An Asynchronous Distributed Constraint Optimization Approach to Multi-Robot Path Planning with Complex Constraints

ABSTRACT

Multi-robot teams can play a crucial role in many applications such as exploration, or search and rescue operations. One of the most important problems within the multi-robot context is path planning. This has been shown to be particularly challenging, as the team of robots must deal with additional constraints, e.g. inter-robot collision avoidance, while searching in a much larger action space. Previous works have proposed solutions to this problem, but they present two major drawbacks: (i) algorithms suffer from a high computational complexity, or (ii) algorithms require a communication link between any two robots within the system. This paper presents a method to solve this problem, which is both computationally efficient and only requires local communication between neighboring agents. We formulate the multi-robot path planning as a distributed constraint optimization problem. Specifically, in our approach the asynchronous distributed constraint optimization algorithm (Adopt) [15] is combined with sampling-based planners to obtain collision free paths, which allows us to take into account both kinematic and kinodynamic constraints of the individual robots. The paper analyzes the performance and scalability of the approach using simulations, and presents real experiments employing a team of several robots.

Keywords

multi-robot path planning, distributed constraint optimization, sampling-based planner

1. INTRODUCTION

In many applications, the use of a multi-robot team can offer clear advantages when compared to a single robot system (see, for instance, [7, 14]). The tasks that compose a given mission can be shared by the robots in the team, leading to more efficient solutions. For example, different robots could share local information, enhancing the global situation awareness; or a team of robots may be able to perform tasks that a single robot can not accomplish.

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

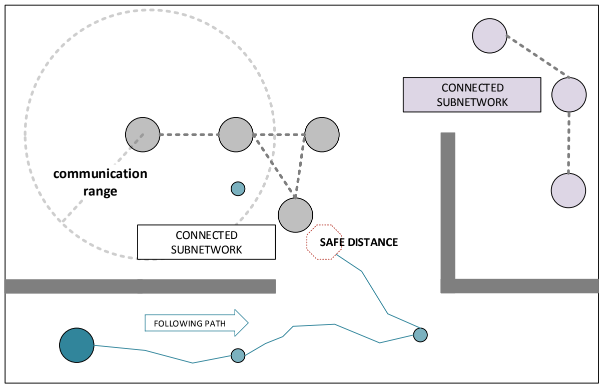


Figure 1: Example of an scenario where multiple robots aim to visit a set of points of interest – stations. On the one hand, robots that are close form a connected subnetwork to find a joint solution to move towards the next station. On the other hand, robots that are far plan their paths individually. This is realized with our proposed algorithm.

At the same time, the use of multiple robots in a mission presents unique challenges. In particular, it requires dealing with shared resources, such as space, time or communication bandwidth; and with additional constraints, such as network connectivity. These challenges need to be considered when determining which tasks and actions each robot has to perform during the mission. In this work, we focus on the space and time constraints that arise when a team of robots needs to plan a set of paths that minimize a global objective function while avoiding collisions between the robots. This can be formulated as a multi-robot path planning problem. Path planning for multiple robots is a common problem that arises in applications such as target tracking [5] or exploration [24]. This problem has been shown to be NP-hard [20]. In the following, we will review the most relevant algorithms that have been proposed in the literature to solve the multi-robot path planning problem.

Several approaches have been proposed in the control theory community to deal with the multi-robot path planning problem. Approaches such as dynamic programming [21], linear programming [2], mixed integer linear programming (MILP) [6], or decentralized model predictive control (DMPC) [9] have been shown to work well but they suffer from a high computational complexity. This makes such approaches intractable for large teams of robots, and for missions that are subject to complex constraints.

Prioritized planning techniques like sequential planning represent a good alternative from the computational perspective [23]. However, they are based on an arbitrary assignment of priorities that could incur in a loss of performance [22]. Van der Berg *et al.* solve this problem by proposing an online method to determine the robots' priorities based on local information [22]. Sequential planning algorithms require that agents need to wait for their neighbors' response each time they have to take a decision. This is in general not desirable as it would result in a performance degradation as the number of robots in the system grows. Consensus based approaches have been also considered in the literature [19, 16]. However, they are suited to sparse environments where only infrequent planning is required.

Efficient algorithms that assume a discretized representation of the environment and robot's state space have been recently proposed in the literature [27, 8, 26, 25]. Specifically, the algorithm in [25] serves us as inspiration for our approach. In [25], the authors employ the concept of subdimensional expansion; i.e. initially each of the robots plans a path individually, and then coordinates its motion with the other robots as needed. However, the previous works resort to discretization and graph-based path planning algorithms, which face computational complexity problems when dealing with robots with complex kinematic and kinodynamic constraints.

Sampling based path planning algorithms are natural candidates to account for arbitrary robot dynamics. Specifically, we are interested in Rapidly-exploring Random Trees (RRT) [11]. Several works for multi-robot cooperation have employed the RRT algorithm to plan the robots' paths [28, 18, 13]. However, none of them focus on the specific problem of path planning for multiple robots.

The work from Desaraju and How is the one that is closest to our proposed approach [3]. They employ the closed-loop RRT algorithm [10] in combination with a token based approach to solve the multi-robot path planning problem in a decentralized manner. However, the authors assume that the network that the agents conform is fully connected. In contrast, in this work, we propose an algorithm that only requires local communication between robots to solve the multi-robot path planning problem. Like in [25] our algorithm is based on a two steps approach: (i) first, the robots plan several paths individually using the RRT algorithm, (ii) second, the robots communicate with their neighbors as needed to find the set of paths that minimize an user-defined utility function. This second step is realized by formulating the problem as a distributed constrained optimization problem [12].

The remainder of this paper is organized as follows. Section II states formally the problem. Section III summarizes the distributed constraint optimization approach that we employ to achieve the multi-robot coordination. We describe in Section IV our distributed multi-robot path planning algorithm. Sections V and VI present the simulations and experiments performed, and is then followed by the conclusions.

2. PROBLEM STATEMENT

Let us consider a network of N robots. Each robot $i = 1, 2, ..., N$ aims to visit, in a predefined order, the set of stations $\mathcal{S}_i = \{\mathbf{s}_{i,1}, \mathbf{s}_{i,2}, ..., \mathbf{s}_{i,n_i}\}$ (for instance, a set of places to take samples), with n_i the number of stations that agent i aims to traverse. We define the path between two stations $\mathbf{s}_{i,j}$ and $\mathbf{s}_{i,l}$ as $\mathcal{P}_{\mathbf{s}_{i,j},\mathbf{s}_{i,l}}$ with $j = [1, 2, ..., n_i - 1]$ and $l = j + 1$. The set of paths that link the stations $\mathcal{S}_i$ is given by $\mathcal{P}_{\mathcal{S}_i} = [\mathcal{P}_{\mathbf{s}_{i,1},\mathbf{s}_{i,2}}, \mathcal{P}_{\mathbf{s}_{i,2},\mathbf{s}_{i,3}}, ..., \mathcal{P}_{\mathbf{s}_{i,n_i-1},\mathbf{s}_{i,n_i}}]$.

The goal of agent i is to find the set of paths $\mathcal{P}_{\mathcal{S}_i}$ that minimizes function $f(\mathcal{P}_{\mathcal{S}_1}, \mathcal{P}_{\mathcal{S}_2}, ..., \mathcal{P}_{\mathcal{S}_N})$. This function is a global function that encodes the inter-dependencies of all the agents that conform the network. In this work, we define the function $f(\cdot)$ as the total distance traveled by the robots while avoiding collisions. Let us remark that a large class of functions can be considered within the framework proposed in this paper, as we will describe in Section 3.

In this work, we consider the following constraints and simplifying assumptions:

1. Each robot's global position is known exactly and noise-free. We assume that there exists an external positioning system that provides us with a highly accurate localization, e.g., a Global Positioning System for outdoor scenarios.

2. The borders and obstacles that define the environment are *a priori* known.

3. Robots initially know neither about the presence nor about the location of other robots.

4. Robots can only communicate locally; i.e. two robots can communicate if they are neighbors. Here we consider two robots as neighbors if they are separated less than a distance r_c. This corresponds to a disc communication model.

5. Stations $\mathbf{s}_{i,j}, \mathbf{s}_{i,j+1}$ can only be separated a maximum distance $\frac{r_c}{2}$. This simplifying assumption allows us to easily cast our problem within the distributed constraint optimization algorithm that is described in Section 3. We discuss possible solutions to relax this assumption in Section 4.

We propose in this paper an algorithm that is able to solve the aforementioned problem for robots whose motion can be planned with a RRT-like algorithm [11]. This corresponds to a large class of robots that includes both kinematic and kinodynamic constraints.

3. ASYNCHRONOUS DISTRIBUTED CONSTRAINT OPTIMIZATION

Distributed constraint optimization problems (DCOP) have emerged as one of the most relevant frameworks to solve complex multi-robot coordination problems [12]. Specifically, in this work we propose the use of the asynchronous distributed constraint optimization algorithm (Adopt) [15] to tackle the multi-robot path planning problem described in Section 2. In contrast to the other algorithms reviewed in [12], Adopt provides theoretical guarantees on the global solution optimality while keeping communication between agents asynchronous and localized. This is our motivation to incorporate Adopt into our algorithm. The three key ideas behind Adopt are the following:

1. Decision making based on local information.

2. Efficient reconstruction of previously explored solutions. This speeds up the algorithm execution since previous solutions can be reused to meet the updated constraints imposed by the other robots.

3. Built-in termination detection. Once the algorithm converges, a termination message is sent through the network. We can exploit this feature to trigger the next step of our algorithm.

In Adopt, each robot can control a decision variable x_i that can take values from domain $\mathcal{D}_i = \{\mathcal{D}_{i,1}, \mathcal{D}_{i,2}, ..., \mathcal{D}_{i,k_i}\}$, with k_i the number of elements in the domain. Let us denote the set of robots – variables – for which we aim to solve the optimization problem as $\mathcal{V}$. The goal of the robots is to minimize the global objective function $g(\mathcal{D})$, where $\mathcal{D}$ denotes a possible assignment for the variables. For example, for a team of three robots $i = 2, 3, 6$ with identical domain size $k_i = 4$, a possible assignment for the variables could be $\mathcal{D} : \{x_2 : \mathcal{D}_{2,1};\ x_3 : \mathcal{D}_{3,4};\ x_6 : \mathcal{D}_{6,1}\}$. Adopt only allows us to solve problems that involve binary constraints between robots. We denote a constraint between robots i and j as $g_{i,j}(\mathcal{D}_{i,l}, \mathcal{D}_{j,m})$, with $l \in [1, 2, ..., k_i]$, $m \in [1, 2, ..., k_j]$. The goal is to find the optimal assignment $\mathcal{D}^*$ that minimizes the following function:

$$g(\mathcal{D}) = \sum_{x_i, x_j \in \mathcal{V}} g_{i,j}(\mathcal{D}_{i,l}, \mathcal{D}_{j,m}), \qquad (1)$$

in a distributed fashion, where each agent i is only in control of its own variable x_i. The robots participating in Adopt must conform a connected subnetwork; i.e. there must be a direct or indirect communication path between any two robots. Let us empathize that this framework allows us to formulate a large class of objective functions. For more details about the properties that equation (1) must meet, please refer to [15].

The Adopt algorithm takes as input a depth-first search (DFS) tree that encodes the constraints between different robots. The vertices of this tree represent the robots that aim to solve the optimization problem, and the edges define the constraints between robots. In this work, we have implemented the algorithm proposed in [1] to create the DFS tree.

The algorithm from [1] is fully distributed. However, one robot – the leader – must trigger the DFS tree creation. The leader will become then the root of the DFS tree. Here, for leader election we have implemented the YO-YO algorithm [17] because it is well suited to connected subnetworks of arbitrary topology, as it is the case in this work.

Let us point out that the proposed multi-robot path planning algorithm does not require that all robots belong to the same network. In contrast, it allows robots to be organized in several disconnected subnetworks. Specifically, within each subnetwork, all robots must be in direct or indirect communication as the Adopt algorithm requires. For example, in Figure 1 we show a system with 8 agents. One agent (green) is not in communication with the rest of the system. The remaining robots form two connected subnetworks with 4 (grey) and 3 (violet) robots. Notice that the two subnetworks are disconnected. Such a situation is handled with the proposed algorithm.

4. DISTRIBUTED MULTI-ROBOT PATH PLANNING

We describe in this section our proposed algorithm. First, we present an overview of the algorithm execution. To this end, we propose a state machine that controls the robot's behavior within a complex mission. Second, we describe in detail the elements that compose our algorithm for the particular case of a connected subnetwork.

4.1 Algorithm Overview

In this paper we propose an algorithm that allows multiple robots to jointly plan their motion in a distributed fashion. Instead of optimizing the complete function $f(\cdot)$ described in Section 2, we optimize the problem in a station to station basis; i.e. robots only cooperate to plan their paths between their current station and the next one. On the one hand, this problem relaxation allows us to keep the algorithm complexity bounded and independent of the number of stations. On the other hand, solving the problem for the complete set of stations is not necessarily optimal; imagine, for example, that the mission requires incorporating a new robot to the system. Then, the robots should replan their complete paths to consider the new constraints imposed by this new robot, which would make the optimization for the full set of stations useless.

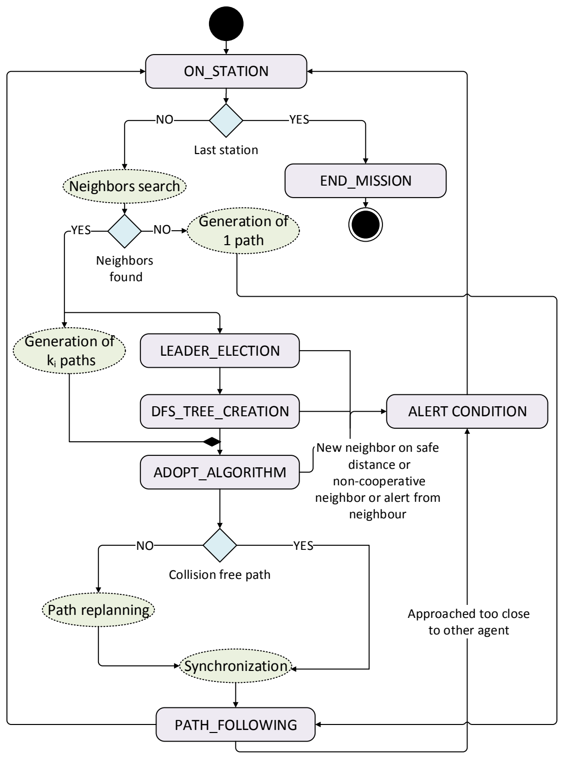


Figure 2: State machine that describes the algorithm's execution.

Figure 2 shows the state machine that controls the algorithm's execution. First, when a robot reaches an station, it will search for neighbors in the environment by sending an identification message. Second, neighboring robots will create connected subnetworks. Notice that, within a large environment, several isolated subnetworks may be created. Then, each subnetwork will execute the proposed multi-robot path planning algorithm described in Section 4.2. This is realized in two steps: First, the robots plan several paths individually towards the next station. To this end, we employ in this work the RRT algorithm [11]. Let us remark that this choice is motivated by the fact that, due to the random nature of the RRT, the algorithm delivers different solutions each time we run it, which is a requirement for our algorithm. Second, the robots cooperate to find the combination of paths that minimize a user-defined objective function while avoiding inter-agent collisions.

Each of the robots plans k_i paths to the next station. However, this does not guarantee the existence of a collision-free solution. Therefore, we introduce a replanning strategy to deal with this problem.

The last step of the algorithm for a connected subnetwork consists of a synchronization step that forces the robots to start at the same time. This is required to avoid collisions. The robots will continue the algorithm's execution until they reach the last station. The aforementioned procedures are explained in further detail in Section 4.2.

Let us also emphasize that in the first step of the algorithm the robot could find no neighbors; for instance, in large environments or for robots that have a small communication range. Then, the robot will plan its path individually and follow it till the next station. This is a natural approach since robots that are not in communication range can not cooperate to find a joint solution. However, it could happen that while the robots are moving they find another robots. Then, an alert condition will be triggered and robots involved will stop following their paths and will execute the second step of the algorithm; i.e. creation of a connected subnetwork.

4.2 Multi-Robot Path Planning for a Connected Subnetwork

This section explains in detail the procedures that the robots follow once they have created a connected subnetwork.

First, each of the robots will generate k_i different paths between their current position and their next station using the RRT algorithm. We denote the set of paths – domain – of robot i as $\mathcal{D}_i$. For example, the domain of robot 3 with domain size $k_3 = 3$, which aims to travel from station $\mathbf{s}_{3,4}$ to station $\mathbf{s}_{3,5}$ would be $\mathcal{D}_3 : \{\mathcal{D}_{3,1} = \mathcal{P}^{(1)}_{\mathbf{s}_{3,4},\mathbf{s}_{3,5}},\ \mathcal{D}_{3,2} = \mathcal{P}^{(2)}_{\mathbf{s}_{3,4},\mathbf{s}_{3,5}},\ \mathcal{D}_{3,3} = \mathcal{P}^{(3)}_{\mathbf{s}_{3,4},\mathbf{s}_{3,5}}\}$.

Second, the robots will elect a leader, create the DFS tree required by Adopt, and start the execution of Adopt. We employ Adopt to find the set of paths from the robots' domain that minimize a user-defined function while avoiding collision between robots. We consider that two paths $\mathcal{D}_{i,l}$ and $\mathcal{D}_{j,m}$ are in collision if at the same time instant they are separated a distance smaller than r_s, which we denote safety distance.

Specifically, in this work our goal is to minimize the total distance traveled by the robots; i.e. the algorithm will select the shortest paths from the ones "sampled" by the RRT. We denote the distance associated to path $\mathcal{D}_{i,l}$ as $\texttt{Dist}(\mathcal{D}_{i,l})$. Following the notation introduced in Section 3, we can define the constraint function $g_{i,j}(\cdot)$ between robots i and j for the collision-free case as follows:

$$g_{i,j}(\mathcal{D}_{i,l}, \mathcal{D}_{j,m}) = \frac{\texttt{Dist}(\mathcal{D}_{i,l})}{|\mathcal{N}_i|} + \frac{\texttt{Dist}(\mathcal{D}_{j,m})}{|\mathcal{N}_j|} \tag{2}$$

with $l \in [1, 2, ..., k_i], m \in [1, 2, ..., k_j]$, and $|\mathcal{N}_i|, |\mathcal{N}_j|$ the number of neighbors of robots i, j. In case there is a collision between the two paths, $g_{i,j}(\mathcal{D}_{i,l}, \mathcal{D}_{j,m})$ will take a value equal to infinite. Let us remark that this definition of the utility function allows the robot to find a collision-free solution that minimizes the total traveled distance for subnetworks of arbitrary topology. For example, let us consider a subnetwork of 3 robots where robot 1 is connected to 2, robot 2 is connected to 1 and 3, and robot 3 is connected to 2. This results in $|\mathcal{N}_1| = 1, |\mathcal{N}_2| = 2, |\mathcal{N}_3| = 1$. We can then easily check by substituting those values in equations (2) and (1) that function (1) is equal to the total distance traveled by all robots.

We have mentioned in the previous subsection that it is possible that there exists no solution given the domain proposed by the robots. Therefore, we have introduced a replanning step. This will be also triggered if Adopt does not converge before an user-defined time threshold. Since agents are already ordered in a DFS tree, we can exploit this hierarchy for the replanning process. This is triggered by the tree root, which will send its best path to all its descendants. Once a node receives the path proposed by its father it will calculate a path that does not collide with the ones proposed by its ancestors; i.e. all nodes that link it to the root. Specifically, we calculate the path using the RRT algorithm and considering the ancestors' paths as spatial-temporal obstacles. This process continues till we reach the robots at the leaves of the DFS tree. After planning their paths, they will send through the tree a message back to the root informing that the replanning process is completed. Once this message reaches the root, the Adopt algorithm terminates.

Finally, the robots execute the synchronization procedure. As we have previously mentioned, collisions between paths are checked both in the spatial and temporal dimension. On the one hand, the consideration of the temporal dimension reduces the number of potential collisions, as we are adding an additional dimension. On the other hand, this requires that the robots are synchronized; i.e. they must start following their paths at the same time instant. The root of the DFS tree is the one that will trigger the path following procedure. This message will be sent through the tree. Once a node receives this message it will start following its path. Since the message must travel through the network, we assume there will be a small delay between different nodes that will increase with the tree's depth. Let us remark that this can be taken into account by increasing the safety distance proportionally to the expected delay, which can be calculated given the communication protocol and the DFS tree topology.

In Section 2, we have introduced a simplifying assumption that limits the maximum distance between two stations; this corresponds to assumption 5. This is motivated by the fact that Adopt can only accept binary constraints. Then, by limiting the planning horizon to $\frac{r_c}{2}$ we can guarantee that robots that are not in communication range will never collide. This assumption could be easily removed by defining an intermediate station between the two considered stations. The robot would plan first to this intermediate station and then to the final one.

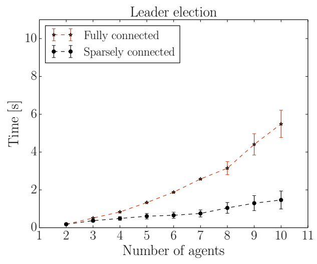
5. SIMULATIONS AND DISCUSSION OF RESULTS

We validate our proposed algorithm in two steps. First, we validate the algorithm for a connected subnetwork (Section 4.2) by performing Montecarlo simulations. Specifically, we evaluate the performance of the leader election, the DFS tree creation, and the distributed assignment of paths. Second, we illustrate with an example the algorithm's behaviour for a complex system – composed of several subnetworks – as described in Section 4.1. Here, and without loss of generality, we consider a holonomic robot in order to abstract the robot's motion from the algorithm's behavior. We carry out all simulations in a central computer but in a decentralized fashion using the robot operating system (ROS)[1]. This decentralization is possible because of the nature of ROS, where nodes run in different threads.

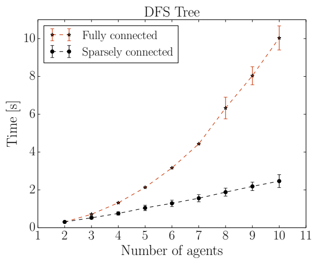
5.1 Leader Election and Depth-First Search Tree Creation

First we evaluate the convergence time of both the leader election and DFS tree creation algorithms as we increase the number of agents in the subnetwork. This is crucial to understand the scalability of the proposed algorithm.

We perform simulations for two types of subnetworks: (i) a fully connected subnetwork where all robots can communicate with each other; (ii) a sparsely connected subnetwork where each robot has a maximum number of 5 neighbors. This last one corresponds to a more realistic scenario that we could encounter in a mission that takes place within a large environment. For each of the simulated number of agents, we repeat the simulation 100 times. Let us add that for the sparsely connected case a new random network is generated each time. Figure 3 shows the resulting average and variance for the different scenarios.



(a) Leader election.



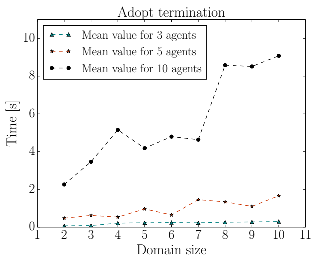
(b) DFS tree creation.

Figure 3: Leader election and DFS tree creation. Fully and sparsely connected subnetworks.

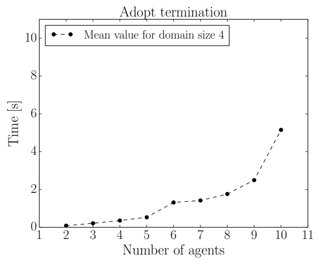
We can conclude that the complexity of both algorithms grow exponentially for a fully connected subnetwork. However, the complexity is linear for a sparsely connected subnetwork, which makes both algorithms suitable for real world missions.

5.2 Distributed Assignment of Paths

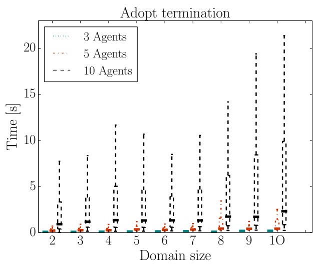
Second, we analyze the performance of the Adopt algorithm as we vary the number of robots in the subnetwork, and the number of paths in the robot's domain. In both cases, we consider a sparse subnetwork with the same properties as in the previous subsection. We repeat each of the simulations 100 times.



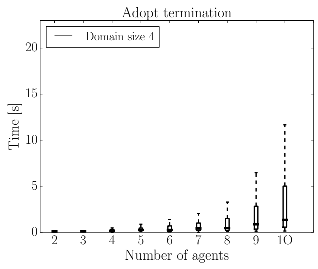
(a) Average performance with the number of paths in the domain.



(b) Average performance with the number of agentsk.



(c) Performance with the number of paths in the domain.



(d) Performance with the number of agents.

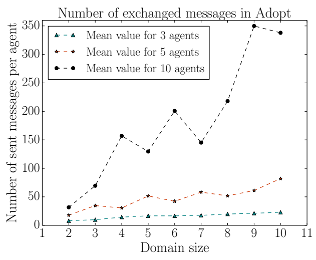
Figure 4: Distributed Assignment of Paths. (a,c) Performance with the number of paths in the domain for a network with 3, 5 and 10 agents. (b,d) Performance with the number of agents given a fixed domain composed of 4 paths.

Figures 4a,4c and 5a,5c show the time and the total number of exchanged messages that the Adopt algorithm required to converge to the optimal solution. We present the average and a box plot representation of the data. These simulations were carried out for a domain size – number of paths – ranging between 2 and 10, and we considered $3, 5$ and 10 robots. We observe that for a small number of robots the algorithm's complexity remains quasi constant respect to the domain size. However, for a large number of robots (10) the performance increases linearly.

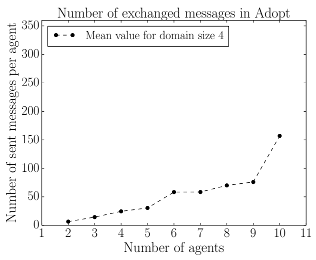
The explanation for such behavior can be understood by analyzing Figures 4b,4d and 5b,5d. Here we show the algorithm's complexity as we vary the number of robots given a fixed domain composed of 4 paths. We can confirm that the algorithm's complexity grows exponentially as the theoretical analysis of the algorithm points out [15]. Moreover, the box plot representation confirms that the algorithm's complexity is highly dependent of the network topology. This is one of the main reasons why we have introduced the replanning process in our algorithm; if the Adopt takes too long to converge, the replanning method will be triggered to find a feasible solution.

Attending to results we can conclude that: (i) the use of Adopt within a small subnetwork results in a low computational complexity, and (ii) for a large subnetwork we should either consider alternative algorithms [12] or introduce additional constraints into the algorithm's design to avoid the creation of such large networks.

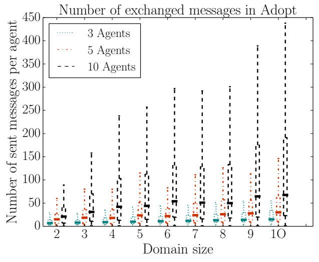
[1]ROS - Robot Operating System. http://wiki.ros.org/.



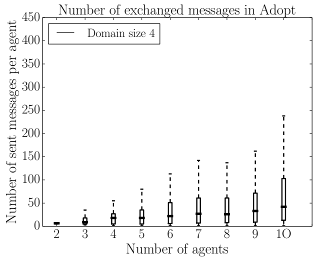
(a) Average number of exchanged messages with the number of paths in the domain.



(b) Average number of exchanged messages with the number of agents.



(c) Exchanged messages with the number of paths in the domain.



(d) Exchanged messages with the number of agents.

Figure 5: Distributed Assignment of Paths. (a,c) Number of exchanged messages with the number of paths in the domain for a network with 3, 5 and 10 agents. (b,d) Number of exchanged messages with the number of agents given a fixed domain composed of 4 paths.

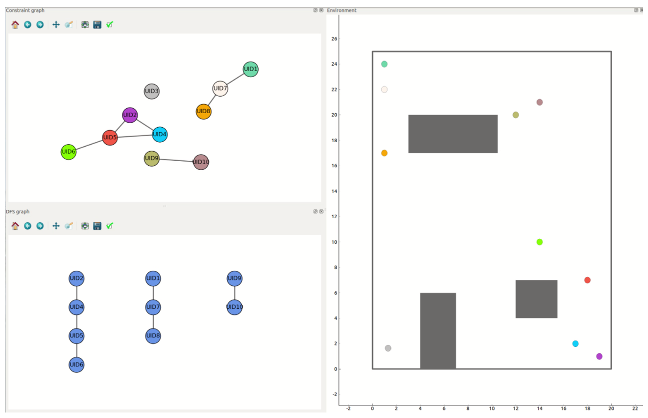
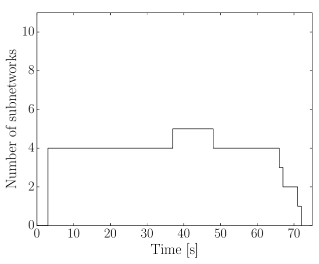
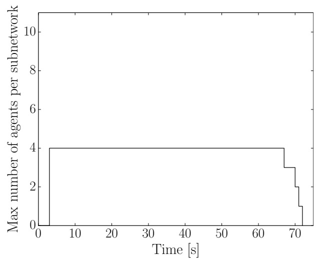


Figure 6: Agents' initial configuration. We show the communication network, DFS tree of each of the subnetworks, and agents' location in the environment.



(a) Number of subnetworks.



(b) Maximum number of agents per subnetwork.

Figure 7: Number of subnetworks and maximum number of agents per subnetwork for a typical scenario with 10 agents.

5.3 Multi-Robot Path Planning for a Complex System

Finally, we show one example of the whole system's behavior for a typical scenario with 10 agents. In a typical situation, given a limited communication radius, the agents' communication topology is divided into several subgraphs (subnetworks), and agents can not communicate with all the rest of the agents that compose the system. In this case, agents will execute the algorithm described in Section 4.1.

We show in Figure 6 the initial configuration of the robots. In concrete, we depict: (i) resulting communication network, (ii) DFS tree that corresponds to each of the subnetworks, and (iii) agents' location in the environment.

Given the agents' configuration depicted in Figure 6, we assign several stations to each of the agents and then simulate the agents' behavior as they try to reach the stations. We show in Figure 7 the evolution with time of the number of subnetworks and maximum number of agents per subnetwork that results as the agents move. We observe that the maximum number of agent's per subnetwork is 4, although we have a system of 10 agents. This property allows us to keep the number of agents per subnetwork low and therefore reduce the computational load of each of the agents. Attending to the results, we can conclude that the algorithm is scalable with the number of agents as in a real situation agents will be in general sparsely distributed on the environment as shown in Figures 1 and 6.

6. EXPERIMENTS AND DISCUSSION OF RESULTS

We also validate the proposed algorithm in an experiment with three holonomic robots (see Figure 8). The robots are a modified version of the commercially available Slider platform by Commonplace Robotics. Due to its four mecanum wheels, the platform is able to perform omnidirectional movements following input commands for forward, lateral and rotational velocities. Let us remark that more complex motion models could be considered within this framework. However, we choose a holonomic robot to abstract the robot's motion from the algorithm's capabilities. Like we did for the simulations, here we also run the algorithm in a central computer in a decentralized fashion, and then we send the corresponding waypoints to the robot using ROS with a WiFi connection. Each robot is equipped with a Raspberry Pi that runs the robot's controller to guide the robot to the desired position. We employ a commercial motion capture system (Vicon) to provide positioning information to the robots.

We assume the robots initially conform a connected network. Agents one and two must visit three stations ($\mathbf{s}_{i,2}$ to $\mathbf{s}_{i,4}$, $i = 1, 2$), while agent three must only visit two ($\mathbf{s}_{3,2}$ and $\mathbf{s}_{3,3}$) from their initial locations ($\mathbf{s}_{i,1}$, $i = 1, 2, 3$). We set a domain size of two for each of the agents; i.e. each agent will propose to its neighbors two possible paths to travel between stations.

Tables 1,2,3 show a summary of the algorithm's execution

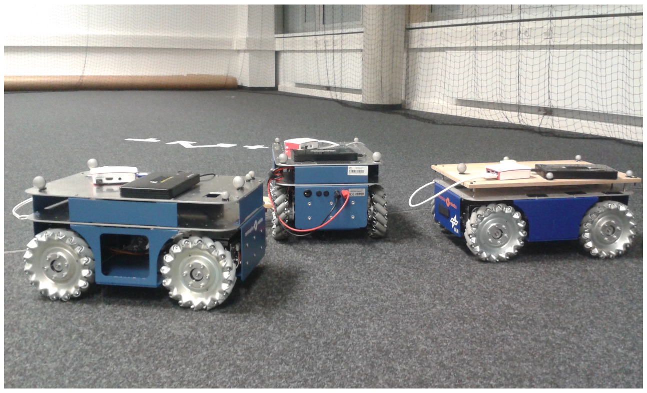


Figure 8: The three holonomic robots employed to carry out the experimental validation of the proposed algorithm.

	$\mathbf{s}_{i,1} \rightarrow \mathbf{s}_{i,2}$	$\mathbf{s}_{i,2} \rightarrow \mathbf{s}_{i,3}$	$\mathbf{s}_{i,3} \rightarrow \mathbf{s}_{i,4}$
$\lvert\mathcal{N}_i\rvert$	2	1	0
$t_p[s]$	0.06	0.18	0.17
$t_l[s]$	0.42	0.12	-
$t_d[s]$	0.74	0.31	-
$\texttt{Dis}[s]$	$\texttt{Dis}(\mathcal{D}_{i,1}) = 6.87$ $\texttt{Dis}(\mathcal{D}_{i,2}) : 10.51$	$\texttt{Dis}(\mathcal{D}_{i,1}) = 5.01$ $\texttt{Dis}(\mathcal{D}_{i,2}) = 5.92$	-
$t_a[s]$	0.58	0.9	-
$\mathcal{D}_i^*$	$\mathcal{D}_{i,1}$	$\mathcal{D}_{i,1}$	-

Table 1: Summary of algorithm execution of agent $i = 1$.

	$\mathbf{s}_{i,1} \rightarrow \mathbf{s}_{i,2}$	$\mathbf{s}_{i,2} \rightarrow \mathbf{s}_{i,3}$	$\mathbf{s}_{i,3} \rightarrow \mathbf{s}_{i,4}$
$\lvert\mathcal{N}_i\rvert$	2	1	0
$t_p[s]$	0.16	0.47	0.12
$t_l[s]$	0.51	0.03	-
$t_d[s]$	0.43	0.01	-
$\texttt{Dis}[s]$	$\texttt{Dis}(\mathcal{D}_{i,1}) = 7.91$ $\texttt{Dis}(\mathcal{D}_{i,2}) = 10.33$	$\texttt{Dis}(\mathcal{D}_{i,1}) = 1.94$ $\texttt{Dis}(\mathcal{D}_{i,2}) = 1.1$	-
$t_a[s]$	0.58	0.8	-
$\mathcal{D}_i^*$	$\mathcal{D}_{i,1}$	$\mathcal{D}_{i,2}$	-

Table 2: Summary of algorithm execution of agent $i = 2$.

	$\mathbf{s}_{i,1} \rightarrow \mathbf{s}_{i,2}$	$\mathbf{s}_{i,2} \rightarrow \mathbf{s}_{i,3}$
$\lvert\mathcal{N}_i\rvert$	2	0
$t_p[s]$	0.31	0.13
$t_l[s]$	0.5	-
$t_d[s]$	0.21	-
$\texttt{Dis}[s]$	$\texttt{Dis}(\mathcal{D}_{i,1}) = 3.35$ $\texttt{Dis}(\mathcal{D}_{i,2}) = 4.52$	- -
$t_a[s]$	0.58	-
$\mathcal{D}_i^*$	$\mathcal{D}_{i,1}$	-

Table 3: Summary of algorithm execution of agent $i = 3$.

for the three robots. We use the following notation: $|\mathcal{N}_i|$ is the number of neighbors of robot i at the origin station; t_p, t_l, t_d, t_a is the time to calculate the path, leader election, DFS tree, and Adopt, respectively; `Dis` is the path length, calculated in seconds, needed to traverse the path towards the goal station; $\mathcal{D}_i^*$ is the choice of agent i resulting from Adopt. According to the results, we can conclude that the robots were able to perform the assigned tasks. In addition, the cooperation between robots took, in the worst case, less than 2 seconds.

A video that includes the experiment together with a simulation showing the algorihtm's behaviour can be found under https://vimeo.com/184840217.

7. CONCLUSIONS AND FUTURE WORK

This paper has presented a novel approach for multi-robot path planning. We propose an algorithm that formulates the problem as a distributed constraint optimization problem (DCOP). The algorithm works as follows: first, the robots that can communicate through the network plan different paths towards their next stations, and then they employ the DCOP solver Adopt to determine the best assignments of paths among the robots that compose the team. This assignment is made according to a user-defined objective function. Specifically, in this paper, we seek to optimize the total traveled distance, and the main constraint is collision avoidance between robots. However the proposed method is more general; i.e. it could be used to optimize alternative utility functions and to incorporate additional constraints.

The experiments illustrate how our approach is able to obtain the best solution in small teams of robots, which is also corroborated by the simulations. At the same time, the simulations show that, while Adopt is a distributed algorithm, the number of messages and the time to converge to a solution grows more than linearly with the number of robots involved in the optimization process.

As future work we plan to apply our approach to the informative path planning problem for exploration tasks [24]. Moreover, we aim to compare the results of the Adopt solution with other approaches for multi-robot coordination, like the max-sum algorithm [4].

8. REFERENCES

[1] B. Awerbuch. A new distributed depth-first-search algorithm. *Information Processing Letters*, 20(3):147–150, 1985.

[2] N. Ayanian and V. Kumar. Decentralized feedback controllers for multiagent teams in environments with obstacles. *IEEE Transactions on Robotics*, 26(5):878–887, 2010.

[3] V. R. Desaraju, J. P. How, V. R. Desaraju, and J. P. How. Decentralized path planning for multi-agent teams with complex constraints. *Auton Robot*, 32:385–403, 2012.

[4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 639–646. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[5] S. K. Gan, R. Fitch, and S. Sukkarieh. Online decentralized information gathering with spatial-temporal constraints. *Autonomous Robots*, 37(1):1–25, 2014.

[6] D. Habib, H. Jamal, and S. A. Khan. Employing multiple unmanned aerial vehicles for co-operative path planning. *International Journal of Advanced Robotic Systems*, 10:1–10, 2013.

[7] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung, D. F. Wolf, G. S. Sukhatme, and D. C. MacKenzie. Adaptive teams of autonomous aerial and ground robots for situational awareness. *Journal of Field Robotics*, 24:991–1014, 2007.

[8] K. Kim, J. Campbell, W. Duong, Y. Zhang, and G. Fainekos. DisCoF + : Asynchronous DisCoF with Flexible Decoupling for Cooperative Pathfinding in Distributed Systems. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 369—-376. IEEE, 2015.

[9] Y. Kuwata and J. P. How. Cooperative distributed robust trajectory optimization using receding horizon MILP. *IEEE Transactions on Control Systems Technology*, 19(2):423–431, 2011.

[10] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. P. How. Motion planning in complex environments using closed-loop prediction. In *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit*, 2008.

[11] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[12] A. R. Leite, F. Enembreck, and J.-P. A. Barthès. Distributed constraint optimization problems: Review and perspectives. *Expert Systems with Applications*, 41(11):5139–5157, 2014.

[13] D. Levine, B. Luders, and J. How. Information-Theoretic Motion Planning for Constrained Sensor Networks. *Journal of Aerospace Information Systems*, 10(10):476—-496, 2013.

[14] I. Maza, F. Caballero, J. Capitan, J. M. de Dios, and A. Ollero. A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities. *Journal of Field Robotics*, 28(3):303–328, 2011.

[15] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180, 2005.

[16] O. Purwin, R. D'Andrea, and J. W. Lee. Theory and implementation of path planning by negotiation for decentralized agents. *Robotics and Autonomous Systems*, 56(5):422–436, 2008.

[17] N. Santoro. *Design and analysis of distributed algorithms*, volume 56. John Wiley & Sons, 2006.

[18] Z. G. Saribatur, E. Erdem, and V. Patoglu. Cognitive factories with multiple teams of heterogeneous robots: Hybrid reasoning for optimal feasible global plans. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):2923–2930, 2014.

[19] P. Scerri, S. Owens, B. Yu, and K. Sycara. A decentralized approach to space deconfliction. *FUSION 2007 - 2007 10th International Conference on Information Fusion*, 2007.

[20] P. Surynek. An Optimization Variant of Multi-Robot Path Planning is Intractable. *AAAI*, pages 1261–1263, 2010.

[21] J. Swigart and S. Lall. An explicit state space solution for a decentralized two-player optimal linear-quadratic regulator. *Proc.\ American Control Conference (ACC.2010)*, 1:6385–6390, 2010.

[22] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. *Robotics: Science and Systems V*, 2009.

[23] P. Velagapudi, K. Sycara, and P. Scerri. Decentralized prioritized planning in large multirobot teams. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 4603–4609, 2010.

[24] A. Viseras Ruiz, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino. Decentralized multi-agent exploration with online-learning of gaussian processes. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016.

[25] G. Wagner and H. Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[26] C. Wei, K. V. Hindriks, and C. M. Jonker. Multi-robot cooperative pathfinding: A decentralized approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8481 LNAI(PART 1):21–31, 2014.

[27] Y. Zhang, K. Kim, and G. Fainekos. *Discof: Cooperative pathfinding in distributed systems with limited sensing and communication range.* 2016.

[28] Z. Ziyang, G. Chen, Z. Qiannan, and D. Ruyi. Cooperative Path Planning for Multiple UAVs Formation. *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, 210016:469–473, 2014.