Distributed Dynamic Sensor Assignment of Multiple Mobile Targets

Eduardo Montijano, Danilo Tardioli, Alejandro R. Mosteo

***Abstract*— Distributed scalable algorithms are sought in many multi-robot contexts. In this work we address the dynamic optimal linear assignment problem, exemplified as a target tracking mission in which mobile robots visually track mobile targets in a one-to-one capacity. We adapt our previous work on formation achievement by means of a distributed simplex variant, which results in a conceptually simple consensus solution, asynchronous in nature and requiring only local broadcast communications. This approach seamlessly tackles dynamic changes in both costs and network topology. Improvements designed to accelerate the global convergence in the face of dynamically evolving task rewards are described and evaluated with simulations that highlight the efficiency and scalability of the proposal. Experiments with a team of three Turtlebot robots are finally shown to validate the applicability of the algorithm.**

I. Introduction

Teams of robots are becoming a realistic proposition for many real-world tasks, like exploration, search and rescue, surveillance, etc. In many cases, robots can perform only one activity at a time, like visiting a place, following a target, or fulfilling a role in a formation [1]. The need to assign tasks or roles to robots is a particular form of multi-robot task allocation, in which an optimization problem is solved. Many approaches optimize the sum of individual costs or rewards as a way of improving the aggregate team performance. The static case of assigning one task per robot, with known costs or utilities, is known as the linear assignment or maximum bipartite matching problem. This optimization has been long known to be centrally solvable in polynomial time with, e.g., the Hungarian method [2].

The increasing desire for resilient, large fleets of robots prompts the use of distributed solutions with partial communication [3]. An early example for the described problem appears in [4], although an implicit complete communication graph is used in this case. The DCOP framework [5] is a general approach to distributed constraint optimization, tailored to NP-complete problems. Distributed methods based on auctions have also been deeply studied in this context [6]–[8]. A distributed task-swapping technique to reach the optimal assignment was presented in [9]. A distributed Hungarian method with convergence bounds was recently presented in a robotic context in [10], being applied to the multi-robot routing with time windows variant.

In some problems, however, there is the need to continuously re-evaluate the assignment of tasks to robots, in what was termed the *iterated assignment* in the classical taxonomy in [11] for multi-robot task allocation. This situation arises in any mission where new information can be ascertained, or that have task rewards that evolve with time. A paradigmatic example is the multi-target tracking problem [12], in which sensors or robots have to be assigned to uncooperative moving targets. The effect of changing task rewards was addressed in a centralized way by the Hungarian method in [13]. Uncertainty in a set of static costs was treated in a distributed fashion in [14]. A distributed method based on sequential single-item auctions was proposed in [15].

The algorithm discussed in this work is based on a distributed simplex initially described in [16] for static costs and modified in our earlier work in [17] for optimal formations. The main advantage of a simplex formulation for this problem is that it can be easily extended to handle time-varying costs without the need for re-initialization, i.e., the algorithm can be run continuously and seamlessly, updating the costs as new information arrives, providing, at each communication round, the best possible outcome given the current information that each robot has. Besides, the algorithm only requires local communications and works well with changing neighborhoods.

In particular, the contributions in this paper are: firstly, the adaptation of our previous algorithm in [17] to a different problem. while in that work there were no moving targets or robots, but evolving pose estimations, herein we consider dynamically moving entities. The former problem is simpler in the sense that once convergence is achieved there is no change in the optimal assignment, whereas the movement of robots and targets makes the problem treated here open-ended. Secondly, we add a reward-estimation capability to reduce the impact of outdated information in the assignment process. Finally, we deploy our algorithm in actual robots that mimic a distributed target tracking mission with panning cameras.

The following section describes the problem formalization. Section III describes the static distributed Simplex algorithm. Our algorithm to consider the dynamic assignment case is presented in Section IV. Simulations and a real experiment are discussed in Sections V and VI, before the conclusion.

II. Problem formulation

In this paper, we will consider a team of N robots, $\mathcal{V} = \{1, \dots, N\}$, with limited communication capabilities

E. Montijano is with Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza, Spain. E-mail: *emonti@unizar.es*

D. Tardioli and A. R. Mosteo are with Centro Universitario de la Defensa, Zaragoza, Spain and Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza, Spain. E-mail: {*dantard*}{*amosteo*}*@unizar.es*

This work was supported in part by the Spanish projects DPI2016-76676-R-AEI/FEDER-UE, PGC2018-098817-A-I00, CUD2018-03, DGA-T45_17R/FSE and in part by the Office of Naval Research Global project ONRG-NICOP-N62909-19-1-2027. We are grateful for this support.

modeled with a connected and undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each robot represents a node in the graph, whereas an edge, $(i, j) \in \mathcal{E}$, represents the possibility of robots i and j to exchange information. We denote by $\mathcal{N}_i$ the neighbors of robot i, i.e., $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$, and $|\mathcal{N}_i|$ its cardinality.

The objective of the robots is to observe a set of N *mobile* targets *over time*. Following standard nomenclature in classic assignment problems, we define the set of assignation variables $x_{ij}(t),\ i, j \in \{1, \dots, N\}$, where $x_{ij}(t) = 1$ implies that robot i is assigned to observe target j at time t, and $x_{ij}(t) = 0$ otherwise. Here, the time will be considered discrete, understanding each time step as the time required for a communication round of robots with their direct neighbors.

Let $r_{ij}(t)$ be the *time-varying* reward associated to assigning robot i to target j at time t, which we consider that each robot is able to compute by means of onboard perception, e.g., by means of the blob size within an image [18]. We do not make any assumption on the motion of the robots or the targets, nor we assume that robots need to know any positions. Each robot initially has access to its own rewards. The knowledge of other robots' rewards can only be obtained by means of communications, which can only be done with neighbors in $\mathcal{G}$. On the other hand, we assume an implicit common knowledge among the robots regarding the identities of the targets, namely, target j is the same for all robots. This can be achieved by means of a distributed data association initialization, e.g., [19].

We also impose the typical constraints on linear assignment problems, i.e., at each time each robot can only be assigned to one target and each target should have one robot assigned. This is formally described by

$$\begin{aligned} &\sum_{j=1}^{N} x_{ij}(t) = 1,\ \forall i \in \{1, \dots, N\}, \\ &\sum_{i=1}^{N} x_{ij}(t) = 1,\ \forall j \in \{1, \dots, N\},\ \text{and} \\ &x_{ij}(t) = \{0, 1\},\ \forall i, j \in \{1, \dots, N\}, \end{aligned} \tag{1}$$

for all t.

Therefore, the objective of the paper is to obtain, at each time step, the optimal assignment in terms of overall reward, satisfying the constraints given in (1). Formally speaking, this is:

$$\begin{aligned} \underset{x_{ij}(t)}{\text{maximize}} \quad & \sum_{t} \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij}(t) r_{ij}(t), \\ \text{subject to} \quad & (1) \text{ for all } t. \end{aligned} \tag{2}$$

For a given t the problem has been thoroughly researched, as discussed in the introduction. The key assumption here is that the rate of change of the rewards is considered to be of the same order of magnitude as the communication frequency, which implies that existing centralized or distributed static assignment algorithms will be only able to provide outdated solutions, as rewards become obsolete over time. The objective of this paper is precisely to account for both variations in rewards and topology, looking for the best possible assignment at each time in a distributed fashion, exploiting the information of the past.

III. Distributed Simplex

The solution presented in this paper exploits the distributed implementation of the simplex algorithm given in [16]. For the sake of completeness, we briefly summarize this algorithm.

A. Centralized simplex

Assume first constant rewards, r_{ij}, describing the benefit obtained by assigning robot i to target j. The concatenation of all the rewards is denoted by $\mathbf{r} = (r_{11}, \dots, r_{NN})^T$. Then, the linear program is:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbf{r}^T \mathbf{x}, \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \end{aligned}$$

where $\mathbf{x} \in \{0, 1\}^{N^2}$ is the vector with all the decision variables and

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_N \otimes \mathbf{1}_N^T \\ \mathbf{1}_N^T \otimes \mathbf{I}_{N-1} \end{pmatrix} \in \mathbb{R}^{2N-1 \times N^2}, \text{ and } \mathbf{b} = \mathbf{1}_{2N-1},$$

code the constraints given in (1), with $\mathbf{I}_N$ the identity matrix of dimension $N \times N$, $\mathbf{1}_N$ the column vector of dimension N with all its components equal to one and $\otimes$ is used to denote the Kronecker product. The second block of the matrix $\mathbf{A}$ is of dimension $N - 1$, instead of N as in (1), to satisfy linear independence of all the constraints.

Since the problem is under-constrained, there are infinite possible optimal solutions. The idea behind simplex is to find a partition of $2N - 1$ columns of $\mathbf{A}$, $\mathbf{B}^*$, such that they form a basis of the space whose associated decision variables yield the optimal reward. The value of the decision variables is equal to $\mathbf{x}_{\mathbf{B}^*} = \mathbf{B}^{*-1}\mathbf{b}$, while the rest of the decision variables are set to zero. The algorithm finds this base by swapping the elements that belong to $\mathbf{x}_\mathbf{B}$, forcing the overall reward to grow with each change and converging to the optimal solution in a finite number of changes.

The particular problem of assignment is degenerated, because there are several decision variables in the optimal basis that take the value zero[1]. Thus, the algorithm is slightly modified by the method described in [20], based on lexicographic perturbation of vectors. This nonstandard simplex method guarantees that no cycling can occur in degenerate problems, and that for dual degenerate ones the same optimal basis will be chosen by all robots in the distributed setup.

B. Distributed simplex

The distributed simplex algorithm defines a partition $\mathcal{A} = \{\mathbf{A}^{[1]}, \dots, \mathbf{A}^{[N]}\} \equiv \mathbf{A}$ of the full problem columns so that each robot knows initially only the columns relative to its own reward for every target, plus a set of artificial columns constructed using the big-M method. In our case the rewards

[1]Note that only N decision variables take the value of one

associated to the artificial columns need to be smaller than any possible rewards, e.g., the value -1. From that set, an initial basis, $\mathbf{B}^{[i]}$, is computed locally by each robot using the simplex algorithm.

At each communication round, robots broadcast to neighbors only the columns forming their current optimal basis, $\mathbf{B}^{[i]}$, with their associated rewards. Then, each robot solves to optimality the subset simplex formed by the columns it is aware of, namely its permanent set $\mathbf{A}^{[i]}$, its current basis, $\mathbf{B}^{[i]}$, and the bases received in the last round, $\mathbf{B}^{[\mathcal{N}_i]}$, using as starting basis its current one. The algorithm is guaranteed to converge in a finite number of communication rounds to the optimal, same basis for all the robots [16], $\mathbf{B}^{[i]} = \mathbf{B}^*, \forall i$.

IV. Distributed Dynamic Simplex

Unfortunately, in our problem, the distributed simplex described in the previous section cannot be used "as is" because the rewards evolve with time at the same pace that the robots exchange their basis. Moreover, there are other important aspects that we need to consider.

First of all, since we assume a common knowledge among robots regarding the identities of the targets, we can consider that the matrix $\mathbf{A}$ is fixed and known by all the robots, even when the rewards of other robots are still unknown to each one. This implies that robots do not need to exchange the columns of $\mathbf{A}$ that form their basis, but instead their indices, (i, j), together with the associated rewards.

On the other hand, since the rewards evolve with time, and each robot is only able to update the rewards associated to its permanent set of columns, these values need to be periodically exchanged as a part of the current optimal solution that each robot is handling. The simplest solution here would be to broadcast to neighbors all the current rewards, besides those available from other robots, but this would incur in unnecessarily large bandwidth requirements. Instead, it seems better to only send the rewards of the elements that belong to the current optimal basis for each robot, i.e., send $\mathbf{r}^{[i]}_{\mathbf{B}^{[i]}}(t)$ to the neighbors at each communication round, together with the indices of those values.

However, this information is not enough for the method to work. On the one hand, it might happen that a robot receives two different values of the same reward, because of the different lengths of different paths of $\mathcal{G}$. On the other hand, and more importantly, some reward $r_{ij}(t)$ can reach at some point a sufficiently high value that it is not further propagated, if it leaves the optimal basis in the robot where it has been computed. This value can prevent other columns with current reward values from entering into the solution of other robots, resulting in stuck sub-optimal solutions.

A. Adding time-to-live to the rewards

We address these two issues following the same procedure that we used in a formation control problem in [17], attaching to each reward an additional value containing the time when it was computed. For robot k this is:

$$a_{ij}(t) = \begin{cases} 0 & \forall (i,j) \in \mathbf{A}^{[k]} \\ a_{ij}(t-1) + 1 & \text{otherwise} \end{cases} . \qquad (3)$$

When a robot receives this information from its neighbors, it chooses the newest possible reward to guarantee that the error with respect to the current ones is as small as possible. Additionally, in order to prevent old rewards to get stuck in the basis of the robots, we eliminate the rewards that are old enough, making them artificially small. We consider that one reward is old whenever $a_{ij}(t) > N$, assuming the worst case, in which the diameter of $\mathcal{G}$ is maximum. Thus, the rewards of robot k at iteration t are:

$$r_{ij}(t) = \begin{cases} r_{ij}(t) & \forall (i,j) \in \mathbf{B}^{[k]} \\ \arg\min a_{ij}^{[\mathcal{N}_k \cup k]}(t) & (i,j) \notin \mathbf{B}^{[k]} \wedge \\ & \min a_{ij}^{[\mathcal{N}_k \cup k]}(t) \leq N \\ -1 & \min a_{ij}^{[\mathcal{N}_k \cup k]}(t) > N \end{cases} \qquad (4)$$

These rewards can be used to make an update on the local optimal basis, solving locally an instance of the LexSimplex algorithm, as in the original method in [16]. Then, each robot can decide locally which target to track by finding the target j such that $x_{ij} \in \mathbf{x}_{\mathbf{B}^{[i]}}(t)$ is equal to one.

B. Predicting current rewards

It is noteworthy that the rewards associated to non-neighboring robots in (4) will not be the current rewards for those robots but rather the rewards at time $t - a_{ij}(t)$. Under the assumption that rewards evolve smoothly with time, robots can use the knowledge from previous communication rounds to obtain an estimation of the current value of the rewards from their past values.

The key idea is to replace the second case in (4) for another value more representative of the current reward. In particular, in this paper we use a simple first order extrapolation model

$$\hat{r}_{ij}(t) = r_{ij}(t_{ij}) + a_{ij}(t)\left(r_{ij}(t_{ij}) - r_{ij}(t_{ij} - 1)\right) \qquad (5)$$

where t_{ij} is a shortcut for $t - a_{ij}(t)$, as an initial test of the potential of estimation in the problem. Nevertheless, our algorithm could use any existing estimation or extrapolation technique [21], [22], being onboard computation the only limitation to account for.

C. Full Algorithm

The iterative method is schematized in Algorithm 1. Note that the whole procedure is fully distributed, in the sense that the robots only use their own information and the one provided by direct neighbors in the communication graph. The communication demands of the algorithm grow linearly with the size of the network, sending 3 data at each communication round for each column in $\mathbf{B}^{[i]}(t)$: the indices in $\mathbf{A}$, the rewards and their ages. Finally, computationally speaking, each robot needs to solve a simplex algorithm at each round, which current processors can handle efficiently for large values of N.

V. Simulations

To assess the performance of the allocation algorithm in dynamically changing situations we ran randomized simulations with the following parameters. The number of robots

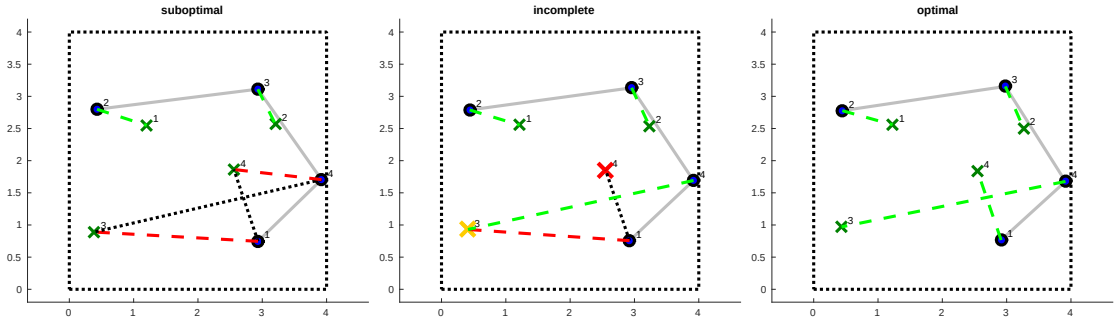


Fig. 1. Three consecutive simulation steps that show possible allocation status during convergence. On the left, robots (blue dots) 1 and 4 have wrongly allocated to themselves (red dashed lines) each other optimal target (X markers). Thus, the allocation is suboptimal but all targets have an assigned tracker. In the center, robot 4 already has found its optimal target, but robot 1 still wants the same target for itself. Thus, target 3 (yellow) has two trackers while target 4 (red) has no assigned tracker. Finally, on the right, all targets are assigned to its optimal trackers. Gray solid lines denote communication neighbors.

Algorithm 1 Distributed Dynamic Assignment (Robot i)

Use simplex to obtain an initial basis, $\mathbf{B}^{[i]}$, from $\mathbf{A}^{[i]}$
while true **do**
  Compute own rewards $r_{ij}, j = 1, \ldots, N$
  Update age of all the rewards with eq. (3)
  Send indices of $\mathbf{B}^{[i]}$, $\mathbf{r}^{[i]}_{\mathbf{B}^{[i]}}$ and $\mathbf{a}^{[i]}_{\mathbf{B}^{[i]}}$
  Receive information from neighbors
  Update rewards using eqs. (4)-(5)
  Compute a new basis, $\mathbf{B}^{[i]}$, using LexSimplex
  Assign j such that $x_{ij} \in \mathbf{x}_{\mathbf{B}^{[i]}}$ and $x_{ij} = 1$
end while

and targets was in $N = \{4, 8, 16\}$. Communication range was always 2 units, while the scenario size was a square of side length equal to N. Both robots and targets moved with a random linear velocity in the $0.1 \ldots 0.5$ units/second range and angular velocity in $\{-\pi/3 \ldots \pi/3\}$ rad/s, bouncing off the scenario limits. Fig. 1 shows three situations in the small simulation size. Each run lasts for 600 iterations of 0.1 seconds of simulated time. Each scenario combination was run a hundred times, preserving the random seeds across scenarios. Every simulated iteration corresponds too with one iteration of Algorithm 1.

The reason for the random movement of robots is that the objective of the simulations is to observe the evolution of the distributed assignment. In the same way, the reward function is not really critical; an exponentially decreasing reward was used, defined as

$$r_{ij}(t) = e^{-||\mathbf{p}_i(t) - \mathbf{q}_j(t)||} \tag{6}$$

where $\mathbf{p}_i$ and $\mathbf{q}_j$ are the poses of robots and targets.

Besides the effect of increasingly large N, three algorithm variants were evaluated: the regular algorithm, including the prediction described in Sec. IV-B (*predictive* in the figures), the same algorithm without the prediction (*not predictive*), and the distributed simplex from [16] (*static*), which does not consider dynamic rewards. We run this last algorithm in the following way: whenever the globally optimal assignment changes, we reset the baseline algorithm. As long as the optimal solution remains valid for enough iterations, the static algorithm will converge to it. In other words, for lack of a dynamic algorithm, a team could use a static one at fixed time intervals, hoping that solutions arrive faster than the ground truth changes.

Fig. 2a shows the percentage of assignments that are optimal at the current time, both globally and for individual robots. It is noteworthy that for large teams global distributed optimality is elusive; this is due to the short-livedness of the optimal assignments. Nonetheless, individual robots have for most of the time their globally optimal target assigned. Essentially, the solution is individually correct for most robots, except for locally transient assignments whenever the optimal solution changes.

Fig. 2b captures the convergence speed, in the sense of how many communication rounds are required to re-achieve the optimal solution when it changes. The values depicted are the quotient between non-optimal iterations and total ground truth changes of optimal assignment. Notice that the values can be optimistic when ground truth optimal assignments last for less iterations than required to converge, which will happen more as N increases.

Finally, Fig. 2c shows the number of local simplex pivoting operations in average that each robot locally performs per iteration. In other words, it may serve as a proxy of CPU load, which may be important in low-power devices. The benefits of the dynamic simplex are clear, as can be expected, besides its intrinsic anytime advantage that precludes the need to periodically replan of a static algorithm. However, the predictive rewards increase the pivots with N.

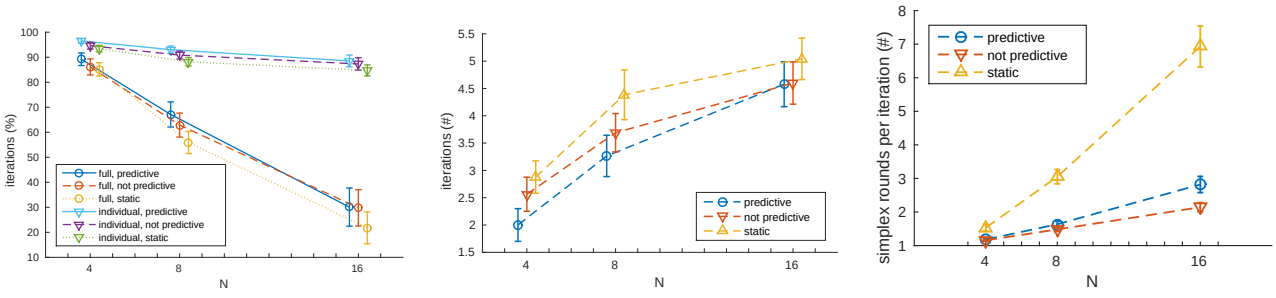
VI. Experiments

A. Setup

The experiment consisted in tracking 3 people moving freely using 3 cameras mounted on mobile robots distributed through a laboratory, as it is shown in Fig. 4[2]. The robots stayed in a fixed position, so that they could rotate on their own axes to point to the targets with their on-board cameras.

Since the computation of the rewards and the specific perception of the targets are not the core of this paper,

[2]We refer the reader to the accompanying video of this experiment for a complete visualization.



(a) (b) (c)

Fig. 2. Average results from simulations. (a) Percent of iterations where the distributed assignment is optimal. (b) Iterations in which the distributed solution is not optimal per ground truth count of optimal assignment changes. (c) Simplex operations per robot and iteration. The abscissa N values have been slightly shifted for non-overlapping visualization, although they always correspond to the closest labeled integer.

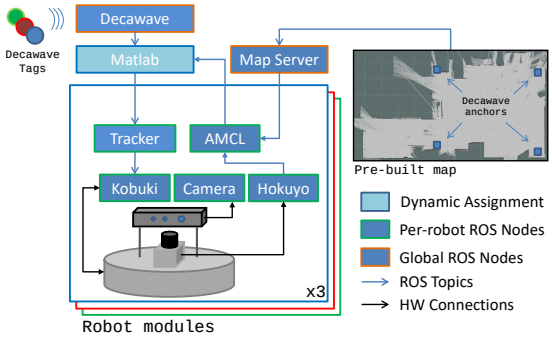


Fig. 3. Software architecture.

we opted for deriving them from the relative positions between the robots and the people. In particular, each robot was localized using the Adaptive Monte Carlo Localization (AMCL) algorithm and the people were localized using a DecaWave Indoor Location system.

From these positions, the rewards were computed using the same function as in the simulation results, decreasing exponentially with the distance between the robot and the target. This was carried out using the same Matlab implementation as in section V in which the distributed nature of the system is simulated by software. The individual assignments computed by our algorithm were fed back to each robot which, using the pose information, rotated its camera to put the assigned target in the center of its field of view.

The software architecture of the experiment is schematized in Fig. 3. From the bottom up, three ROS nodes —*Kobuki*, *Hokuyo* and *Camera* — manage the TurtleBot II platform equipped with an on-board Intel NUC Core i7 computer, a Hokuyo URG-04LX LiDAR sensor and a Microsoft Kinect RGBD sensors used as standard RGB camera for visualization purposes. The first two nodes provide odometry and laser readings to the *AMCL* node, which is in charge of localizing the robots on a pre-built map (obtained previously to the experiments) provided by the *Map Server* node.

The target positions were provided by the DecaWave Indoor Location system. It consists in a GPS-like system with two categories of devices: anchors and tags. The former must be positioned in advance in known locations, taking the place of satellites in the analogy. The tags are instead the mobile receptors. These compute their own position using a triangulation technique using the time-of-flight of Ultra Wide Band (UWB) frames exchanged with the anchors. These positions were acquired and published by the *Decawave* node, using one additional DecaWave device as listener.

Finally, the assignment computed in Matlab was published in ROS and used to provide goal orientations to the *Tracker* nodes that generate the angular velocities for the cameras.

B. Evaluation

Table I shows some of the same metrics that were used in simulations: team and individual optimality percent, and average iterations and wall time to convergence. Measurements were taken since the first person starts to move until the last one stops, lasting for 900 iterations and 100 seconds. The results in the table show that the algorithm is able to obtain both individually and globally the optimal solution above 90% of the time.

Optimality %		Convergence	
Team	Robot	Iterations	Time
91.8 %	95.7 %	2.0	0.22 s

TABLE I
EXPERIMENT METRICS

A practical analysis of the tracking quality shows that the real-world behavior of the algorithm is also satisfactory. Considering the field of vision (FOV) of the RGB camera of the Microsoft Kinect sensor used which is of about 62°, the targets have been visible during about 95.15% of the experiment. Figure 5 shows graphically this metric for one of the robots: the blue line shows the error of the system, obtained as the difference between the angular reference sent to the *Tracker* node and the actual orientation of the robot for the whole experiment. It is possible to observe that the graph only sporadically exceeds the dotted lines at ±31° (half of the FOV) coinciding with assignments that forced a notable orientation adjustment.

VII. Conclusion

This work presented a solution to the dynamic linear assignment problem with dynamic network topology applied

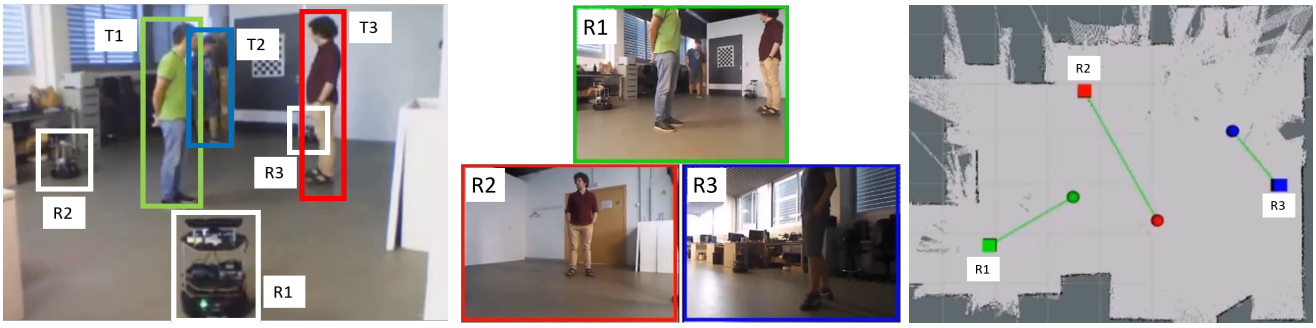


Fig. 4. Overview of the experiments. In the left figure there is a view of the experiment from an external camera with the three robots and the three targets. For a better interpretation, each target is assigned a different color. The middle figure shows the views of the three cameras. The target assigned to each camera is identified by the color of the outside rectangle. The right plot shows the visualization of the same instant in rviz, including links with the assignment.

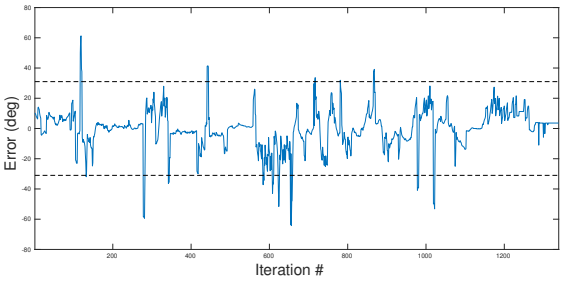


Fig. 5. Angular error of Robot #1 during one of the experiments. The dotted line represents the field of vision of the Kinect sensor used.

to a multi-robot tracking mission. We presented and tailored a distributed simplex algorithm which is able to consistently provide optimal assignments for the reward function chosen during most of the running time in the evaluated conditions. Whenever the optimal assignment changes, our solution converges in a fraction of the iterations that would be needed with a static solution approach, which in turn reduces computational load and sub-optimal assignment periods. Validation experiments with a team of three robots serve to put in context the time scales and demonstrate its applicability in a real setup. As a closing assessment, the algorithm has in our opinion many desirable properties for robotic teams: from the theoretical point of view, it is scalable, optimal, and exhibits fast optimal tracking capabilities. In regard to implementation, it requires only local asynchronous message broadcast, eliminating complex topology management, and it is not CPU-intensive during steady-state periods.

References

[1] A. R. Mosteo, E. Montijano, and D. Tardioli, "Optimal role and position assignment in multi-robot freely reachable formations," *Automatica*, vol. 81, pp. 305–313, 2017.

[2] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[3] D. Tardioli, A. R. Mosteo, L. Riazuelo, J. L. Villarroel, and L. Montano, "Enforcing network connectivity in robot team missions," *The International Journal of Robotics Research*, vol. 29, no. 4, pp. 460–480, 2010.

[4] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 133–149, 1990.

[5] R. Zivan, H. Yedidsion, S. Okamoto, R. Glinton, and K. Sycara, "Distributed constraint optimization for teams of mobile sensing agents," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 3, pp. 495–536, May 2015.

[6] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.

[7] M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *47th IEEE Conference on Decision and Control*, 2008, pp. 1212–1217.

[8] H. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, Aug 2009.

[9] L. Liu, N. Michael, and D. A. Shell, "Communication constrained task allocation with optimized local task swaps," *Autonomous Robots*, vol. 39, no. 3, pp. 429–444, Oct 2015.

[10] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, Aug 2017.

[11] B. P. Gerkey and M. J. Matari, "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] L. E. Parker, "Cooperative robotics for multi-target observation," *Intelligent Automation & Soft Computing*, vol. 5, no. 1, pp. 5–19, 1999.

[13] G. A. Korsah, A. T. Stentz, and M. B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27, July 2007.

[14] L. Liu and D. A. Shell, "Assessing optimal assignment under uncertainty: An interval-based algorithm," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 936–953, 2011.

[15] A. Farinelli, L. Iocchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Autonomous Robots*, vol. 41, no. 6, pp. 1321–1345, 2017.

[16] M. Burger, G. Notarstefano, F. Allgower, and F. Bullo, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[17] E. Montijano and A. R. Mosteo, "Efficient multi-robot formations using distributed optimization," in *IEEE 53th Conference on Decision and Control*, 2014, pp. 6167–6172.

[18] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006.

[19] E. Montijano, R. Aragues, and C. Sagues, "Distributed data association in robotic networks with cameras and limited communications," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1408–1423, Dec 2013.

[20] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski, "Lexicographic perturbation for multiparametric linear programming with applications to control," *Automatica*, vol. 43, no. 10, pp. 1808–1816, 2007.

[21] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[22] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.