Citation: Javier G. Martin, Muhammad Hanif, Takeshi Hatanaka, José M. Maestre & Eduardo F. Camacho. "Predictive Receding-Horizon Multi-Robot Task Allocation with Moving Tasks". 2022 European Control Conference (ECC), 12-15 Julio 2022 – London (UK). IEEE. ISBN 978-3-9071-4407-7. http://doi.org/10.23919/ECC55457.2022.9838127

Predictive Receding-Horizon Multi-Robot Task Allocation with Moving Tasks*

Javier G. Martin, Muhammad Hanif, Takeshi Hatanaka, *Senior Member, IEEE,* Jose M. Maestre, *Senior Member, IEEE,* and Eduardo F. Camacho, *Fellow, IEEE*

***Abstract*— This paper addresses a multi-robot task allocation (MRTA) towards moving tasks and presents a novel computationally efficient predictive allocation algorithm that requires solving a linear program (LP) problem. Following the receding horizon control policy, the present algorithm repeats the optimization of future task assignments within an *allocation horizon* while predicting the evolution of the system. The online optimization is formulated so that the assignment problem is reduced exactly to an LP. The algorithm is also compared with other traditional methods, namely, the greedy approach and a genetic algorithm (GA). Our results show that the algorithm herein proposed outperforms the greedy approach for small prediction horizons and has significantly lower computational load than GA.**

I. INTRODUCTION

Multi-robot systems (MRS) comprise a set of robots that work collaboratively to perform tasks. Among their applications, MRS can be used for mapping [1], [2], surveillance [3]–[5], maintenance [6], and also as robotic sensor networks (RSN), a particular case of wireless sensor networks (WSN) where sensors are mounted on mobile robots [7], [8].

A common challenge in MRS management is the so-called multi-robot task allocation (MRTA) problem [9], [10], where a set of tasks are assigned to the robots in the most efficient way. This problem can be addressed both with decentralized and centralized approaches. While the decentralized approach tends to be more scalable and robust to communication failures, the centralized one benefits from its access to broader information to generate allocations closer to the optimal one.

According to the taxonomy proposed in [11], MRTA problems can be classified for the number of simultaneous tasks per robot (single-task robot (ST) vs. multi-task robot (MT)), the number of robots per task (single-robot task (SR) vs. multi-robot task (MR)), and the availability of information to plan future allocations (instantaneous assignment (IA) vs. time-extended assignment (TA)). This taxonomy has been further developed in [12] by considering the inter-dependencies between tasks, e.g., for the cases of in-schedule (ID) and cross-schedule (XD) dependencies.

TA MRTA problems become more challenging when tasks move as in [13], where an algorithm based on predator dynamics is proposed to deal with this issue, and [14], where manipulators with limited communications must operate over dynamic tasks. In this work, a novel linear approach is presented to address the XD[ST-SR-TA] MRTA problem with moving tasks taking into consideration the order in which robots perform the tasks. To this end, an algorithm that predicts the evolution of the tasks and robots as it is done in the model predictive control (MPC) framework is proposed [15], [16]. MPC is a control technique that uses a model to predict the response of a system and to compute optimal control signals during a certain time horizon. At each time step, the first control action of the sequence of inputs is applied after its corresponding update using the most recent information available. Predictive control methods in the MRS field have been proposed in other works, e.g., [17], where a decentralized model-predictive control strategy is used to address the observation of multiple moving targets; [18], where a non-linear model predictive control strategy is used to dynamically set the formation leader, and [19], where it is used to compute the time to change the assignment of tasks to robots in XD[ST-SR-TA] MRTA problems with static tasks.

The rest of this paper is organized as follows. In Section II, the problem formulation is mathematically expressed. In Section III, the proposed algorithm is introduced. Section IV presents the case study where the algorithm is tested and the corresponding results are discussed. Finally, in Section V, the conclusions of this work are detailed and some future research lines are given.

Notation: $1_{a\times b}$ denotes a matrix of ones with a rows and b columns; $0_{a\times b}$ represents a matrix of zeros with a rows and b columns; I_a stands for the identity matrix with a rows and columns; and $\mathrm{diag}(x)$ is employed to denote a diagonal matrix containing vector x as its main diagonal.

II. PROBLEM STATEMENT

Let us consider a set of N heterogeneous mobile robots, $\mathcal{R} = \{r_1, r_2, ..., r_{\mathrm{N}}\}$, which must complete a set of M moving tasks, $\mathcal{S} = \{s_1, s_2, ..., s_{\mathrm{M}}\}$, following a given sequence specified by ordinal indixes contained in set $\mathcal{K} = \{1, 2, ..., \mathrm{K}\}$. The speed and the position of robot $r_i \in \mathcal{R}$ are denoted as $v_i \in \mathbb{R}$ and $p_i \in \mathbb{R}^2$, respectively. On the

*This work has been funded by the European Research Council (ERC) under the Advanced Grant OCONTSOLAR (grant agreement number 789051), Grant PID2020-119476RB-I00 funded by MCIN/AEI/ 10.13039/501100011033, and Junta de Andalucía (grant P18-HO-4713). Finally, funding from VI PPIT-US is also gratefully acknowledged.

J. G. Martin, J. M. Maestre and E. F. Camacho are with the Department of Systems and Automation Engineering, University of Seville, Spain {jgarmar, efcamacho,pepemaestre}@us.es and M. Hanif and T. Hatanaka are with the Dept. of Systems and Control Engineering, Major in Systems and Control Engineering, Tokyo Institute of Technology hanif.m.aa@m.titech.ac.jp, hatanaka@sc.e.titech.ac.jp

other hand, the changing location and velocity of each task $s_j \in \mathcal{S}$ can be predicted and are denoted by $q_j \in \mathbb{R}^2$ and $u_j \in \mathbb{R}^2$, respectively. Likewise, task $s_j \in \mathcal{S}$ has a relevance quantified by a positive scalar ϕ_j and requires an operation time τ_j to be performed (besides the time employed to reach the task position). For simplicity, τ_j is considered independent of the robot that performs the task. Likewise, any time spent performing task s_j will be subtracted from τ_j in further allocations. Finally, we define $b_i \in [0, 100]$ as the State of Charge (SOC) of robot r_i, and w_i as its discharge rate, which will be used to avoid the assignment of unfeasible tasks from an energetic viewpoint. Moreover, we may want to prioritize some robots over others, e.g., unmanned ground vehicles (UGVs) over unmanned aerial vehicles (UAVs) because UAVs have less autonomy. To this end, we define the penalty λ_i for the use of robot r_i.

A. Allocation Variables

The allocation, i.e., the assignment of robots to tasks, is described by a set of Boolean variables δ_{ijk} that are set to 1 when the $k-$th mission of robot r_i is to perform task s_j (0 otherwise). For convenience, x is defined as the aggregated vector $x = [\delta_{ijk}]_{i\in\mathcal{R},j\in\mathcal{S},k\in\mathcal{K}}$. The following constraints need to be imposed in the optimization

$$\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{K}} \delta_{ijk} = 1 \quad j \in \mathcal{S}, \tag{1}$$

$$\sum_{j=1}^{\mathrm{M}} \delta_{ijk} \leq 1 \quad i \in \mathcal{R}.\ k \in \mathcal{K}, \tag{2}$$

to ensure that all tasks are performed (1), and to avoid that robots are simultaneously assigned more than one task per mission slot (2).

An example of the described variables can be seen in Fig. 1, where tasks have been considered static.

B. Distance and Time Computations

The distance between robot r_i and task s_j is defined as $d_{ij} = ||q_j - p_i||_2$, and the time for robot r_i to reach task s_j becomes $t_{ij} = \frac{d_{ij}}{v_i}$, as shown in Fig. 2.

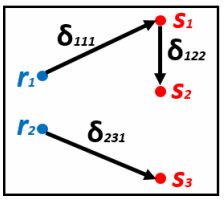

Fig. 1: r_1 performs first s_1 and then s_2; r_2 performs task s_3. Let t_1, t_2 and t_3 be the completion time of s_1, s_2, and s_3 respectively. Then, t_1 is the time it takes for r_1 to reach s_1 plus τ_1; t_2 is t_1 plus the time it takes r_1 to go from s_1 to s_2 plus τ_2; and t_3 is the time it takes for r_2 to reach task s_3 plus τ_3. Likewise, the distance traveled by r_1 is the distance from its initial position to s_1 plus the distance from s_1 to s_2; and the distance traveled by r_2 is that from its initial position to s_3.

Let us introduce now the mean time to perform the $k-$th task as

$$t^k = \frac{1}{\mathrm{M}\cdot\mathrm{N}} \cdot \sum_{i=1}^{\mathrm{N}}\sum_{j=1}^{\mathrm{M}} (t_{ijk} + \tau_j) \quad k \in \mathcal{K}, \tag{3}$$

where t_{ijk} is the time it takes for robot r_i to perform task s_j as its $k-$th mission.

The computation of the time to complete a task s_j, say t_j, requires to know all the previous tasks carried out by the robot that performs s_j. To overcome this issue, we approximate the accumulated time before the beginning of the $k-$th mission as

$$t^k_{\mathrm{a}} = \sum_{\mu=1}^{k-1} t^{\mu} \qquad k \in \mathcal{K}. \tag{4}$$

Thus, an estimation of t_j can be calculated as

$$\begin{aligned} \mathrm{t}_j &= \sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{K}} \delta_{ijk} \cdot t^{\mathrm{E}}_{ijk} \qquad j \in \mathcal{S}, \\ \mathrm{t}^{\mathrm{E}}_{ijk} &= t^k_{\mathrm{a}} + t_{ijk} + \tau_j \qquad i \in \mathcal{R},\ j \in \mathcal{S},\ k \in \mathcal{K}, \end{aligned} \tag{5}$$

with t^{E}_{ijk} as the estimated time if task s_j is allocated to robot r_i as its $k-$th mission.

Finally, we can estimate the distance traveled by the robot r_i as

$$d_i = \sum_{i=1}^{\mathrm{N}}\sum_{j=1}^{\mathrm{M}}\sum_{k=1}^{\mathrm{K}} d_{ijk} \cdot \delta_{ijk} \qquad i \in \mathcal{R}. \tag{6}$$

where d_{ijk} is the estimated distance traveled for robot r_i if allocated with task s_j as its $k-$th mission.

C. Energetic Feasibility

Using (5), it is possible to define a family of functions

$$\beta_{ijk} = \begin{cases} 1 & \text{if } b_i - w_i \cdot t^{\mathrm{E}}_{ijk} \geq 0 \\ -1 & \text{if } b_i - w_i \cdot t^{\mathrm{E}}_{ijk} < 0 \end{cases} \tag{7}$$

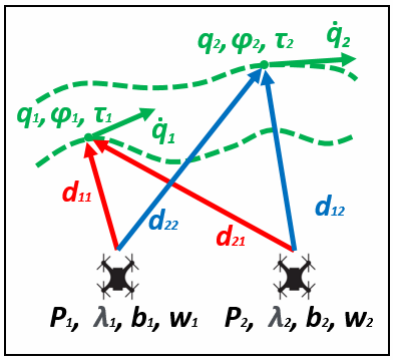

Fig. 2: Example with 2 robots and 2 tasks. Distances from robots to task s_1 are in red, and to task s_2 in blue. Task trajectories can be seen in green.

to assess whether it will be feasible from an energetic viewpoint for robot r_i to perform task s_j as its $k-\text{th}$ mission. This allows us to formulate a new constraint to ensure that tasks will only be assigned to robots with enough SOC:

$$\beta_{ijk} \cdot \delta_{ijk} \quad \geq \quad 0 \; i \in \mathcal{R} \; j \in \mathcal{S}, \; k \in \mathcal{K}, \tag{8}$$

D. Optimization Goals

The core of the multi-criteria objective function employed by our algorithm takes into account both the time in which tasks are performed and the distance traveled by the robots and it is defined as

$$J_c = \sum_{j=1}^{\mathrm{M}} \phi_j \cdot t_j(x) + \sum_{i=1}^{\mathrm{N}} \lambda_i \cdot d_i(x), \tag{9}$$

where $\lambda_i \in \mathbb{R}$ is the penalty for the use of robot r_i; d_i is the distance traveled by the robot r_i, which can be estimated by means of (6); $\phi_j \in \mathbb{R}$ is the penalty for the time employed to complete task s_j; and t_j is the completion time in which task s_j is finished, which can be estimated by means of (5).

Then, the cost function (9) can be transformed into

$$J_s = \sum_{i=1}^{\mathrm{N}} \sum_{j=1}^{\mathrm{M}} \sum_{k=1}^{\mathrm{K}} k \cdot \delta_{ijk} \cdot (\lambda_i \cdot d_{ijk} + \phi_j \cdot t^{\mathrm{E}}_{ijk}), \tag{10}$$

where the multiplication by k enforces that a robot can only be assigned a task as its $k-\text{th}$ mission if it already has another task allocated in its $(k-1)-\text{th}$ mission slot.

Considering the feasibility of the fulfillment of all tasks, we can find two possible cases:

1) All the tasks can be performed. When $\mathrm{N} \cdot \mathrm{K} \geq \mathrm{M}$, there are enough robots to complete the tasks in the allocation horizon and the MRTA problem can be formulated as:

$$\begin{array}{cc} \min\limits_{\delta_{ijk} \in \{0,1\}} & J_s \\ \text{s.t.} & \text{(1), (2), (8).} \end{array} \tag{11}$$

   Note that if $\mathrm{K} = \mathrm{M}$ all allocations can be explored (including those using the same robot to perform all the tasks). Thus, if $\mathrm{K} < \mathrm{M}$ optimality can be lost, although the computational cost of the problem decreases (the number of decision variables in the problem is $\mathrm{N} \cdot \mathrm{M} \cdot \mathrm{K}$).

2) Some tasks cannot be completed when $\mathrm{N} \cdot \mathrm{K} < \mathrm{M}$, the problem must be changed considering the following constraints

$$\sum_{i=1}^{\mathrm{N}} \sum_{k=1}^{\mathrm{K}} \delta_{ijk} \leq 1 \quad j \in \mathcal{S}, \tag{12}$$

$$\begin{cases} \sum\limits_{j=1}^{\mathrm{M}} \delta_{ijk} = 1 \text{ if } \max_{i,k}(\beta_{ijk}) = 1 \\ \sum\limits_{j=1}^{\mathrm{M}} \delta_{ijk} = 0 \text{ if } \max_{i,k}(\beta_{ijk}) = -1 \end{cases} \begin{array}{l} i \in \mathcal{R} \\ k \in \mathcal{K} \end{array}, \tag{13}$$

   to ensure that not all tasks need to be fulfilled in the allocation horizon (maintaining that they can be performed only once) and that robots are not idle (unless they do not have enough battery to perform any of the remaining missions). The problem can be then formulated as

$$\begin{array}{cc} \min\limits_{\delta_{ijk} \in \{0,1\}} & J_s \\ \text{s.t.} & \text{(12), (13), (8),} \end{array} \tag{14}$$

III. RECEDING HORIZON TASK ALLOCATION ALGORITHM

In this section, the LP relaxation of the problem presented in Section II is detailed and the proposed predictive multi-robot task allocation algorithm is presented.

A. LP relaxation

Both (11) and (14) can be transformed into an equivalent LP problem to reduce the computational cost. The equivalent LP problem is as follows:

$$\begin{array}{cc} \min\limits_{x \geq 0} & c \cdot x \\ \text{s.t.} & A \cdot x = b \end{array} \tag{15}$$

where $c = [c_i]_{i \in \mathcal{R}}$ and $x^{\mathrm{T}} = [x_i^{\mathrm{T}}]_{i \in \mathcal{R}}$, with

$$c_i^{\mathrm{T}} = \begin{bmatrix} 1 \cdot (\lambda_i \cdot d_{i11} + \phi_1 \cdot t^{\mathrm{E}}_{i11}) \\ 2 \cdot (\lambda_i \cdot d_{i12} + \phi_1 \cdot t^{\mathrm{E}}_{i12}) \\ \vdots \\ \mathrm{K} \cdot (\lambda_i \cdot d_{i1\mathrm{K}} + \phi_1 \cdot t^{\mathrm{E}}_{i1\mathrm{K}}) \\ 1 \cdot (\lambda_i \cdot d_{i21} + \phi_2 \cdot t^{\mathrm{E}}_{i21}) \\ 2 \cdot (\lambda_i \cdot d_{i22} + \phi_2 \cdot t^{\mathrm{E}}_{i22}) \\ \vdots \\ \mathrm{K} \cdot (\lambda_i \cdot d_{i2\mathrm{K}} + \phi_2 \cdot t^{\mathrm{E}}_{i2\mathrm{K}}) \\ \vdots \\ 1 \cdot (\lambda_i \cdot d_{i\mathrm{M}1} + \phi_{\mathrm{M}} \cdot t^{\mathrm{E}}_{i\mathrm{M}1}) \\ 2 \cdot (\lambda_i \cdot d_{i\mathrm{M}2} + \phi_{\mathrm{M}} \cdot t^{\mathrm{E}}_{i\mathrm{M}2}) \\ \vdots \\ \mathrm{K} \cdot (\lambda_i \cdot d_{i\mathrm{MK}} + \phi_{\mathrm{M}} \cdot t^{\mathrm{E}}_{i\mathrm{MK}}) \end{bmatrix}, \quad x_i = \begin{bmatrix} \delta_{i11} \\ \delta_{i12} \\ \vdots \\ \delta_{i1\mathrm{K}} \\ \delta_{i21} \\ \delta_{i22} \\ \vdots \\ \delta_{i2\mathrm{K}} \\ \vdots \\ \delta_{i\mathrm{M}1} \\ \delta_{i\mathrm{M}2} \\ \vdots \\ \delta_{i\mathrm{MK}} \end{bmatrix}.$$

and the matrixes A and b contain the corresponding constraints. For simplicity, we consider only the problem expressed in (11) because problem (14) can be relaxed in the same manner. Then, these matrixes A and b can be written as

$$A = \left[\begin{array}{c|c} R & 0 \\ \hline V & I \end{array}\right], \quad b = \begin{bmatrix} 1_{\mathrm{M}+\mathrm{N}\cdot\mathrm{K}\times 1} \\ 0_{\mathrm{N}\cdot\mathrm{M}\cdot\mathrm{K}\times 1} \end{bmatrix}, \tag{16}$$

with

$$R = \begin{bmatrix} R_1 & R_2 & \cdots & R_{\mathrm{N}} \end{bmatrix},$$

$$R_i = \begin{bmatrix} 1_{1\times\mathrm{K}} & 0_{1\times\mathrm{K}} & \cdots & 0_{1\times\mathrm{K}} \\ 0_{1\times\mathrm{K}} & 1_{1\times\mathrm{K}} & \cdots & 0_{1\times\mathrm{K}} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times\mathrm{K}} & 0_{1\times\mathrm{K}} & \cdots & 1_{1\times\mathrm{K}} \end{bmatrix},$$

$$V = \left[\begin{array}{ccccc} V_1 & V_2 & \cdots & V_{\mathrm{N}} \\ \hline \multicolumn{4}{c}{\mathrm{diag}([\beta_{111} \cdots \beta_{\mathrm{NMK}}])} \end{array}\right],$$

$$V_i = \begin{bmatrix} 0_{\mathrm{K}\times\mathrm{K}} & 0_{\mathrm{K}\times\mathrm{K}} & \cdots & 0_{\mathrm{K}\times\mathrm{K}} \\ \vdots & \vdots & & \vdots \\ I_{\mathrm{K}} & I_{\mathrm{K}} & \cdots & I_{\mathrm{K}} \\ \vdots & \vdots & & \vdots \\ 0_{\mathrm{K}\times\mathrm{K}} & 0_{\mathrm{K}\times\mathrm{K}} & \cdots & 0_{\mathrm{K}\times\mathrm{K}} \end{bmatrix}.$$

Note that the $i-\text{th}$ block row in V_i is the one containing the I_{K} matrices.

Problem (15) is known to be equivalent to (11) in the sense of having the same optimizer if the matrix A is totally unimodular and every element of b is an integer [20]. The latter condition is obviously satisfied in the present case. Regarding the former, the following lemma is shown to be true, which is equivalent to total unimodularity of A (Theorem 19.3 [20]).

Lemma 1 *Matrix A satisfies that $\exists \xi \in \{0, \pm 1\}$ s.t. $A \cdot \xi \in \{0, \pm 1\}$.*

Proof: Let us define

$$\begin{array}{rcl} \xi & = & \left[\begin{array}{cccccc|c} \xi_1 & -\xi_2 & \xi_3 & \cdots & \pm\xi_{\mathrm{N}} & -1_{1\times(\mathrm{M}+1)\cdot\mathrm{N}\cdot\mathrm{K}} \end{array} \right], \\ \xi_i & = & \left[\begin{array}{cccccc} 1_{1\times\mathrm{K}} & -1_{1\times\mathrm{K}} & 1_{1\times\mathrm{K}} & \cdots & \pm 1_{1\times\mathrm{K}} \end{array} \right], \end{array} \tag{17}$$

where ξ_{N} is positive if N is odd and negative otherwise.

That is, the first K elements of ξ are equal to 1, the following K ones are equal to -1, and this pattern is repeated until the $\mathrm{K} \cdot \mathrm{M}-\text{th}$ element. Note that the last term will be positive if K is odd and negative otherwise.

Then, it is easy to see that $\left[\begin{array}{c|c} R & 0 \end{array} \right] \cdot \xi \in \{0, 1\}$, and that $\left[\begin{array}{c|c} V & I \end{array} \right] \cdot \xi \in \{0, \pm 1\}$. Thus, $A \cdot \xi \in \{0, \pm 1\}$, which completes the proof. ■

Therefore, we can solve the LP (15) rather than the integer program (11).

B. Predictive Receding-Horizon Multi-Robot Task Allocation (PMRTA) Algorithm

To solve the problem, we need d_{ijk}, t_{ijk}, t_{a}^k, and t_{ijk}^{E} which can be obtained following Algorithm 1.

After computing d_{ijk} and t_{ijk} $\forall k$, the LP problem expressed in (15) is solved and the assignment corresponding to $k = 1$ is applied. Once a task is completed, the algorithm is restarted in an event-driven fashion after updating the positions of robots and tasks, and the battery status. In this way, predictions and allocations are constantly updated based on the current information. See Algorithm 2, where the proposed predictive receding-horizon multi-robot task allocation (PMRTA) algorithm is described.

Algorithm 1: d_{ijk}, t_{ijk}, t_{a}^k and t_{ijk}^{E} estimation.

Let $k = 1$;
Initialize the estimated position of robots using their current position, $p'_{ik} = p_i$;
Initialize the estimated position of tasks using their current position, $q'_{jk} = q_j$;
while $k \leq \mathrm{K}$ **do**
  Compute ℓ_{ijk} using p'_{ik}, q'_{jk}, v_i, and u_j;
  Compute $d_{ijk} = ||\ell_{ijk} - p'_{ik}||_2 \ \forall i \in \mathcal{R},\ j \in \mathcal{S}$;
  Compute t_{ijk} as $t_{ijk} = \frac{d_{ijk}}{v_i}\ i \in \mathcal{R},\ j \in \mathcal{R}$;
  Use t_{ijk} and τ_j to compute t^k using (3);
  Compute t^k by means of (3);
  Compute t_{a}^{k+1} by means of (4);
  Estimate the position of tasks in $k+1$, q'_{jk+1}, using q'_{jk}, u_j and the time obtained in t_{a}^k;
  Compute t_{ijk}^{E} $\forall i \in \mathcal{R},\ j \in \mathcal{S}$ by means of (5);
  Remove from $\mathcal{S}$ the task with the lowest estimated mean time in k, since we will consider this is the task that has been completed;
  Compute the position of the robots in $k+1$, p'_{ik+1}, by means of
  $$p'_{ik+1} = p'_{ik} + \sum_{j=1}^{|\mathcal{S}|+1} t_{\mathrm{a}}^k \cdot v_i \cdot \frac{q'_j - p_i}{||q'_j - p_i||_2} \cdot \frac{1}{|\mathcal{S}| + 1};$$
end

Algorithm 2: Predictive Receding-Horizon Multi-Robot Task Allocation (PMRTA)

while $|\mathcal{S}| > 0$ **do**
  Update the real position of the robots;
  Update the real position of the tasks;
  Update the real state of the batteries;
  Update the predicted position of the tasks and the robots, d_{ijk}, t_{ijk}, t^k, t_{a}^k and t_{ijk}^{E} using Algorithm 1;
  Solve the allocation using (15);
  Apply the part of the allocation corresponding to $k = 1$;
  Complete the corresponding task and update the number of remaining tasks, $|\mathcal{S}| = |\mathcal{S}| - 1$;
end

IV. Results

A case study with 4 robots and 16 tasks moving northeast at different speeds is employed using a sample time of 0.1 seconds. The parameters of the robots and the tasks have been generated randomly and can be seen in Table I.

An assessment of the effects of the allocation horizon K in the results and the increase of the computational load with K can be seen in Table II. For comparison purposes, J has been computed following (9) for each simulation using the *real* distance traveled by each robot and the *real* time in which the tasks were finished. Note that the worst value is obtained with $\mathrm{K} = 1$, which is equivalent to the greedy approach. The best allocation occurs when $\mathrm{K} = 3$ and stabilizes after $\mathrm{K} = 4$, obtaining the same allocation disregarding the value of K and the computational burden increases linearly up to 2.02 seconds. We have compared it also with the allocation obtained by solving (15) once obtaining that the computational burden is similar to that of $\mathrm{K} = 1$ but the performance decreases significantly, i.e., the periodic recomputing of the allocation is necessary. This phenomenon is to be expected since the predictions employed by the LP are based on several simplifying assumptions. Finally, the allocations obtained using $\mathrm{K} = 3$ are shown in Fig. 3. There are 16 allocations because there are 16 tasks and a new allocation is computed every time a task is completed.

The results are also compared with those achieved by the Genetic Algorithm (GA) method presented in [21], which

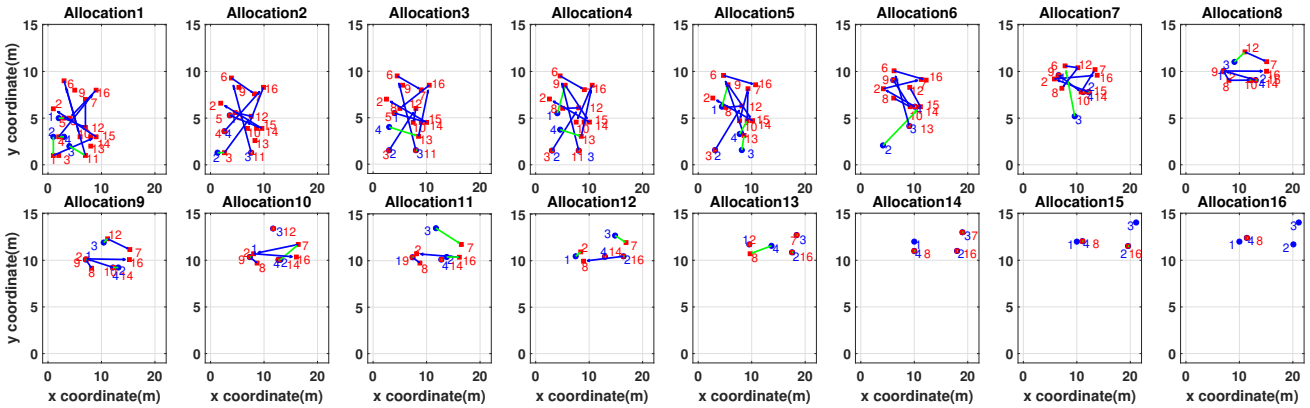

Fig. 3: The lines represent the allocation obtained in each iteration. The green line represents the part of the allocation corresponding to $k = 1$, i.e., the part of the allocation which is applied, while the blue lines represent the part of the allocation for $k > 1$. In the first allocation, robot 2 is sent to task 1, robot 4 is sent to task 4, robot 3 is sent to task 11, and robot 1 is sent to task 5. The second allocation occurs when robot 2 completes task 1. In this allocation, robots 1, 3, and 4 keep on completing the current tasks while robot 2 is sent to task 3. Note that in the previous allocation, the second assignment for robot 2 was not task 3 but task 15. The third allocation is similar to the second one, but robot 4 finishes and is sent to a new task (task 13). In the fourth allocation, we can see how task 13 is reallocated to robot 4 (since it has not reached the task yet), and, in the fifth allocation, we can see that the proposed algorithm redirects robot 4 to task 10 and task 13 is reassigned to robot 3. There are 16 allocations since each time a task is completed a new one is computed.

TABLE I: Parameters of the robots in the case study

Robots	$p_i(0)$	$\|\|v_i\|\|$	λ_i	$b_i(0)$	w_i
r_1	(2,5)	4	1	100	0.2
r_2	(1,3)	8	2	80	0.5
r_3	(4,2)	8	1	40	0.4
r_4	(3,3)	16	2	30	0.1
Tasks	$q_j(0)$	$\dot{q}_j$		τ_j	ϕ_j
s_1	(1,1)	(1,1)		1	3
s_2	(1,6)	(3,2)		2	1
s_3	(2,1)	(2,1)		3	5
t_4	(2,3)	(2,2)		2	4
t_5	(3,5)	(2,1)		2	2
t_6	(3,9)	(3,1)		1	3
t_7	(7,7)	(4,2)		4	1
t_8	(4,5)	(2,2)		5	1
t_9	(5,8)	(1,1)		7	6
t_{10}	(6,3)	(3,3)		1	1
t_{11}	(7,1)	(2,1)		2	4
t_{12}	(7,4)	(2,4)		1	3
t_{13}	(8,2)	(1,2)		4	5
t_{14}	(8,3)	(2,3)		9	6
t_{15}	(9,3)	(2,3)		3	2
t_{16}	(9,8)	(3,1)		5	1

solves

$$\min_{U} \quad J^{\mathrm{GA}} = \sum_{j=1}^{\mathrm{M}} \phi_j \cdot t_j(U) + \sum_{i=1}^{\mathrm{N}} \lambda_i \cdot d_i(U) + \gamma(U)$$
$$\text{s.t.} \qquad u_i(n) \in \mathcal{S} \cup \{0\} \ \forall \ i, n, \tag{18}$$

where $U = [u_1(1), u_1(2), .., u_1(M), u_2(1), ..., u_{\mathrm{N}}(M)]$ represents the complete allocation and $u_i(n) \in \mathcal{S} \cup \{0\}$, stands for the $n-$th allocated task of robot r_i. Note that $u_i(n) = 0$ when robot r_i is idle. Here, ϕ_j and λ_i respectively correspond to the priority given to a certain task s_j and the penalty for using robot r_i; $t_j(U)$ and $d_i(U)$ correspond to nonlinear functions regarding the time that it takes to complete task s_j and the distance traveled by robot r_i in a given allocation U. Function $\gamma(U)$ is a soft restriction ensuring the energetic feasibility (no robot has a negative battery level after the allocation) and that every task is performed once. Finally, GA parameters, which have been tuned by trial and error to achieve a good balance between performance and computational load, can be seen in Table III. Note that given the heuristic nature of GA, the problem has been solved 10 times to average the results.

TABLE II: Results of the case-study for different K

K	1	2	3	4	···	16	16 (no recom.)
J	609.14	519.38	484.02	504.35	···	504.35	945.79
r_i	d_i (m)						
r_1	12.95	26.40	11.20	14.97	···	14.97	17.49
r_2	31.27	28.24	26.08	32.72	···	32.72	25.32
r_3	22.12	12.65	28.48	22.54	···	22.54	24.60
r_4	32.01	28.52	24.74	23.55	···	23.55	30.23
s_j	t_j (s)						
s_1	1.2	1.2	1.2	4.4	···	4.4	15.5
s_2	2.2	2.2	12	13.3	···	13.3	31.5
s_3	12.4	4.3	4.3	3.2	···	3.2	3.6
s_4	4	2	2	2.3	···	2.3	4.3
s_5	2	2.3	2.1	6.4	···	6.4	26.3
s_6	5.4	3.8	8.1	8	···	8	33.4
s_7	6.7	16.3	14.1	12.3	···	12.3	38.9
s_8	7.4	7.7	16.7	16.7	···	16.7	45.5
s_9	14.7	10.9	9.9	11	···	11	20.4
s_{10}	1.3	10.2	9.4	10.7	···	10.7	39.8
s_{11}	9.1	6.9	2.3	2.4	···	2.4	4.1
s_{12}	2.5	3.4	9.5	9.3	···	9.3	21.2
s_{13}	12.3	8.7	6.4	4.5	···	4.5	5.1
s_{14}	18.1	16.1	11.4	11.4	···	11.4	21.4
s_{15}	9	11.2	8.3	7.8	···	7.8	9.1
s_{16}	7.9	15.6	14.8	16.1	···	16.1	27.9
	Computational time (s)						
	0.74	0.79	0.86	0.93	···	2.02	0.76

TABLE III: GA tuning parameters

Pop. size	Max Gen.	Max Stall Gen.	$\mathcal{G}_{\mathrm{C}}$	$\mathcal{G}_{\mathrm{E}}$	$\mathcal{G}_{\mathrm{M1}}$	$\mathcal{G}_{\mathrm{M2}}$
400	50	10	0.7	0.3	0.5	0.5

The computational load of GA is around 60 seconds and J^{GA} is in the range $[600, 650]$, i.e., GA barely reaches the

performance achieved by the greedy approach despite its significantly higher computational load. However, it must be taken into account that, unlike the greedy approach and the predictive approach, the implementation of GA here lacks the capability of leaving tasks incomplete.

To validate the proposed method, we have generated 1000 random scenarios with $\mathrm{N} \in [1, 10]$, and $\mathrm{M} \in [1, 20]$, located in random spots of a 15×15 m square area. These scenarios have been solved using GA and PMRTA $\forall \mathrm{K} \in [1, \mathrm{M}]$. The rest of the parameters are contained in the following ranges: $v_i \in [4, 20]$ m/s; $\lambda \in (0, 5)$; $b_i \in [0, 100]$; $w_i \in (0, 1)$; u_j constant with $||u_j|| \in (0, 4]$ m/s and random direction; $\tau_j \in (0, 10)$; and $\phi_j \in (0, 10)$ m/s.

Our results show that when GA is constrained to achieve an allocation in the same time as the proposed algorithm it achieves a worse allocation in most cases (90.15%). In Fig. 4, it can be seen that PMRTA is superior in almost all cases with small K ($\mathrm{K} > 1$), although its performance decreases for larger prediction horizons until it finally becomes worse than the greedy algorithm ($\mathrm{K} = 1$). On the other hand, when GA has unlimited time to converge (the stopping criterion was set to 30 generations with no improvement), it outperforms the proposed algorithm in most cases, as expected. However, even in these unfair conditions, the proposed algorithm occasionally beats GA. Also, note that for $\mathrm{K} = 2$ PMRTA outperforms the greedy algorithm and is able to get better results than GA in 18.9% of the cases (the greedy algorithm outperforms the unlimited GA only 7% of the simulations).

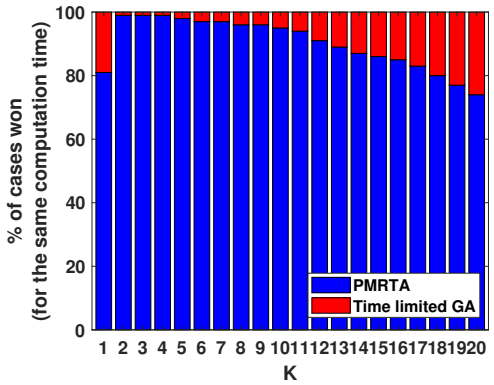

Fig. 4: Performance comparison between PMRTA and time limited GA.

V. CONCLUSIONS

A predictive MRTA algorithm that takes advantage of the available information regarding the system evolution has been presented. In particular, the proposed algorithm computes the best allocation based on the predicted system evolution for a given horizon and applies the first element of the calculated sequence in a receding horizon manner.

Our results show that the proposed algorithm outperforms well-established heuristical methods such as the greedy algorithm and GA in time-limited optimizations. Also, the proposed algorithm performs remarkably better with small horizons than with larger ones due to the use of a simple internal model for the predictions. To overcome this issue, we plan to iterate the allocation until convergence is obtained and to employ myopic cost functions in future works. Also, in future works, experiments with real robots in more challenging environments will be carried out.

REFERENCES

[1] F. Nex and F. Remondino, "UAV for 3D Mapping Applications: A Review," *Applied geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[2] P. Maini and P. Sujit, "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, 2015, pp. 1370–1377.

[3] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "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, 2009, pp. 82–85.

[4] A. Puri, "A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance," *Department of computer science and engineering, University of South Florida*, pp. 1–29, 2005.

[5] H. Dan, J. Yamauchi, T. Hatanaka, and M. Fujita, "Control barrier function-based persistent coverage with performance guarantee and application to object search scenario," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2020, pp. 640–647.

[6] J. Franko, S. Du, S. Kallweit, E. Duelberg, and H. Engemann, "Design of a multi-robot system for wind turbine maintenance," *Energies*, vol. 13, no. 10, p. 2552, 2020.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[8] A. Koubâa and A. Khelil, *Cooperative Robots and Sensor Networks*. Springer, 2014.

[9] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-art," in *Cooperative Robots and Sensor Networks*. Springer, 2015, pp. 31–51.

[10] W. Dai, H. Lu, J. Xiao, Z. Zeng, and Z. Zheng, "Multi-robot dynamic task allocation for exploration and destruction," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 455–479, 2020.

[11] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[12] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-Robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[13] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu, "Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach," *Automatica*, vol. 100, pp. 75–81, 2019.

[14] L. Jin and S. Li, "Distributed task allocation of multiple robots: A control perspective," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 693–701, 2016.

[15] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[16] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Springer science & business media, 2013.

[17] J. Kuhn, C. Reinl, and O. Von Stryk, "Predictive control for multi-robot observation of multiple moving targets based on discrete-continuous linear models," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 257–262, 2011.

[18] B. Augusto de Holanda, S. P. Madruga, A. V. Brito, and T. P. Nascimento, "Dynamic leader allocation in multi-robot systems based on nonlinear model predictive control," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 359–376, 2020.

[19] M. Gaggero, D. Di Paola, A. Petitti, and L. Caviglione, "When time matters: Predictive mission planning in cyber-physical scenarios," *IEEE Access*, vol. 7, pp. 11 246–11 257, 2019.

[20] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[21] J. G. Martin, J. R. D. Frejo, R. García, and E. F. Camacho, "Multi-Robot Task Allocation Problem with Multiple Non-Linear Criteria Using Branch and Bound and Genetic Algorithms," *Intelligent Service Robotics*, 2021.