Auctioning over Probabilistic Options for Temporal Logic-Based Multi-Robot Cooperation under Uncertainty

Philipp Schillinger[1,2], Mathias Bürger[1] and Dimos V. Dimarogonas[2,3]

***Abstract*— Coordinating a team of robots to fulfill a common task is still a demanding problem. This is even more the case when considering uncertainty in the environment, as well as temporal dependencies within the task specification. A multi-robot cooperation from a single goal specification requires mechanisms for decomposing the goal as well as an efficient planning for the team. However, planning action sequences offline is insufficient in real world applications. Rather, due to uncertainties, the robots also need to closely coordinate during execution and adjust their policies when additional observations are made. The framework presented in this paper enables the robot team to cooperatively fulfill tasks given as temporal logic specifications while explicitly considering uncertainty and incorporating observations during execution. We present the effectiveness of our ROS implementation of this approach in a case study scenario.**

I. Introduction

In multi-robot scenarios, goal specifications often involve complex constraints on the behavior of the robots, including temporal relations between their actions. *Linear Temporal Logic* (LTL) [2], including its fragment *co-safe LTL* [11], is an established formalism to express such specifications. It is well understood how provably correct action plans can be derived from LTL specifications [3, 23].

One reason for the success of LTL in robotics results from the possibility to integrate temporal constraints with probabilistic models [2], in particular with *Markov Decision Processes* (MDPs) [20, 25] to model uncertainties such as unknown durations of actions or stochastic events. However, efficient planning algorithms for coordinating multiple robots in a non-deterministic environment to solve tasks with temporally dependent specifications are still missing.

We propose here an efficient auction algorithm for solving co-safe LTL tasks in uncertain environments, relying on the concept of *options* [20, 24] to represent sub-tasks. Auction algorithms are well-established for efficient allocation of tasks to agents [14]. However, since such algorithms usually assume independent tasks, they are not directly applicable to tasks with temporal dependencies [19, 4].

[1]Bosch Center for Artificial Intelligence, Renningen, Germany. {philipp.schillinger, mathias.buerger}@de.bosch.com

[2]KTH Centre for Autonomous Systems and ACCESS Linnaeus Center, EES, KTH Royal Institute of Technology, Stockholm, Sweden. {schillin, dimos}@kth.se

[3]Supported by the H2020 ERC Starting Grant BUCOPHSYS, the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF), and the Knut och Alice Wallenberg Foundation (KAW).

This work was supported by the EU H2020 Research and Innovation Programme under GA No. 731869 (Co4Robots).

Approach. A goal specification is given as co-safe LTL formula (Sec. III, IV) and a team of robots is assumed, each modeled as an MDP (Sec. V). Based on the automaton representation of the LTL formula, each robot computes its *options*, i.e., sub-policies that ensure progress towards the overall goal (Sec. IV, VI). Computing these options is computationally attractive as it requires to solve a much smaller probabilistic planning problem. A distributed auction scheme is proposed to efficiently allocate the options to robots (Sec. VII). An important property of the proposed auction algorithm is that it allows the robots to prepare tasks that have temporal dependencies on other tasks. During the execution, both the allocation and the policies are updated to incorporate additional knowledge (Sec. VIII).

In this paper, we assume that no *critical* events can occur, that is, events leading to specification violation. Still, a specification may require certain reactions to potentially occurring events. The proposed approach is *top-down* in the sense that it does not require the tasks to be assigned to particular robots a-priori. Instead, the tasks are derived from one global co-safe LTL specification and allocated to available robots depending on the robot's initial conditions.

Contributions. (1) We propose a decentralized framework for multi-robot planning and execution under uncertainty to guarantee satisfaction of a single global co-safe LTL specification. To the best of our knowledge, a framework of this scope has not been previously presented. (2) We propose an auction algorithm suitable for tasks which are not independent, but include temporal dependencies.

As an illustrative example, we consider the following scenario for a multi-robot system in an office environment. To allow the robots to access a restricted room, they first need confirmation from two persons for whom only some possible locations are known. Such a scenario in particular requires consideration of uncertainty regarding the location of the persons and close coordination between the robots. We conclude the paper with a case study evaluation of our ROS implementation in this scenario.

II. Related Work

Decentralized control approaches allow to scale systems to increasing numbers of agents. For example, [16] uses a *Decentralized MDP* definition to model a system with sparse interaction between the robots, but assumes a known task allocation. To allocate tasks among multiple robots, [14] discusses several auction strategies with focus on efficiency. [10] presents a sequential single-item auction to

efficiently allocate tasks, which we adopt and extend in this paper. [5] proposes consensus-based bundled auctions to decide assignment of multiple tasks at the same time. However, our problem requires a sequential task allocation.

In general, planning from an LTL specification allows to define complex tasks with temporal dependencies. [12] presents a single-robot planner which combines an LTL specification with an MDP, allowing to plan policies under uncertainty. Similarly, [6] plans single-robot policies which maximize the probability of LTL satisfaction. [13] formulates a *Timed MDP* to consider variable action durations instead of single-step actions. Alternatively, [21] presents a learning approach to satisfy the LTL specification.

Already without considering allocation, extending LTL planning to multi-robot teams requires to consider dependencies between the robots. [9] does not coordinate the robots in advance, but assumes locally assigned temporal logic tasks and communicates reactively to arising conflicts, similar to [27]. [18] also assumes pre-assigned tasks, but considers uncertainty in advance and clusters the robots into teams. In [23], we automatically identify independent tasks and also optimize allocation, but do not consider uncertainty.

In the case that a single specification is given, it needs not only to be decided how to allocate the tasks, but also how the specification can be decomposed into multiple tasks in the first place. In the context of MDPs, [24] introduces *options* as an additional layer of abstraction, enabling a way of decomposition. [1] constructs a *Partially Observable MDP* for robots in a multi-agent system and manually defines such options to improve planning efficiency. [15] manually defines options to improve learning for complex LTL specifications. In contrast, we propose here to construct options automatically from a given LTL specification.

The above approaches each address only some of the considered aspects. We are not aware of a comparable multi-robot framework to efficiently coordinate robots from a single LTL goal specification while considering stochastic events in the environment and uncertain action durations.

III. Preliminaries

A *syntactically co-safe Linear Temporal Logic* (scLTL) formula [11] is given in the form

$$\phi = \top \mid \pi \mid \neg\pi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi_1 \mid \phi_1 \,\mathcal{U}\, \phi_2.$$

$\top$ denotes the Boolean constant *true*, $\pi \in \Pi$ is an atomic proposition which is either true or false, and ϕ_1, ϕ_2 are themselves scLTL formulas. The semantics of the above operators "not" $\neg$, "and" $\wedge$, "or" $\vee$, "next" $\bigcirc$, and "until" $\mathcal{U}$ are defined over sequences of sets of propositions $\sigma\colon \mathbb{N} \to 2^{\Pi}$. $\sigma(t)$ "satisfies" ϕ as follows and trivially, $\sigma(t) \vDash \top$.

- $\sigma(t) \vDash \pi$ if proposition $\pi \in \sigma(t)$
- $\sigma(t) \vDash \neg\pi$ if proposition $\pi \notin \sigma(t)$
- $\sigma(t) \vDash \phi_1 \wedge \phi_2$ if both $\sigma(t) \vDash \phi_1$ and $\sigma(t) \vDash \phi_2$
- $\sigma(t) \vDash \phi_1 \vee \phi_2$ if any of $\sigma(t) \vDash \phi_1$ or $\sigma(t) \vDash \phi_2$
- $\sigma(t) \vDash \bigcirc\phi_1$ if next $\sigma(t+1) \vDash \phi_1$
- $\sigma(t) \vDash \phi_1 \,\mathcal{U}\, \phi_2$ if eventually $\exists T \geq t\colon \sigma(T) \vDash \phi_2$ and until then $\forall t' \in [t, T)\colon \sigma(t') \vDash \phi_1$

In addition, we define the derived operators "implies" $\pi \Rightarrow \phi_1 := \neg\pi \vee \phi_1$ and "eventually" $\Diamond\phi_1 := \top \,\mathcal{U}\, \phi_1$.

Any scLTL formula ϕ can be translated into a Deterministic Finite Automaton (DFA) [2, 8].

Definition 1 (DFA)**.** *A DFA is given by $\mathcal{D} = (Q, q_0, \alpha, \delta, F)$ consisting of **(1)** a set of states Q, **(2)** an initial state $q_0 \in Q$, **(3)** an input alphabet α, **(4)** a transition function $\delta\colon Q \times \alpha \to Q$, **(5)** a set of accepting states $F \subseteq Q$.*

Then, ϕ is satisfied by a sequence σ if a run $\rho\colon \mathbb{N}_0 \to Q$ with $\rho(0) = q_0$ and $\delta(\rho(i-1), \sigma(i)) = \rho(i), \forall i$ exists and is *accepting*, given by $\exists i\colon \rho(i) \in F$.

Furthermore, we employ in this paper the following two standard definitions from graph theory [17].

Definition 2 (Strongly Connected Component)**.** *In a directed graph $D = (V, E)$, a strongly connected component $C \in \mathrm{SCC_D}$ is a set of vertices such that each vertex in the set can be reached from any other vertex of the set.*

We use the notation $\mathrm{SCC}\colon V \to \mathrm{SCC_D}$ to denote the strongly connected component including a vertex $v \in V$. In particular, $v' \in \mathrm{SCC}(v)$ means that both v and v' are in the same strongly connected component.

Definition 3 (Vertex Boundary)**.** *In a directed graph $D = (V, E)$, the vertex boundary of a set of vertices $S \subseteq V$ is another set of vertices $\mathrm{VBD}(S) \subset V$ such that $S \cap \mathrm{VBD}(S) = \emptyset$ and for each $v' \in \mathrm{VBD}(S)$, there exists a $v \in S$ with a directed edge $(v, v') \in E$.*

IV. LTL Progress

The standard definition of a DFA provides a notion about how a sequence can be continued in order to eventually satisfy the given scLTL formula [2, 3]. However, the DFA does not quantify the "progress" towards satisfaction achieved by following a particular transition. While it is in general not possible to quantify such a measure in a purely logical context, we define here the term *progress* for our use as follows, using the above graph theory definitions.

Definition 4 (LTL Progress)**.** *Given the DFA $\mathcal{D}$ of an scLTL formula ϕ; a transition from state q to q' makes progress towards satisfaction of ϕ if no path exists which leads back to q, starting from q'. Otherwise, the transition makes no progress and is called reversible.*

In this section, we present an approach to identify those transitions which make a progress in a given DFA $\mathcal{D}$. For this purpose, we assign a progress level $L\colon Q \to \mathbb{N}$ to each state $q \in Q$, similar to the idea of a *Progressive Function* as presented in [26, Def. 8] or an *Energy Function* as in [7, Def. 3.3]. We choose the progress level of a state such that $L(q) < L(q')$ if a transition from q to q' makes progress and $L(q) = L(q')$ if not, for all pairs $q, q' \in Q$ which have a transition $\delta(q, \cdot) = q'$ in $\mathcal{D}$.

Algorithm 1 calculates such an implementation of the progress level function L. The algorithm starts with the initial state q_0 and its respective progress $L(q_0) = 0$. In

Algorithm 1 DFA Progress Level

Input: DFA $\mathcal{D}$ for which the levels should be determined
Output: Progress level function $L\colon Q \to \mathbb{N}$

Notation Remarks:
$\mathrm{SCC}_{\mathcal{D}}$ – Strongly connected components of $\mathcal{D}$, see Def. 2
$\mathrm{VBD}\colon 2^Q \to 2^Q$ – Vertex boundary, see Def. 3
$Q_{\mathrm{L}}\colon \mathbb{N} \to 2^Q$ – States of given level, initially $Q_{\mathrm{L}}(i) := \emptyset, \forall i$

1: $Q_{\mathrm{todo}} \leftarrow \{q_0\}$; $l \leftarrow 0$
▶ Determine possible levels for all states, see Lem. 1
2: **while** $Q_{\mathrm{todo}} \neq \emptyset$ **do**
3: $\quad Q_{\mathrm{next}} \leftarrow \emptyset$
4: $\quad$ **for all** $C \in \mathrm{SCC}_{\mathcal{D}}\colon C \cap Q_{\mathrm{todo}} \neq \emptyset$ **do**
5: $\quad\quad Q_{\mathrm{L}}(l) \leftarrow Q_{\mathrm{L}}(l) \cup C$
6: $\quad\quad Q_{\mathrm{next}} \leftarrow Q_{\mathrm{next}} \cup \mathrm{VBD}(C)$
7: $\quad Q_{\mathrm{todo}} \leftarrow Q_{\mathrm{next}}$; $l \leftarrow l + 1$
▶ Assign highest possible level, see Lem. 2
8: **for all** $q \in Q$ **do**
9: $\quad L(q) \leftarrow \max\{l\colon q \in Q_{\mathrm{L}}(l)\}$
10: **return** L

the first part of the algorithm (lines 3-9), the states of all reachable strongly connected components are determined and labeled with a corresponding level increasing over the iterations. Afterwards, in the second part (lines 10-11), the highest progress level is assigned to each state.

Figure 1 shows an example DFA and the progress levels at each state resulting from Algorithm 1. Again, please note that the specific numbers have no particular meaning[1], only their relation is what matters here.

It can be shown that the first part always terminates after a finite number of iterations without specific assumptions regarding the structure of $\mathcal{D}$.

Lemma 1 (Finite Labeling)**.** *The while-loop in line 3 of Algorithm 1 terminates after at most $|Q| - 1$ iterations.*

Proof. Given any two states $q_i, q_j \in Q$, consider the following three cases. **(1)** *There exists a cycle containing q_i and q_j.* Then, q_i and q_j belong to the same strongly connected component and thus, both are labeled during the same iteration. **(2)** *There exists a shortest path from q_i to q_j of length n edges, but no path from q_j to q_i.* Since the set of considered states in each iteration is given by the vertex boundary of the previous iteration, q_j is labeled at most n iterations after q_i. **(3)** *There exists no path between q_i and q_j.* At least one of q_i, q_j is not labeled during the while-loop. The number of iterations does not increase in this case. Consequently, the upper bound on the number of iterations is given by the longest, non-cyclic path starting from the initial state q_0 and its length is bounded by $|Q| - 1$. □

Based on the assumption that every state is reachable from the initial state q_0, which is a given property of the DFA $\mathcal{D}$ by construction [8], it can further be shown that indeed all states $q \in Q$ are labeled with at least one level during the first part of Algorithm 1.

[1]In fact, we provide an interpretation in Sec. VII, Thm. 1 specific to the numbers assigned by Alg. 1. But this can be safely ignored here.

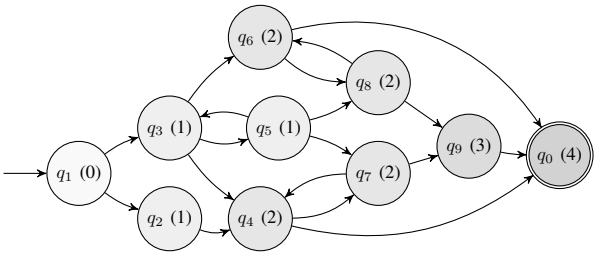

Fig. 1. Example DFA structure (simplified for illustration) for the formula $\Diamond(\pi_1 \wedge \Diamond\pi_2) \wedge \Diamond\pi_3 \wedge \Box(\pi_4 \Rightarrow \Diamond\pi_1)$. The numbers in parenthesis at each state denotes its progress level towards satisfaction of the specification.

Lemma 2 (State Labeling)**.** *For all $q \in Q$ there exists an $l \in \mathbb{N}$ such that $q \in Q_{\mathrm{L}}(l)$, given that there exists a path from q_0 to q.*

Proof. Given any two states $q_i, q_j \in Q$, only the following two cases are possible under the given assumption. **(1)** *There exists a cycle containing q_i and q_j.* **(2)** *There exists a shortest path from q_i to q_j of length n edges, but no path from q_j to q_i.* In both cases, the states q_i and q_j are labeled as shown in the proof of Lemma 1. □

V. Model Construction

To model the multi-robot system and its probabilistic dynamics, we first model each robot individually by constructing a respective *Markov Decision Process* (MDP). Actions in the constructed MDP correspond to capabilities of the robot. While these capabilities are considered atomic in the following, they can practically correspond to complex implementations like navigation to a certain waypoint. Consequently, we assign a cost (or negative reward) to actions denoting their desirability, for example based on the time required for execution and an evaluation of the action outcome. Such an approach is similar to, e.g., [6, 12].

First, the capabilities of each robot are modeled as a deterministic transition system.

Definition 5 (Agent Automaton)**.** *An agent automaton is given by $\mathcal{A} := (S_{\mathcal{A}}, s_{0,\mathcal{A}}, A_{\mathcal{A}})$ consisting of **(1)** a set of states $S_{\mathcal{A}}$, **(2)** an initial state $s_{0,\mathcal{A}} \in S_{\mathcal{A}}$, **(3)** a set of actions $A_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times S_{\mathcal{A}}$.*

In addition, we define a labeling function $\lambda_{\mathcal{A}}\colon S_{\mathcal{A}} \to 2^{\Pi_{\mathcal{A}}}$ over the states of $\mathcal{A}$ given a set of propositions $\Pi_{\mathcal{A}}$. The label of a state $s_{\mathcal{A}} \in S_{\mathcal{A}}$ denotes the set of propositions $\lambda(s_{\mathcal{A}}) \subseteq 2^{\Pi_{\mathcal{A}}}$ which are true in $s_{\mathcal{A}}$.

Uncertainties in the environment and resulting from action execution can only be observed by the robot. Therefore, we model these events in a so-called *sensing model*.

Definition 6 (Sensing Model)**.** *Observable events are modeled by $\mathcal{S} = (S_{\mathcal{S}}, s_{0,\mathcal{S}}, E_{\mathcal{S}}, \Pi_{\mathcal{S}}, P_{\mathcal{S}})$ consisting of **(1)** states $S_{\mathcal{S}}$ of the observed environment, **(2)** an initial state $s_{0,\mathcal{S}} \in S_{\mathcal{S}}$, **(3)** events $E_{\mathcal{S}} \subseteq S_{\mathcal{S}} \times S_{\mathcal{S}}$ excluding self transitions, **(4)** propositions $\Pi_{\mathcal{S}}$, **(5)** probabilities $P_{\mathcal{S}}\colon E_{\mathcal{S}} \times \Pi_{\mathcal{S}} \to [0, 1]$.*

The sensing model is similar to the agent automaton, but here, transitions represent a prior sensing belief that a certain event will be observed, e.g., a person or an object is found. The probabilities $P_\mathcal{S}$ are assumed to be known and can, for example, be learned during long-term operation. The set of propositions $\Pi_\mathcal{S}$ has two functions here. First, the event probabilities $P_\mathcal{S}$ depend on $\Pi_\mathcal{S}$. Second, we again define a labeling function $\lambda_\mathcal{S}\colon S_\mathcal{S} \to 2^{\Pi_\mathcal{S}}$ similar to above.

Combing both $\mathcal{A}$ and $\mathcal{S}$, an MDP can be constructed describing the probabilistic outcomes of actions.

Definition 7 (Agent MDP)**.** *An agent MDP is given by $\mathcal{M} = (S_\mathcal{M}, s_{0,\mathcal{M}}, A_\mathcal{M}, T_\mathcal{M}, D_\mathcal{M})$ consisting of **(1)** states $S_\mathcal{M} = S_\mathcal{A} \times S_\mathcal{S}$, **(2)** an initial state $s_{0,\mathcal{M}} = (s_{0,\mathcal{A}}, s_{0,\mathcal{S}})$, **(3)** actions $A_\mathcal{M} = \{((s_\mathcal{A}, s_\mathcal{S}), (s'_\mathcal{A}, s'_\mathcal{S})) \in S_\mathcal{M} \times S_\mathcal{M}\colon (s_\mathcal{A}, s'_\mathcal{A}) \in A_\mathcal{A}\}$, **(4)** transition probabilities $T_\mathcal{M}\colon S_\mathcal{M} \times A_\mathcal{M} \times S_\mathcal{M} \to [0,1]$ defined as below, **(5)** probabilistic action durations $D_\mathcal{M}\colon A_\mathcal{M} \to \mathbb{R}^2_+$.*

The transition probabilities are defined as follows based on the event probabilities.

$$T_\mathcal{M}\colon ((s_\mathcal{A}, s_\mathcal{S}), a, (s'_\mathcal{A}, s'_\mathcal{S})) \mapsto \begin{cases} P_\mathcal{S}(s_\mathcal{S}, s'_\mathcal{S}) & \text{if } s_\mathcal{S} \neq s'_\mathcal{S} \\ 1 - p_s & \text{otherwise} \end{cases}$$

where $a \in A_\mathcal{M}$ denotes an action leading to $(s'_\mathcal{A}, s'_\mathcal{S})$ and $p_s = \sum_{s' \in \mathrm{NB}(s)} P_\mathcal{S}(s, s')$ with neighbors $\mathrm{NB}(s) = \{s' \in S_\mathcal{S}\colon (s, s') \in E_\mathcal{S}\}$ denotes the probability of any event being observed. Consequently, we assume for the sensing model $\mathcal{S}$ that $p_s \leq 1$ for all $s \in S_\mathcal{S}$.

The duration $D_\mathcal{M}\colon a_\mathcal{M} \mapsto (\mu, \sigma^2)$ of an action $a_\mathcal{M}$ is approximated by a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

Labels of $\mathcal{M}$ follow from the labeling functions of $\mathcal{A}$ and $\mathcal{S}$, given by $\lambda_\mathcal{M}\colon (s_\mathcal{A}, s_\mathcal{S}) \mapsto \lambda_\mathcal{A}(s_\mathcal{A}) \cup \lambda_\mathcal{S}(s_\mathcal{S})$.

Finally, we combine $\mathcal{D}$ and $\mathcal{M}$ to relate a specification given as co-safe LTL formula with the system dynamics. It is well-known [2] that the DFA $\mathcal{D}$ and system MDP $\mathcal{M}$ can be combined to a single model which on the one hand describes the physical system as given by $\mathcal{M}$ and on the other hand incorporates the specification as given by $\mathcal{D}$.

Definition 8 (Product MDP)**.** *The combination of an Agent MDP and a DFA is given by $\mathcal{P} = (S_\mathcal{P}, s_{0,\mathcal{P}}, A_\mathcal{P}, T_\mathcal{P}, D_\mathcal{P}, F_\mathcal{P})$ with **(1)** states $S_\mathcal{P} = Q \times S_\mathcal{M}$, **(2)** an initial state $s_{0,\mathcal{P}} = (q_0, s_{0,\mathcal{M}})$, **(3)** actions $A_\mathcal{P} = \{((q, s_\mathcal{M}), (q', s'_\mathcal{M}))\colon (s_\mathcal{M}, s'_\mathcal{M}) \in A_\mathcal{M}, \delta(q, \lambda(s_\mathcal{M})) = q'\}$, **(4)** transition probabilities $T_\mathcal{P}\colon S_\mathcal{P} \times A_\mathcal{P} \times S_\mathcal{P} \to [0,1]$, **(5)** probabilistic action durations $D_\mathcal{P}\colon A_\mathcal{P} \to \mathbb{R}^2_+$, **(6)** a set of accepting states $F_\mathcal{P} = \{(q, s_\mathcal{M})\colon q \in F\}$.*

$T_\mathcal{P}$ and $D_\mathcal{P}$ are defined by a projection onto $\mathcal{M}$, given by $T_\mathcal{P}\colon ((q, s_\mathcal{M}), \alpha, (q', s'_\mathcal{M})) \mapsto T_\mathcal{M}((s_\mathcal{M}, \alpha_\mathcal{M}, s'_\mathcal{M}))$ and $D_\mathcal{P}\colon ((q, s_\mathcal{M}), (q', s'_\mathcal{M})) \mapsto D_\mathcal{M}((s_\mathcal{M}, s'_\mathcal{M}))$ Since the DFA $\mathcal{D}$ is deterministic, such a projection preserves the probabilistic property of $T_\mathcal{M}$.

In fact, constructing $\mathcal{P}$ is known to be computationally expensive [2], especially when considering uncertainty. For this reason, we show in the following how an explicit construction of the full model can be avoided.

VI. Option Definition

To enable allocation of tasks to agents, we define a set of options $O_\mathcal{P}$ over the Product MDP $\mathcal{P}$ following the notation of [24]. As each agent constructs its own model $\mathcal{P}$, also the options $O_\mathcal{P}$ are defined separately for each of the agents.

Definition 9 (Options)**.** *An option $o_i \in O_\mathcal{P}$ is given by the tuple $o_i = (\mathcal{I}_i, \pi_i, \beta_i)$ consisting of **(1)** an initiation set $\mathcal{I}_i \subseteq S_\mathcal{P}$ where o_i becomes available, **(2)** a policy $\pi_i\colon S_\mathcal{P} \times A_\mathcal{P} \to [0,1]$ denoting the probability of selecting an action when being in a particular state, **(3)** a termination condition $\beta_i\colon S_\mathcal{P} \to [0,1]$ denoting the probability that o_i terminates.*

In particular, we define the options $o_i \in O_\mathcal{P}$ to be associated with a certain target DFA state $q_i \in Q$. Since the DFA tracks the mission progress as discussed in Section IV, q_i can be seen as denoting a certain progress in satisfying the overall mission. Consequently, following o_i can be interpreted as choosing q_i as an intermediate goal on the way to finally reach an accepting state in the DFA, which implies satisfaction of the scLTL specification.

To make sure that each option makes progress towards satisfaction of the scLTL specification, the initiation set $\mathcal{I}_i$ needs to be chosen carefully. To address this, we employ the notion of task progress introduced in Section IV. The respective initiation set is then given by

$$\begin{aligned} \mathcal{I}_i = \{(q, s_\mathcal{M}) \in S_\mathcal{P}\colon L(q) < L(q_i) \\ \wedge\, L(q) = L(q_p), \forall q_p \in Paths(q, q_i)\} \end{aligned} \tag{1}$$

where $Paths(q, q_i)$ denotes the set of states in paths from q to q_i. The equality constraint $L(q) = L(q_p)$ is added for technical reasons to avoid redundant options.

Furthermore, the termination condition is defined as

$$\beta_i\colon (q, s_\mathcal{M}) \mapsto \begin{cases} 1 & \text{if } q = q_i \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Since each option $o_i \in O_\mathcal{P}$ is defined to lead to a corresponding DFA state $q_i \in Q$, the policy π_i needs to be planned such that q_i is eventually reached.

A particular pair of $\mathcal{I}_i$ and β_i implies a limited subset of states reachable while following o_i. In practice, this subset is much smaller than the full state space $S_\mathcal{P}$. Based on this limited state space, we define the construction of a so-called *Option MDP* as a local substitute of the full Product MDP $\mathcal{P}$. This allows us to avoid the computationally expensive construction of $\mathcal{P}$ and plan π_i more efficiently.

In particular, we define the Option MDP as follows, assuming a particular mission progress by being in the DFA state $q_c \in Q$ given that $(q_c, s_\mathcal{M}) \in \mathcal{I}_i$ for all $s_\mathcal{M} \in S_\mathcal{M}$.

Definition 10 (Option MDP)**.** *The Option MDP corresponding to option o_i assuming the current DFA state q_c is given by $\mathcal{O}_i^c = (S_\mathcal{O}, S_{0,\mathcal{O}}, A_\mathcal{O}, T_\mathcal{O}, D_\mathcal{O})$ consisting of **(1)** states $S_\mathcal{O} = \{(q, s_\mathcal{M}) \in S_\mathcal{P}\colon q \in \{q_i\} \cup \mathrm{SCC}(q_c)\}$, **(2)** a set of initial states $S_{0,\mathcal{O}} = \{(q, s_\mathcal{M}) \in S_\mathcal{P}\colon q = q_c\}$, **(3)** actions $A_\mathcal{O} = \{(s_\mathcal{P}, s'_\mathcal{P}) \in A_\mathcal{P}\colon s_\mathcal{P}, s'_\mathcal{P} \in S_\mathcal{O}, \beta(s_\mathcal{P}) \neq 1\}$, **(4)** transition probabilities $T_\mathcal{O}\colon S_\mathcal{O} \times A_\mathcal{O} \times S_\mathcal{O} \to [0,1]$, **(5)** action durations $D_\mathcal{O}\colon A_\mathcal{O} \to \mathbb{R}^2_+$.*

While $T_{\mathcal{O}}$ and $D_{\mathcal{O}}$ are defined as for $\mathcal{P}$, the state space of $\mathcal{O}$ is much smaller. Specifically, only states in the strongly connected component of DFA state q_c are considered by the Option MDP, as well as the target DFA state q_i. Still, this limitation of the state space does not limit the possible policies π_i for execution of o_i.

Lemma 3 (Completeness)**.** *Given an option $o_i = (\mathcal{I}_i, \pi_i, \beta_i)$ and a current DFA state q_c such that $(q_c, s_{\mathcal{M}}) \in \mathcal{I}_i$ for an arbitrary $s_{\mathcal{M}} \in S_{\mathcal{M}}$. The resulting Option MDP $\mathcal{O}_i^c$ is complete in the sense that there exists a π_i such that a state s with $\beta_i(s) = 1$ can be reached whenever such a π_i exists in the full Product MDP $\mathcal{P}$.*

Proof. Given by Equation (1), $(q_c, s_{\mathcal{M}}) \in \mathcal{I}_i$ implies that $L(q_c) = L(q_p), \forall q_p \in Paths(q_c, q_i)$ with $Paths(q_c, q_i)$ as defined for Equation (1). In fact, Algorithm 1 admits the even stronger implication that $q_p \in \mathrm{SCC}(q_c)$ since $L(q_c) = L(q_p)$ implies that $q_p \notin \mathrm{VBD}(\mathrm{SCC}(q_c))$. Consequently, a state $s = (q_i, s_{\mathcal{M}})$ can be reached in $\mathcal{O}$ whenever it can be reached in $\mathcal{P}$ and $\beta_i(s) = 1$. □

To finally plan the policies π_i, standard algorithms [2, 20] such as *Value Iteration* can be employed on the Option MDP $\mathcal{O}_i^c$ and the duration $\widehat{D}_{\pi,i}$ required by π_i can be estimated.

When the option o_i including its policy π_i is known, it is possible to reduce the Option MDP to an *absorbing Markov Chain* (MC) [2] in order to predict the state in which o_i will terminate. The set of absorbing states is given by $\{s \in S_{\mathcal{O}} \colon \beta_i(s) = 1\}$. To calculate the termination probability over states, the transitions of the MC can be expressed in matrix form P by arranging the entries of $[P^T]_{jk} := T_{\mathcal{O}_i}(s_j, \pi_i(s_j), s_k)$ such that

$$P = \begin{bmatrix} Q & R \\ 0 & I_m \end{bmatrix} \tag{3}$$

where $Q \in \mathbb{R}^{n \times n}$ denotes transition probabilities within the set of transient, i.e., not absorbing, states and $R \in \mathbb{R}^{n \times m}$ transition probabilities from a transient state to an absorbing state. I_m is the identity matrix of size m, here denoting self transitions within the absorbing states. Then, the probability distribution over states in which π_i will terminate can be calculated by the transformation

$$B = (I_n - Q)^{-1} R \tag{4}$$

for absorbing states and is zero for transient states.

VII. Task Allocation

Based on the results of the previous sections, the Product MDP $\mathcal{P}$, of which we want to avoid its computationally expensive construction, can be abstracted to a *Semi MDP* [24] according to the options $O_{\mathcal{P}}$. Instead of planning a policy in $\mathcal{P}$, the robots can subsequently select one of the options $o_i \in O_{\mathcal{P}}$ according to $\mathcal{I}_i$, follow its policy π_i until termination given by β_i, and then select the next option.

At this point, the system can incorporate a multi-robot team as follows. Instead of using only one robot to follow a certain policy, multiple robots of the team can each select one of the options and execute its respective policy in parallel. It remains to allocate the options to the robotic agents such that satisfaction of the scLTL goal is guaranteed.

Algorithm 2 Sequential Single-Option Auctions (SSO)

Input: Agent MDP $\mathcal{M}$, mission DFA $\mathcal{D}$
Output: Next option o for this agent
Notation Remarks:
ρ – index of the agent which executes this algorithm
$\widehat{s} \in [0,1]^{|S_{\mathcal{M}}|}$ – estimate of the current state, with $\sum_i \widehat{s}_i = 1$
D_r – total duration of options assigned to robot r
$q_{i^*} \in Q$ – DFA state resulting from task i^*

$q_c, \widehat{s} \leftarrow initialize()$ ▷ Current state, Eq. (5)
while $q_c \notin F$ **do** ▷ Repeat until satisfaction
▶ Evaluate available options, c.f. Sec. VI
  for all $i \colon (q_c, s_{\mathcal{M}}) \in \mathcal{I}_i, \forall s_{\mathcal{M}}$ **do** ▷ Eq. (1)
    $\pi_i, \widehat{D}_{\pi,i} \leftarrow planPolicy(\mathcal{O}_i^c)$ ▷ Def. 10
    $d'_{\rho i} \leftarrow estimateDuration(\widehat{s}, \widehat{D}_{\pi,i})$ ▷ Eq. (6)
    $d_{\rho i} \leftarrow \max(\bigcup_{r \neq \rho}\{D_r\} \cup \{D_\rho + d'_{\rho i}\})$ ▷ Lem. 4
▶ Select best assignment of next task
  $\{d_{ri}\} \leftarrow communicate(\{d_{\rho i}, \forall i\})$
  $r^*, i^* \leftarrow \arg\min\{d_{ri}\}$ ▷ Minimize duration
▶ Predict state after executing the assigned task
  $q_c \leftarrow q_{i^*}$ ▷ Update team progress
  if $r^* = \rho$ **then**
    if o not set **then** $o \leftarrow (\mathcal{I}_{i^*}, \pi_{i^*}, \beta_{i^*})$ ▷ Def. 9
    $\widehat{s} \leftarrow updateState(\widehat{s})$ ▷ Eq. (7)
return o ▷ If o is not set, agent ρ will do nothing

To plan this allocation, we follow the idea of the *sequential single-item auction* algorithm [14, 10], which provides an efficient decentralized allocation approach. However, the usual assumption of this algorithm that the single tasks are independently executable cannot be made here and thus, this algorithm cannot be applied in its original form. Instead, we see from the initiation set $\mathcal{I}_i$ that the respective option o_i can only be chosen at a subset of states. Consequently, o_i can have a *dependency* on other options meaning that its initiation set $\mathcal{I}_i$ can only be reached by executing these specific other options before.

In Algorithm 2, we propose a method where the agents auction over the options they can choose, while respecting their dependencies. Each agent in the algorithm keeps track of the predicted global DFA state $q_c \in Q$ and estimates its system state in the Agent MDP $\mathcal{M}$. Then, the agents synchronously repeat a three phase procedure.

In the first phase, the agents plan and evaluate their available options, based on their state estimate. The resulting estimated durations of completing the respective options form the bids of the agents. In the second phase, the agents communicate their bids and select the cheapest bid. In the third phase, after the winning bid has been selected, all agents update the mission progress to the target of the winning option. Also, the single agent that won the bid estimates in which system state distribution its selected

option will terminate. This procedure is repeated until an accepting state is reached. Alternatively, it can be terminated as soon as every agent is assigned at least one option.

In detail, progress and state estimate $\widehat{s} \in [0,1]^{|S_{\mathcal{M}}|}$ are initialized by the respective current states from $\mathcal{D}$ and $\mathcal{M}$, given by the current DFA state q_c and the system state

$$\widehat{s}_{s_{\mathcal{M}}} = \begin{cases} 1 & \text{if } s_{\mathcal{M}} = s_{0,\mathcal{M}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\widehat{s}_{s_{\mathcal{M}}}$ denotes the probability of being in state $s_{\mathcal{M}}$ according to the estimate $\widehat{s}$. In line 5, the durations $d'_{\rho i}$ of reaching q_i are calculated for each of the options o_i available given the current mission progress q_c, based on the duration estimates $\widehat{D}_{\pi,i}$ given by

$$d'_{\rho i} = \sum_{s \in S_{\mathcal{M}}} \widehat{s}_s \widehat{D}_{\pi,i}(s). \tag{6}$$

However, the resulting total duration to complete o_i, denoted by $d_{\rho i}$, needs as well to consider the maximum over the durations required by all agents. This accounts for possible dependencies on the options assigned to other agents and, as shown by Lemma 4 below, reflects that an option cannot be completed with lower duration than its dependencies.

Next (line 7), the agents share their required durations $d_{\rho i}$ and select a winning assignment by minimizing these durations. Since this assignment will change the DFA state q_c, only a single assignment is decided in each iteration.

The state update in line 12 is calculated by first calculating the transformation matrix B given by Equation (4). Then, the updated state estimate $\widehat{s}'$ is given by

$$\begin{aligned} \widehat{s}'_{[n]} &= 0_{\mathrm{n}} \\ \widehat{s}'_{[m]} &= \widehat{s}_{[m]} + B^{\mathrm{T}} \widehat{s}_{[n]} \end{aligned} \tag{7}$$

where 0_{n} denotes the n-dimensional zero vector, $\widehat{s}_{[n]}$ the n entries in $\widehat{s}$ corresponding to transient states and $\widehat{s}_{[m]}$ the m entries corresponding to absorbing states.

We conclude by closer investigating two key aspects of Algorithm 2. First, the influence of the dependency on previously completed options is incorporated as follows.

Lemma 4 (Option Dependencies)**.** *Assume an assignment of options to agents by Alg. 2 and D_r denoting the respective total duration for agent $r \in \{1, \dots, N\}$. Furthermore, denote by R the set of robots with highest duration $D_r = D_{\max} = \max(D_1, \dots, D_N)$ for all $r \in R$ and by $d'_{\rho i}$ the additional duration for robot ρ to execute an option o_i. Then, at least one of the following statements is true:*

1) $\forall \rho \colon D_\rho + d'_{\rho i} \geq D_{\max}$

2) o_i depends on o_j and o_j assigned to any $r \in R$

Proof. In line 8, the option with minimal increase on the total execution duration is selected in each bidding phase. Since o_i has not been assigned in an earlier iteration, either the cost increase by assigning o_i is higher than for the already assigned options (case 1), or o_i has not been available for assignment in an earlier iteration. The latter case implies a dependency of o_i on the option o_j last assigned to any $r \in R$ and leading to $D_{\max}$ (case 2). □

Second, it can be shown that Algorithm 2 terminates after a finite number of iterations and an upper bound on the number of required options can be calculated.

Theorem 1 (Finite Satisfiability)**.** *Given the set of available options as defined by the initiation set in Equation* (1)*, the number of subsequent options required to satisfy $\mathcal{D}$ is finite and an upper bound is given by $|Q| - 1$.*

Proof. As given by Equation (1), each option increases the level $L(q)$ of its goal state q. Consequently, at most $L_{\max} := \max\{L(q), q \in Q\}$ options are selected. Furthermore, Lemma 1 implies $L_{\max} \leq |Q| - 1$. □

Consequently, the progress level assigned by Algorithm 1 can indeed be understood as denoting an upper bound on the required number of options which need to be executed in order to achieve a mission progress corresponding to the respective DFA state.

VIII. Task Execution

After the allocation terminates, each agent is either assigned one option o_i for which it can follow the policy π_i, or it is not required at the moment. However, during the auction, the robots assume future DFA states to plan policies which they can follow to prepare future parts of the mission. Consequently, it still needs to be guaranteed that following these policies does not violate the mission specification in its actual current DFA state.

For this reason, the set of actions A_π over which the policy π_i is defined are restricted in the following two ways for all options assuming a future task progress. First, all actions that would violate the specification for the current progress q are removed. This corresponds to limiting the set of actions to only actions which satisfy the self-loop condition at q, given by

$$\widetilde{A}_\pi = \{(s, s') \in A_{\mathcal{O}} \colon \lambda(s') \vDash \delta(q, q)\}. \tag{8}$$

Second, a policy depending on anticipated future progress cannot be completed before the required progress has actually been made. Therefore, also actions which make progress in the mission are removed, given by

$$A_\pi = \{((q, s), (q', s')) \in \widetilde{A}_\pi \colon q' \in \mathrm{SCC}(q)\}. \tag{9}$$

As a consequence of both Equations (8) and (9), the robots will prepare their respective options as much as they can and then wait for the other robots if required.

Due to the uncertainty in the environment, it is unclear in advance how exactly the execution will continue in the future. To address this, the robots can update their followed policies in two ways during execution, each including again execution of Algorithm 2 to update allocation.

1) Progress. Whenever the option of an agent terminates, the current state $q \in Q$ of each other agent needs to be updated. Since this causes the state of each agent to change, the previously followed option is interrupted.

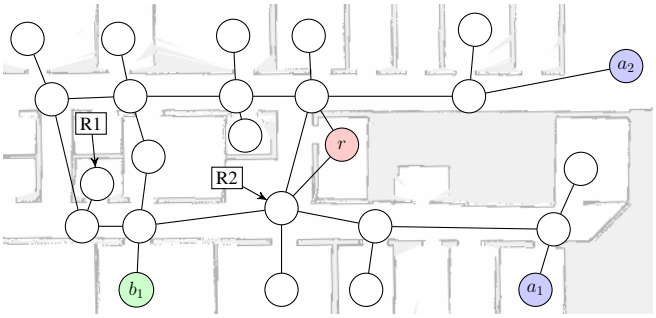

Fig. 2. Topological map of the environment to represent navigation actions, while nodes are labeled with atomic propositions.

2) Observation. We incorporate the knowledge collected by the robots during execution. As discussed above, the event probabilities P_S given in Definition 6 denote the prior expectation that the respective event can be observed. Therefore, the robots temporarily update these probabilities during execution to reflect their actual observations.

This procedure of planning policies according to the options, allocation of these options to the robots, and execution to reach the next DFA state is repeated until an accepting DFA state is reached to satisfy the scLTL goal.

IX. Case Study

The theoretical framework presented in this paper has been implemented as a multi-robot planning system in ROS. We use the open-source components *Spot*[2] [8] for LTL translation and *FlexBE*[3] [22] for action implementation and execution, as well as *Gazebo*[4] for simulation. The robots communicate via ROS topics for the auction algorithm and for environment updates, e.g., by using a multi-master setup when executed on the real robots.

We illustrate application of the proposed framework in a case study scenario based on the environment depicted in Figure 2. The proposition r denotes a room with access restrictions which requires permission by two persons, provided for example by entering an access code via the touch display of the robots. The locations denoted by a_i and b_i indicate where the required persons can be found with some probability. For this case study, we assume that the persons can be found at a_2 and b_1, but this is not known to the robots in advance. Furthermore, we assume for simplicity that the expected duration of a navigation action is proportional to the distance between two locations.

The goal to access room r while respecting the access restrictions is expressed by the scLTL formula

$$\phi = \Diamond r \wedge (\neg r \,\mathcal{U}\, a_\mathrm{c}) \wedge (\neg r \,\mathcal{U}\, b_\mathrm{c})$$

where a_c, b_c denote that the confirmation has been given. The DFA constructed from ϕ is shown in Figure 3.

We get the following qualitative results after sending the scLTL mission to the robots[5]. Initially, R1 gets assigned

[2]See http://spot.lrde.epita.fr [3]See http://flexbe.github.io
[4]See http://gazebosim.org
[5]A video showing the execution of this case study and the two variations discussed below is provided under https://youtu.be/OJ9E2fs79qE.

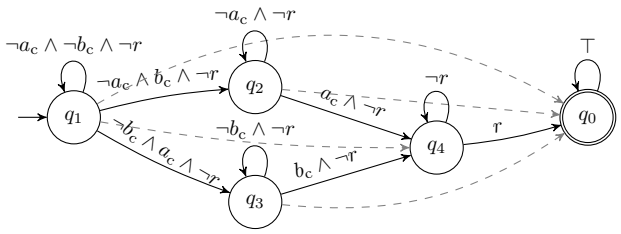

Fig. 3. DFA constructed from ϕ. Transitions are simplified to those which are feasible in the given environment, e.g., a_c and b_c are never true at the same MDP state (infeasible transitions are indicated by dashed arrows).

option o_{12} leading from q_1 to q_2 and R2 gets assigned the option o_{24} based on the assumed task progress q_2. Consequently, the policy of R1 leads the robot to b_1 and the policy of R2 leads towards a_1. Note here that R2 decides to go to a_1 instead of a_2 because it is closer and we assume the same probabilities. Also, note that R2 only moves towards a_1, but its initial policy does not let it actually reach a_1, resulting from Equation (8).

The first replanning (Fig. 4 left) occurs after R1 reached b_1, observed b_c, and thus, completed its option to reach mission progress q_2. Since R2 is already close to a_1, it gets assigned the first option o_{24} which it is now allowed to complete, while R1 gets assigned the option o_{40} to then complete the mission. Now, assuming that a_c cannot be observed at a_1, R2 fails to complete o_{24} as expected and updates the event probability $P_S(e_{a_\mathrm{c}}, a_1) = 0$ to indicate that a_c is not observed at a_1 (Fig. 4 middle).

Resulting from the model update, R1 is now the robot with a lower cost solution for o_{24} by going to a_2. Instead, R2 gets assigned o_{40}. Since R2 reaches the waypoint next to c faster than R1 can complete o_{24}, it can be observed that R2 waits for R1 (Fig. 4 right). After R1 finished o_{24}, R2 keeps the assigned o_{40}, which it already almost completed, and finishes the mission by entering c.

A considerable benefit of the presented coordination approach is that the option auctions enable the robots to anticipate the future state of the others and consequently, prepare future parts of the mission. To quantify this benefit, we compared our approach "SSO" to a simpler allocation strategy were no mission progress is predicted and each robot is simply assigned the one available option which it can satisfy with lowest duration, denoted by "NBO". As shown in Table I, a significantly lower mission completion time can be achieved by our approach. While the particular speed-up depends on the specific mission, the advantage becomes especially relevant for increasingly complex goals.

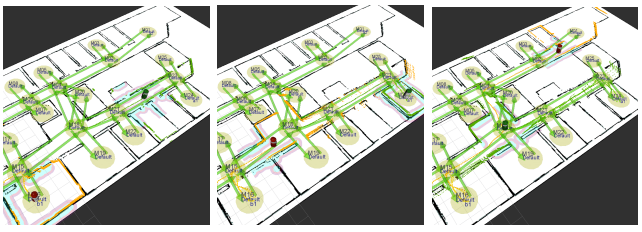

Fig. 4. RViz screenshots of the simulated system during execution of ϕ.

TABLE I

TIME REQUIRED BY THE APPROACHES TO FULFILL THE MISSION ϕ.

Approach	Robots	Mission Time
NBO	2	140.3 sec
SSO (ours)	2	98.5 sec
SSO (ours)	5	49.4 sec

In addition, we show that SSO scales to more agents due to the decentralized nature of the auction allocation. Indeed, the time required for the mission can be reduced by adding more agents. Note that *Mission Time* in Table I includes all time required for planning and coordination. Specifically, in the two-robots case, a robot required at most 3.1 seconds of planning time in total, using on average 1.8% CPU during these times. Although the planning time is slightly higher in the five-robots case (5.1 seconds, 1.3% CPU), potentially influenced by simulating the additional robots, this indicates that the computational complexity of more robots is clearly not as limiting as it would be the exponential size of a centralized product between the robots.

X. CONCLUSIONS

The framework presented in this paper was motivated by the need for close coordination of multi-robot teams under uncertainty, in particular for an efficient algorithm to plan policies for fulfilling single temporal logic mission specifications given to the team as whole. Using the proposed methods, a scenario as presented in the case study can automatically be executed by the robots without the need to manually assign parts of the specification to individual agents. In particular, it is not required to manually define sub-tasks of the specification and additionally available robots can already prepare future parts of the mission. Furthermore, updating the planned policies to include observations during execution enables the robots to flexibly adapt to the uncertainties in the environment. Nevertheless, considering uncertainty about specification violation and thus, incorporating a probability of option failure in the task allocation procedure, remains a topic for future work.

REFERENCES

[1] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *IEEE Int. Conf. on Robotics and Automation*, pages 1241–1248, 2015.

[2] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press Cambridge, 2008.

[3] C. Belta, B. Yordanov, and E. A. Gol. *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.

[4] R. E. Burkard and E. Cela. Linear assignment problems and extensions. In *Handbook of combinatorial optimization*, pages 75–149. Springer, 1999.

[5] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *Transactions on Robotics*, 25(4):912–926, 2009.

[6] X. C. D. Ding, S. L. Smith, C. Belta, and D. Rus. LTL control in uncertain environments with probabilistic satisfaction guarantees. *IFAC*, 44(1):3515–3520, 2011.

[7] Xuchu Ding, Mircea Lazar, and Calin Belta. LTL receding horizon control for finite deterministic systems. *Automatica*, 50(2):399–408, 2014.

[8] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *ATVA*. Springer, 2016.

[9] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[10] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain. The power of sequential single-item auctions for agent coordination. In *National Conference on Artificial Intelligence*, volume 21, page 1625. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[11] O. Kupferman and M. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[12] B. Lacerda, D. Parker, and N. Hawes. Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. In *Int. Conf. on Intelligent Robots and Systems*, pages 1511–1516. IEEE, 2014.

[13] B. Lacerda, D. Parker, and N. Hawes. Multi-Objective Policy Generation for Mobile Robots Under Probabilistic Time-Bounded Guarantees. *ICAPS 17*, 2017.

[14] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain. Auction-Based Multi-Robot Routing. In *Robotics: Science and Systems*, pages 343–350. Rome, Italy, 2005.

[15] X. Li and C. Belta. A Hierarchical Reinforcement Learning Method for Persistent Time-Sensitive Tasks. *arXiv preprint arXiv:1606.06355*, 2016.

[16] F. S. Melo and M. Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

[17] M. Mesbahi and M. Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[18] A. Nikou, J. Tumova, and D. V. Dimarogonas. Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. In *20th World Congress of the Int. Fed. of Automatic Control*, 2017.

[19] D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2): 774–793, 2007.

[20] S. M. Ross. *Applied Probability Models with Optimization Applications*. Holden Day, San Francisco, 1970.

[21] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *CDC*, pages 1091–1096. IEEE, 2014.

[22] P. Schillinger, S. Kohlbrecher, and O. von Stryk. Human-Robot Collaborative High-Level Control with Application to Rescue Robotics. In *IEEE Int. Conf. on Robotics and Automation*, Stockholm, Sweden, May 2016.

[23] P. Schillinger, M. Bürger, and D. V. Dimarogonas. Multi-Objective Search for Optimal Multi-Robot Planning with Finite LTL Specifications and Resource Constraints. In *IEEE Int. Conf. on Robotics and Automation*. Singapore, 2017.

[24] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[25] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[26] J. Tumova and D. V. Dimarogonas. A receding horizon approach to multi-agent planning from local LTL specifications. In *Am. Control Conference*, pages 1775–1780. IEEE, 2014.

[27] K. W. Wong and H. Kress-Gazit. Let's talk: Autonomous conflict resolution for robots carrying out individual high-level tasks in a shared workspace. In *Int. Conf. on Robotics and Automation*, pages 339–345. IEEE, 2015.