 is particularized taking into account both the robot models and the plant layout.

4.1 Robot fleet

Since the fleet is comprised by both aerial and ground unmanned robots, there are different characteristics of speed and energy consumption or recharging rates for each type of robot. Likewise, different security energy levels have been considered for each type of robot. The parameters considered for UAVs and UGVs can be seen in Table 4. Note that V_{mean} is the mean speed of the robots in the XY plane and $V_{Z\text{mean}}$ is the landing/taking off speed. A different number of UGVs and UAVs is considered for the simulations in order to analyze the time required to solve problems of different sizes.

4.2 Solar plant layout

An area of 63 ha (1180×530 m) has been chosen as a model of a generic thermosolar power plant of approximately 30 MW, where several obstacles have been placed. It is assumed that no robot can move through the power plant for security reasons and, as a consequence, there is a big square forbidden area for all types of robots in the central upper part of the map corresponding to the steam generation, turbine and generator (see Fig. 10).

Table 4 Value of the parameters of the robots

Parameter	UGVs	UAVs
V_{mean}	1.5 m/s	10 m/s
$V_{Z\text{mean}}$	0 m/s	3 m/s
Discharge rate	0.005% per second	0.1% per second
Charge rate	0.0025% per second	0.03% per second
Security energy	10%	20%

Note that discharge and charge rates refer to operating and charging time, respectively

In the case of aerial robots, it is also assumed that they are neither allowed to fly over the collectors for identical reasons, so the only allowed paths are those which do not go through the small vertical rectangular obstacles which represent the collectors.

In the case of ground robots, manifolds are considered insurmountable and to overcome them they must exclusively go through the authorized crossing-points, which can be considered as bottlenecks for moving from one part of the plant to another.

This area has been meshed into a grid of 434 spots where measurements can be taken. This grid has been created by placing the measurement spots in the intersection of 35 columns and 13 rows. Columns have been placed in the space between collectors, while rows have been placed taking into account that there must be rows in the horizontal passing places and that the space between rows must be similar to the space between columns.

In Fig. 10 and in the rest of figures used in this paper, the red circles represent the measurement spots, the red squares represent the charge stations and the different forbidden areas are represented by:

- Black rectangles in the case they affect all the robots. This area is where the offices, parking lots, deposits, thermal energy storage (TES), and energy generation zone (including the steam generation, the turbine, and the electric generator) are located.
- Green rectangles in the case they affect only UAVs. This rectangles represent the solar collectors and are comprised of parabolic mirrors.
- Blue rectangles in the case they affect only UGVs. This rectangles represent the manifolds that distribute the HTF to the collectors and back to the steam generator.

Notice that, even though UAVs cannot fly over the collectors, their potential to move through the plant is bigger than in the case of UGVs, which need to go through a crossing pass that can act as a bottleneck to go from one sector to

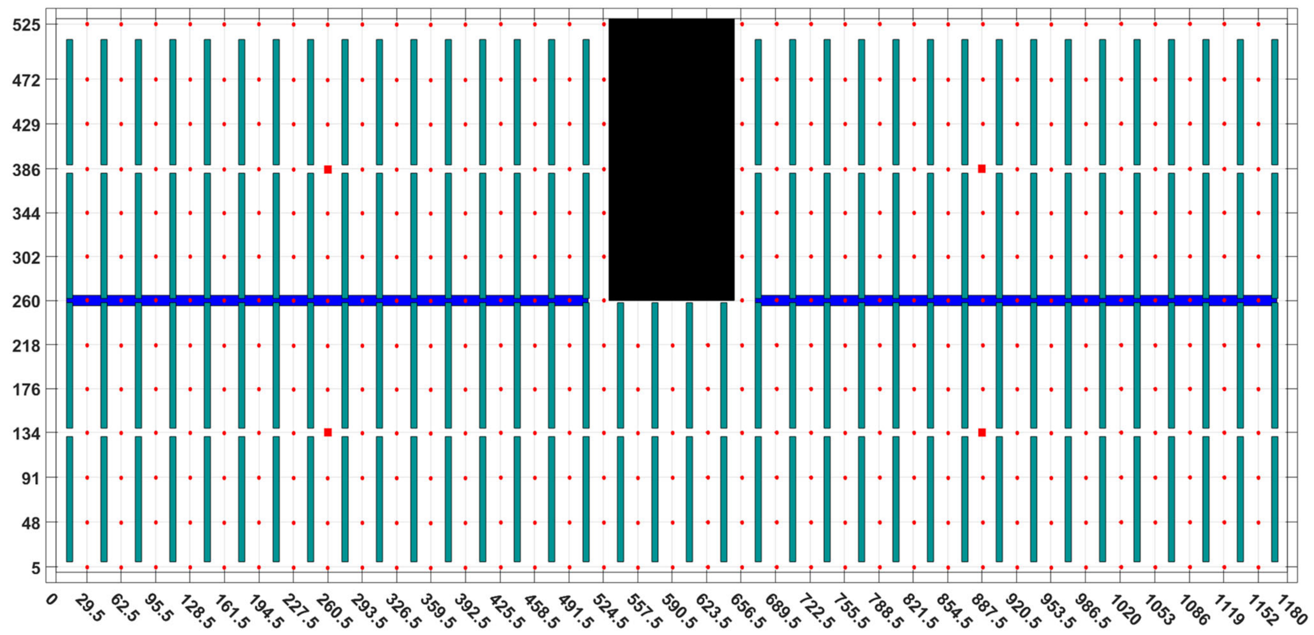

Fig. 10 Solar plant layout. The solar collectors are represented by vertical green rectangles. There is a grid of the 434 measurement spots (red dots). There are three zones or *sectors* splitted by the manifolds [(blue rectangles)] and the power plant main [zone (black rectangle)]: upper left sector, upper right sector and lower sector. There are also 4 charge stations (red squares) 1 in each upper sector and 2 in the lower sector (color figure online)

another. On the other hand, UGVs have a higher durability than UAVs, as they have a longer-lasting battery and consume less power.

4.3 Cost function

In this paper, the distances between each robot and each task and the distance between each task and the other tasks have been pre-calculated. This consideration can be done since the plant is a structured environment and the starting positions of the robots, the positions of the tasks and the possible paths are known. Thus, we only need to pre-compute these distances for the actual positions of the robots and the tasks at the moment of solving the MRTA problem.

Making use of these distances and the speed and discharging rate of the robots, it is possible to pre-calculate all the times and energy needed by a robot to go from its starting position to a task, or from one task to another. Besides, an initial delay time, t_i^o, is considered for each robot since a robot can be considered for an allocation even if it is currently charging of finishing a previous task. These initial delay times can be estimated knowing the speed, the state of the robot and the parameters of the allocated task (in case the robot is finishing a previous task); or the charging rate and the energy level (in case the robot is charging). These parameters can be consulted in Table 4.

Once these data are obtained and there is a proposed U, the process of obtaining the distance traveled by each robot, $d_i(U)$, is as simple as adding the distances that each robot i must do due to its allocated tasks, i.e., the distance from its starting position to the task allocated as its first task, the distance from its first allocated task to the second allocated task, etc. This can be mathematically expressed by:

$$d_i(U) = \sum_{k=1}^{M} \hat{d}_{ik}(U) \quad \forall i$$
$$\hat{d}_{ik}(U) = \begin{cases} D^{\mathrm{RT}}_{i,u_i(1)} & \text{if } k = 1 \\ D^{\mathrm{TT}}_{u_i(k-1),u_i(k)} & \text{if } k > 1 \end{cases} \quad \forall i \tag{9}$$

where $\hat{d}_{ik}(U)$ is the distance traveled by robot i as a result of the kth allocated task, D^{RT}_{ab} contains the distance between robot a starting position and task b, and D^{TT}_{ab} are the distances between task a and task b.

Similarly, the process of checking if the allocation is feasible from an energetic point of view, $\gamma_1(U) = 1$, is simple as well, since the initial energy of the robots, E_i^o, and the different needed energies have been pre-calculated. This can be mathematically expressed by:

$$E_i(U) = \sum_{k=1}^{M} \hat{E}_{ik}(U) \quad \forall i$$

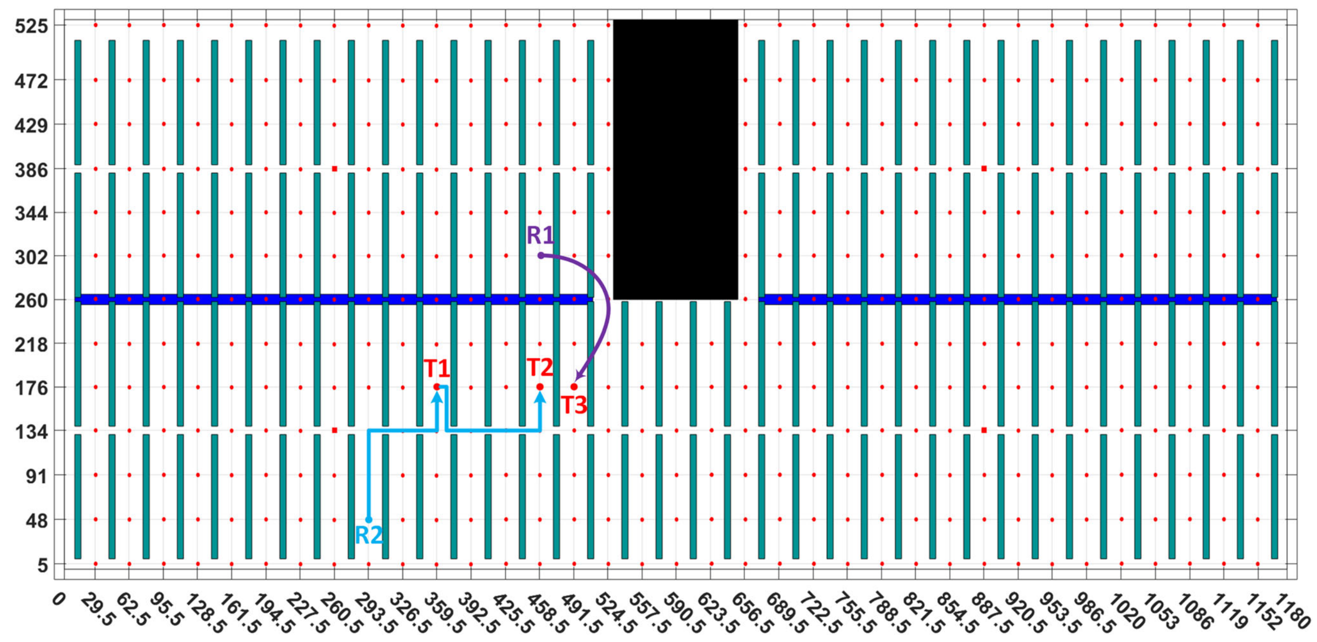

Fig. 11 Example scenario

$$\hat{E}_{ik}(U) = \begin{cases} E^{\mathrm{RT}}_{i,u_i(1)} & \text{if } k = 1 \\ E^{\mathrm{TT}}_{u_i(k-1),u_i(k),i} & \text{if } k > 1 \end{cases} \quad \forall i$$

$$\gamma_1(U) = \sum_{i=1}^{N} \delta_i(U)$$

$$\delta_i(U) = \begin{cases} 0 & \text{if } E_i^o - E_i(U) \geq E^{\mathrm{s}} \\ 1 & \text{if } E_i^o - E_i(U) < E^{\mathrm{s}} \end{cases} \quad \forall i \qquad (10)$$

where $E_i(U)$ is the estimated energy consumed by robot i after finishing all its allocated tasks, $\hat{E}_{ik}(U)$ is the estimated energy consumed by robot i as a result of the kth allocated task, E^{RT}_{ab} contains the estimated energy that robot a needs to move from its starting position to task b, E^{TT}_{abc} contains the estimated energy that robot c needs to move between task a and task b, and E^{s} is the security energy level, i.e., the minimum energy needed to reach a charging station.

However, the process of obtaining the delay time of each task, $t_j(U)$, is a little bit different, not only is the time needed to reach the position of the task relevant but also the accumulated time of the robot which is performing task j. The operating time of each robot is defined and set to the initial delay calculated previously, i.e., $t_i(U) = t_i^o \; \forall i$. Once the initial values are set, we must go through each robot queue of allocated tasks updating $t_i(U)$ by adding the necessary time to reach the following task and the operation time of this task, t_j^{op}. Each time a task j is considered the delay time of task j is set to the current value of the operating time of robot i, i.e., $t_j(U) = t_i(U_j^*)$. This can be mathematically expressed by:

$$t_j(U) = \sum_{k=1}^{N_j+1} \hat{t}_{R_jk}(U) + \sum_{k=1}^{N_j+1} t^{\mathrm{op}}_{u_{R_j}(k)} \quad \forall j$$

$$\hat{t}_{R_jk}(U) = \begin{cases} T^{\mathrm{RT}}_{R_j,u_{R_j}(1)} & \text{if } k = 1 \\ T^{\mathrm{TT}}_{u_{R_j}(k-1),u_{R_j i}(k),R_j} & \text{if } k > 1 \end{cases} \quad \forall j \qquad (11)$$

where R_j represents the robot to which task j has been allocated, N_j is the number of tasks allocated to robot R_j before task j, $\hat{t}_{R_jk}(u_{R_j})$ is the traveling time of robot R_j as a result of the k^{th} allocated task; T^{RT}_{ab} contains the time that robot a needs to get from its starting position to task b, and T^{TT}_{abc} contains the time that robot c needs to go from task a to task b.

Notice that, even though the amount of energy needed to fulfill the task has been neglected, adding it would not modify substantially the proposed approach.

Example 2 In Fig. 11, we can see an example with 2 robots (an UGV and an UAV) and 3 tasks. The allocation consists of robot 2 performing sequentially tasks 2 and 3, and robot 1 performing task 3, i.e., $U^* = \begin{bmatrix} 3 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}$. In this case, distance traveled by robot 2 is $d_1(U^*) = D^{\mathrm{RT}}_{1,3}$ and distance traveled by robot 2 is $d_2(U^*) = D^{\mathrm{RT}}_{2,1} + D^{\mathrm{TT}}_{1,2}$. Analogously, the corresponding consumed energies are $E_1(U^*) = E^{\mathrm{RT}}_{1,3}$ and $E_2(U^*) = E^{\mathrm{RT}}_{2,1} + E^{\mathrm{TT}}_{1,2}$. On the other hand, the time required to complete the tasks in this allocation are $t_1(U^*) = T^{\mathrm{RT}}_{2,1} + t_1^{\mathrm{op}}$, $t_2(U^*) = t_1(U^*) + T^{\mathrm{TT}}_{1,2,2} + t_2^{\mathrm{op}}$ and $t_3(U^*) = T^{\mathrm{RT}}_{1,3} + t_3^{\mathrm{op}}$. Thus, since all the tasks are allocated and none of the tasks is allocated more than once ($\gamma_2(U^*) = 0$) and assuming $E_1(U^*) \leq E_1^o - E^s$ and $E_2(U^*) \leq E_2^o - E^s$ ($\gamma_1(U^*) = 0$), the cost function will be: $J = \delta_1 \cdot (T_{2,1}^{\mathrm{RT}} + t_1^{\mathrm{op}}) + \delta_2 \cdot (T_{2,1}^{\mathrm{RT}} + t_1^{\mathrm{op}} + T_{1,2,2}^{\mathrm{TT}} + t_2^{\mathrm{op}}) + \delta_3 \cdot (T_{1,3}^{\mathrm{RT}} + t_3^{\mathrm{op}}) + \lambda_1 \cdot (D_{1,3}^{\mathrm{RT}}) + \lambda_2 \cdot (D_{1,2}^{\mathrm{RT}} + D_{1,2}^{\mathrm{TT}})$.

4.4 Scenarios

To test both, the GA and B&B approaches, a set of different sized scenarios are simulated:

- Scenario I (S-I): in this scenario, there are 4 robots (2 UGVs and 2 UAVs) and a set of 6 objective spots (tasks), which according to (3) correspond to 60,480 possible solutions.
- Scenario II (S-II): in this scenario, there are 6 robots (3 UGVs and 3 UAVs) and a set of 8 objective spots (tasks), which according to (3) correspond to 51,891,840 possible solutions.
- Scenario III (S-III): in this scenario, there are 10 robots (6 UGVs and 4 UAVs) and 9 objective spots (tasks), which according to (3) correspond to $1.7643 \cdot 10^{10}$ possible solutions. In this scenario, it has been considered that all the robots are concentrated in an area of the plant and all tasks are concentrated in the opposite area of the plant. .
- Scenario IV (S-IV): in this scenario, there are 5 robots, 2 UGVs and 3 UAVs, and 15 objective spots (tasks), which according to (3) correspond to $5.0685 \cdot 10^{15}$ possible solutions. This scenario models a common situation where there are several tasks per robot.

These scenarios are determined by the amount of robots and their types, locations, starting energy, delay on being prepared in the starting point, and penalty for using each one; and the amount of tasks, their locations and the operating time that it takes to perform them. The parameters that define these scenarios can be seen in Table 5 for robots and in Table 6 for Tasks. Note that in Table 5, G stands for ground robots and A for Aerial robots.

4.5 GA tuning

The GA has been tuned with the parameters shown in Table 7. These parameters have been chosen heuristically. Given the heuristic nature of the GA, it has been run 10 times in each of the scenarios described in the previous subsection.

Table 5 Robot parameters in the four scenarios

Parameter	Type of robot				Starting spot (coordinate x, coordinate y)				λ_i				E_i^o (%)				t_i^o (s)			
Scenario	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV
Robot$_1$	G	G	G	A	(227.5, 386)	(161.5, 429)	(326.5, 176)	(1052.5, 48)	2	2	2	1	98	84	32	18	15	4	10	9
Robot$_2$	G	G	G	G	(260.5, 176)	(821.5, 48)	(182.5, 472)	(986.5, 386)	2	2	1	1	94	44	92	8	10	0	2	10.5
Robot$_3$	A	G	A	A	(293.5, 48)	(1100, 412)	(62.5, 344)	(953.5, 218)	1	1	2	1	30	76	26	53	1	2	3	11
Robot$_4$	A	A	A	A	(1151.5, 48)	(755.5, 344)	(29.5, 302)	(359.5, 5)	1	1	1	1	50	40	28	61	0	4	15	5
Robot$_5$	–	A	G	G	–	(260.5, 525)	(392.5, 472)	(1151.5, 134)	–	1	2	1	–	25	45	76	–	7	2	8.5
Robot$_6$	–	A	G	–	–	(953.5, 472)	(227.5, 302)	–	–	1	1	–	–	88	32	–	–	0	2.5	–
Robot$_7$	–	–	A	–	–	–	(491.5, 525)	–	–	–	1	–	–	–	25	–	–	–	5	–
Robot$_8$	–	–	A	–	–	–	(161.5, 386)	–	–	–	2	–	–	–	26	–	–	–	0	–
Robot$_9$	–	–	G	–	–	–	(29.5, 429)	–	–	–	2	–	–	–	12	–	–	–	0	–
Robot$_{10}$	–	–	G	–	–	–	(194.5, 472)	–	–	–	1	–	–	–	77	–	–	–	8	–

Table 6 Tasks parameters in the four scenarios

Parameter	Objective spot (coordinate x, coordinate y)				δ_j				t_j^{op} (s)			
Scenario	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV	S-I	S-II	S-III	S-IV
Task$_1$	(392.5, 302)	(227.5, 176)	(821.5, 218)	(161.5, 218)	3	2	5	8	10	40	0	30
Task$_2$	(458.5, 302)	(689.5, 472)	(1085.5, 134)	(29.5, 176)	1	3	2	6	10	0	0	23
Task$_3$	(1151.5, 472)	(491.5, 134)	(1019.5, 48)	(524.5, 344)	5	5	2	3	30	20	40	69
Task$_4$	(623.5, 176)	(623.5, 5)	(1151.5, 48)	(788.5, 5)	1	4	4	1	20	40	60	48
Task$_5$	(788.5, 344)	(1085.5, 344)	(986.5, 218)	(161.5, 91)	2	2	5	7	20	70	20	93
Task$_6$	(854.5, 48)	(359.5, 48)	(920.5, 5)	(425.5, 134)	1	2	4	2	10	80	25	109
Task$_7$	–	(62.5, 429)	(1052.5, 91)	(491.5, 91)	–	10	2	8	–	30	30	57
Task$_8$	–	(656.5, 472)	(1151.5, 5)	(788.5, 429)	–	2	1	6	–	50	10	21
Task$_9$	–	–	(1118.5, 218)	(161.5, 302)	–	–	4	1	–	–	20	107
Task$_{10}$	–	–	–	(887.5, 302)	–	–	–	6	–	–	–	68
Task$_{11}$	–	–	–	(326.5, 48)	–	–	–	3	–	–	–	21
Task$_{12}$	–	–	–	(590.5, 5)	–	–	–	3	–	–	–	23
Task$_{13}$	–	–	–	(920.5, 344)	–	–	–	6	–	–	–	38
Task$_{14}$	–	–	–	(1052.5, 386)	–	–	–	3	–	–	–	11
Task$_{15}$	–	–	–	(260.5, 48)	–	–	–	1	–	–	–	109

Table 7 GA tunning parameters

Pop. size	$\mathcal{G}_{\text{E}}$	$\mathcal{G}_{\text{C}}$	$\mathcal{G}_{\text{M1}}$	$\mathcal{G}_{\text{M2}}$	Max Gen.	Max Stall Gen.
100	0.1	0.7	0.5	0.5	50	30

5 Results

This section shows and discusses the simulation results on the application of the proposed algorithms for the proposed scenarios.

5.1 Scenario I

In this case, the best allocation obtained from solving Scenario I using the GA, which is also the most frequently obtained allocation among the GA repetitions, is the optimal allocation, obtained with the B&B algorithm and that can be seen in Fig. 12. The optimal allocation implies that robot 1 and 2 stay still, robot 3 performs task 1 and task 2 sequentially, and robot 4 performs the rest of the tasks in the following order: 3, 5, 6 and 4.

The best initial solution obtained for Scenario I, which corresponds to the assignment problem for distances [see Eq. (6)], is shown in Fig. 13. As it can be easily seen in the figure, there are some similarities and differences with the achieved allocation after applying GA. The most important difference is that, after applying the GA, the UGVs are not allocated to any task and the tasks that were previously performed by UGVs are now distributed to the UAVs.

Moreover, in those cases in which the optimal allocation cannot be achieved by combining the initial solutions, the GA makes use of the mutation strategy to jump out of the local minima of the cost function.

5.2 Scenario II

As for Scenario I, the allocation obtained from solving Scenario II by using the B&B algorithm (i.e., the optimal allocation) coincides with the best allocation obtained using the GA. The allocation is shown in Fig. 12.

However, in contrast to the previous scenario, the most repeated allocation among the repetitions is not the optimal one, reaching a value of cost function close but slightly larger to the optimal one (Fig. 14).

5.3 Scenario III

The B&B algorithm cannot be applied to Scenario III because of the combinatorial explosion due to the size of the problem. As a consequence, it is not possible to know if the best allocation provided by the GA is the optimal one. The best allocation obtained from solving Scenario III by using the GA is shown in Fig. 15.

As it can be seen in Fig. 15, in this scenario robots are concentrated in a sector of the plant while tasks are located in another sector. As a consequence, only two aerial robots are assigned to the entire set of tasks, implying that the obtained solution highly differs from the ones obtained for the set of initial solutions.

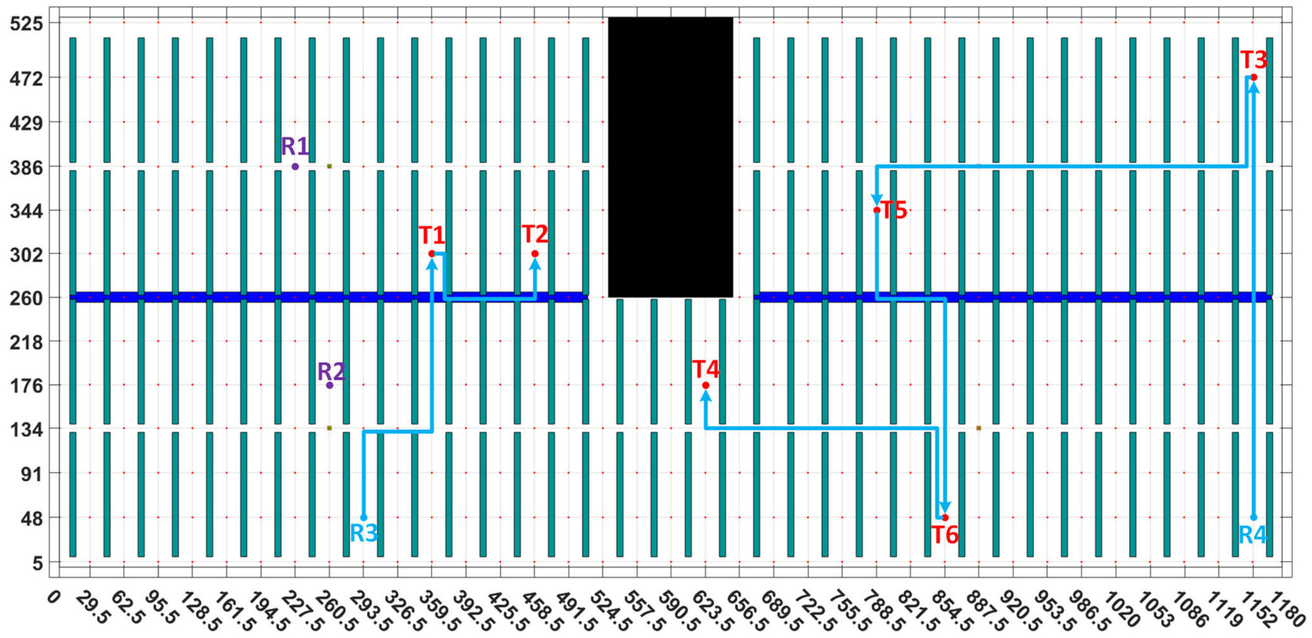

Fig. 12 Optimal allocation for Scenario I. Note that in the image the ground robots appear in purple while the aerial robots appear in blue (color figure online)

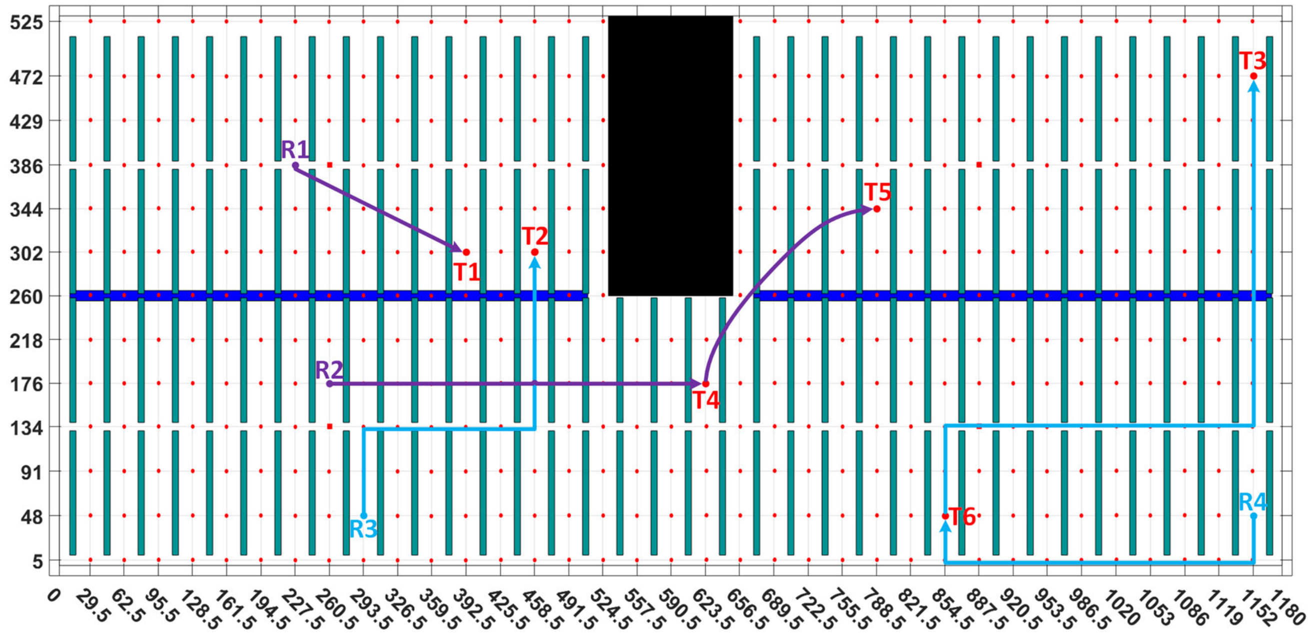

Fig. 13 Allocation using assignment problem using distances with all robots

5.4 Scenario IV

As in the previous scenario, the optimal solution cannot be found due to the size of the problem. The best allocation obtained from solving Scenario III by using the GA is shown in Fig. 16).

In this allocation, most tasks are performed by UAVs, while UGVs remain as backup robots performing only those tasks that are near their initial locations (14 and 4). Note that in the allocation obtained the route followed by robot 4 is quite strange, leaving task 9 for the end and attending tasks 2 and 5 before. However, it must be taken into account that the allocation obtained is not the optimal one (since the GA is a metaheuristic algorithm) and that, in this case, this allocation makes sense since there are other parameters to be considered in addition to the distance. As we can seen in Table 6,

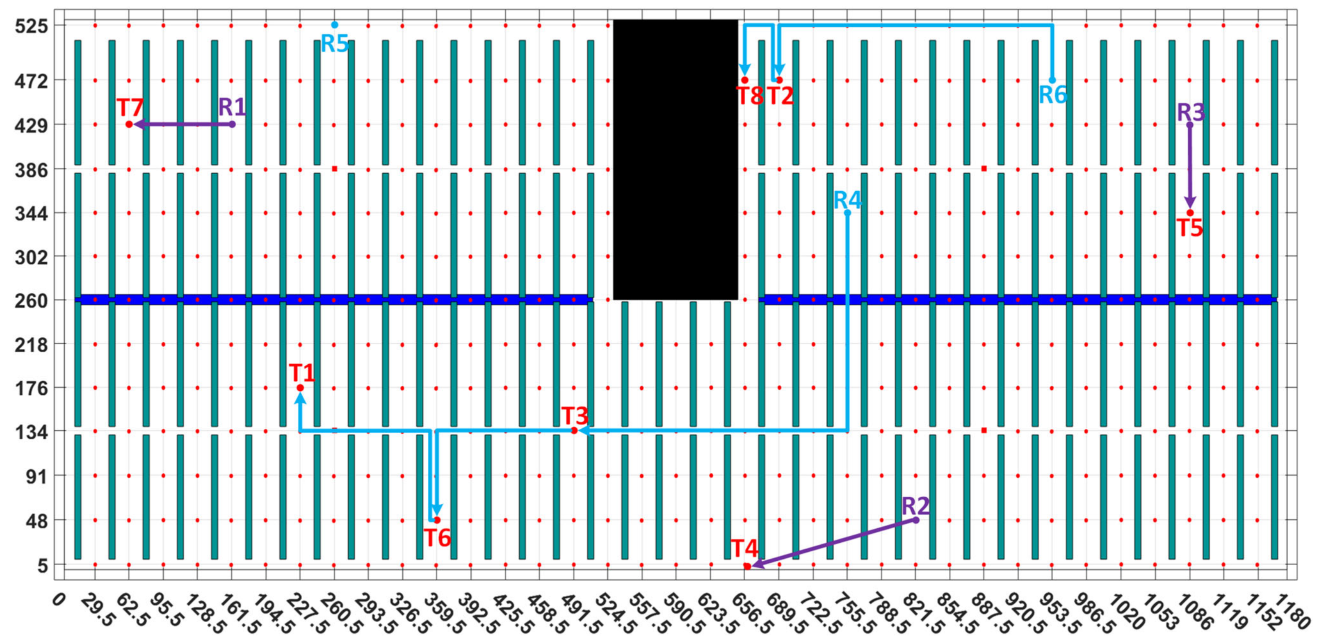

Fig. 14 Allocation with B&B and best allocation with Genetic algorithm in Scenario II

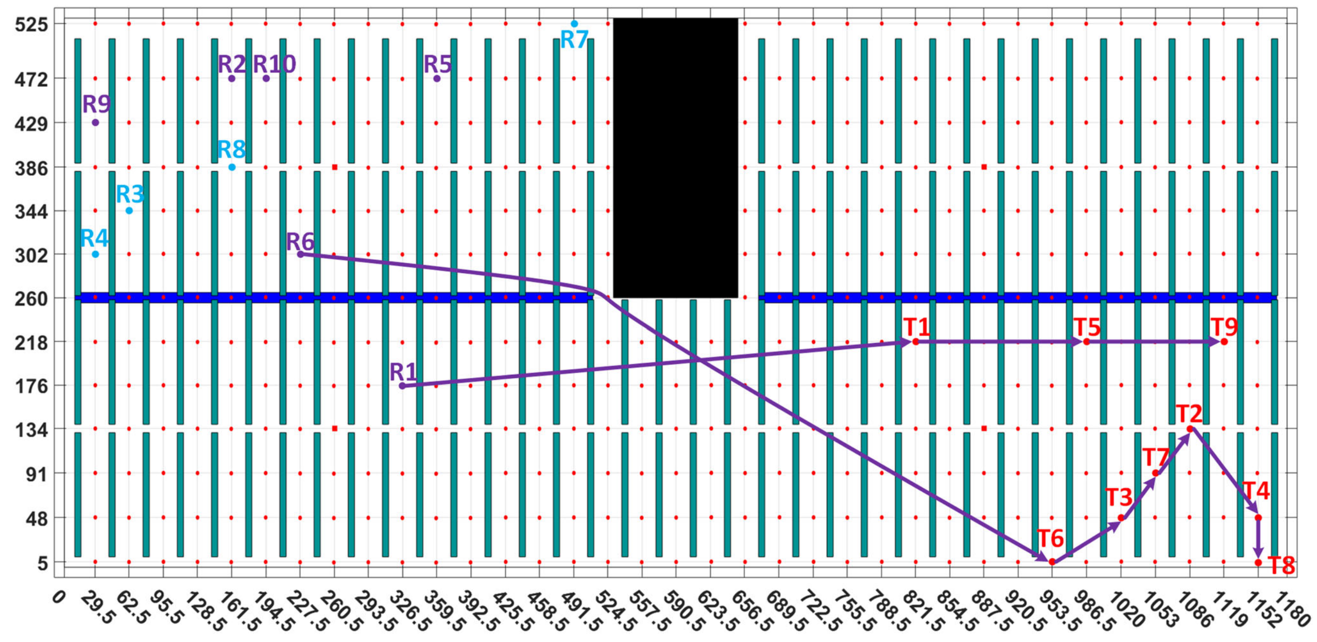

Fig. 15 Allocation with Genetic algorithm in Scenario III

task 9 has low importance ($\delta_9 = 1$) and the time required to fulfill it is long ($t_9^{\mathrm{op}} = 107$ s). Thus, performing it first would delay significantly other tasks allocated to robot 4.

5.5 Performance comparison

In Table 8, all the values of J can be compared. In the first column, there are the values of the best solutions among the initial solutions. In the second and third column, there are the best J values and the mean J values among all the repetitions, of the solutions obtained using the GA. In the last column, there are the values of the solutions obtained using B&B and therefore the optimal values for each scenario.

It can be seen that, in those scenarios where the optimal value using B&B could be obtained, this value is achieved by the GA most times, getting very near when it is not. It can also be checked that, even though B&B ensures an optimal

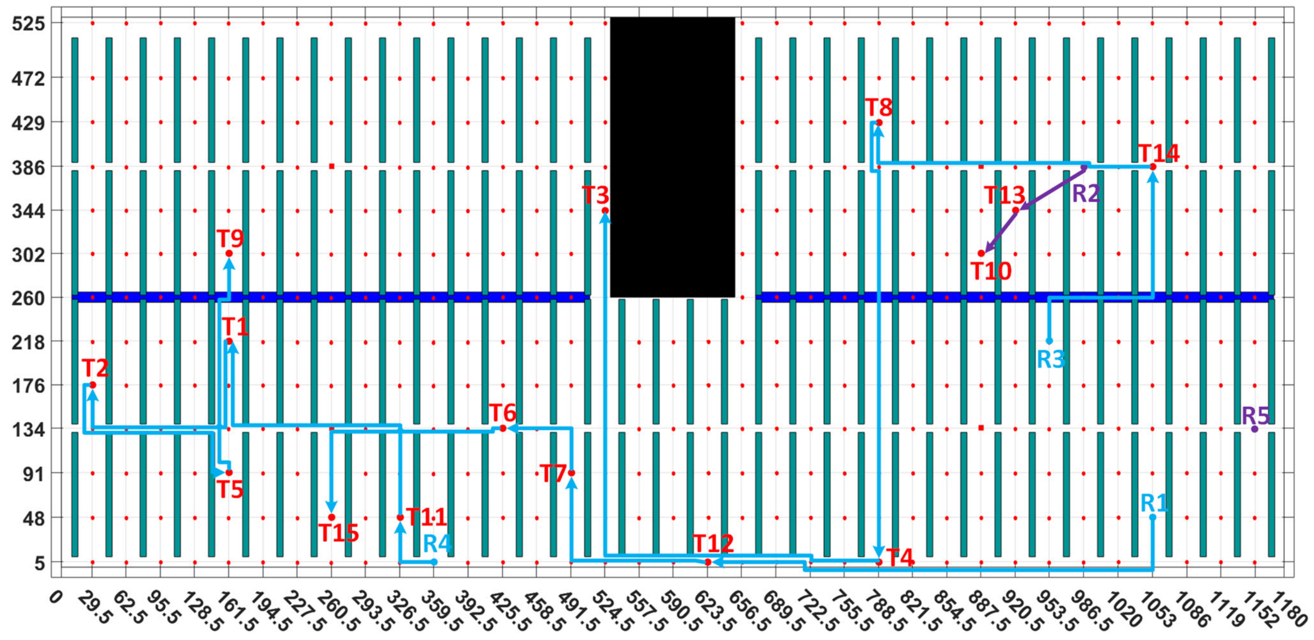

Fig. 16 Allocation with Genetic algorithm in Scenario IV

Table 8 Cost function for the best initial solution (J^{ini}), best ($J^{\text{GA}}_{\text{best}}$) and mean ($J^{\text{GA}}_{\text{mean}}$) cost functions obtained using the GA and optimal cost function obtained using B&B (J^{opt})

	Initial solutions	GA		B&B
J	J^{ini}	$J^{\text{GA}}_{\text{best}}$	$J^{\text{GA}}_{\text{mean}}$	J^{opt}
Scenario I	5490.81	3285.08	3366.02	3285.08
Scenario II	12488.67	5182.70	5457.36	5182.70
Scenario III	23823.12	19505.29	20141.02	–
Scenario IV	59952.30	53390.35	53762.06	–

Notice that the values of J for the GA correspond to the minimum and the mean value of applying the GA 10 times, so even though the minimum value coincides with the optimal value, this does not mean that the GA always reaches it (since the mean value is over the optimal one)

solution, it is only computationally feasible in small-sized problems.

As an example of the performance of the GA, the evolution of the best J values and the mean J values among the repetitions in a single run of the GA can be observed in Fig. 17.

5.6 Computational costs

The different computational costs using an Intel(R) Core(TM) i7-8700 CPU with 3.20 GHz processor can be compared in Fig. 18 for the mean times required for the GA. As it can be seen, the time required for solving scenario III is 2.27 s. Given that the mean time required for solving the cost function on this scenario is $3.2 \cdot 10^{-4}$ s in the time it takes to solve the scenario using the GA, the cost function could be solved around 7000 times, which contrasts with the $1.5579 \cdot 10^{13}$ possible solutions that there are in this scenario.

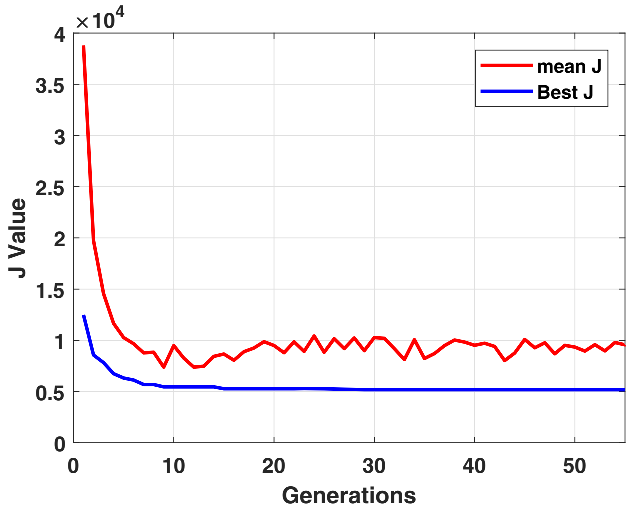

Fig. 17 Genetic evolution in Scenario II

The computation time required for solving the set of initial solutions is negligible since it is below 10^{-2} s for the four scenarios considered. In the B&B algorithm, the lower the bound is, the faster the algorithm is, and so, the computation time varies with the bound obtained from the set of initial solutions. However, only the lower bound has been considered with a computation time of 5.8 s for the first scenario and 206,040 s for the second one, i.e., 2.38 days. As previously stated, solving the third and fourth scenarios using B&B is not practicable due to the exponential increase in the computation times.

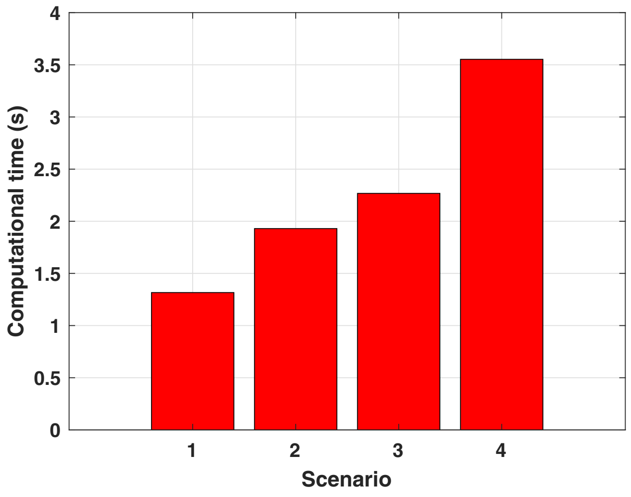

Fig. 18 Total computation times in each Scenario

It is also important to remark that, even though in small scenarios with few robots and tasks the B&B algorithm takes a similar time to the genetic algorithm and reaches an optimum solution, the computational cost increases exponentially with the size of the problem as it corresponds to a NP-Hard problem.

5.7 Monte Carlo studio

To validate the efficiency of our method, we have generated 20 different random scenarios for each problem size from 1 to 8 robots and from 4 to 8 tasks, i.e., 640 scenarios. Each scenario has been solved 50 times in the case of the GA from where we are taking the mean and the minimum value of J among the iterations. The J values obtained with both methods, GA and B&B, are shown in Fig. 19, where it can be seen that in many cases the optimal allocation is among the initial solutions. Also, it can be seen that in a few cases the mean GA is nearer the initial solutions than the optimal allocation, as it corresponds to a metaheuristical method.

Notice that B&B was only computed for cases where the number of solutions would not exceed 10^5 and that the GA tuning parameters, including the population size, were maintained constant and equal to those in Table 7.

The mean improvement of the GA over the initial solutions is 9.48% when considering the mean value among the iterations and it ascends to 12.61% when considering the minimum value among the iterations. Also, we can define *optimality* by how close the obtained J is to the one obtained in the B&B (in the cases in which it has been computed) in reference to the one of the initial solutions (disregarding all the cases in which the optimal solution is among the initial ones). Under this definition, the mean *optimality* considering the mean value among the iterations is 93.17%, while considering the minimum value among the iterations it increases up to 93.81%.

6 Conclusions

In this work, the MRTA problem has been formulated in the context of a thermosolar power plant where a MRS formed by various types of robots (ground and aerial) is in charge of collecting the DNI data. Under this approach, we assume a list of the tasks to be performed along with the time it takes to complete them and an associated priority or urgency. Besides, we also prioritize the use of a certain type of robot over the other or what is the same, to minimize the weighted distance traveled by the robots. Thus, we proposed a multi-criteria cost function that evaluates allocations taking into account the time needed to perform each task and the distance traveled by robots. However, the algorithms proposed in this paper do not vary when considering other criteria.

In this cost function, the constraints associated with energetic feasibility and to the completion of the entire set of tasks only once have been included as soft constraints. In order to solve this problem, discrete variables that code the allocation in a compact way have been used, which, combined with the B&B algorithm or the GA designed for the formulation in this work, allow to accelerate the computation of a feasible (if possible) and optimal (or quasi-optimal) allocation.

The B&B algorithm has proved to be able to obtain the optimal allocation. Moreover, it can be computed faster than the genetic algorithm if the size of the problem does not exceed a certain limit. However, as expected, the calculation time in this method increases with the size of the problem addressed, getting to a point where the B&B algorithm cannot be computed in a reasonable time. For medium- and large-scale problems, the proposed GA algorithm allows to obtain a suboptimal solution in a shorter period of time.

The proposed algorithm has been tested considering a fleet composed of UGVs and UAVs and a 63 ha (1180 × 530 m), 30 MW thermosolar plant with a known structured layout, in 4 different sized scenarios (varying the number of robots of each type, the number of tasks and the starting parameters).

The results obtained show that the optimal solution can be computed using a B&B algorithm in a short time (less than 10 s) for problems with between 4 and 5 robots and 5 or 6 tasks, i.e., problems with around 10^6 possible allocations according to Eq. (2). For larger sizes, the results show that the GA provides a good solution that is quite close to the optimal one (in Scenario II the GA achieves an allocation with cost 5454.36 while the optimal allocation cost is 5182.70).

The simulation results also show that the proposed set of initial solutions used as initial population improves the speed of the GA method at the same time that it provides a floor for it, and it also allows to obtain an initial bound of the cost function to be used by the B&B algorithm. The observed increase in the computation speed becomes more remarkable as the problem size increases.

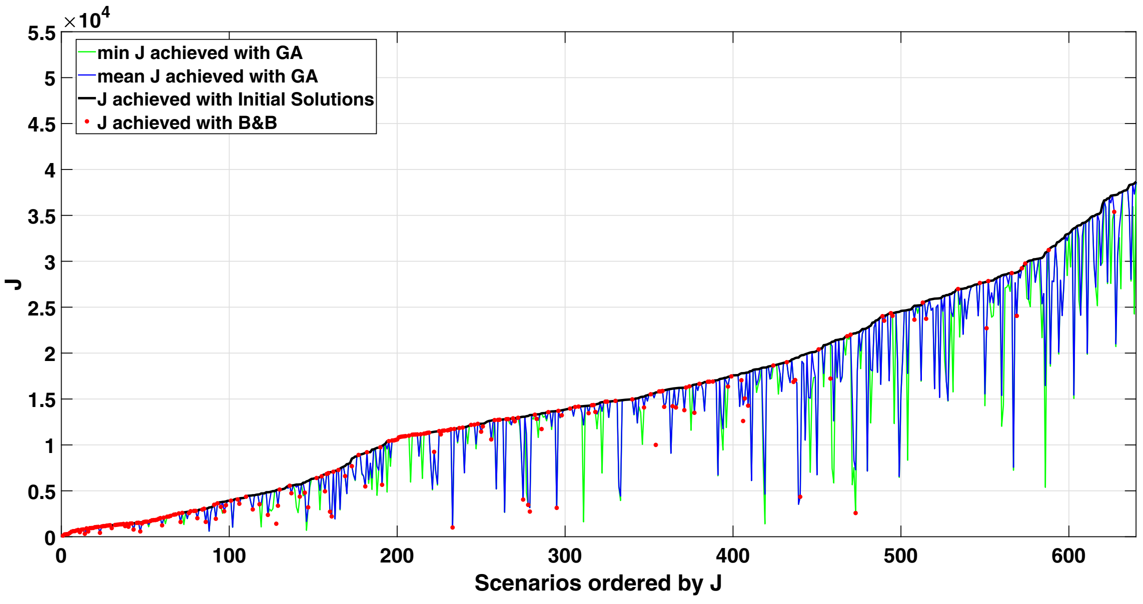

Fig. 19 In the figure, the different random scenarios have been ordered by the value of J obtained from the initial solutions (note that, due to the different parameters considered in the problem, there can be scenarios with similar sizes but very different values of J and vice versa). We can see the mean J value and the minimum J value obtained among the 50 GA iterations, respectively, in dark blue and in green. As expected, the mean value is contained between the minimum and the best J obtained by solving the initial solutions (black). The J obtained using B&B is represented by red dots and is always the minimum value (color figure online)

Future research lines for this approach might include improving the GA developed, making use of a wider part of the genetic pool to create the descendant individuals and trying different strategies as *Roulette Wheel*, testing it with real robots in the context of a distributed estimation of the radiation, the use of clusterings to split large groups of robots and tasks into smaller, affordable ones which can be solved by using the B&B algorithm and the testing of other meta-heuristic algorithms such as the Cuckoo Search with this formulation.

Acknowledgements This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Advanced Grant OCONTSOLAR, Grant agreement No 789051)". The authors also gratefully acknowledge the financial support by the European Research Council [Advanced Grant OCONTSOLAR number 789051] and the Spanish Ministry of Science, Innovation, and Universities [Grant number IJC2018-035395-I].

Author contributions J. G. Martin: Idea, code, writing of the paper, review and submission. J. R. D. Frejo: Idea, code, writing of the paper, review. R. A. Garcia: Idea, code, writing of the paper, review. E. F. Camacho: Idea, writing of the paper, review.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No 789051)".

Code availability Code is available by request to the first author.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Consent to Participate J. G. Martin, J. R. D. Frejo, R. A. Garcia and E. F. Camacho consent to be part of the developed work in this paper.

Consent to Publish J. G. Martin, J. R. D. Frejo, R. A. Garcia and E. F. Camacho consent the journal to publish this work.

References

1. Jin M, Lee J, Tsagarakis NG (2016) Model-free robust adaptive control of humanoid robots with flexible joints. IEEE Trans Ind Electron 64(2):1706–1715

2. Patidar V, Tiwari R (2016) Survey of robotic arm and parameters. In: 2016 International conference on computer communication and informatics (ICCCI)
3. Quaglia G, Visconte C, Scimmi LS, Melchiorre M, Cavallone P, Pastorelli S (2020) Design of a UGV powered by solar energy for precision agriculture. Robotics 9(1):13
4. Tokekar P, Hook JV, Mulla D, Isler V (2016) Sensor planning for a symbiotic UAV and UGV system for precision agriculture. IEEE Trans Rob 32(6):1498–1511
5. Maini P, Sujit PB (2015) On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications. In: 2015 International conference on unmanned aircraft systems (ICUAS). IEEE, pp 1370–1377
6. Nex F, Remondino F (2014) UAV for 3D mapping applications: a review. Appl Geomat 6(1):1–15
7. Puri A (2005) A survey of unmanned aerial vehicles (UAV) for traffic surveillance. University of South Florida, Department of Computer Science and Engineering, pp 1–29
8. Semsch E, Jakob M, Pavlicek D, Pechoucek M (2009) Autonomous UAV surveillance in complex urban environments. In: 2009 IEEE/WIC/ACM International joint conference on web intelligence and intelligent agent technology, vol 2. IEEE, pp 82–85
9. Ropero F, Muñoz P, R-Moreno MD (2019) TERRA: a path planning algorithm for cooperative UGV-UAV exploration. Eng Appl Artif Intell 78:260–272
10. Cortés J, Egerstedt M (2017) Coordinated control of multi-robot systems: a survey. SICE J Control Meas Syst Integr 10(6):495–503
11. Rizk Y, Awad M, Tunstel EW (2019) Cooperative heterogeneous multi-robot systems: a survey. ACM Comput Surv (CSUR) 52(2):1–31
12. Khamis A, Hussein A, Elmogy A (2015) Multi-robot task allocation: a review of the state-of-the-art. Cooperative robots and sensor networks. Springer, Berlin, pp 31–51
13. Gerkey BP, Mataric M (2003) A formal framework for the study of task allocation in multi-robot systems. Int J Robotic Res—IJRR
14. Gerkey BP, Mataric MJ (2003) Multi-robot task allocation: analyzing the complexity and optimality of key architectures. ICRA 3:3862–3868
15. Yan Z, Jouandeau N, Cherif AA (2013) A survey and analysis of multi-robot coordination. Int J Adv Robotic Syst 10(12):399
16. Gerkey BP, Mataric MJ (2004) Are (explicit) multi-robot coordination and multi-agent coordination really so different. In: Proceedings of the AAAI spring symposium on bridging the multi-agent and multi-robotic research gap, pp 1–3
17. Dias MB, Zlot R, Kalra N, Stentz A (2006) Market-based multi-robot coordination: a survey and analysis. Proc IEEE 94(7):1257–1270
18. Gerkey BP, Mataric MJ (2002) Sold!: auction methods for multi-robot coordination. IEEE Trans Robot Autom 18(5):758–768
19. Choi H, Brunet L, How JP (2009) Consensus-based decentralized auctions for robust task allocation. Trans Robotics 25(4):912–926
20. Lee D-H (2018) Resource-based task allocation for multi-robot systems. Robot Auton Syst 103:151–161
21. Horst R, Pardalos PM, Van Thoai N (2000) Introduction to global optimization. Springer, Berlin
22. Gale D (1989) The theory of linear economic models. University of Chicago press, Chicago
23. Atay N, Bayazit B (2006) Mixed-integer linear programming solution to multi-robot task allocation problem. In: All computer science and engineering research (WUCSE-2006-54)
24. Darrah M, Niland W, Stolarik B (2005) Multiple UAV dynamic task allocation using mixed integer linear programming in a SEAD mission. In: Infotech@ Aerospace, p 7164
25. Juedes D, Drews F, Welch L, Fleeman D (2004) Heuristic resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems. In: 18th IEEE proceedings of international parallel and distributed processing symposium, p 117
26. Wang J, Gu Y, Li X (2012) Multi-robot task allocation based on ant colony algorithm. J Comput 7(9):2160–2167
27. Huang L, Ding Y, Zhou MC, Jin Y, Hao K (2018) Multiple-solution optimization strategy for multirobot task allocation. IEEE Trans Syst Man Cybern Syst
28. Xue F, Dong T, You S, Liu Y, Tang H, Chen L, Yang X, Li J (2020) A hybrid many-objective competitive swarm optimization algorithm for large-scale multirobot task allocation problem. Int J Mach Learn Cybern 1–15
29. Xue F, Tang H, Su Q, Li T (2019) Task allocation of intelligent warehouse picking system based on multi-robot coalition. KSII Trans Internet Inf Syst 13(7)
30. Asma A, Sadok B (2019) PSO-based dynamic distributed algorithm for automatic task clustering in a robotic swarm. Procedia Comput Sci 159:1103–1112
31. Choudhury BB, Biswal BiB (2011) A PSO based multi-robot task allocation. Int J Comput Vis Robotics 2(1):49–61
32. Li X, Ma H-X (2008) Particle swarm optimization based multi-robot task allocation using wireless sensor network. In: 2008 International conference on information and automation. IEEE, pp 1300–1303
33. Jose K, Pratihar DK (2016) Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. Robot Auton Syst 80:34–42
34. Camacho EF, Berenguel M (2012) Control of solar energy systems. IFAC Proc Vol 45(15):848–855
35. Frejo JRD, Camacho EF (2020) Centralized and distributed model predictive control for the maximization of the thermal power of solar parabolic-trough plants. Sol Energy 204:190–199
36. Sánchez AJ, Gallego AJ, Escaño JM, Camacho EF (2018) Event-based MPC for defocusing and power production of a parabolic trough plant under power limitation. Sol Energy 174:570–581
37. Sánchez AJ, Gallego AJ, Escaño JM, Camacho EF (2018) Temperature homogenization of a solar trough field for performance improvement. Sol Energy 165:1–9
38. Anis Koubâa, Abdelmajid Khelil (2014) Cooperative robots and sensor networks. Springer, Berlin
39. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38(4):393–422
40. Martin JG, García RA, Camacho EF (2021) Event-MILP-based task allocation for heterogeneous robotic sensor network for thermosolar plants. J Intell Robot Syst 102(1):1
41. Maurtua I, Susperregi L, Fernández A, Tubío C, Perez C, Rodríguez J, Felsch T, Ghrissi M (2014) MAINBOT-mobile robots for inspection and maintenance in extensive industrial plants. Energy Procedia 49:1810–1819
42. Bellman RE (1961) Dynamic programming treatment of the traveling salesman problem
43. Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. Local Search Comb Optim 1(1):215–310
44. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Q 2(1–2):83–97
45. Lawler EL, Wood DE (1966) Branch-and-bound methods: a survey. Oper Res 14(4):699–719
46. Kartal B, Nunes E, Godoy J, Gini M (2016) Monte Carlo tree search with branch and bound for multi-robot task allocation. In: The IJCAI-16 workshop on autonomous mobile service robots
47. Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. IEEE Trans Comput Intell AI Games 4(1):1–43
48. Goldberg DE (2006) Genetic algorithms. Pearson Education India

49. Gil E, Bernardine JM, Stentz DA (2011) Time-extended multi-robot coordination for domains with intra-path constraints. Auton Robot 30(1):41–56
50. Tolmidis AT, Petrou L (2013) Multi-objective optimization for dynamic task allocation in a multi-robot system. Eng Appl Artif Intell 26(5–6):1458–1468
51. Rajmohan M, Sundar R, Baskaran R et al (2018) Multi-objective optimisation of multi-robot task allocation with precedence constraints. Def Sci J 68(2):175–182
52. Liu C, Kroll A (2016) Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems. Springerplus 5(1):1361