Probabilistic Plan Synthesis for Coupled Multi-Agent Systems [⋆]

Alexandros Nikou [*] **Jana Tumova** [**] **Dimos V. Dimarogonas** [*]

[*] *ACCESS Linnaeus Center, School of Electrical Engineering and KTH Center for Autonomous Systems, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden.*
E-mail: {*anikou, dimos*}*@kth.se*
[**] *School of Computer Science and Communication, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden.*
E-mail: {*tumova*}*@kth.se*

Abstract: This paper presents a fully automated procedure for controller synthesis for multi-agent systems under the presence of uncertainties. We model the motion of each of the N agents in the environment as a Markov Decision Process (MDP) and we assign to each agent one individual high-level formula given in Probabilistic Computational Tree Logic (PCTL). Each agent may need to collaborate with other agents in order to achieve a task. The collaboration is imposed by sharing actions between the agents. We aim to design local control policies such that each agent satisfies its individual PCTL formula. The proposed algorithm builds on clustering the agents, MDP products construction and controller policies design. We show that our approach has better computational complexity than the centralized case, which traditionally suffers from very high computational demands.

1. INTRODUCTION

Over the last few years, the field of control of multi-agent systems under high-level task specifications has been gaining attention. In this work, we aim to impose specific probability bounds to each robot in order to satisfy one specification task formula. Such formulas can be "The probability of a robot to periodically survey regions A, B, C, avoid region D is always more than 0.9", or "The probability of a robot to visit regions A, B, C, in this order is more that 0.8".

Temporal properties for multi-agent plan synthesis under Linear Temporal Logic (LTL) as well as Metric Interval Temporal Logic (MITL) formulas has been considered e.g., in [Guo and Dimarogonas, 2013, Ulusoy et al., 2013, Kantaros and Zavlanos, 2016, Quottrup et al., 2004, Karaman and Frazzoli, 2008, Nikou et al., 2016, 2017b,a].

Most of the existing formal synthesis frameworks are based on the discretization of the agent's motion in a partitioned environment to a finite Transition System (TS) (the process is called abstraction) under the following assumptions. First, the measurements of the current state are accurate. Second, the transition system is either purely deterministic (namely, each control action enables a unique transition) or purely nondeterministic (namely, each control action enable multiple transitions). However, in realistic applications of robotic systems, noisy sensors and actuators can cause both of the aforementioned assumptions to fail. Motivated by this, we aim to model the multi-agent system in a probabilistic way such that the above issues are taken into consideration.

Some recent works model the system in a probabilistic way and imposes high-level specifications, given in Linear Temporal Logic (see e.g., [Ding et al., 2011, Wolff et al., 2012, Ding et al., 2014, Fu et al., 2015, Wang et al., 2015]). In [Montana et al., 2016], the authors modeled the system with an MDP and computed policies such that the satisfaction of a formula given in MITL, is maximized. Other works model the system in a probabilistic way with Markov Decision Processes (MDPs) and introduce high-level tasks in PCTL (see e.g., [Ayala et al., 2011, Lahijanian et al., 2012, Wu and Lin, 2016]). However, all these works are restricted to single agent planning and they are not expendable to multi-agent systems in a straightforward way since in the multi-agent case potential couplings may occur among the agents.

By extending these works to multi-agent systems, in order to develop an approach in which the system noise, model errors and external disturbances is explicitly considered, in this work we consider that each agent is modeled as a MDP and the task specifications are given in PCTL formulas. Motivated by the fact that in real applications, the agents (robots) are required to collaborate with each other to perform a task, we assume that there are agents in the system that are dependent to each other. They need to communicate, collaborate through sharing a common action in order to achieve a desired task.

The main contribution of the paper is to develop a strategy for controlling a general framework of N individual MDPs with respect to individual agents' specifications given in PCTL formulas. The proposed solution can handle the dependencies between the agents by considering agents clustering and MDP product construction and has provably better complexity than the centralized approach. When applied to robotic systems, our approach provides a framework for multi-robot control from temporal logic specifications with probabilistic guarantees. To the best of the authors' knowledge this is the first work that addresses the cooperative task planning for multi-agent systems under probabilistic temporal logic specifications in the presence of dependencies between the agents. Due to space constraints, a more detailed version of this paper

[⋆] This work was 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.

that contains examples for definitions, proofs of theorems, and omitted calculations, can be found in [Nikou et al., 2017c].

2. NOTATION AND PRELIMINARIES

Denote by $\mathbb{N}$ the set of natural numbers. Given a set S, denote by $|S|$ its cardinality and by 2^S the set of all its subsets. Denote by $\bigtimes_{i=1}^{N} S_i = S_1 \times \ldots \times S_N$ the n-th fold Cartesian product of the sets $S_1, \ldots, S_N$. An *atomic proposition* χ is a statement that is either True ($\top$) or False ($\bot$).

Definition 1. A *probability distribution* over a countable set S is a function $\sigma : S \to [0,1]$ satisfying $\sum_{s \in S} \sigma(s) = 1$. Define by $\Sigma(S)$ the set of all probability distributions over the set S.

Definition 2. A Discrete Time Markov Chain (DTMC) $\mathcal{D}$ is a tuple (S, s_0, P) where: S is a finite set of states; $s_0 \in S$ is the initial state; $P : S \times S \to [0,1]$ is the transition probability matrix where for all $s \in S$ it holds that $\sum_{s' \in S} P(s, s') = 1$.

Definition 3. A Markov Decision Process (MDP) $\mathcal{M}$ is a tuple (S, s_0, Act, T) where: S is a finite set of states; $s_0 \in S$ is the initial state; Act is a finite set of actions (controls); $T : S \to 2^{Act \times \Sigma(S)}$ is the transition probability function.

Definition 4. (Control Policy) A *control policy* $\mu : FPath \to Act$ of an MDP model $\mathcal{M}$ is a function mapping a finite path $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} s_n$, of $\mathcal{M}$ onto an action in $\mathcal{A}(s_n)$ and specifies for every finite path, the next action to be enabled. If a control policy depends only on the last state of the finite path ρ, then it is called a *stationary policy.*

Denote by M the set of all control policies. Under a control policy $\mu \in M$, an MDP becomes a DTMC $\mathcal{D}_\mu$ (see Def. 2). Let $IPath_\mu \subseteq IPath$ and $FPath_\mu \subseteq FPath$ denote the set of infinite and finite paths that can be produced under the control policy μ. For each policy $\mu \in M$, a probability measure $Prob_\mu$ over the set of all paths (under the control policy μ) $IPath_\mu$ is induced. A probability measure $Prob_\mu^{fin}$ over the set of paths $FPath_\mu$ for a finite path ρ, is defined as: $Prob_\mu^{fin}(\rho) = 1, \text{if } |\rho| = 0$ and $Prob_\mu^{fin}(\rho) = P(s_0, s_1)P(s_1, s_2) \ldots P(s_{n-1}, s_n)$, otherwise, where $P(s_k, s_{k+1}), k \geq 0$ are the corresponding transition probabilities in $\mathcal{D}_\mu$.

Probabilistic Computational Tree Logic (PCTL) [Hansson and Jonsson, 1994] is used to express properties of MDPs. PCTL formulas can be recursively defined as follows:

$$\varphi := \top \mid \chi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathcal{P}_{\bowtie p}[\psi], \qquad (state\ formulas)$$
$$\psi := \bigcirc\varphi \mid \varphi_1 \, \mathcal{U}^{\leq k} \, \varphi_2, \qquad (path\ formulas)$$

where $\chi \in Act$ is an action, $\bowtie = \{<, >, \leq, \geq\}$, $p \in [0,1]$ and $k \in \mathbb{N} \cup \{\infty\}$. In the syntax above, we distinguish between state formulas φ and path formulas ψ, which are evaluated over states and paths, respectively. A property of a model will always be expressed as a state formula; path formulas only occur as the parameter of the probabilistic path operator $\mathcal{P}_{\bowtie p}[\psi]$. Intuitively, a state s satisfies $\mathcal{P}_{\bowtie p}[\psi]$ (we write $s \models \mathcal{P}_{\bowtie p}[\psi]$) if there exists a control policy μ under which the probability of all paths starting from s is in the range of the interval $\bowtie p$. For path formulas, we allow the "next" ($\bigcirc$) operator which is true if the state formula φ is satisfied in the next state and the "bounded until" ($\mathcal{U}^{\leq k}$) which is true if φ_2 is satisfied within k steps and φ_1 holds up until that point. The unbounded "until" operator $\mathcal{U}$ is the same as $\mathcal{U}^{\leq k}$ as $k \to \infty$.

For preliminaries background regarding Discrete Time Markov Chains, Markov Decision Processes, PCTL and Probabilistic Control Synthesis, we refer the reader to [Nikou et al., 2017c, Section 2].

3. PROBLEM FORMULATION

3.1 System Model and Abstraction

Consider a multi-agent team with $N \geq 2$ agents operating in the bounded workspace $\mathcal{W} \subseteq \mathbb{R}^n$. Let $\mathcal{V} = \{1, \ldots, N\}$ denote the index set of the agents. The workspace $\mathcal{W} = \bigcup_{\ell \in \mathcal{W}} \gamma_\ell$ is partitioned using a finite number (assume W) of regions of interest $\gamma_1, \ldots, \gamma_W$. Denote by γ_ℓ^i the fact that the agent i is occupying the region γ_ℓ, where $i \in \mathcal{V}, \ell \in \mathcal{W}$.

We assume that each agent is programmed with a small set of feedback control primitives, which are not completely reliable, allowing it to move inside each region and from a region to an adjacent regions. It is also assumed that the probabilities of these transitions are known.

Assumption 1. We assume here that an abstraction of the dynamics of each robot into a MDP is given, and that a low level continuous time controller that allows each robot to transit from one region γ_ℓ to an adjacent region γ_l with $\ell, l \in \mathcal{W}$, can be designed.

This modeling has been also considered in [Lahijanian et al., 2012].

Definition 5. The motion of each agent $i \in \mathcal{V}$ in the workspace can be described by an MDP $\mathcal{M}_i = (S_i, s_0^i, Act_i, T_i)$ where $S_i = \left\{\gamma_1^i, \gamma_2^i, \ldots, \gamma_W^i\right\}$ is the set of states of agent i. The number of states for each agent is $|S_i| = W$, meaning that S_i includes all regions within $\mathcal{W}$; $s_0^i \in S_i$ is the initial state of agent i (the initial region where agent i may start). Note that the initial state is known and deterministic i.e. we know exactly the region from which each agent starts its motion; Act_i is a finite set of actions (controls); $T_i : S_i \to 2^{Act_i \times \Sigma(S_i)}$ is the transition probability function.

Remark 1. We investigate here, under which conditions two or more agents are visiting simultaneously a specific region of the workspace. Consider the $\{i_1, \ldots, i_c\} \subseteq \mathcal{V}, c \geq 2$ agents of the multi-agent system. Let $r_{i_1} = s_0^{i_1} s_1^{i_1} s_2^{i_1} \ldots s_k^{i_1} \ldots s_n^{i_1}, \ldots, r_{i_c} = s_0^{i_c} s_1^{i_c} s_2^{i_c} \ldots s_k^{i_c} \ldots s_n^{i_c}$, be the finite paths of length n that are executed by the corresponding MDPs $\mathcal{M}_1, \ldots, \mathcal{M}_c$, respectively, where $s_z^j \in S_z, \forall z \geq 0, j \in \{i_1, \ldots, i_c\}$. Then, if there exists $k \geq 1$ such that for all the k-th elements of the above runs (in the same position at every path) it holds that $s_k^{i_1} = \cdots = s_k^{i_c} = s_k^{\text{meet}}$, then we say that the agents $\{i_1, \ldots, i_c\}$ are visiting simultaneously the same region s_k. If there does not exist such region s_k^{meet}, then the agents cannot meet simultaneously to one region.

3.2 Handshaking Actions

The motivation for introducing dependencies in the multi-agent system comes from real applications where more than one agents (robots) need to collaborate with each other in oder to perform a desired task. For example, imagine two aerial manipulators that are required to meet and grasp an object simultaneously and deliver it to a specific location in a warehouse.

In order to be able to introduce dependencies in the actions between the agents, we write the action set of each agent as: $Act_i = \{\Pi_i, \hat{\Pi}_i\}, i \in \mathcal{V}$, where Π_i is a finite set of actions that the agent i need to execute in collaboration with other agents (*handshaking* actions) and $\hat{\Pi}_i$ is a finite set of actions that the agent i executes independently of the other agents (*independent* actions). For the independent actions it holds that: $\hat{\Pi}_i \cap \hat{\Pi}_j = \emptyset, \forall i \neq j, i, j \in \mathcal{V}$.

The independent actions can always be executed without any constraints. On the other hand, for the handshaking actions, we have the following requirements: First, the agents are required to meet and occupy the same region of the workspace (not necessarily a specific region); Once they meet, they need to execute simultaneously the same action; All the agents that share an action are required to execute it in order for the task to be completed properly.

Formally, the handshaking actions are defined as follows:

Definition 6. (Handshaking Actions) Let $\{i_1, \ldots, i_c\} \subseteq \mathcal{V}, c \geq 2$ be a set of agents that need to collaborate in order to execute simultaneously a task under the action α. The following two properties should hold in order for α to be well-posed handshaking action: (1) $\alpha \in \bigcap_{i \in \{i_1, \ldots, i_c\}} \Pi_i$; (2) Let the following finite paths of length n: $r_{i_1} = s_0^{i_1} \xrightarrow{\alpha_0^{i_1}} \ldots \xrightarrow{\alpha_{k-1}^{i_1}} s_k^{i_1} \xrightarrow{\alpha} s_{k'}^{i_1} \ldots \xrightarrow{\alpha_{n-1}^{i_1}} s_n^{i_1} \ \ldots \ r_{i_c} = s_0^{i_c} \xrightarrow{\alpha_0^{i_c}} \ldots \xrightarrow{\alpha_{k-1}^{i_c}} s_k^{i_c} \xrightarrow{\alpha} s_{k'}^{i_c} \ldots \xrightarrow{\alpha_{n-1}^{i_c}} s_n^{i_c}$ be executed by the MDPs $\mathcal{M}_{i_1}, \ldots, \mathcal{M}_{i_c}$ respectively. Here, $s_k^{i_1}, \ldots, s_k^{i_c}$ are the regions that the $i_1, \ldots, i_c$ should occupy in order to execute the handshaking action α simultaneously. Then, there should exist at least one $k \geq 0$ such that $s_k^{i_1} = \cdots = s_k^{i_c} = s_k^{\text{meet}}$ and $\delta(s_k^j, \alpha, s_{k'}^j) > 0$ for at least one $s_{k'}^j \in \text{Post}(s_k^j, \alpha)$ for every $j \in \{i_1, \ldots, i_c\}$.

Notice that the same condition for a state s_k^{meet} as in condition (2) was mentioned in Remark 1, but here the existence of a common action α is also required. It should be also noted that every region of the workspace in which the agents can potentially meet, can serve as a region that a handshaking action can be executed (if such an action exists).

3.3 Dependencies

Suppose that one agent i receives a cooperative task that involves other agent's $j \in \mathcal{V} \backslash \{i\}$ participation. This means that both agents need to execute the same action at the same region so as for the task to be performed. The dependencies are formally defined as:

Definition 7. The agents $i, j \in \mathcal{V}$ are called *dependent* if one the following statements holds: (1) Agent i depends on agent j if $\Pi_i \cap \Pi_j \neq \emptyset$; (2) Agent i depends on agent j if $\Pi_i \cap \Pi_j \neq \emptyset$; (3) There exist at least one region $s_k^{\text{meet}}, k \geq 0$ of the workspace such that the second condition of Def. 6 holds.

Conditions (1),(2) can be checked by comparing all the elements of the sets $\Pi_i, \Pi_j, \forall i, j \in \mathcal{V}$ one by one. Condition (3) can be checked by using graph search algorithms.

Remark 2. It should be noticed from the above definitions that all the agents that share an action, they are required to meet and execute it simultaneously.

Remark 3. Due to fact that the control policies are defined over finite paths, the handshaking actions are defined with respect to finite paths as well. Therefore, the graph search algorithm for condition (3) is searching into a finite graph.

Assumption 2. It is assumed that there exists at least 2 agents that are dependent. Otherwise, there exist no dependencies between the agents and the problem that is later defined can be straightforwardly solved by solving the controller synthesis methodology of Section 4 for each agent independently.

3.4 Problem Statement

We define here the problem that we aim to solve in this paper:

Problem 1. Given N agents performing in the workspace $\mathcal{W}$, under the Assumptions 1, 2, individual task specification formulas $\varphi_1, \ldots, \varphi_N$ over the actions $\Pi_i \cup \hat{\Pi}_i, i \in \mathcal{V}$ given in PCTL, synthesize individual control policies $\mu_1, \ldots, \mu_N$ (if there exists one) which guarantee the satisfaction of the formulas $\varphi_1, \ldots, \varphi_N$ respectively.

4. PROBLEM SOLUTION

4.1 Overview

An overview of the proposed solution is given as follows:

- Step 1: First, the dependencies among the agents are modeled as a dependency graph (see Section 4.2). The agents are split into clusters and each cluster contains the agents that are dependent according to Def. 7.
- Step 2: For each cluster of agents, the mutual specification φ_m and the product MDP $\widetilde{M}$ are defined (see Section 4.3).
- Step 3: By utilizing the controller synthesis algorithms of Section 4.6, we design a control policy $\widetilde{\mu}$ of each cluster that guarantees the satisfaction of φ_m (if such a control policy exists). We provide in Sec. 4.4 the definition of successful control policies, which project onto local control policies $\mu_1, \ldots, \mu_N$ for each agent, which finally are a solution to Problem 1.

An algorithm describing all the steps of the proposed procedure is given in Section 4.5. Probabilistic model checking algorithms, which can compute all the control policies under which a PCTL formula φ_m is satisfied, are presented in Section 4.6. The computational complexity of the proposed framework is discussed in Section 4.7.

4.2 Modeling the Dependencies

Based on the dependency relation of the Def. 7, the dependency graph associated with the handshaking actions $\Pi_i, i \in \mathcal{V}$ is defined as follows:

Definition 8. The *dependency graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is an undirected graph that consists of the set of vertices $\mathcal{V}$ in which each of the agents is node for the graph and the edge set $\mathcal{E}$ which is defined as follows: $\mathcal{E} = \{\{i, j\} : i \text{ is dependent to } j \text{ and } i, j \in \mathcal{V}, i \neq j\}$.

In order to proceed, the following definition is required:

Definition 9. [Mesbahi and Egerstedt, 2010] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Then every graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$ if called a *subgraph* of the graph $\mathcal{G}$.

Definition 10. The set $\mathcal{C} = \{C_\ell : \ell \in \mathbb{M}\} \subseteq \mathcal{V}$, where $\mathbb{M} = \{1, \ldots, m\}$, forms a *dependency cluster* if and only if $\forall i, j \in \mathcal{C}$ there is a path from node i to node j in the dependency graph $\mathcal{G}$.

Define the function $f : \mathcal{V} \rightarrow \mathbb{M}$ which maps each agent to the index of the cluster that it belongs to. It can be observed that $\bigcup_{\ell\in\mathbb{M}} C_\ell = \mathcal{V}$ and $\sum_{\ell\in\mathbb{M}} |C_\ell| = |\mathcal{V}| = N$.

Each agent $i \in \mathcal{V}$ for which there exist no $j \in \mathcal{V}$ such that $j \in C_{f(i)}$ will be called an *independent agent.* For an independent agent it holds that $|C_{f(i)}| = 1$.

From Definition 10, it follows that every dependency cluster $C_\ell \in C, \ell \in \mathbb{M}$ is the vertex set of the subgraphs $G^{(\ell)} = (\mathcal{C}_\ell, \mathcal{E}_\ell), \mathcal{E}_\ell \subseteq \mathcal{E}, \ell \in \mathbb{M}$ of the system graph $\mathcal{G}$.

Loosely speaking, two agents belong to the same cluster when they are directly dependent or transitively dependent by a dependency chain. An example of a dependency graph and dependency clusters is given as follows: According to the mathematical derivation above, Assumption 2 is modified as follows:

Assumption 3. There exists at least one dependency cluster $C_\ell, \ell \in \mathbb{M}$ (as was defined in Definition 10) of the dependency graph $\mathcal{G}$ of the under consideration multi-agent system, which contains at least two dependent agents. I.e., there exists $\ell \in \mathbb{M}$ such that: $|C_\ell| = 2, \text{if } N = 2$ and $|C_\ell| \in [2, N-1], \text{if } N > 2$.

By employing the above computation, the initial multi-agent system is modeled as m subgraphs $\mathcal{G}^\ell, \ell \in \mathbb{M}$ which capture the dependencies between the agents, as they are defined in Def. 7. This forms a convenient modeling of the system's dependencies in order to compute the product MDP of every subsystem $\ell \in \mathbb{M}$ in the next Section.

4.3 Product Markov Decision Process

Define here the *mutual specification* of a cluster of agents C_ℓ as:

$$\varphi_m^\ell = \bigwedge_{i\in C_\ell} \varphi_i, \ell \in \mathbb{M}, \tag{1}$$

over the set of actions $\bigcup_{i\in C_\ell} \left(\Pi_i \cup \hat{\Pi}_i\right)$. If the satisfaction of φ_m^ℓ for each cluster C_ℓ is guaranteed, it holds by definition that the satisfaction of all the individual formulas $\varphi_i, i \in \mathcal{I}$ is guaranteed as well. Thus, a method for finding a team control policy that guarantees the satisfaction of $\varphi_m^\ell, \ell \in \mathbb{M}$ should be provided.

In the sequel, we construct a product MDP that captures the collaborative behavior of all the agents within a cluster. Having $\widetilde{M_\ell}$, allow us to synthesize a control policy $\widetilde{\mu}$ for C_ℓ, which guarantees the satisfaction of the collaborative formula φ_m^ℓ. Subsequently, the team control policy $\widetilde{\mu}_\ell$ can be projected onto the local agents' control policies $\mu_1, \ldots, \mu_N$ which are a solution to Problem 1.

Definition 11. (Product MDP) The *product MDP* $\widetilde{\mathcal{M}_\ell}$ for the cluster of agents C_ℓ is a tuple $(\widetilde{S}_\ell, \widetilde{s}_0^\ell, \widetilde{Act_\ell}, \widetilde{T}_\ell)$ where: $\widetilde{S}_\ell = \bigtimes_{i\in C_\ell} S_i$ is the set of states; $\widetilde{s}_0^\ell = \bigtimes_{i\in C_\ell} s_0^i$ is the initial state; $\widetilde{Act_\ell} = \bigcup_{i\in C_\ell} Act_i = \bigcup_{i\in C_\ell} \left\{\Pi_i, \hat{\Pi}_i\right\}$ is the set of actions; $\widetilde{T}_\ell : \widetilde{S}_\ell \rightarrow 2^{\widetilde{Act_\ell}\times\Sigma(\widetilde{S}_\ell)}$ is the transition probability function for the product system. Similar to δ of Def. 3, we define $\widetilde{\delta}(\widetilde{s}, \widetilde{\alpha}, \widetilde{s}') \in [0, 1]$ the probability of transitioning from the state $\widetilde{s}$ to the state $\widetilde{s}'$ under the action $\widetilde{\alpha}$. Let $C_\ell = \{i_1, \ldots, i_{|C_\ell|}\}$ be an enumeration of the agents of the cluster C_ℓ. Then, $\widetilde{\delta}_\ell$ is defined as follows: (1) $\widetilde{\delta}((s_{i_1}, \ldots, s_{i_{|C_\ell|}}), \alpha, (s'_{i_1}, \ldots, s'_{i_{|C_\ell|}})) = \prod_{j\in C_\ell} \delta_j(s_j, \alpha, s'_j)$, if $\alpha \in \bigcap_{j\in C_\ell} \mathcal{A}(s_j)$; (2) $\widetilde{\delta}((\widetilde{s}_{i_1}, \ldots, \widetilde{s}_{k_1}, \ldots, \widetilde{s}_{k_\nu}, \ldots, \widetilde{s}_{i_{|C_\ell|}}),$ $\alpha, (\widetilde{s}_{i_1}, \ldots, \widetilde{s}'_{k_1}, \ldots, \widetilde{s}'_{k_\nu}, \ldots, \widetilde{s}_{i_{|C_\ell|}})) = \prod_{j=1}^{\nu} \delta_{k_j}(s_{k_j}, \alpha, s'_{k_j})$ if $\alpha \in \left[\bigcap_{j=1}^{\nu} \mathcal{A}(s_{k_j})\right] \Bigg\backslash \left[\bigcup_{z\in C_\ell\setminus\{k_1,\ldots,k_\nu\}} \mathcal{A}(s_z)\right]$, for $k_j \in C_\ell, j \in \{1, \ldots, \nu\}$.

Intuitively, (1) denotes that all the agents $i_1, \ldots, i_{|C_\ell|}$ of the cluster $|C_\ell|$ are located in the states $s_{i_1}, \ldots, s_{i_{|C_\ell|}}$ respectively, and they are simultaneously transiting to the states $s'_{i_1}, \ldots, s'_{i_{|C_\ell|}}$ with action α. (2) denotes that among all the agents of the cluster C_ℓ, only the agent $\{k_1, \ldots, k_\nu\} \subsetneq C_\ell$ are transiting simultaneously to the states $s'_{k_1}, \ldots, s'_{k_\nu}$ respectively. (2) can not be handshaking action since for the handshaking action all the agents of the cluster should transit simultaneously to the next state. In order for (1) to be a handshaking transition according to Def. 6 it is required also that $s_{i_1} = \ldots = s_{i_{|C_\ell|}}$.

Remark 4. In the case of a cluster $\ell \in \mathbb{M}$ that contains an independent agent $i \in \mathcal{I}$ with the property $|C_\ell| = |C_{f(i)}| = 1$, the product MDP $\widetilde{\mathcal{M}_\ell}$ coincides with the individual MDP $\mathcal{M}_i$ of Def. 5 ($\widetilde{\mathcal{M}_\ell} \equiv \mathcal{M}_i$).

The infinite path $\widetilde{r}$, the finite path $\widetilde{\rho}$, the control policy $\widetilde{\mu}$ and the set of all infinite and finite paths $\widetilde{FPath}$ and $\widetilde{IPath}$, are defined similarly to Sec. 2.

4.4 Designing the Control Policies $\widetilde{\mu}$

The product MDP $\widetilde{M_\ell}, \ell \in \mathbb{M}$ of each cluster captures the paths and the control policies of the agents that belong to the same cluster and they are required to collaborate for achieving a task or acting independently. By employing the controller synthesis algorithms (see Section 4.6), the control policies $\widetilde{\mu}_\ell$ for the team of agents in each cluster can be designed. The algorithms can compute all the control policies $\widetilde{\mu}_\ell$ that guarantees the satisfaction of formula φ_m^ℓ from (1). What remains is to project these policies onto the individual control policies of the agents of each cluster in such a way that they serve as a solution to Problem 1.

Consider a cluster of agents $C_\ell = \{i_1, \ldots, i_{|C_\ell|}\}$. A control policy $\widetilde{\mu}_\ell(\widetilde{\rho}) = \widetilde{\mu}_\ell(\widetilde{s}_0\widetilde{s}_1 \ldots \widetilde{s}_n) \subseteq \widetilde{Act}$ for the finite path $\widetilde{\rho} = \widetilde{s}_0\widetilde{s}_1 \ldots \widetilde{s}_n$ of length n, where $\widetilde{s}_k = (s^k_{i_1}, \ldots, s^k_{i_{|C_\ell|}}), k \in \{1, \ldots, n\}$, projects onto the local individual control policies $\mu_j(s^1_j, \ldots, s^n_j), j \in \{i_1, \ldots, i_{|C_\ell|}\}$, of the agents $\{i_1, \ldots, i_{|C_\ell|}\}$ of the cluster $C_\ell, \ell \in \mathbb{M}$. Note that: $\mu_j \subseteq \widetilde{Act}\Bigg|_j \subseteq Act_j, j \in \{i_1, \ldots, i_{|C_\ell|}\}$, and $\widetilde{Act}\Bigg|_j$ is the set of actions of the agent j that are appearing in the s et $\widetilde{Act}$.

The set $\widetilde{\mu}(\widetilde{\rho})$ contains control policies that are either handshaking or independent. Let us also define the following set of handshaking actions: $\text{Succ}(\alpha, \ell) = \{\alpha \in \Pi_{i_1} \cap \cdots \cap \Pi_{i_{|C_\ell|}} : \alpha \in \widetilde{\mu}_\ell(\widetilde{\rho})\}$, which is the subset of $\widetilde{\mu}(\widetilde{\rho})$ that contains the handshaking actions. We need to search now if all the projections $\mu_{i_j}, \forall j \in \{1, \ldots, |C_\ell|\}$ follow the handshaking rules of Def. 6.

Definition 12. (Successful Control Policy) Let $\widetilde{\mu}_\ell(\widetilde{\rho}) = \widetilde{\mu}_\ell(\widetilde{s}_0\widetilde{s}_1 \ldots \widetilde{s}_n) \subseteq \widetilde{Act}$ be a control policy of a cluster

Algorithm 1 - SolveProblem1($\cdot$)

```
1:  Input: MDPs: $\mathcal{M}_1, \ldots, \mathcal{M}_N$;
2:         PCTL Formulas: $\varphi_1, \ldots, \varphi_N$
3:  Output: $\mu_1, \ldots, \mu_N$
4:
5:  $\mathcal{C} = \{C_\ell, \ell \in \mathbb{M}\} = \text{checkDepend}(Act_1, \ldots, Act_N)$
6:  $\varphi_m^\ell = \bigwedge_{i \in C_\ell} \varphi_i$
7:  for $z \in C_\ell = \{i_1, \ldots, i_{|C_\ell|}\}$ do
8:      $\widetilde{M}_\ell = \text{product}(\{\mathcal{M}_j, j \in C_\ell\})$
9:      $SP(\ell) = \text{controlSynthesis}(\widetilde{M}, \varphi_m^\ell)$
10:     for $\widetilde{\mu}_\ell \in SP(\ell)$ do
11:         $\{\mu_1, \ldots, \mu_N\} = \text{projection}(\widetilde{M}_\ell)$
12:         if succPolicy($\{\mu_1, \ldots, \mu_N\}$) = $\top$ then
13:             solFound = 1
14:             return $\{\mu_1, \ldots, \mu_N\}$          ▷ Solution found
15:         else
16:             go to 12        ▷ Search other control policies
17:         end if
18:     end for
19:     if solFound $\neq$ 1 then
20:         Problem 1 has no solution
21:     end if
22: end for
```

C_ℓ. The control policy $\widetilde{\mu}_\ell(\widetilde{\rho})$ is called *successful* if for all $\alpha \in \text{Succ}(\alpha, \ell)$ it holds that $s_{i_1}^n = \cdots = s_{i_{|C_\ell|}}^n$ and $\delta(s_j^n, \alpha, (s_j^n)') > 0$ for at least one $(s_j^n)' \in \text{Post}(s_j^n, \alpha), j \in \{i_i, \ldots, i_{|C_\ell|}\}$.

Let $SP(\ell) = \left\{\widetilde{\mu}_\ell(\widetilde{\rho}) \subseteq \widetilde{Act} : \widetilde{M}_\ell \models \varphi_m^\ell\right\}, \ell \in \mathbb{M}$, denotes the set of all the control policies that guarantee the satisfaction of the formula φ_m^ℓ. All the control policies of SP needs to be checked if they are successful. If $SP(\ell) = \emptyset$ for at least one $\ell \in \mathbb{M}$, then the Problem 1 has no solution. The set $SP(\ell)$ is computed by employing the algorithms of Section 4.6.

4.5 Proposed Algorithm

The proposed procedure of solving Problem 1 can be shown in Algorithm 1. The function checkDepend() determines the dependent agents according to Def. 7. The product and projection that were introduced in Sec. 4.3, 4.4, are computed by the functions product(), projection() respectively. The algorithms of Sec. 4.6 are incorporated in the function controlSynthesis(). By employing Def. 12, the function succPolicy() determines if a sequence of control policies are successful.

4.6 Algorithms for Probabilistic Control Synthesis

We are investigating here algorithms of computing all the control policies $\widetilde{\mu}_\ell \in SP(\ell)$. Once these control policies are found, then by following Algorithm 1, the individual policies $\mu_j, j \in \{i_1, \ldots, i_{|C_\ell|}\}$ can be designed and the Problem 1 is solved (if there exists a solution). For more details about the algorithms we refer to [Rutten et al., 2004, Kwiatkowska et al., 2007, Forejt et al., 2011, Bianco and Alfaro, 1995, Lahijanian et al., 2012]

First, define $\text{Sat}(\varphi_m^\ell) = \{s \in S : s \models \varphi_m^\ell\}$ as the set of states that satisfy φ_m^ℓ. Then we have: $\text{Sat}(\top) = S$, $\text{Sat}(\pi) = \{s \in S : \pi \in \mathcal{A}(s)\}$, $\text{Sat}(\neg\varphi_m^\ell) = S \backslash \text{Sat}(\varphi_m^\ell)$, $\text{Sat}(\varphi_{m,1}^\ell \wedge \varphi_{m,2}^\ell) = \text{Sat}(\varphi_{m,1}^\ell) \cap \text{Sat}(\varphi_{m,2}^\ell)$ for two PCTL formulas $\varphi_{m,1}^\ell$, $\varphi_{m,2}^\ell$. Define also the minimum and the maximum probabilities of satisfying the formula under the control policy μ for a starting state s:

$$Prob_{\max}(s, \psi) = \sup_{\mu \in M} \{Prob_\mu(s, \psi)\}, \tag{2a}$$

$$Prob_{\min}(s, \psi) = \inf_{\mu \in M} \{Prob_\mu(s, \psi)\}. \tag{2b}$$

where M is set of all control policies. It has been proved in [Bianco and Alfaro, 1995], that the model checking problem problem of the operator $\mathcal{P}_{\bowtie p}[\psi]$ can be reduced to the computation of (2a), (2b) according to the following:

- If $\bowtie = \{\geq, >\}$ then
$$s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow Prob_{\min}(s, \psi) \bowtie p \tag{3}$$
- If $\bowtie = \{\leq, <\}$ then
$$s \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow Prob_{\max}(s, \psi) \bowtie p \tag{4}$$

For the controller synthesis of the path operators $\mathcal{P}_{\bowtie p}[\bigcirc \varphi_m^\ell]$, $\mathcal{P}_{\bowtie p}[\varphi_m^\ell \mathcal{U}^{\leq k} \varphi_m^\ell]$, we utilize the Algorithms 2,3 respectively as follows:

Algorithm 2) If the formula is $\varphi_m^\ell = \mathcal{P}_{\bowtie p}[\bigcirc \varphi_{m,1}^\ell]$, initially the maximum probability of satisfying φ_m^ℓ at the state $s \in S$:

$$Prob_{\max}(s, \varphi_m^\ell) = \max_{\alpha \in \mathcal{A}(s)} \left(\sum_{s' \in Sat(\varphi_{m,1}^\ell)} \delta(s, \alpha, s') \right), \tag{5}$$

is computed for every $s \in S$. By replacing max with min in (5), $Prob_{\min}(s, \bigcirc \varphi_{m,1}^\ell)$ can be computed. Define the vector $\Phi(s) = 1$, if $s \in Sat(\varphi_{m,1}^\ell)$ or $\Phi(s) = 0$, otherwise. Perform now the matrix multiplication $X = T \cdot \Phi$. X is a vector whose entries are the probabilities of satisfying $\bigcirc \varphi_1$, where each row corresponds to a state-action pair. After obtaining the vector X, eliminate the state-actions pairs whose probabilities are not in the range of $\bowtie p$ by taking into consideration the conditions (3), (4). This operation determines all the states $s \in S$ and all the actions $\mu \in M$ that satisfy the formula φ_m^ℓ.

Algorithm 3) For the formula of the form $\phi_m^\ell = \mathcal{P}_{\bowtie p}\left[\varphi_{m,1}^\ell \, \mathcal{U}^{\leq k} \, \varphi_{m,2}^\ell\right]$, define by $S^{\text{yes}} = \text{Sat}(\varphi_{m,2}^\ell)$, $S^{no} = S \backslash \left[\text{Sat}(\varphi_{m,1}^\ell) \cup \text{Sat}(\varphi_{m,2}^\ell)\right]$, and $S^{rem} = S \backslash (S^{yes} \cup S^{\text{no}})$ the states that always satisfy the specification, the states that never satisfy the specification and the remaining states, respectively. Compute the maximum probability of satisfying φ_m^ℓ at the state $s \in S$ as: $Prob_{\max}(s, \varphi_m^\ell) = 1$ or 0, if $s \in S^{\text{yes}}$ or $s \in S^{\text{no}}$ respectively. For $s \in s \in S^{\text{rem}}$ and $k \geq 0$ compute recursively the following:

$$Prob_{\max}(s, \varphi_m^\ell, k) = \max_{\alpha \in \mathcal{A}(s)} \left(\sum_{s' \in S^{rem}} \delta(s, \alpha, s') Prob_{\max}(s, \varphi_m^\ell, k-1) + \sum_{s' \in S^{yes}} \delta(s, \alpha, s') \right) \tag{6}$$

with $Prob_{\max}(s, \varphi_m^\ell, 0) = 0$. The computation can be carried out in k iterations, each similar to the process of Algorithm 2. By replacing max with min in (6), $Prob_{\min}(s, \varphi_m^\ell, k)$ can be computed.

Algorithm 4) The form $\phi_m^\ell = \mathcal{P}_{\bowtie p}\left[\varphi_{m,1}^\ell \, \mathcal{U} \, \varphi_{m,2}^\ell\right]$ is in fact the same as $\phi_m^\ell = \mathcal{P}_{\bowtie p}\left[\varphi_{m,1}^\ell \, \mathcal{U}^{\leq k} \, \varphi_{m,2}^\ell\right]$ as $k \to \infty$.

With this approach, Algorithm 3 can be used to solve this problem.

Remark 5. The resulting control strategies of the aforementioned algorithms are stationary. Therefore, the control policies $\widetilde{\mu}_\ell(\widetilde{s}_1\widetilde{s}_2\dots\widetilde{s}_n)$ depend only to the state $\widetilde{s}_n$.

4.7 Computational Complexity

According to [Baier et al., 2008], the model checking of an MDP $\mathcal{M}$ is polynomial in the number of states of $\mathcal{M}$ and linear in the length of the formula φ. Denote by $|\varphi|$ the length of the formula φ in terms of the number of the operator it has e.g., $|\mathcal{P}_{\geq 0.5}[\bigcirc\{red\}]| = 2$. The complexity can be expressed mathematically as $\mathcal{O}(\text{poly}(|\mathcal{M}|)|\varphi|\kappa(\varphi))$, where $\kappa(\varphi) = \max\{k : \phi_1\mathcal{U}^{\leq k}\varphi_2\}$, φ_1, φ_2 are subformulas of φ and $\varphi_1\mathcal{U}^{\leq k}\varphi_2$ are possible until operators involving in φ. Define also $\text{poly}(n) = 2^{\mathcal{O}(\log(n))}$. If φ does not contain a bounded until operator then $\kappa(\varphi) = 1$. The number of states of the the product MDP in the centralized solution is $|\widetilde{S}| = \prod_{i\in\mathcal{I}}|S_i| = W^N$ and the corresponding complexity is in the class of $O = \mathcal{O}\big(2^{\mathcal{O}(N\log(W))}\cdot|\varphi_m^\ell|\cdot\kappa(\varphi_m^\ell)\big)$, where φ_m^ℓ as it is defined in (1) for $|C_\ell| = N$. The worst case complexity of the proposed framework is when 1 agent is independent and the other $N-1$ agents are dependent to each other. Then, there are two clusters $\ell\in\{1,2\}$: the first contains the independent agent and the other one contains the remaining agents. The corresponding MDPs have $|\widetilde{S}_\ell| = |\bigtimes_{i\in C_\ell} S_i| = W^{|C_\ell|}, \ell\in\{1,2\}$ states i.e., W, W^{N-1} states respectively. Thus, the worst case complexity of our framework is: $\widetilde{O} = \mathcal{O}\big(2^{\mathcal{O}(\log(W))}\cdot|\varphi_m^1|\cdot\kappa(\varphi_m^1) + 2^{\mathcal{O}((N-1)\log(W))}\cdot|\varphi_m^2|\cdot\kappa(\varphi_m^2)\big)$. The best case complexity of the proposed framework is when every agent is dependent to at most one other agent. Formally, if N is odd number then $|C_{\ell'}| = 1$ for only one $\ell'\in\mathbb{M}$ and $|C_\ell| = 2, \forall\ell\in\mathbb{M}\backslash\{\ell'\}$. In this case, the best case complexity is in the class: $\bar{O} = \mathcal{O}\Bigg(\sum_{\ell\in\mathbb{M}\backslash\{\ell'\}}[2^{\mathcal{O}((N-1)\log(W))}\cdot|\varphi_m^\ell|\cdot\kappa(\varphi_m^\ell)] + 2^{\mathcal{O}(\log(W))}\cdot|\varphi_m^{\ell'}|\cdot\kappa(\varphi_m^{\ell'})\Bigg)$. If N is even number, then previous summation in performed in all the elements $\ell\in\mathbb{M}$. In total, it holds that: $\bar{O} < \widetilde{O} < O$ which verifies that our proposed framework achieves significantly better computational complexity than the centralized one.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a systematic method for designing control policies for multi-agent systems. We assume that the system is under the presence of model uncertainties and actuation failures, thus the modeling is performed through MDPs. The agents are divided into dependency clusters which indicate the team of agents that they need to share an action in order to achieve a desired task. With the proposed framework, each agent is guaranteed to perform a task given in PCTL formulas. The computational complexity of the proposed framework is significantly better than the complexity of the centralized framework. Future efforts will be devoted towards performing the abstraction of the stochastic system which is given according to Assump. 1.

REFERENCES

Ayala, A., Andersson, S., and Belta, C. (2011). Temporal Logic Control in Dynamic Environments with Probabilistic Satisfaction Guarantees. *IROS.*

Baier, C., Katoen, J., and Larsen, K.G. (2008). Principles of model checking. *MIT Press.*

Bianco, A. and Alfaro, L. (1995). Model Checking of Probabilistic and Nondeterministic Systems. *ICFSTTCS.*

Ding, X., Smith, S., Belta, C., and Rus, D. (2011). LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees. *IFAC WC.*

Ding, X., Smith, S., Belta, C., and Rus, D. (2014). Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *TAC.*

Forejt, V., Kwiatkowska, M., Norman, G., and Parker, D. (2011). Automated Verification Techniques for Probabilistic Systems. *FMENSS.*

Fu, J., Han, S., and Topcu, U. (2015). Optimal control in markov decision processes via distributed optimization. *55th IEEE Conference on Decision and Control (CDC).*

Guo, M. and Dimarogonas, D. (2013). Reconfiguration in Motion Planning of Single-and Multi-Agent Systems Under Infeasible Local LTL Specifications. *CDC.*

Hansson, H. and Jonsson, B. (1994). A Logic for Reasoning about Time and Reliability. *FA of Computing.*

Kantaros, Y. and Zavlanos, M. (2016). A Distributed LTL-Based Approach for Intermittent Communication in Mobile Robot Networks. *ACC.*

Karaman, S. and Frazzoli, E. (2008). Vehicle Routing Problem with Metric Temporal Logic Specifications. *CDC*, 3953–3958.

Kwiatkowska, M., Norman, G., and Parker, D. (2007). Stochastic Model Checking. *ISFMDCCSS.*

Lahijanian, M., Andersson, S., and Belta, C. (2012). Temporal Logic Motion Planning and Control with Probabilistic Satisfaction Guarantees. *TRO.*

Mesbahi, M. and Egerstedt, M. (2010). Graph Theoretic Methods in Multiagent Networks. *Princ. Univer. Press.*

Montana, F., Liu, J., and Dodd, T. (2016). Sampling-Based Stochastic Optimal Control with metric interval temporal logic specifications. *CCA.*

Nikou, A., Andersson, S., and Dimarogonas, D.D. (2017a). Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications. *IFAC WC.*

Nikou, A., Boskos, D., Tumova, J., and Dimarogonas, D.D. (2017b). Cooperative Planning Synthesis for Coupled Multi-Agent Systems Under Timed Temporal Specifications. *ACC, Seattle, WA, USA.*

Nikou, A., Tumova, J., and Dimarogonas, D. (2016). Cooperative Task Planning of Multi-Agent Systems Under Timed Temporal Specifications. *ACC*, 13–19.

Nikou, A., Tumova, J., and Dimarogonas, D.D. (2017c). Probabilistic Plan Synthesis for Coupled Multi-Agent Systems. *https://arxiv.org/abs/1704.01432.*

Quottrup, M., Bak, T., and Zamanabadi, R. (2004). Multi-Robot Planning: A Timed Automata Approach. *ICRA*, 5, 4417–4422.

Rutten, J., Kwiatkowska, M., Norman, G., and Parker, D. (2004). Mathematical Techniques for Analyzing Concurrent and Probabilistic systems. *Am. Math. Society.*

Ulusoy, A., Smith, S., Ding, X., Belta, C., and Rus, D. (2013). Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints. *IJRR.*

Wang, J., Ding, X., Lahijanian, M., Paschalidis, I., and C. Belta, C.A. (2015). Temporal Logic Motion Control Using Actor–Critic Methods. *IJRR.*

Wolff, E., Topcu, U., and Murray, R. (2012). Robust Control of Uncertain Markov Decision Processes with Temporal Logic Specifications. *CDC.*

Wu, B. and Lin, H. (2016). Counterexample-Guided Distributed Permissive Supervisor Synthesis for Probabilistic Multi-Agent Systems Through Learning. *ACC.*