Decentralized Multi-Robot Task Allocation with Time Window and Ordering Constraints

Elina Suslova and Pooyan Fazli

***Abstract*— The multi-robot task allocation problem comprises task assignment, coalition formation, task scheduling, and routing. We propose a distributed constraint optimization formalism to allocate tasks to a team of robots. The tasks have time window and ordering constraints. Each robot creates a simple temporal network to maintain the tasks in its schedule. The proposed algorithm forms efficient coalitions among robots to accomplish the tasks more efficiently as a result of their collective abilities. We conduct extensive experiments to assess the performance of the proposed algorithm and compare it against a benchmark auction-based approach. The results show that our algorithm completes more tasks in 100% of the settings, increases the task completion rate and task completion frequency by 3.3% and 5.4%, respectively, and reduces the task execution time by 37.7% on average.**

I. INTRODUCTION

Multi-robot systems provide robust, flexible, and efficient solutions for tackling real-world applications, such as search and rescue [1], patrol and monitoring [2], and distributed servicing tasks [3]. Multi-robot task allocation (MRTA) is a challenging problem that involves task allocation, coalition formation, task scheduling, and routing. MRTA aims to recruit the best single or multiple robots to accomplish the tasks while maximizing performance. In variants of the MRTA problem, limitations are imposed on both robots and tasks, such as capability; capacity; and temporal, spatial, hard, and soft constraints. Constraints reduce the possible solution set for the task allocation problem. The goal of this work is to address the MRTA problem with time window and ordering constraints on the tasks.

Tasks and robots are spatially distributed in the environment. In order to complete as many tasks as possible while satisfying their time window and ordering constraints, each robot must be provided with a schedule to complete the maximum number of tasks in the minimum time. Moreover, by forming teams or coalitions, robots can achieve such tasks more efficiently as a result of their collective abilities.

There are various reasons for forming coalitions among robots to complete the tasks [1]. First, the task workload may be high, so a single robot may not be able to perform the given task within the specified time window. Second, coalition among robots leads to faster completion of tasks; therefore, the robots will have enough time to attempt other tasks in the environment. Third, the travel distance to the task may be too long for a single robot to reach the task in time and complete it individually within the specified time window. Hence, it is critical that the processes of coalition formation are managed effectively among robots.

Elina Suslova and Pooyan Fazli are with the Department of Computer Science, San Francisco State University, San Francisco, CA 94132, USA {esuslova,pooyan}@sfsu.edu

The distributed constraint optimization problem (DCOP) is a powerful framework for modeling many real-world problems involving collaborative multi-agent systems [4]. In DCOP, agents coordinate through local communication to choose values in a decentralized manner that optimize the team's global objective function. Despite the vast literature on DCOP, we are unaware of any work that has extended these models to handle tasks with time window and ordering constraints. Thus, the contributions of the paper are as follows:

- We present the first DCOP formulation of the multi-robot task allocation problem with time window and ordering constraints.
- The proposed DCOP framework facilitates coalition formation among robots to maximize the task completion rate and frequency while minimizing the average task execution time.
- We empirically evaluate the performance of the proposed DCOP framework against a benchmark auction-based method and show that our algorithm completes more tasks in 100% of the settings, increases the task completion rate and task completion frequency by 3.3% and 5.4%, respectively, and reduces the task execution time by 37.7% on average.

II. BACKGROUND AND STATE OF THE ART

Centralized approaches [5] are able to find optimal solutions for the MRTA problem. However, MRTA is an NP-hard problem, and as the number of robots or tasks increases, the problem becomes intractable. For this reason, approximated or greedy methods [6] are used extensively to solve the MRTA problem. Centralized methods suffer from a single point of failure, poor scalability, and need to generate a new global solution every time the attributes of the environment, tasks, and robots change. On the other hand, decentralized approaches are more robust to unreliable communication and robot failures and can repair the solution locally, however, they do not guarantee optimality. Below, we will discuss the two common classes of decentralized methods addressing the MRTA problem.

A. Market-based Approaches

Market- and auction-based approaches [7] are popular methods for distributed task allocation in multi-robot systems. An auctioneer announces a task, and each robot uses its local information to compute a bid, which is an estimate

of the robot's expected cost or utility of performing the task. The auctioneer collects the bids and selects the winning robot that will be responsible for executing the task. McIntire et al. [8] introduced an iterated sequential single-item auction algorithm to allocate tasks with ordering constraints. Nunes et al. [9] extended the previous work to accommodate tasks with both time window and ordering constraints. Despite the vast literature on market-based approaches, we are unaware of any work that has extended these methods to integrate coalition formation to handle the task allocation problem with time window and ordering constraints.

B. DCOP-based Approaches

DCOP algorithms rely on local message passing between robots to find solutions but do not, to our knowledge, currently handle tasks with time window and ordering constraints. Farinelli et al. [10] used the DCOP framework and the max-sum algorithm to coordinate low-power embedded devices in a decentralized manner. Ramchurn et al. [1] used DCOP for task allocation in the search and rescue domain. They introduced fast max-sum, which is a more efficient and robust variation of the max-sum algorithm, to solve the task allocation problem and facilitate coalition formation among various types of agents: ambulance, fire brigade, and police agents. As with a real-world disaster scenario, tasks have deadlines and are spatially distributed, and agents must synchronize their arrival time at victim locations. In addition, previous work [11] improved the DCOP model to reduce the communication overhead among the agents in the rescue domain.

III. PROBLEM STATEMENT

Let $\mathcal{R} = \{r_1, r_2, ..., r_{|\mathcal{R}|}\}$ be a finite set of robots and $\mathcal{K} = \{k_1, k_2, ..., k_{|\mathcal{K}|}\}$ be a finite set of tasks with time window and ordering constraints. The time window constraint specifies the time interval within which a task needs to be performed. In particular, the time window determines the earliest start time (EST) and the latest finish time (LFT) of the task. For example, suppose a task must be performed between 9:00 am and 2:00 pm. Each task may have one or more predecessor and successor tasks. More precisely, executing a task is possible only if all the predecessor tasks are completed first. $k_i \prec k_j$ denotes that k_i precedes k_j, or k_i must be completed before any robot can start executing k_j. Each task has a location (L) associated with it, and a robot must be present at the location to execute the task. The initial locations of robots and tasks are chosen randomly on the map. Moreover, each task has a duration (D) and a type (τ). Each robot can perform only one task at a time, and robots are heterogeneous in the sense that each robot can execute only predefined types of tasks. Robots can perform each task individually or form teams or coalitions. When multiple robots $\mathcal{C}_{k_i} \subseteq \mathcal{R}$ collaborate on one task k_i, the task duration D_{k_i} is divided by the number of robots, $\frac{\mathrm{D}_{k_i}}{|\mathcal{C}_{k_i}|}$, and robots in the coalition contribute to the task equally. Furthermore, robots in a coalition do not have to work on a task simultaneously; they are allowed to perform the task partially and then move to the next allocated task, leaving the rest to other robots in the coalition.

In brief, there are tasks in the environment whose locations, durations, types, time windows, and orders are given to the set of robots. Each robot's duty is to go to its allocated task's location, perform the task completely or partially, and then move to the next assigned task on the map. Tasks are allocated and scheduled before the execution is started.

In order to assess the performance of the algorithms, we define three evaluation metrics: 1) task completion rate (TCR), 2) task completion frequency (TCF), and 3) average task execution time (ATET). The evaluation metrics are defined as below:

1) $\mathrm{TCR} = \frac{|\hat{\mathcal{K}}|}{|\mathcal{K}|} \times 100,$
2) $\mathrm{TCF} = \frac{|\hat{\mathcal{K}}|}{m},$
3) $\mathrm{ATET} = \frac{1}{|\mathcal{K}|} \sum_{i=1}^{|\mathcal{K}|} \mathrm{ET}_{k_i},$

where $|\hat{\mathcal{K}}|$ is the number of completed tasks, $|\mathcal{K}|$ is the total number of tasks, m or *makespan* is the latest finish time of the last task, and ET_{k_i} is the execution time of task k_i. We believe that TCF is a better metric than makespan to assess the performance in the MRTA problem because different solutions do not always complete an equal number of tasks. One algorithm may schedule and complete more tasks, resulting in a higher makespan, and another algorithm may complete fewer tasks, resulting in a lower makespan. The goal is to maximize the task completion rate (TCR) and task completion frequency (TCF) and minimize the average task execution time (ATET).

IV. PROPOSED APPROACH

We present an approach to form coalitions of robots to accomplish a set of tasks that are constrained by time windows and have dependencies with other tasks. The overall scheme of the approach is shown in Figures 1–4.

A. Precedence Graph

Tasks $\mathcal{K}$ are given to the set of robots $\mathcal{R}$ through a precedence graph. The precedence graph $\mathcal{G}_{\mathcal{P}} = (\mathcal{K}, \mathcal{E}_P)$ is a directed acyclic graph (DAG) with nodes $\mathcal{K}$ corresponding to tasks and edges $\mathcal{E}_P$ representing the ordering constraints between the tasks. A directed edge $e_{ij} \in \mathcal{E}_P$ indicates that task k_i should be completed before performing task k_j or in other words $k_i \prec k_j$. Figure 1 shows a sample precedence graph with eight tasks. The precedence graph is divided into layers, such that there is no ordering constraint between the tasks in each layer. Hence, each layer contains a set of tasks that can be executed independently. Initially, tasks in the first layer are allocated through the DCOP formulation detailed in the next section. After assigning the tasks in the first layer, they are removed from the precedence graph, and the tasks in the next layer are allocated. This process continues until all the tasks in the precedence graph are assigned.

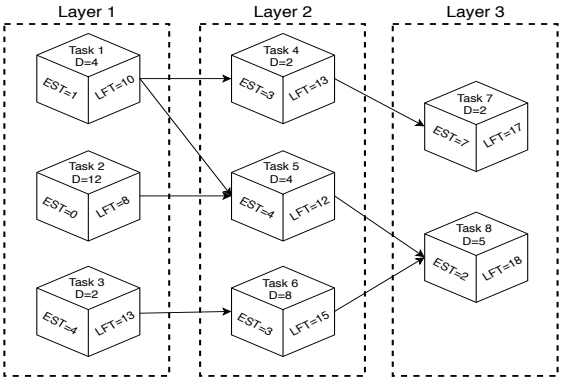

Fig. 1: Precedence graph. Each box represents a task with ID, earliest start time (EST), latest finish time (LFT), and duration (D). Edges show the ordering constraints.

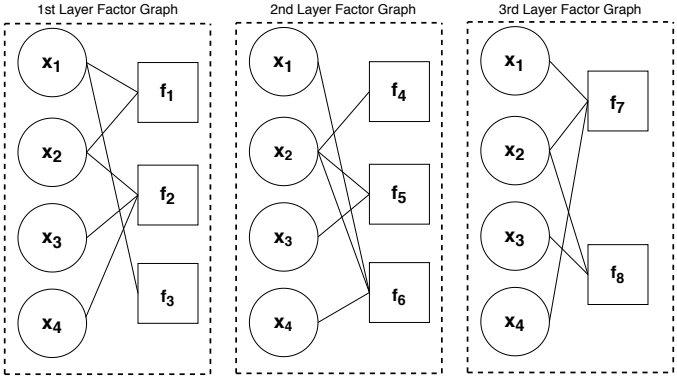

Fig. 2: Factor graphs. The variable nodes $\mathcal{X}$ represent the robots $\mathcal{R}$ (shown by circles), and the function nodes $\mathcal{F}$ represent the tasks $\mathcal{K}$ (shown by squares).

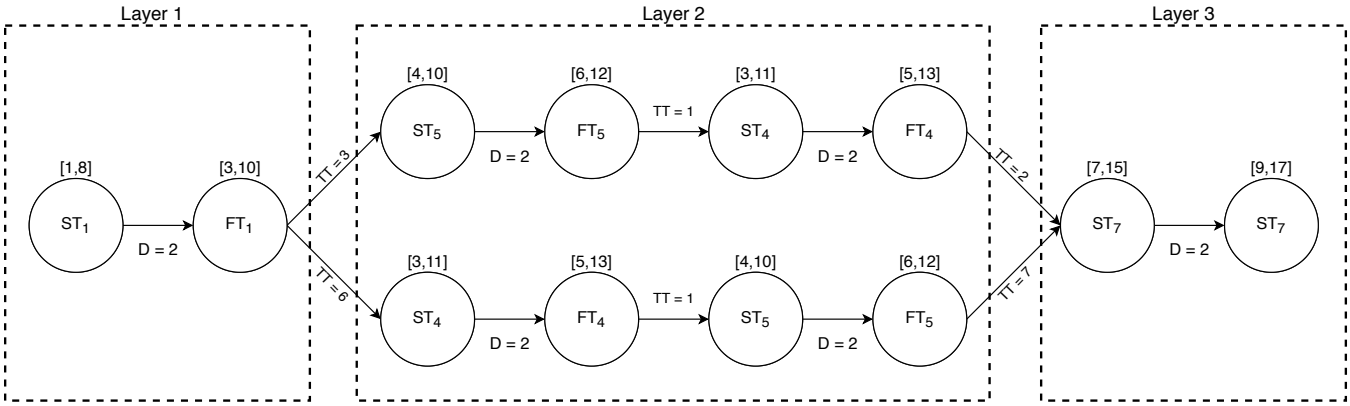

Fig. 3: The simple temporal network (STN) formation for robot 2. The STN contains the time points (ST: start time, FT: finish time), duration (D), travel time (TT), and time window and ordering constraints of the four tasks assigned to robot 2. Task 1 is executed, and then there is a choice of completing task 4 or task 5 first. Task 7 is done at the end of the schedule.

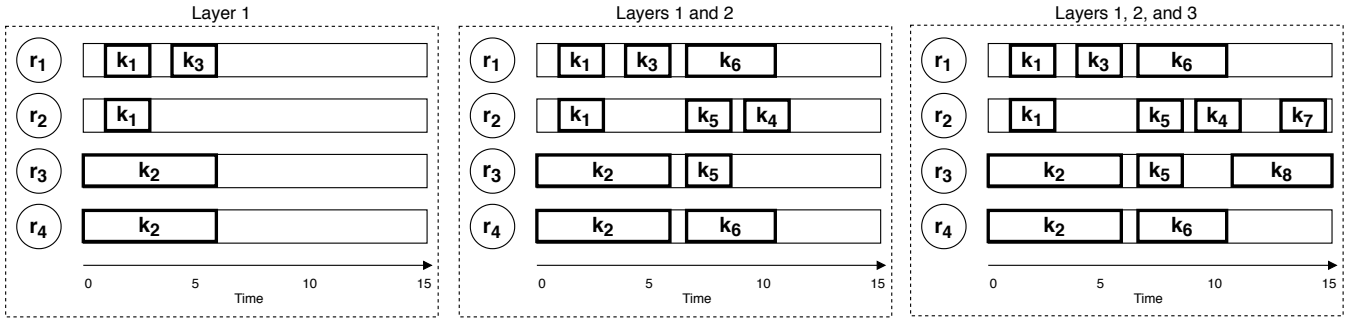

Fig. 4: Final schedules for robot 1 to 4. $\mathcal{R}$ is the set of robots, and $\mathcal{K}$ is the set of tasks.

B. Distributed Constraint Optimization Problem

Distributed constraint optimization (DCOP) provides a powerful framework to model multi-robot collaboration and coordination problems. DCOP generalizes the distributed constraint satisfaction problem (DisCSP). In DisCSP, the constraints are all hard, meaning that the solution must satisfy all of them. However, real-world problems often contain soft constraints too, which need to be satisfied as much as possible. DCOP handles both types of constraints.

A distributed constraint optimization problem (DCOP) is formally defined by a tuple $\langle \mathcal{R}, \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, where:

- $\mathcal{R} = \{r_1, \ldots, r_{|\mathcal{R}|}\}$ is a finite set of robots.
- $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$ is a finite set of variables, where $|\mathcal{X}| = |\mathcal{R}|$ in our problem.
- $\mathcal{D} = \{\mathcal{D}_{x_1}, \ldots, \mathcal{D}_{x_{|\mathcal{X}|}}\}$ is a set of finite domains for the variables in $\mathcal{X}$, with $\mathcal{D}_{x_i}$ being the domain of variable x_i.
- $\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{F}|}\}$ is a finite set of cost functions, where $|\mathcal{F}| = |\mathcal{K}|$ in our problem. Each cost function is defined over a set of variables: $f_i : \prod_{x \in \mathcal{X}_{f_i}} \mathcal{D}_x \rightarrow$

$\mathbb{R}_0^+ \cup \{+\infty\}$, where infeasible assignments have $+\infty$ utility and $\mathcal{X}_{f_i} \subseteq \mathcal{X}$.

To follow the DCOP formulation, each variable $x_i \in \mathcal{X}$ is assigned to a robot $r_i \in \mathcal{R}$, who has the sole responsibility for the variable's value. In our problem, each robot controls exactly one variable. The domain $\mathcal{D}_{x_i}$ of each variable x_i consists of the tasks that the corresponding robot r_i is capable of doing. Each task k_j is represented as a cost function f_j. The function f_j shows the cost of accomplishing the corresponding task with different numbers of robots $\mathcal{C}_{k_j} \subseteq \mathcal{R}$, from no robot to the coalition of all the robots capable of doing task k_j. Each robot knows only about the functions in which it is involved. We compute the cost function f_j for task k_j as follows:

$$f_j = \alpha \times \max_{r_i \in \mathcal{C}_{k_j}} (m(r_i, k_j)) + (1-\alpha) \times \sum_{r_i \in \mathcal{C}_{k_j}} tt(\mathrm{L}_{r_i}, \mathrm{L}_{k_j}), \tag{1}$$

where $m(r_i, k_j)$ is the latest finish time of the last task in robot r_i's schedule if task k_j were to be assigned to the robot, $tt(\mathrm{L}_{r_i}, \mathrm{L}_{k_j})$ is the time it takes for robot r_i to travel from its current location to task k_j's location, and α is a hyperparameter set to 0.7 in our experiments. The objective is to find a complete value assignment for the variables of $\mathcal{X}$ (denoted by $\mathbf{x}$) that minimizes the following global cost function:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{j=1}^{|\mathcal{F}|} f_j(\mathbf{x}_{f_j}), \tag{2}$$

where $\mathbf{x}_{f_j}$ is the partial value assignment for the variables of $\mathcal{X}_{f_i} \subseteq \mathcal{X}$.

C. DCOP Representation: Factor Graph

There are different ways to represent a DCOP problem. The factor graph [12] is a bipartite cyclic graph chosen to represent the problem. In the factor graph $\mathcal{G}_\varphi = (\mathcal{X} + \mathcal{F}, \mathcal{E}_\varphi)$, the variable nodes $\mathcal{X}$ represent the robots (shown by circles in Figure 2), and the function nodes $\mathcal{F}$ represent the tasks (shown by squares in Figure 2). An undirected edge $e_{ij} \in \mathcal{E}_\varphi$ between a variable node x_i and a function node f_j indicates that robot r_i is capable of performing task k_j. Each robot has a local view of the factor graph that includes only its immediate neighbors. Figure 2 shows three factor graphs for the three layers of the precedence graph in Figure 1.

D. Solving DCOP: Max-Sum Algorithm

Solving DCOP exactly is NP-hard [13], and for this reason approximate methods such as max-sum [10] are used to solve the optimization problem. The max-sum algorithm is an incomplete inference-based method that iteratively performs message passing on the factor graph corresponding to the DCOP. In each iteration, every variable node sends messages to all the function nodes that it connects to (Equation 3), and every function node sends messages to all the variable nodes that it is connected to in the graph (Equation 4).

- The message from variable x_i to function f_j is

$$q_{i \to j}(x_i) = \beta_{ij} + \sum_{k \in \mathcal{M}_i \setminus j} s_{k \to i}(x_i), \tag{3}$$

where $q_{i \to j}$ is the message sent from variable node x_i to function node f_j, β_{ij} is a scalar chosen such that $\sum_{x_i} q_{i \to j}(x_i) = 0$, and $\mathcal{M}_i$ is the set of indices of all the function nodes connected to variable node x_i in the factor graph.

- The message from function f_j to variable x_i is

$$s_{j \to i}(x_i) = \min_{\mathbf{x}_{f_j} \setminus i} [f_j(\mathbf{x}_{f_j}) + \sum_{k \in \mathcal{N}_j \setminus i} q_{k \to j}(x_k)], \tag{4}$$

where $s_{j \to i}$ is the message sent from function node f_j to variable node x_i, $\mathcal{N}_j$ is the set of indices of all the variable nodes connected to function node f_j, and $\mathbf{x}_{f_j} \setminus i \equiv \{x_k \mid k \in \mathcal{N}_j \setminus i\}$.

This process continues until the messages converge or a fixed number of iterations is reached. Each robot r_i then selects the best task by aggregating the cost values received from its neighboring robots through the adjacent function nodes:

$$x_i^* = \operatorname*{argmin}_{x_i} \sum_{j \in \mathcal{M}_i} s_{j \to i}(x_i). \tag{5}$$

It may be the case that there are more tasks than robots in a layer of the precedence graph, or a task is not assigned to any robot. The max-sum algorithm is run on each layer for a fixed number of iterations so that all the tasks in the layer are processed. The algorithm processes and allocates the tasks in the precedence graph layer by layer. Max-sum has a time complexity of $O(d^l)$ [4], where $d = \max_{\mathcal{D}_{x_i} \in \mathcal{D}} |\mathcal{D}_{x_i}|$ is the size of the largest domain (i.e., the largest number of tasks a robot may perform), and $l = \max_{r_i \in \mathcal{R}} |N_{r_i}|$ is the largest number of neighboring robots.

E. Managing Schedules: Simple Temporal Networks

To maintain a schedule for the tasks, each robot creates a simple temporal network (STN) [14]. An STN is a graph $\mathcal{G}_\sigma = (\mathcal{T}, \mathcal{E}_\sigma)$ in which nodes represent the start time (ST) or finish time (FT) of the tasks, and edges show the duration (D) of a task or travel time (TT) between two tasks. The start time of a task k_i should be scheduled in the interval $[\mathrm{EST}_{k_i}, \mathrm{LFT}_{k_i} - \frac{\mathrm{D}_{k_i}}{|\mathcal{C}_{k_i}|}]$, and the finish time should be scheduled in the interval $[\mathrm{EST}_{k_i} + \frac{\mathrm{D}_{k_i}}{|\mathcal{C}_{k_i}|}, \mathrm{LFT}_{k_i}]$.

Each robot r_i uses its STN to calculate the cost of a task k_j if the task were to be assigned to the robot. The robot's corresponding variable node x_i uses this value to initiate the message with the function node f_j:

$$\alpha \times m(r_i, k_j) + (1-\alpha) \times tt(\mathrm{L}_{r_i}, \mathrm{L}_{k_j}). \tag{6}$$

Whenever a new task is assigned to a robot, the robot checks every possible position in its STN and inserts the task at the position that keeps the network consistent and leads to

the lowest makespan. To find a solution and a valid schedule for the STN, all the tasks should be assigned start-time and finish-time points that fulfill all their time window, ordering, and travel constraints. The Floyd-Warshall algorithm is used to solve the STN in $O(n^3)$ polynomial time, where n is the number of time points in the network. Figure 3 demonstrates the gradual formation of the complete STN for robot 2 for the sample precedence graph of Figure 1. Figure 4 shows the final schedule for each robot after solving their associated STNs.

V. Experiments and Results

A. Experimental Setup

We generate random precedence graphs using the method presented by Melancon et al. [15]. The map used in the experiments is a 100×100 grid in a $2D$ coordinate plane. We set our scheduling *time frame* (TF) based on the number of tasks in the environment. The time frame determines that all the scheduled tasks must be completed within a fixed number of steps (e.g., 5000 steps). Tasks are generated with a set of parameters, including location (L), time duration (D), earliest start time (EST), latest finish time (LFT), and type (τ). The *time percentile* (TP) detailed below determines the EST of a task in the scheduling time frame. Tasks are distributed randomly throughout the map. For each task, the time duration is sampled uniformly from the integer interval $[100, 1000]$. The EST and LFT of each task are sampled uniformly from the following intervals:

$$\text{EST} = [0, \text{TF} \times \text{TP}], \tag{7}$$

$$\text{LFT} = \min(\text{EST} + \text{D} + rand(0, \text{TF}), \text{TF}), \tag{8}$$

where $rand(0, \text{TF})$ generates a random integer between 0 and the set time frame. The LFT is assigned the cap value of time frame if the generated finished time exceeds the time frame.

We define three different time percentiles for the EST of tasks. In the first case, we assign the EST of all tasks to be scheduled within the initial 25% of the scheduling time frame. This setting provides the most flexible time window constraints, giving sufficient time for each task to be performed and allowing robots to schedule and complete more tasks. In the second case, we assign the EST of all tasks to be within the initial 50% of the scheduling time frame. This setting provides more restricted time window constraints for the robots compared to the previous case. In the third case, we assign the EST of all tasks to be within the initial 75% of the scheduling time frame. Using this setting, robots face the most restricted temporal constraints for the tasks.

We define two types of tasks in the environment and randomly assign a type to each task to identify whether the task can be performed by a certain robot. Each robot is randomly assigned to perform one of the two types of tasks or both. We make sure that there is at least one robot for each type of task. Our experiments also examine another parameter, the number of robots involved in the MRTA problem. For each experiment, we use 4, 6, or 8 robots to complete the tasks. In addition, we consider a different number of tasks, i.e., 50, 100, and 200 in the experiments. We set the scheduling time frame for 50 tasks to 5000 steps, for 100 tasks to 7500 steps, and for 200 tasks to 10000 steps.

In summary, we conducted the experiments with the following settings:

- number of robots: 4, 6, and 8
- number of precedence graphs per setting: 25
- number of tasks per precedence graph: 50, 100, and 200
- scheduling time frame: 5000, 7500, and 10000 steps
- EST percentile of the tasks: 25%, 50%, and 75%
- task duration: $[100, 1000]$

We compared our proposed DCOP algorithm with an auction-based approach from the literature called prioritized iterated auction (PIA) [8], [9]. PIA handles both time window and ordering constraints imposed on the tasks, but can only be used in non-coalition scenarios. Ours is the first work to extend coalition formation to handle tasks with time window and ordering constraints.

B. Results

We conducted extensive experiments to assess the performance of the proposed algorithm. Tables I, II, and III show the results of the experiments under different settings, averaged over 25 randomly generated precedence graphs.

Overall, DCOP allocated more tasks compared to PIA in 100% of the settings, increasing the task completion rate (TCR) by 3.3% on average. Moreover, DCOP outperformed PIA under task completion frequency (TCF) in 100% of the cases, increasing the TCF by 5.4% on average. The formation of coalitions in DCOP reduced the average task execution time (ATET) compared to PIA by 37.7% on average, and DCOP outperformed PIA under this metric in 100% of the settings.

In addition, we examined the earliest start time (EST) parameter in the experiments. If the EST of tasks is scheduled early (i.e., within the initial 25% of the scheduling time frame), the time window of the tasks will be longer and more flexible on average. Hence, robots schedule and complete more tasks in this setting. When the EST of tasks is assigned to be in the later percentiles of the scheduling time frame (50% and 75%), the time window will be shorter and less flexible on average. Under these settings, it is more challenging for robots to schedule and complete the tasks.

VI. Conclusion and Future Work

We presented the first DCOP formulation of the multi-robot task allocation problem with time window and ordering constraints. The proposed method forms efficient coalitions among robots to maximize the task completion rate and frequency while minimizing the task execution time. The method outperforms an auction-based approach under different evaluation metrics. For future work, we plan to extend this work in various directions:

# of Tasks	# of Robots	DCOP (TCR)	PIA (TCR)	DCOP (TCF)	PIA (TCF)	DCOP (ATET)	PIA (ATET)
50	4	**65.52**	62.00	**0.166**	0.158	**367**	519
50	6	**85.92**	83.04	**0.219**	0.211	**294**	554
50	8	**95.20**	93.28	**0.251**	0.249	**237**	571
100	4	**52.28**	48.08	**0.175**	0.162	**403**	535
100	6	**72.20**	66.20	**0.242**	0.223	**336**	567
100	8	**88.16**	83.00	**0.296**	0.285	**275**	581
200	4	**37.96**	32.84	**0.190**	0.165	**387**	541
200	6	**53.74**	48.60	**0.269**	0.244	**337**	544
200	8	**68.52**	63.70	**0.343**	0.319	**302**	552

TABLE I: Results for tasks with EST within the first 25% of the scheduling time frame.

# of Tasks	# of Robots	DCOP (TCR)	PIA (TCR)	DCOP (TCF)	PIA (TCF)	DCOP (ATET)	PIA (ATET)
50	4	**64.80**	61.44	**0.163**	0.155	**392**	502
50	6	**84.64**	81.52	**0.215**	0.207	**309**	533
50	8	**93.84**	92.08	**0.240**	0.239	**254**	552
100	4	**50.28**	45.64	**0.168**	0.153	**411**	549
100	6	**69.76**	63.84	**0.233**	0.214	**338**	571
100	8	**85.52**	81.20	**0.286**	0.273	**297**	574
200	4	**37.86**	34.86	**0.190**	0.175	**383**	496
200	6	**52.24**	49.90	**0.262**	0.250	**342**	503
200	8	**65.04**	63.34	**0.326**	0.318	**297**	529

TABLE II: Results for tasks with EST within the first 50% of the scheduling time frame.

# of Tasks	# of Robots	DCOP (TCR)	PIA (TCR)	DCOP (TCF)	PIA (TCF)	DCOP (ATET)	PIA (ATET)
50	4	**63.68**	60.08	**0.161**	0.152	**369**	510
50	6	**84.96**	80.72	**0.215**	0.204	**252**	468
50	8	**88.96**	88.56	**0.225**	0.224	**238**	515
100	4	**50.52**	48.12	**0.169**	0.161	**411**	521
100	6	**68.36**	65.56	**0.229**	0.220	**342**	551
100	8	**82.40**	81.08	**0.275**	0.271	**259**	495
200	4	**35.84**	33.58	**0.179**	0.168	**441**	510
200	6	**49.76**	48.08	**0.249**	0.241	**363**	521
200	8	**60.60**	59.52	**0.303**	0.298	**301**	539

TABLE III: Results for tasks with EST within the first 75% of the scheduling time frame.

- **Scalability:** Scalability is a major challenge in DCOPS due to communication overhead among the robots. We plan to work on improved variations of the max-sum algorithm to reduce this overhead [1].
- **Heterogeneity:** The algorithm should be able to handle various forms of heterogeneity, such as different motion or sensing capabilities of the robots.
- **Dynamic environments:** The method should be able to adapt to changes in the environment without having to be completely re-run. For example, robots or tasks can be added to or removed from the environment.
- **Robustness:** The algorithm should be robust to failures, such as robot action failure and unreliable communication.

REFERENCES

[1] S. Ramchurn, A. Farinelli, K. Macarthur, M. Polukarov, and N. Jennings, "Decentralised coordination in robocup rescue," *The Computer Journal*, vol. 53, no. 9, pp. 1447–1461, 2010.

[2] P. Fazli, A. Davoodi, and A. K. Mackworth, "Multi-robot repeated area coverage," *Autonomous Robots*, vol. 34, no. 4, p. 251–276, 2013.

[3] J. Ji, P. Fazli, S. Liu, T. Pereira, D. Lu, J. Liu, M. Veloso, and X. Chen, "Help me! sharing of instructions between remote and heterogeneous robots," in *International Conference on Social Robotics (ICSR)*, 2016, pp. 786–795.

[4] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *Journal of Artificial Intelligence Research*, vol. 61, pp. 623–698, 2018.

[5] S. D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings, "Coalition formation with spatial and temporal constraints," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010, pp. 1181–1188.

[6] M. C. Gombolay, R. Wilcox, and J. A. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," in *Robotics: Science and Systems (RSS)*, 2013, pp. 49–56.

[7] 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, 2006.

[8] M. McIntire, E. Nunes, and M. Gini, "Iterated multi-robot auctions for precedence-constrained task scheduling," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2016, pp. 1078–1086.

[9] E. Nunes, M. McIntire, and M. Gini, "Decentralized allocation of tasks with temporal and precedence constraints to a team of robots," in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2016, pp. 197–202.

[10] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *International Conference on Autonomous Agents and Multiagent systems (AAMAS)*, 2008, pp. 639–646.

[11] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. A. Rodriguez-Aguilar, "Efficient inter-team task allocation in robocup rescue," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015, pp. 413–421.

[12] F. R. Kschischang, B. J. Frey, and H. . Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[13] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, vol. 161, no. 1, pp. 149–180, 2005.

[14] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[15] G. Melançon, I. Dutour, and M. Bousquet-Mélou, "Random generation of directed acyclic graphs," *Electronic Notes in Discrete Mathematics*, vol. 10, pp. 202–207, 2001.