Control Engineering Practice 123 (2022) 105130

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Security-preserving multi-agent coordination for complex temporal logic tasks☆

Xinyi Yu [a,b], Xiang Yin [a,b], Shaoyuan Li [a,b,*], Zhaojian Li [c]

[a] Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
[b] Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China
[c] Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Keywords:
Discrete event systems
Formal methods
Multi-agent systems
Cyber-security

ABSTRACT

This paper investigates the coordination of multiple agents for high-level tasks described by linear temporal logics (LTL). The general purpose for multi-agent coordination is to synthesize a plan such that the LTL task is achieved optimally. In addition to the standard requirement on the correctness of the plan, we further investigate the potential *information leakage* of each agent during the operating process. Specifically, we consider the scenario where the behavior of each individual agent is partially monitored by a passive intruder modeled as an outside observer or an *eavesdropper*. The security constraint requires that the intruder can never identify for sure that some specific individual agent is carrying out some sub-tasks of significant importance. To this end, we model the mobile capability of the agent team by a global labeled transition system. To describe information-flow security constraint, motivated by the generic notion of opacity, two different types security requirements are proposed for each individual agent. An effective coordination algorithm is proposed that synthesizes an optimal global plan for the entire agent team such that the global LTL task can be achieved, while the security of each individual agent is preserved. The proposed framework is implemented in real-world experiment and is also demonstrated by several case studies.

1. Introduction

1.1. Background and motivation

Tasks planning are the keys towards the collaborations and coordination of multi-agent teams. Previous researches on multi-agent coordination mainly focus on low-level tasks such as collision avoidance and point-to-point navigation (Karaman & Frazzoli, 2011; LaValle, 2006). Recent years, temporal logics, such as linear temporal logics (LTL) and computation tree logics (CTL) have drawn considerable attention in the agent planning literature due to their rich capabilities in describing complex high-level planning objectives such as liveness, safety and surveillance (Basile, Chiacchio, & Di Marino, 2019; Fanti, Mangini, Pedroncelli, & Ukovich, 2018; Kress-Gazit, Lahijanian, & Raman, 2018). By abstracting the motion capabilities of autonomous agents to the discrete domain, automata-theoretic approaches can be used to effectively synthesize high-level plans such that the temporal logic task specifications can be fulfilled. Such high-level plans can be then implemented by low-level hybrid controllers to control the physical dynamic of the system, which provides a fully-automated and correct-by-construction formal planning framework.

In order to achieve complex global tasks, each agent in the team may need to communicate with each other via networks, e.g., to share its local information, or may need to acquire/transmit information from/to a third party, e.g., to download/upload data. However, malicious cyber-attacks may be performed on the communication networks such that some critical information of the system can be leaked, which will threaten the security and privacy of each agent. Roughly speaking, cyber-attacks can be classified into active attacks and passive attacks. For active attacks, the intruder can actively disrupt the communication channels between each agent. This includes, e.g., denial-of-service attacks and deception attacks; the interested reader is referred to the comprehensive (Ding, Han, Ge, & Wang, 2020) for more recent developments on this topic. In this paper, we consider the case of passive attacks. Specifically a passive intruder will not affect the system directly. Instead, it will passively observe some information output from the system for the purpose of inferring and reasoning about the behavior of the system.

Taking manufacturing factory as a motivating example, let us consider a working scenario where a team of agents want to assemble a machine collaboratively as shown in Fig. 1. In particular, one agent

☆ This work was supported by the National Key Research and Development Program of China (2018AAA0101700) and the National Natural Science Foundation of China (62061136004, 62173226, 61833012) and Shanghai Jiao Tong University Scientific and Technological Innovation Funds, China.

* Corresponding author at: Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail addresses: yuxinyi-12@sjtu.edu.cn (X. Yu), yinxiang@sjtu.edu.cn (X. Yin), syli@sjtu.edu.cn (S. Li), lizhaoj1@egr.msu.edu (Z. Li).

https://doi.org/10.1016/j.conengprac.2022.105130
Received 18 August 2021; Received in revised form 14 January 2022; Accepted 16 February 2022
Available online 7 March 2022

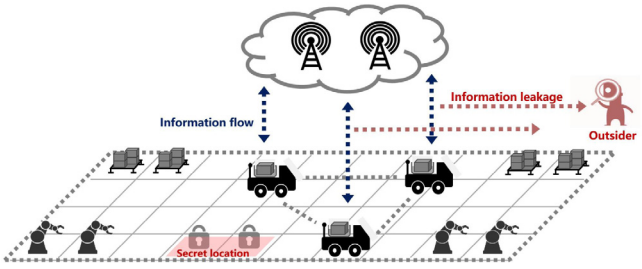


Fig. 1. Illustration of problem background.

in the team needs to go to the secure station in red base to get the blueprint and the other agents need to collect different pieces of components and transfer them to each corresponding places. At the same time, some collective behavior and communication information of the agent team may be partially observed by an outsider, e.g., via using some additional sensors or eavesdropping communications. In this case, which agent is carrying out the blueprint becomes a very sensitive secure information of the entire system. If the outsider is able to figure out this information by its information available, then potential malicious attack might be performed on the specific agent targeted.

Therefore, security and privacy issues have been becoming important concerns in agent networks and security constraints also need to be considered in the task planning and coordinating algorithm. Essentially, this requires the designer to achieve a tradeoff between the *utility* of the system in terms of the LTL task and the *security* of the system in terms of its generated information-flow. This paper works along this line of research by developing security-preserving task planning algorithms for autonomous multi-agent systems.

1.2. Related works

High-level task planning for mobile agents using formal methods has drawn considerable attention in the literature; see, e.g., Kloetzer and Belta (2009), Kress-Gazit, Fainekos, and Pappas (2009), Plaku and Karaman (2016). In the context of task coordination for multi-agent systems, following the sensor-based framework proposed in Kress-Gazit et al. (2009), Raman and Kress-Gazit (2014) proposes an approach for synthesizing reactive controllers of a team of agents. In Ulusoy, Smith, Ding, Belta, and Rus (2013), the authors consider the optimal path planning problem under temporal logic constraints. In Cai, Peng, Li, and Kan (2020), Kantaros, Malencia, Kumar, and Pappas (2020), LTL task planning under partially unknown environment has also been investigated. In order to mitigate the computational complexity in multi-agent planning, efficient planning algorithms have also been proposed, see, e.g., distributed approach (Guo & Dimarogonas, 2015; Moarref & Kress-Gazit, 2020), sampling-based approach (Kantaros & Zavlanos, 2020; Oh, Cho, Choi, & Oh, 2020) and Petri nets-based approach (Kloetzer & Mahulea, 2020; Mahulea & Kloetzer, 2017). Other types of formal methods such as the supervisory control of discrete-event systems have also been applied to multi-agent system for the purpose of ensuring high-level specifications (Goryca & Hill, 2013; Hill & Lafortune, 2017; Mahulea, Kloetzer, & González, 2020) and has been successfully applied to, e.g., warehouse systems (Tatsumoto, Shiraishi, Cai, & Lin, 2018) and agricultural field agents (Ju & Son, 2019). However, above mentioned works only focus on the correctness of high-level behaviors; security issues during the coordination are not considered.

More recently, some researchers have started to investigate how to preserve information-flow security in the agent task planning problem; see, e.g., Hadjicostis (2018), Ma and Cai (2021), O'Kane and Shell (2015), Saboori and Hadjicostis (2011), Wang, Nalluri, and Pajic (2020), Yang, Yin, Li, and Zamani (2020). In Hadjicostis (2018), Saboori and Hadjicostis (2011), the notion of opacity is adopted to characterize the information-flow security requirement of the system and the planning task is described as a reachability problem. In our recent work, we further consider a general class of LTL tasks under the opacity constraint. Efficient algorithms without building the observer structure are proposed. In O'Kane and Shell (2015), the authors address the problem of designing information transmission policies for agents subject to discreetness constraints. It is shown in Wang et al. (2020) certain types of security constraints in the planning problem can be encoded as HyperLTL specifications. These works are closely related to our setting as the security requirements are all motivated from the general notion of opacity (Ji, Yin, & Lafortune, 2019; Lafortune, Lin, & Hadjicostis, 2018; Lin, 2011; Tong, Li, Seatzu, & Giua, 2018). However, the above mentioned works only consider the security issue in a single overall agent system. Here, we focus on the multi-agent scenario where the security and privacy for each individual agent in the team are considered. Such a scenario is much more sophisticated and different types of security customized to the multi-agent setting are proposed to capture different security requirements.

In summary, existing works on multi-agent high-level coordination mainly focus on the correctness of the tasks; the information-flow security issue has not been considered. For secure-aware task planning, existing notions of security as well as the corresponding planning algorithms are only suitable for single-agent systems. How to coordinate multi-agent systems for high-level specifications while preserving security is still a challenging problem that has not yet been fully solved. The main difficulty in this problem is how to define security notions that are suitable for multi-agent systems and how to compute the optimal strategies that preserve security. These issues will be tackled in this work.

1.3. Our contributions

Motivated by the security and privacy concerns in multi-agent systems, in this paper, we formulate and solve a security-aware path planning problem for complex temporal logic tasks. Following the standard framework, we consider a multi-agent system whose motion capability is modeled as a global labeled transition system, which is the composition of each individual agent. Agents need to collaboratively achieve a global task specified by an LTL formula. We consider a passive intruder monitoring the system's behavior via a generic observation mapping. Our goal is to generate a plan for the agent team such that the LTL task is achieved with the least cost while some secure information of each individual agent is preserved against the outside observer. To formal capture the information-flow security issue for each individual agent during the entire task cooperation, we proposed two new security requirements. Specifically, we use a particular set of states to model the secret behavior of each individual agent. Our goal is to make sure that the intruder can never determine for sure that a specific agent has visited secret locations, which is referred to as the Type I security, or cannot uniquely suspect any specific agent for executing the secret behavior, which is referred to as the Type II security.

Since the system is partially observed by the intruder, standard methodologies for handling this type of problem require the subset construction to track consistent information. However, this yields computational issues as the subset construction results in an exponentially state–space in the size of the global systems. Our main approach is based on a computationally efficient procedure by constructing a new structure called the multiple-labeling-GTS, which synchronizes the entire system with its several copy patterns based on the observation from the intruder: half copy patterns for Type I security and a half for Type II security. By performing shortest path searches over the product of the multiple-labeling-GTS and the Büchi automaton characterizing the LTL specification, an optimal plan that is both secure and specification achieving is synthesized. We show that the proposed security-preserving planning algorithm is both sound and complete.

Table 1
Main symbols and definitions.

Symbol	Definition
$\mathcal{I}$	Index set of agents
k	Total number of agents
Π	Set of all regions in the workshop
$\mathcal{AP}$	Set of atomic propositions
T_i	Transition system for agent $i \in \mathcal{I}$
T_g	Transition system for global system
τ_g	Infinite path in T_g
$trace(\tau_g)$	Trace of infinite path τ_g
$Path^{\omega}(T_g)$	Set of all infinite paths in T_g
$Path^{*}(T_g)$	Set of all finite paths in T_g
$J(\tau_g)$	Cost of finite path τ_g
φ	LTL formula
A_{φ}	Nondeterministic Büchi automaton for φ
H	Output function
Y	Set of output symbols
Π_s	Set of secret states
Π_{ns}	Set of non-secret states
$\hat{J}(\tau_g)$	Total cost of an infinite path τ_g
$\hat{T}_g$	Labeling-GTS
V	Multiple-labeling-GTS
$T_{\otimes}$	Product system
$Goal(T_{\otimes})$	Set of all accepting states
$S(T_{\otimes})$	Set of all secure accepting states

1.4. Organization

The rest of the paper is organized as follows. The main symbols and definitions are summarized in Table 1. The system model is described in Section 2 and we present the privacy requirements and formulate the problem in Section 3. Section 4 focuses on the solution algorithm of the problem and Section 5 discusses the correctness and complexity of the proposed algorithm. The overall framework is demonstrated by the simulation and experiment in Section 6 and Section 7 concludes the whole work.

2. System model

In this section, we introduce the formal model for the mobility of the multi-agent system as well as the LTL-based complex task requirement.

2.1. Global transition system

We consider a team of k heterogeneous agents and denote by $\mathcal{I} = \{1, \dots, k\}$ as the index set of agents. All agents are working in the same workspace $\Pi \subseteq \mathbb{R}^2$, which is partitioned as N regions $\Pi = \{\pi_1, \dots, \pi_N\}$. The motion capability of each agent $i \in \{1, \dots, k\}$ in the workspace is abstracted as a finite weighted transition system (wTS), which is a 6-tuple

$$T_i = (\Pi_i, \Pi_{i,0}, \rightarrow_i, w_i, \mathcal{AP}_i, L_i),$$

where Π_i is a finite set of states representing all regions, $\Pi_{i,0} \subseteq \Pi_i$ is a set of initial states from which the agent may start, $\rightarrow_i \subseteq \Pi_i \times \Pi_i$ is a transition relation, $w_i : \Pi_i \times \Pi_i \rightarrow \mathbb{R}_+$ is a cost function specifying the moving cost between different regions, $\mathcal{AP}_i$ is a set of atomic propositions specifying some important properties of our interest, and $L_i : \Pi_i \rightarrow 2^{\mathcal{AP}_i}$ is a labeling function that specifying what properties hold at each region. The readers are referred to Belta, Yordanov, and Gol (2017), Kloetzer and Belta (2007) for how to construct wTS for physical systems. In principal, the wTS T_i is determined by both the connectivity of the workspace, the motion ability and the mapping information of the agent i. Note that T_i and T_j may be different even though they work in the same workspace.

In this work, we assume that all agents are running synchronously, which can be implemented by using a global clock and each agent needs to confirm the completion of other agent's movement before the next period. Under this setting, the mobility of the entire agent team can be modeled by a global transition system (GTS), denoted by T_g, which is the synchronous product of each local agent T_i.

Definition 2.1 (*GTS*). Given k weighted transition systems $T_i = (\Pi_i, \Pi_{i,0}, \rightarrow_i, w_i, \mathcal{AP}_i, L_i)$, $i \in \{1, \dots, k\}$, the global transition system $T_g = T_1 \times \cdots \times T_k$ is a 6-tuple

$$T_g = (\Pi_g, \Pi_{g,0}, \rightarrow_g, w_g, \mathcal{AP}_g, L_g),$$

where,

- $\Pi_g = \Pi_1 \times \cdots \times \Pi_k$ is the set of global states;
- $\Pi_{g,0} = \Pi_{1,0} \times \cdots \times \Pi_{k,0}$ is the set of initial global states;
- $\rightarrow_g \subseteq \Pi_g \times \Pi_g$ is the transition relation defined by: for any $\pi_g = (\pi^1, \dots, \pi^k), \pi'_g = (\pi'^1, \dots, \pi'^k) \in \Pi_g$, we have $(\pi_g, \pi'_g) \in \rightarrow_g$ if and only if $(\pi^i, \pi'^i) \in \rightarrow_i, \forall i = 1, \dots, k$;
- $w_g : \Pi_g \times \Pi_g \rightarrow \mathbb{R}_+$ is the cost function defined by: for any $\pi_g = (\pi^1, \dots, \pi^k), \pi'_g = (\pi'^1, \dots, \pi'^k) \in \Pi_g$, we have $w_g(\pi_g, \pi'_g) = w_1(\pi^1, \pi'^1) + \cdots + w_k(\pi^k, \pi'^k)$;
- $\mathcal{AP}_g = \mathcal{AP}_1 \cup \cdots \cup \mathcal{AP}_k$ is the set of atomic propositions;
- $L_g : \Pi_g \rightarrow 2^{\mathcal{AP}_g}$ is the labeling function defined by: for any $\pi_g = (\pi^1, \dots, \pi^k) \in \Pi_g$, we have $L_g(\pi_g) = L_1(\pi^1) \cup \cdots \cup L_k(\pi^k)$.

An infinite path $\tau_g = \tau_g(0)\tau_g(1) \cdots \in \Pi_g^{\omega}$ in T_g is an infinite sequence of states such that $\tau_g(0) \in \Pi_{g,0}$ and $(\tau_g(j), \tau_g(j+1)) \in \rightarrow_g$, for any $j \in \mathbb{N}$. A finite path can be defined accordingly. The trace of an infinite path τ_g denoted by $trace(\tau_g)$ is an infinite sequence of atomic propositions, i.e. $trace(\tau_g) = L_g(\tau_g(0))L_g(\tau_g(1)) \cdots \in (2^{\mathcal{AP}_g})^{\omega}$. Recall that each state in Π_g is a k-tuple and we denote by τ_g^i the path of agent $i \in \{1, \dots, k\}$ in global path τ_g, i.e., $\tau_g^i = \tau_g^i(0)\tau_g^i(1) \cdots \in \Pi_i^{\omega}$, where $\tau_g^i(j)$ is the ith component in $\tau_g(j)$. We denote by $Path^{\omega}(T_g)$ and $Path^{*}(T_g)$ the set of all infinite and finite paths in T_g respectively. Considering the cost function w_g in T_g, the cost of a finite path $\tau_g \in Path^{*}(T_g)$ denoted by $J(\tau_g)$ is defined as:

$$J(\tau_g) = \sum_{j=0}^{|\tau_g|-2} w_g(\tau_g(j), \tau_g(j+1)), \tag{1}$$

where $|\tau_g|$ is the length of the path τ_g. In words, it captures the total cost incurred by all agents during the execution of τ_g.

2.2. Task specification

In multi-agent system, agents need to work collaboratively to achieve a global task specification. In this paper, we consider high-level tasks described by Linear Temporal Logic (LTL) formulas. Specifically, an LTL formula consists of a set of atomic propositions $\mathcal{AP}$ and several Boolean and temporal operators, which is formed according to the following syntax:

$$\varphi ::= True \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid X\varphi \mid \varphi_1 U \varphi_2,$$

where $a \in \mathcal{AP}$ is an atomic proposition, and X, U mean "next" and "until", respectively. The above definitions also induce temporal operators such as $\Diamond$ ("eventually"), $\Box$ ("always") and $\Rightarrow$ ("implication"), where $\Diamond\varphi = True U \varphi$, $\Box\varphi = \neg\Diamond\neg\varphi$ and $\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 U \varphi_2$.

LTL formulae are used to evaluate whether an infinite word satisfies some properties or not. We denote by $words(\varphi) = \{\sigma \in (2^{\mathcal{AP}})^{\omega} : \sigma \vDash \varphi\}$ the set of all infinite words satisfying LTL formula φ, where $\vDash \subseteq (2^{\mathcal{AP}})^{\omega} \times \varphi$ is the satisfaction relation. The interested reader is referred to Baier and Katoen (2008) for more details about the syntax and semantics of LTL, which are omitted here. For any LTL formula φ, there always exists a corresponding Nondeterministic Büchi Automaton (NBA) over $\Sigma = 2^{\mathcal{AP}}$ which is defined as follows and there are many well-studied fast translation algorithms from an LTL formula to its corresponding NBA; see, e.g., Gastin and Oddoux (2001).

Definition 2.2 (*NBA*)**.** A nondeterministic Büchi automaton A is a 5-tuple, $A = (Q, Q_0, \Sigma, \delta, \mathcal{F})$ where Q is a finite set of states; $Q_0 \subseteq Q$ is a set of initial states; Σ is a set of alphabets; $\delta : Q \times \Sigma \to 2^Q$ is a non-deterministic transition function and $\mathcal{F}$ is a set of accepting states.

Given an infinite word $\sigma = a_0 a_1 a_2 \cdots \in \Sigma^\omega$, an infinite run R of A over σ is an infinite sequence $R = q_0 q_1 q_2 \cdots \in Q^\omega$, where $q_0 \in Q_0$, $\delta(q_i, a_i) \in q_{i+1}$ for any $i \in \mathbb{N}$. An infinite run R is called accepting if $Inf(R) \cap \mathcal{F} \neq \emptyset$, where $Inf(R)$ is the set of states that appear in R infinite times. An infinite word σ is said to be accepted if it induces an accepting infinite run. We denote by $\mathcal{L}_w(A) \subseteq \Sigma^\omega$ the accepted language of A, which is the set of accepting words.

In our problem, we consider an LTL formula φ over $\mathcal{AP}_g$ as the global task specification of the multi-agent system. We denote by $A_\varphi = (Q, Q_0, 2^{\mathcal{AP}_g}, \delta, \mathcal{F})$ as the NBA associated with φ, i.e., $words(\varphi) = \mathcal{L}_w(A_\varphi)$.

Remark 1. In practice, some atomic propositions are common to several agents while some are unique to the individual agents. To avoid confusion, we add the superscript to distinguish the atomic propositions of different agents in φ if necessary. For example, $\varphi = a^i U b^j$ means that property a should be satisfied by agent i until agent j completes b.

3. Security requirement and problem formulation

In this section, we describe the observation model of the intruder, provide the definition of security and formulate the synthesis problem.

3.1. Security constraints

Let $T_g = (\Pi_g, \Pi_{g,0}, \to_g, w_g, \mathcal{AP}_g, L_g)$ be the GTS that models the motion capability of the entire multi-agent system. As we discussed earlier, the system may leak information to the outside during its execution due to various reasons. Here, we model the information leakage by a generic output function

$$H : \Pi_g \to Y,$$

where Y is a new set of output symbols. Therefore, after executing any finite internal path $\tau_g = \tau_g(0)\tau_g(1)\cdots\tau_g(n) \in \Pi_g^*$, the outsider (intruder) will observe a finite external path $H(\tau_g) = H(\tau_g(0))H(\tau_g(1))\cdots H(\tau_g(n)) \in Y^*$.

Remark 2. The above observation model is very generic since Y can be an arbitrary set of symbols; the specific form depends on how the outsider is monitoring the system and what information the system is released to the outsider. This model takes many different practical cases into consideration. For example, the outsider may have sensors for specific regions. Therefore, it knows either some specific agent is entering this region or knows some agent is entering but cannot distinguish which one. Also, in some scenarios, the outsider can "listen" the communications between each agent or the communications between each agent to the central controller. This issue can also be captured by our model by setting Y as the communicating events. Then the outsider may identify the location of some agent if this information is exchanged. Take the simulation in Sections 6.1 and 6.2 as examples, the set Y means the column number of each agent's location, i.e. the intruder can only obtain the column number rather than the complete coordinate information via the states' output function.

Here we consider a *passive intruder* modeled as an outside observer having the following capabilities

- it has the full knowledge of the mobility of the agent team. i.e, the GTS T_g; and
- it can observe the external path generated by the system via the output function, i.e., $H(\tau_g)$.

However, it does not know the plan of the agent team or the internal path τ_g. In other words, it needs to infer the actual behavior of each agent using the mobility model and the information-flow observed online.

Remark 3. Intruders with the above two capabilities are very common for multi-agent systems. Regarding the first capability, the mobility of the agent team is essentially the map of the workspace that specifies how each agent can move from one region to another. If the agents are moving in a public area, then the map of the workspace is usually the public information which can be obtained easily by the intruder. Regarding the second capability, note that here we use the output function to model the observation of the intruder. For example, when the agents are moving in a building, the intruder may place sensors or hake cameras at some gates to detect whether or not some agents have gone to some specific rooms. There are also many other practical applications, where the intruders have these two capabilities, e.g., location-based services (Wu, Sankararaman, & Lafortune, 2014) and ship information systems (Xing, Dai, & Liu, 2016).

To formulate the security requirement, in this work, we consider the problem of protecting the identity of the agent who executed some behaviors of significant importance that do not want to be discovered by the intruder. We model such important behaviors by the visit of some secret states

$$\Pi_s \subset \Pi.$$

We also define $\Pi_{ns} = \Pi \backslash \Pi_s$ as the set of non-secret states. For any path $\tau_g^i \in Path^*(T_i)$ by agent $i \in \mathcal{I}$, we write $\tau_g^i \rightsquigarrow \Pi_s$ if $\exists\, n \leq |\tau_g^i| - 1$ such that $\tau_g^i(n) \in \Pi_s$, i.e., agent i has visited secret states when executing τ_g. Similarly, we write $\tau_g^i \nrightsquigarrow \Pi_s$ if $\forall\, n \leq |\tau_g^i| - 1$ such that $\tau_g^i(n) \in \Pi_{ns}$, which means that all the states in τ_g^i are non-secret. We propose two different types of security requirements that captures the individual privacy in terms of the visit of secret states.

Type I Security: a finite path $\tau_g \in Path^*(T_g)$ is said to be secure in Type I if for any agent $i \in \{1, \ldots, k\}$ such that $\tau_g^i \rightsquigarrow \Pi_s$, there exists a finite path $\tau_g' \in Path^*(T_g)$ such that

(1) $H(\tau_g) = H(\tau_g')$;
(2) $\tau_g'^i \nrightsquigarrow \Pi_s$.

The above security requirement says that, for any agent in the team, if it visited a secret state in the path, then there must exist another possible path in which the same agent did not visit a secret state and these two paths have to be indistinguishable from the intruder's point of view. Otherwise, the intruder will know for sure that *some specific agents (one or more) have visited secret states*. In other words, any agent that executes behaviors of significant importance should hold plausible deniability for doing so. Note that, for the case where more than one agent have visited secret states, their paths for denying the secret may be different.

The above Type I security only requires that the intruder cannot identify for sure that agent i has visited secret states. However, the intruder may identify that *agent i is the only possible agent that has visited the secret states*. Then such a scenario will also threaten the security of the system as the intruder may focus its attention on tracking this targeted agent possibly with further actions. This leads to the following definition of Type II security.

Type II Security: a finite path $\tau_g \in Path^*(T_g)$ is said to be secure in type II if for any agent $i \in \{1, \ldots, k\}$ such that $\tau_g^i \rightsquigarrow \Pi_s$, there exists a finite path $\tau_g'' \in Path^*(T_g)$ such that

(1) $H(\tau_g) = H(\tau_g'')$;
(2) $\tau_g''^j \rightsquigarrow \Pi_s$ for some $j \in \{1, \ldots, k\}, j \neq i$.

An infinite path $\tau_g \in Path^\omega(T_g)$ is said to be secure if any of its finite prefixes is secure (in type I or type II).

The above two types of security requirements are *incomparable* in the sense that none implies the other. Which notion to adopt is rather application dependent. Hereafter, we focus on finding a path that satisfies these two requirements simultaneously. Our methodologies can be easily modified to handle each single type of security individually. This issue will be discussed after our synthesis algorithm is presented.

3.2. Problem formulation

As we mentioned earlier, our overall objective is to find an optimal plan of the agent team, i.e., an infinite path in the GTS, such that the LTL specification is fulfilled and the security requirements are also satisfied. Due to the structural property of the Büchi acceptance condition, without loss of generality, we restrict our solution accepting runs to the following prefix–suffix structure, which is also referred to as a *plan*:

$$\tau_g = \pi_{g,0}\pi_{g,1}\dots\pi_{g,r}[\pi_{g,r+1}\dots\pi_{g,n}]^{\omega} \in Path^{\omega}(T_g).$$

Intuitively, $\pi_{g,0}\pi_{g,1}\dots\pi_{g,r+1}$ is the prefix from the initial state to the accepting state and the suffix $\pi_{g,r+1}\dots\pi_{g,n}$ back to itself is repeated infinitely often by the multi-agent system. The total cost of a plan τ_g is defined by:

$$\hat{J}(\tau_g) = \alpha \sum_{i=0}^{r} w_g(\pi_{g,i}, \pi_{g,i+1}) + (1-\alpha) \sum_{i=r+1}^{n-1} w_g(\pi_{g,i}, \pi_{g,i+1}), \tag{2}$$

where $\alpha \in [0, 1]$ is a parameter adjusting the weight of each part. We consider the case of $\alpha = 0.5$ for the sake of simplicity and it can be extended to the general situation.

Then the security-preserving planning problem that we solve in this paper is formulated as follows.

Problem 3.1 (*Optimal LTL Path Planning Problem with Security Constraints*)**.** Given a GTS $T_g = (\Pi_g, \Pi_{g,0}, \to_g, w_g, \mathcal{AP}_g, L_g)$ with an output function $H : \Pi_g \to Y$, LTL formula φ and a set of secret states $\Pi_s \subset \Pi$, determine a plan $\tau_g \in Path^{\omega}(T_g)$ starting from the possible initial states, satisfying the following conditions:

(1) $trace(\tau_g) \vDash \varphi$;

(2) τ_g is secure in Type I and II;

(3) For any other plan $\dot{\tau}_g \in Path^{\omega}(T_g)$ satisfying the above two requirements, we have $\hat{J}(\tau_g) \le \hat{J}(\dot{\tau}_g)$.

Remark 4. According to the definition of security, if all agents never pass through secret states during the whole period when executing τ_g, then this plan is trivially secure since we can choose $\tau_g'' = \tau_g' = \tau_g$. Therefore, in this case, we just need to solve the standard optimal LTL path planning problem. However, in many applications, passing through the secret states may be unavoidable as secret states may represent some important ingredients in the task and we have to take the security constraints into consideration.

4. Planning algorithm solution

In this section, we present the main planning algorithm that solves the formulated problem. Our approach is based on a new transition system called the multiple-labeling-GTS that captures the security constraints effectively. This structure is then composed with the Büchi automaton for the LTL specification in order to form the solution space containing all feasible secure plans, in which an optimal planning problem is solved.

4.1. Multiple-labeling-GTS with security constraints

In Section 2, we construct GTS T_g as the mobility model of entire multi-agent system. Recall that a state $\pi_g \in \Pi_g$ contains k states of k agents, i.e. $\pi_g = (\pi^1, \dots, \pi^k)$. To avoid confusion, in the rest of the paper, we use superscript to represent the ID of each agent and the subscript to denote the state number.

First, we need to augment the GTS T_g by adding a new label for each agent component in order to track the information whether this agent has visited secret states on the state in GTS T_g. The labeling construction is as follows.

Definition 4.1 (*Labeling-GTS*)**.** Given GTS $T_g = (\Pi_g, \Pi_{g,0}, \to_g, w_g, \mathcal{AP}_g, L_g)$ and a set of secret states Π_s, its labeling-GTS $\tilde{T}_g$ is defined by:

$$\tilde{T}_g = (\tilde{\Pi}_g, \tilde{\Pi}_{g,0}, \tilde{\to}_g, \tilde{w}_g, \mathcal{AP}_g, \tilde{L}_g),$$

where

- $\tilde{\Pi}_g = \Pi_g \times \{0,1\}^k$;
- $\tilde{\Pi}_{g,0} = \Pi_{g,0} \times \{a^1\} \times \cdots \times \{a^k\}$, where for each $i \in \{1, \dots, k\}$, if $\pi^i_{g,0} \in \Pi_s$, then $a^i = 1$, and $a^i = 0$ otherwise;
- $\tilde{\to}_g \subseteq \tilde{\Pi}_g \times \tilde{\Pi}_g$ is the transition relation defined by: for any $\tilde{\pi}_g = (\pi_g, a^1, \dots, a^k)$, $\tilde{\pi}_g' = (\pi_g', a'^1, \dots, a'^k) \in \tilde{\Pi}_g$, we have $(\tilde{\pi}_g, \tilde{\pi}_g') \in \tilde{\to}_g$ if the following holds:
  - $(\pi_g, \pi_g') \in \to_g$,
  - for each $i \in \{1, \dots, k\}$, if $\pi_g'^i \in \Pi_s$, then $a'^i = 1$, and $a'^i = a^i$ otherwise;
- $\tilde{w}_g : \tilde{\Pi}_g \times \tilde{\Pi}_g \to \mathbb{R}_+$ is the cost function defined by: for any $\tilde{\pi}_g = (\pi_g, a^1, \dots, a^k)$, $\tilde{\pi}_g' = (\pi_g', a'^1, \dots, a'^k) \in \tilde{\Pi}_g$, we have $\tilde{w}_g(\tilde{\pi}_g, \tilde{\pi}_g') = w_g(\pi_g, \pi_g')$;
- $\tilde{L}_g : \tilde{\Pi}_g \to 2^{\mathcal{AP}_g}$ is the labeling function defined by: for any $\tilde{\pi}_g = (\pi_g, a^1, \dots, a^k) \in \tilde{\Pi}_g$, $\tilde{L}_g(\tilde{\pi}_g) = L_g(\pi_g)$.

Proposition 1. *Given a labeling-GTS $\tilde{T}_g$ and a finite path $\tilde{\tau}_g = \tilde{\tau}_g(0)\tilde{\tau}_g(1)\dots\tilde{\tau}_g(n) \in \tilde{\Pi}_g^{\omega}$, for any agent $i \in \mathcal{I}$, $\tau_g^i \nrightarrow \Pi_s$ if and only if the labeling of $\tilde{\tau}_g^i(n)$ equals to* 1.

Proof. (1) (Sufficiency) $a_n^i = 1 \Rightarrow$ Passing the secret state: From the transition rule in Definition 4.1, we have if $a_n^i = 1$, then there are two situations. The first one is $\pi_{g,n}^i \in \Pi_s$ which means the agent i is in the secret state at time n. The second one is $a_{n-1}^i = 1$ and $\pi_{g,n}^i \notin \Pi_s$, which indicates that the agent i was in the secret state at time $n-1$ or $a_{n-2}^i = 1$. Recursively, we have there is at least one moment that the agent i was in the secret state. (2) (Necessity) Passing the secret state $\Rightarrow a_n^i = 1$: We assume the agent was in the secret state at time k, $k \in \mathbb{N}$, then $a_k^i = 1$. From the transition rule, we have $a_{k+1}^i = 1$ regardless of whether it was in the secret state or not at time $k+1$. It is easy to obtain $a_n^i = 1$ after iterating $n-k$ times. □

If one agent passes the secret state, then for any plan τ_g, in order to satisfy the security constraint, we need to have another two possible strings τ_g' and τ_g'' as defined in security Types I and II, respectively, such that $H(\tau_g) = H(\tau_g') = H(\tau_g'')$. Similarly, if k agents visit secret states, we need at most $2k$ possible strings with the same observation. This essentially asks us to track, from the intruder's point of view, $2k+1$ paths having the same observation. This idea is implemented by the multiple-labeling-GTS defined as follows. For the sake of simplicity, hereafter, we will omit the subscript in $\tilde{\pi}_g$ and write it as $\tilde{\pi}$, since state with upper tilde must be a global state.

Definition 4.2 (*Multiple-labeling-GTS*)**.** Given a labeling-GTS $\tilde{T}_g = (\tilde{\Pi}_g, \tilde{\Pi}_{g,0}, \tilde{\to}_g, \tilde{w}_g, \mathcal{AP}_g, \tilde{L}_g)$, the corresponding multiple-labeling-GTS V is a new wTS

$$V = (X, X_0, \to_v, w_v, \mathcal{AP}_g, L_v),$$

where

- $X \subseteq \tilde{\Pi}_g \times \cdots \times \tilde{\Pi}_g$ is the set of states which is the product of $2k+1$ states in labeling-GTS $\tilde{T}_g$;
- $X_0 = \{(\tilde{\pi}_0, \tilde{\pi}_{0,1}', \dots, \tilde{\pi}_{0,k}', \tilde{\pi}_{0,1}'', \dots, \tilde{\pi}_{0,k}'') \in \tilde{\Pi}_{g,0} \times \cdots \times \tilde{\Pi}_{g,0} : H(\tilde{\pi}_0) = H(\tilde{\pi}_{0,1}') = \cdots = H(\tilde{\pi}_{0,k}') = H(\tilde{\pi}_{0,1}'') = \cdots = H(\tilde{\pi}_{0,k}'')\}$ is the set of initial states;
- $\to_v \subseteq X \times X$ is the transition relation defined by: for any $x = (\tilde{\pi}, \tilde{\pi}_1', \dots, \tilde{\pi}_k', \tilde{\pi}_1'', \dots, \tilde{\pi}_k'') \in X$ and $x^{\cdot} = (\tilde{\pi}^{\cdot}, \tilde{\pi}_1'^{\cdot}, \dots, \tilde{\pi}_k'^{\cdot}, \tilde{\pi}_1''^{\cdot}, \dots, \tilde{\pi}_k''^{\cdot}) \in X$, we have $(x, x^{\cdot}) \in \to_v$ if the following hold:

- $(\tilde{\pi}, \tilde{\pi}^{'}) \in \rightarrow_g$,
- $(\tilde{\pi}^{'}_1, \tilde{\pi}^{'}_1) \in \rightarrow_g, \ldots, (\tilde{\pi}^{'}_k, \tilde{\pi}^{'}_k) \in \rightarrow_g$,
- $(\tilde{\pi}^{''}_1, \tilde{\pi}^{''}_1) \in \rightarrow_g, \ldots, (\tilde{\pi}^{''}_k, \tilde{\pi}^{''}_k) \in \rightarrow_g$,
- $H(\tilde{\pi}^{'}) = H(\tilde{\pi}^{'}_1) = \cdots = H(\tilde{\pi}^{'}_k) = H(\tilde{\pi}^{''}_1) = \cdots = H(\tilde{\pi}^{''}_k)$;

- $w_v : X \times X \rightarrow \mathbb{R}_+$ is the cost function defined by: for any $x = (\tilde{\pi}, \tilde{\pi}^{'}_1, \ldots, \tilde{\pi}^{'}_k, \tilde{\pi}^{''}_1, \ldots, \tilde{\pi}^{''}_k) \in X$ and $x^{'} = (\tilde{\pi}^{'}, \tilde{\pi}^{''}_1, \ldots, \tilde{\pi}^{''}_k, \tilde{\pi}^{'''}_1, \ldots, \tilde{\pi}^{'''}_k) \in X$, we have $w_v(x, x^{'}) = \tilde{w}_g(\tilde{\pi}, \tilde{\pi}^{'})$;
- $L_v : X \rightarrow 2^{\mathcal{AP}_g}$ is the labeling function defined by : for any $x = (\tilde{\pi}, \tilde{\pi}^{'}_1, \ldots, \tilde{\pi}^{'}_k, \tilde{\pi}^{''}_2, \ldots, \tilde{\pi}^{''}_k) \in X$, we have $L_v(x) = \tilde{L}_g(\tilde{\pi})$.

Note that each state $\tilde{\pi} = (\pi^1, \ldots, \pi^k, a^1, \ldots, a^k)$ is a $2k$-tuple. Here, we ignore the labels when using the output function, i.e., $H(\tilde{\pi}) = H((\pi^1, \ldots, \pi^k))$.

The multiple-labeling-GTS essentially contains all possible tuples of an actual global path together with other $2k$ observation-equivalent global paths: the first k paths are used to fulfill Type I security and last k paths are to fulfill the Type II security. This structure will serve as the basis for synthesizing an optimal path.

4.2. Planning algorithm

Recall that $A_\varphi = (Q, Q_0, 2^{\mathcal{AP}_g}, \delta, \mathcal{F})$ is the NBA associated with the entire task specification φ of the multi-agent system. In order to obtain the global path satisfying the LTL task formula φ, we propose a way to "encode" automaton A_φ accepting φ into the solution space multiple-labeling-GTS V, which is detail as follows.

Definition 4.3 (*Product System*). Given multiple-labeling-GTS $V = (X, X_0, \rightarrow_v, w_v, \mathcal{AP}_g, L_v)$, and NBA $A_\varphi = (Q, Q_0, 2^{\mathcal{AP}_g}, \delta, \mathcal{F})$, the product system is a new wTS:

$$T_\otimes = (\Pi_\otimes, \Pi_{\otimes,0}, \rightarrow_\otimes, w_\otimes),$$

where:

- $\Pi_\otimes \subseteq X \times Q$ is the set of states;
- $\Pi_{\otimes,0} = X_0 \times Q_0$ is the set of initial states;
- $\rightarrow_\otimes \subseteq \Pi_\otimes \times \Pi_\otimes$ is the transition relation defined by: for any $\pi_\otimes = (x, q), \pi^{'}_\otimes = (x^{'}, q^{'}) \in \Pi_\otimes$, we have $(\pi_\otimes, \pi^{'}_\otimes) \in \rightarrow_\otimes$ if the following two conditions hold:
  - $(x, x^{'}) \in \rightarrow_v$,
  - $q^{'} \in \delta(q, L_v(x^{'}))$;
- $w_\otimes : \Pi_\otimes \times \Pi_\otimes \rightarrow \mathbb{R}_+$ is the cost function defined by: for any $\pi_\otimes = (x, q), \pi^{'}_\otimes = (x^{'}, q^{'}) \in \Pi_\otimes$, we have $w_\otimes(\pi_\otimes, \pi^{'}_\otimes) = w_v(x, x^{'})$.

The product system further restricts the dynamic of agents such that each movement in $T_\otimes$ should not violate the task specification φ. The satisfaction of the task requires agents to reach accepting states in the product system infinitely often, i.e. the accepting state should appear in the suffix path that forms a cycle such that agents could reach repeatedly. Let $Cycle(T_\otimes) \subseteq \Pi_\otimes$ be the set of all states in some cycles in the product system $T_\otimes$. We denote by set $Goal(T_\otimes)$ all accepting states in $T_\otimes$ and to be specific, the states in $Goal(T_\otimes)$ are in $Cycle(T_\otimes)$ and their last components are in $\mathcal{F}$, i.e.

$$Goal(T_\otimes) = \{(x, q) \in \Pi_\otimes : (x, q) \in Cycle(T_\otimes) \wedge q \in \mathcal{F}\},$$

which will be the first state in suffix and the last state in prefix when calculating the global plan.

Note that visiting states in $Goal(T_\otimes)$ only ensures the LTL specification; the security requirements are not considered. Therefore, we further define $S(T_\otimes) \subseteq Goal(T_\otimes)$ as the set of *secure accepting states* in $T_\otimes$ such that the security requirements are satisfied, i.e.

$$S(T_\otimes) = \left\{ (x, q) \in Goal(T_\otimes) : \begin{array}{c} (\forall i \in \{1, \ldots, k\} : \text{if } a^i = 1) \\ \left[\begin{array}{ll} \text{s.t.} & \exists u, v \in \{1, \ldots, k\}, j \neq i : \\ & a^{'i}_u = 0 \text{ and } a^{''j}_v = 1 \end{array} \right] \end{array} \right\}, \quad (3)$$

where $x = ((\pi_g, a^1, \ldots, a^k), (\pi^{'}_{g,1}, a^{'1}_1, \ldots, a^{'k}_1), \ldots, (\pi^{'}_{g,k}, a^{'1}_k, \ldots, a^{'k}_k), (\pi^{''}_{g,1}, a^{''1}_1, \ldots, a^{''k}_1), \ldots, (\pi^{''}_{g,k}, a^{''1}_k, \ldots, a^{''k}_k))$.

The construction of multiple-labeling-GTS and the set $S(T_\otimes)$ ensures both security types. In words, for any secure global path, if one agent went through a secret state, there must exist an observation-equivalent global path with this agent not passing by the secret state and another observation-equivalent global path with another agent passing by the secret state in the period. Such a multiple-path composition from the initial state to the secure accepting state in $S(T_\otimes)$ should exist in the product of multiple-labeling-GTS and automatons which ensures the global path satisfies the task specification.

Recall that, our problem formulation requires that both types of security should be satisfied. In case we are only interested in a single type of security, we can simple restrict the condition in Eq. (3) by

- $a^{'i}_u = 0$ for Type I security; and
- $\exists j \neq i : a^{''j}_v = 1$ for Type II security.

Furthermore, for each state $\pi_\otimes = ((\pi_g, a^1, \ldots, a^k), (\pi^{'}_{g,1}, a^{'1}_1, \ldots, a^{'k}_1), \ldots, (\pi^{'}_{g,k}, a^{'1}_k, \ldots, a^{'k}_k), (\pi^{''}_{g,1}, a^{''1}_1, \ldots, a^{''k}_1), \ldots, (\pi^{''}_{g,k}, a^{''1}_k, \ldots, a^{''k}_k), q) \in \Pi_\otimes$, we denote by $\Pi_g[\pi_\otimes] = \pi_g$ the first global state of $\pi_\otimes$, i.e., the projection onto GTS. We also write $\Pi_g[\pi_{\otimes 0}\pi_{\otimes 1} \ldots \pi_{\otimes n}] = \pi_{g,0}\pi_{g,1} \ldots \pi_{g,n}$. Also, we denote by $R(\Pi_{\otimes 0})$ the set of all possible states reachable from $\Pi_{\otimes 0}$ in $T_\otimes$.

Based on the prefix–suffix structure introduced in Section 3, we need to find an optimal path in the form of $\Pi_{\otimes 0} \rightarrow (S(T_\otimes) \rightarrow S(T_\otimes))^\omega$ in $T_\otimes$. Note that the cardinality of sets $\Pi_{\otimes 0}$ and $S(T_\otimes)$ may be both greater than one. Therefore, we need to compare costs of all possible paths and choose an optimal one.

The overall solution is formalized by Algorithm 1. Specifically, line 1 constructs the GTS T_g with wTS T_i, $i \in \{1, \ldots, k\}$ and lines 2–3 obtain the labeling-GTS $\tilde{T}_g$ and multiple-labeling-GTS V with the security information, respectively. Line 4 converts the task specification to the corresponding NBA A_φ. Line 5 constructs the product system of V and A_φ. Line 6 aims to determine whether there exists a path satisfying the task specification as well as the security constraints. If agents cannot reach any states in $S(T_\otimes)$ from $\Pi_{\otimes 0}$, then there is no feasible path and "no feasible plan" will be returned. If there exists at least one path, we consider all possible combinations from initial states in $\Pi_{\otimes 0}$ to secure accepting states in $S(T_\otimes)$ in lines 9 and 10. In lines 11 and 12, we calculate the shortest prefix and suffix paths for all combinations by utilizing the shortest path algorithm, e.g. Dijkstra's algorithm (LaValle, 2006). In line 13, among all feasible combinations, we choose the optimal pair $(\pi^*_{\otimes 0}, \pi^*_{\otimes F})$ with the least cost according to Eq. (2) and return the optimal plan τ_g which is the projection onto GTS T_g by $\Pi_g[]$ in line 14.

5. Correctness and complexity

In this section, we prove the correctness of the solution and discuss the complexity of Algorithm 1.

5.1. Correctness of the solution

Theorem 1. *For GTS $T_g = (\Pi_g, \Pi_{g,0}, \rightarrow_g, w_g, \mathcal{AP}_g, L_g)$ with output function $H : \Pi_g \rightarrow Y$, LTL formula φ and a set of secret states Π_s, Algorithm 1 correctly solves the planning problem formulated in Problem 1.*

Proof. Hereafter, we assume that τ_g is the resulting plan returned by Algorithm 1 and the optimal plan is $\tau_g = \Pi_g[\tau_{pre}(\tau_{suf})^\omega]$, where

$$\begin{aligned} \tau_{pre} = \; & ((\tilde{\pi}_0, \tilde{\pi}^{'}_{0,1}, \ldots, \tilde{\pi}^{'}_{0,k}, \tilde{\pi}^{''}_{0,1}, \ldots, \tilde{\pi}^{''}_{0,k}), q_0) \ldots \\ & ((\tilde{\pi}_n, \tilde{\pi}^{'}_{n,1}, \ldots, \tilde{\pi}^{'}_{n,k}, \tilde{\pi}^{''}_{n,1}, \ldots, \tilde{\pi}^{''}_{n,k}), q_n) \\ \tau_{suf} = \; & ((\tilde{\pi}_{n+1}, \tilde{\pi}^{'}_{n+1,1}, \ldots, \tilde{\pi}^{'}_{n+1,k}, \tilde{\pi}^{''}_{n+1,1}, \ldots, \tilde{\pi}^{''}_{n+1,k}), q_{n+1}) \ldots \\ & ((\tilde{\pi}_{n+m}, \tilde{\pi}^{'}_{n+m,1}, \ldots, \tilde{\pi}^{'}_{n+m,k}, \tilde{\pi}^{''}_{n+m,1}, \ldots, \tilde{\pi}^{''}_{n+m,k}), q_{n+m}), \end{aligned}$$

and $q_{n+1} \in \mathcal{F}$.

Algorithm 1: Optimal LTL plan.

Input: LTL formula φ, wTS T_i, $i \in \{1, \dots, k\}$ with output mapping H and the set of secret states Π_s.
Output: Optimal global plan τ_g.

Construct the GTS $T_g = T_1 \times \dots \times T_k = (\Pi_g, \Pi_{g,0}, \rightarrow_g, w_g, \mathcal{AP}_g, L_g)$.
Obtain $\tilde{T}_g$ by labeling T_g, where $\tilde{T}_g = (\tilde{\Pi}_g, \tilde{\Pi}_{g,0}, \tilde{\rightarrow}_g, \tilde{w}_g, \mathcal{AP}_g, \tilde{L}_g)$.
Construct the multiple-labeling-GTS $V = (X, X_0, \rightarrow_v, w_v, \mathcal{AP}_g, L_v)$.
Convert φ to the corresponding NBA A_φ.
Construct the product system $T_\otimes$ of V and A_φ, where $T_\otimes = (\Pi_\otimes, \Pi_{\otimes 0}, \rightarrow_\otimes, w_\otimes)$.
if $R(\Pi_{\otimes 0}) \cap S(T_\otimes) = \emptyset$ **then**
    return "no feasible plan".
else
    for $\pi_{\otimes 0} \in \Pi_{\otimes 0}$ **do**
        for $\pi_{\otimes F} \in S(T_\otimes)$ **do**
            $\tau_{pre}(\pi_{\otimes 0}, \pi_{\otimes F}) = shortpath(\pi_{\otimes 0}, \pi_{\otimes F})$.
            $\tau_{suf}(\pi_{\otimes F}, \pi_{\otimes F}) = shortpath(\pi_{\otimes F}, \pi_{\otimes F})$.
    $(\pi^*_{\otimes 0}, \pi^*_{\otimes F}) = \text{argmin}_{(\pi_{\otimes 0}, \pi_{\otimes F})} \hat{J}(\Pi_g[\tau_{pre}(\pi_{\otimes 0}, \pi_{\otimes F})(\tau_{suf}(\pi_{\otimes F}, \pi_{\otimes F}))^\omega])$.
    return optimal plan $\tau_g = \Pi_g[\tau_{pre}(\pi^*_{\otimes 0}, \pi^*_{\otimes F})(\tau_{suf}(\pi^*_{\otimes F}, \pi^*_{\otimes F}))^\omega]$.

We first prove the resulting global path satisfies the task specification, i.e., $trace(\tau_g) \vDash \varphi$. It is easy to obtain that $\tau_g = \pi_{g,0} \dots \pi_{g,n}(\pi_{g,n+1} \dots \pi_{g,n+m})^\omega$. From the transition rule of the product system $T_\otimes$, we have that $\mu = q_0 \dots q_n(q_{n+1} \dots q_{n+m})^\omega$ is an infinite run induced by an infinite word $trace(\tau_g) = L_g(\pi_{g,0}) \dots L_g(\pi_{g,n})(L_g(\pi_{g,n+1}) \dots L_g(\pi_{g,n+m}))^\omega$. Since $q_{n+1} \in \mathcal{F}$, we have that $Inf(\mu) \cap \mathcal{F} \neq \emptyset$, i.e., $trace(\tau_g) \in \mathcal{L}_{A_\varphi} = words(\varphi)$. Therefore, $trace(\tau_g) \vDash \varphi$.

Second, we show that the global path is secure, i.e., τ_g is secure. If there is no agent visited secret states, then τ_g is secure trivially. Then we consider the case where at least one agent has visited the secret state. We find the resulting path from the initial state $\pi_{\otimes 0} \in \Pi_{\otimes 0}$ to $\pi_{\otimes F} \in S(T_\otimes)$. From the definition of $S(T_\otimes)$ and Theorem 1, we have that if one agent in $\tau_{pre,g}$ has visited secret states, then there must exist $u, v \in \{1, \dots, k\}$ such that this agent did not visit secret states in $\tau'_{pre,g,u}$ and there exists another agent visited secret states in $\tau''_{pre,g,v}$, where $\tau_{pre,g} = \pi_{g,0} \dots \pi_{g,n}$, $\tau'_{pre,g,u} = \pi'_{g,0,u} \dots \pi'_{g,n,u}$ and $\tau''_{pre,g,v} = \pi''_{g,0,v} \dots \pi''_{g,n,v}$; these paths satisfy the second point in Definition of type I and type II security. Furthermore, we have that the composition of $\tau_{pre,g}$, $\tau'_{pre,g,u}$ and $\tau''_{pre,g,v}$ is a part of path in V and from the transition rule of V, it is easy to obtain $H(\tau_{pre,g}) = H(\tau'_{pre,g,u}) = H(\tau''_{pre,g,v})$, which satisfies the first point in both type I and type II security. The case of multiple robots visiting secret states can also be proved in the same way. Therefore, the infinite path τ_g is secure since any of its finite prefix $\tau_{pre,g}$ is secure.

Third, we show that the global path is optimal, i.e., for any secure path $\dot{\tau}_g$ satisfying the task specification φ, we have $\hat{J}(\dot{\tau}_g) \geq \hat{J}(\tau_g)$. Here, we prove it by contradiction. Suppose that $\dot{\tau}_g \in Path^\omega(T_g)$ is a secure path satisfying the task specification and $\hat{J}(\dot{\tau}_g) < \hat{J}(\tau_g)$. Since $\dot{\tau}_g$ is secure, its prefix $\dot{\tau}_{pre,g}$ is secure and there exist other two paths $\dot{\tau}'_{pre,g,u}, \dot{\tau}''_{pre,g,v} \in Path^\omega(T_g)$, $u, v \in \{1, \dots, k\}$ such that $H(\dot{\tau}_{pre,g}) = H(\dot{\tau}'_{pre,g,u}) = H(\dot{\tau}''_{pre,g,v})$. Then according to Definition 4.2, we have that there exists a path $\tau_V \in Path^\omega(V)$ containing labeling $\dot{\tau}_g$, $\dot{\tau}'_{pre,g,u}$ and $\dot{\tau}''_{pre,g,v}$. Furthermore, since it satisfies the task, there exists a path $\tau_\otimes \in Path^\omega(T_\otimes)$ which is the product of V and A_φ. Without loss of generality, we write the $\dot{\tau}_\otimes = \dot{\tau}_{pre}(\dot{\tau}_{suf})^\omega$ as the following structure:

$$\begin{aligned}\dot{\tau}_{pre} = \; & ((\tilde{\pi}_0, \dots, \tilde{\pi}'_{0,u}, \dots, \tilde{\pi}''_{0,v}, \dots), q_0) \dots \\ & ((\tilde{\pi}_n, \dots, \tilde{\pi}'_{n,u}, \dots, \tilde{\pi}''_{n,v}, \dots), q_n) \\ \dot{\tau}_{suf} = \; & ((\tilde{\pi}_{n+1}, \dots, \tilde{\pi}'_{n+1,u}, \dots, \tilde{\pi}''_{n+1,v}, \dots), q_{n+1}) \dots \\ & ((\tilde{\pi}_{n+m}, \dots, \tilde{\pi}'_{n+m,u}, \dots, \tilde{\pi}''_{n+m,v}, \dots), q_{n+m}).\end{aligned}$$

We have that $((\tilde{\pi}_0, \dots, \tilde{\pi}'_{0,u}, \dots, \tilde{\pi}''_{0,v}, \dots), q_0) \in \Pi_{\otimes 0}$ and $\dot{\pi}_{\otimes F} = ((\tilde{\pi}_{n+1}, \dots, \tilde{\pi}'_{n+1,u}, \dots, \tilde{\pi}''_{n+1,v}, \dots), q_{n+1}) \in R(\Pi_{\otimes 0}) \cap S(T_\otimes)$. Then Algorithm 1 returns $\dot{\tau}_g = \Pi_g[\dot{\tau}_{pre}(\dot{\tau}_{suf})^\omega]$ with the least cost instead of τ_g, which is a contradiction. Therefore, τ_g is the optimal path with the least cost.

The above three aspects show that Algorithm 1 is sound that the solution is correct if a global path is returned by the algorithm. Finally we show that Algorithm 1 is also complete, i.e., if "no feasible plan" is returned by Algorithm 1, then there is no solution to Problem 3.1. The proof is similar with the proof of optimal property. We assume that there exists a global secure path $\tau_g \in Path^\omega(T_g)$ such that $trace(\tau_g) \vDash \varphi$ and τ_g is secure. The same as the above proof for optimality, there exists a path $\tau_\otimes \in Path^w(T_\otimes)$ and $R(\pi_{\otimes 0}) \cap S(T_\otimes) \neq \emptyset$. Therefore, it will not return "no feasible plan" in Algorithm 1.

In conclusion, Algorithm 1 is sound and complete in solving the optimal planning problem with security constraints defined in Problem 3.1. □

5.2. Complexity analysis

We conclude this section by analyzing the computational complexity of Algorithm 1. First, we note that the global transition system T_g contains $|\Pi_g| \leq |\Pi|^k$ states, where $|\Pi| = \max_{i \in \{1,\dots,k\}} |\Pi_i|$ is the largest number of states in each wTS T_i, $i \in \{1, \dots, k\}$ and there are at most $|\tilde{\Pi}_g| = 2^k |\Pi_g|$ states after adding labels in labeling-GTS $\tilde{T}_g$. From Definition 4.2, we have that the multiple-labeling-GTS includes at most $|\tilde{\Pi}_g|^{2k+1}$ states. Since there are $|Q|$ states in the corresponding Büchi automaton converted by φ, the product system $T_\otimes$ contains at most $|\Pi_\otimes| = |\tilde{\Pi}_g|^{2k+1}|Q|$ states which increases exponentially with the number of agents. Algorithm 1 involves solving a shortest path search problem over a graph with at most $(2|\Pi|)^{2k^2+2k}|Q|$ states.

Now, we compare the complexity of Algorithm 1 with existing LTL task planning algorithm without security constraints. Specifically, let us consider the standard LTL multi-agent planning algorithm in Guo and Dimarogonas (2015). To generate optimal plannings for each agent, we need to solve a shortest path search problem over a structure with $|\Pi|^k|Q|$ states. The complexity of our algorithm is higher than that of Guo and Dimarogonas (2015) because the order of the number of states increases from k to k^2. However, as we will see in the next section, the algorithm in Guo and Dimarogonas (2015) does not necessarily preserve security. Our algorithm spends higher complexity compared with the existing one mainly because we further take the issue of security into account.

6. Case studies

In this section, three case studies are presented to illustrate the proposed security-preserving path planning algorithms. In particular, we have implemented the proposed algorithms and set up the physical experiment environment. All algorithms are implemented in Python 2.7. The first two case studies are both extended based on the motivating example in Section 1.1 and correspond to security type I and type II respectively. The experiment is presented for the combination of two types security constraints.

Codes and experiment video are available at https://github.com/Xinyi-Yu/Multiagent-LTL-Opacity.

6.1. Case Study 1: Type I security

We use this case study to illustrate the Type I security, i.e., the intruder should not be able to determine for sure that some specific robots have visited secret states.

System Model: Suppose that there are two manufacturing robots R_1 and R_2 working in a 6×6 workspace as shown in Fig. 2. Two robots R_1 and R_2 start from their own sets of initial states (the red area), in lower-left corner and lower-right corner respectively. At each instant, each robot can move left/right/up/down to its adjacent grid or stay

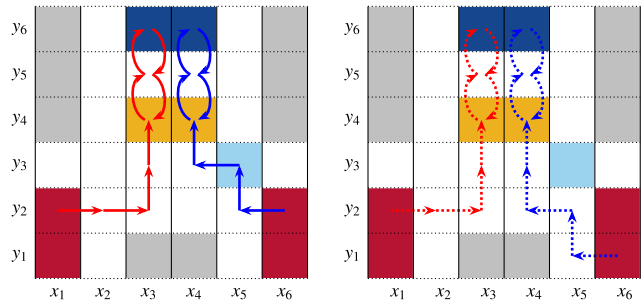


Fig. 2. Simulation results for the case study described in Section 6.1. Left: The actual paths executed by each robot to fulfill the LTL specification and security constraint. Right: Possible paths having the same observations as the actual ones from the intruder's point of view. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

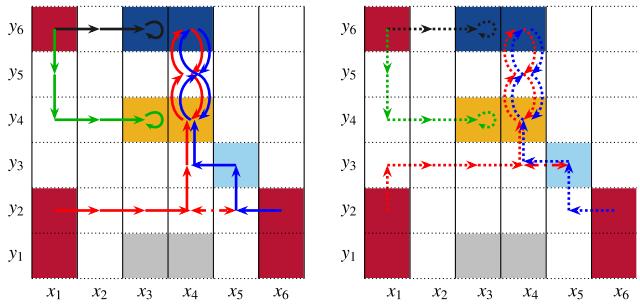


Fig. 3. Simulation results for the case study described in Section 6.2. Left: The actual paths executed by each robot to fulfill the LTL specification and security constraint. Right: Possible paths having the same observations as the actual ones from the intruder's point of view. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at its current grid. The cost for moving vertically one step up/down is fixed as one unit. However, the cost for moving horizontally one step left/right is $0.2i$ if the robot is in the ith row, where $i = 1, \ldots, 6$. Furthermore, there is no cost for staying at where it is without any movement.

The working space has the following regions with special properties of interest:

- an assembly workshop (blue region) where the robots assemble products; we use atomic proposition a^i to denote that robot R_i is in the assembly workshop.
- a warehouse (yellow region) where some components are stored; we use w^i to denote that robot R_i is in the warehouse.
- a command center (cyan region) where the robots can get blueprints for using some special machines in the assembly workshop; we use c^i to denote that robot R_i is in the command center.
- obstacles (gray region) where the robots cannot visit; we use obs to denote that some robot reaches the obstacle.

Planning Objectives: The main task of robots R_1 and R_2 is to deliver components between the warehouse and assembly workshop infinitely often while avoiding all obstacles. Furthermore, one robot needs to go to the command center to obtain the blueprint of the product. Formally, the overall task specification can be expressed by the following LTL formula

$$\varphi = (\Diamond c^1 \vee \Diamond c^2) \wedge \Box\Diamond w^1 \wedge \Box\Diamond a^1 \wedge \Box\Diamond w^2 \wedge \Box\Diamond a^2 \wedge \Box\neg obs.$$

Security Constraints: We assume that there is an intruder that can get the column information of each specific robot at each instant, but does not know its row information. That is, if R_1 and R_2 are at states (x_1, y_1) and (x_2, y_2), respectively, then the intruder observes (x_1, x_2). Here we consider the following security-preserving problem. The multi-robot system does not want the intruder to know for sure a specific robot is carrying the blueprint. Otherwise, e.g., the intruder may hack the robot to disrupt the key operation associated with the blueprint.

Solution: The above setting can be modeled by Problem 3.1. Specifically, the system model can be constructed in the form of GTS T_g and the output function only reads the column information of each robot as described above. The secret state is the command center (x_5, y_3) and the security constraint actually falls into the category of Type I security. Therefore the proposed planning Algorithm 1 in Section 4 can be implemented in this case and the synthesized plan is shown in Fig. 2. Specifically, the paths in the left picture of Fig. 2 are the actual paths executed by each robot that satisfy both the task specification and the security constraint. The paths in the right picture of Fig. 2 are not the actual paths executed by the robots. Instead, they represent those possible paths having the same observation as the left ones from the point of intruder's view but do not visit the secret state so that they explain why the left ones are secure.

Now, let us compare the synthesized plans with the solution synthesized by the standard algorithm for LTL multi-agent task planning without security constraints. Specifically, we consider the algorithm in Guo and Dimarogonas (2015) and the solution generated is exactly the same as ours. However, this similarity is just a coincidence and we will show in the next two examples that the solution provided by Guo and Dimarogonas (2015) may not be secure.

6.2. Case Study 2: Type II security

We use this case study to illustrate the Type II security, i.e., the intruder should not be able to identify that robot i is the only possible robot that has visited the secret states. The system model is the same as that in Case Study 1 described the last subsection. However, here we have four robots R_1, R_2, R_3 and R_4 in the workspace starting from their own initial states (the red area), in lower-left corner (for R_1), lower-right corner (for R_2) and upper-left corner (for R_3 and R_4), respectively.

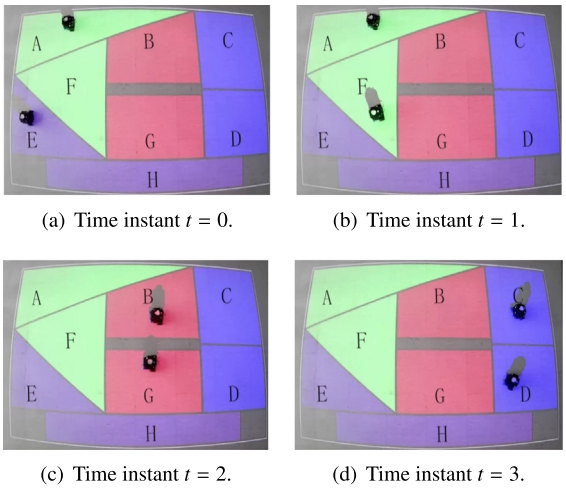
Planning Objectives: The main task of robots R_1 and R_2 is the same as Case Study 1, i.e., to deliver components between the warehouse and assembly workshop infinitely often while avoiding all obstacles. In addition, robot R_3 is responsible for assembling components in the assembly workshop and robot R_4 is responsible for managing the warehouse. Furthermore, robot R_2 needs to get the blueprint of the product. Therefore, the overall task specification becomes

$$\varphi = \Diamond c^2 \wedge \Box\Diamond w^1 \wedge \Box\Diamond a^1 \wedge \Box\Diamond w^2 \wedge \Box\Diamond a^2 \wedge \Box\Diamond a^3 \wedge \Box\Diamond w^4 \wedge \Box\neg obs.$$

Security Constraints: We consider an intruder with the observation function as that in Case Study 1 and the system does not want the intruder to *suspect* that it is robot R_2 the only candidate that has possibly been to the command center. Otherwise, the intruder may focus its attention on tracking the behavior of R_2 in subsequent missions, which may threaten the security of the blueprint information.

Solution: The same as Case Study 1, the above setting can be modeled by Problem 3.1. Specifically, the secret state is also command center (x_5, y_3) with label c^i but this time, the security constraint fits Type II security. Algorithm 1 is implemented in the simulation and the results are shown in Fig. 3. It is worth mentioning that, in order to ensure security, robot R_1 takes a little detour depicted by the thin dash dot line in the left of Fig. 3 so that the intruder may think that R_1 may also possibly take the path shown in the right of Fig. 3 that passes through the command center. Therefore, the intruder cannot suspect R_2 as the only robot carrying the blueprint.

Still, we compare the synthesized plans with solutions obtained by the standard LTL multi-agent task planning algorithm without security constraint (Guo & Dimarogonas, 2015). For this example, the optimal



(a) Time instant $t = 0$. (b) Time instant $t = 1$.

(c) Time instant $t = 2$. (d) Time instant $t = 3$.

Fig. 4. Experiments results for Case Study 3. Four pictures show the positions of two robots at different times. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

plans for robots R_2, R_3 and R_4 are the same as our synthesized plans. However, the plan for robot R_1 is different. Specifically, it will take the same path as Case Study 1, i.e., from state (x_3, y_2), it will move upward directly instead of detouring to (x_5, y_2). However, this plan generated by Guo and Dimarogonas (2015) is not secure in Type II because only R_2 has been to column 5 and the intruder will only suspect R_2 although it cannot make sure R_2 has been to secret states. In contrast, our solution shown in Fig. 3 ensures both the correctness of the task and the security of the system.

6.3. Case Study 3: Type I and II security

The above two case studies illustrate Type I and II security respectively and which security definition to use is problem-dependent. In some cases, Type I and II security should be satisfied simultaneously; we take the following experiment as an example.

System Model: Let us consider a team of two robots R_1 and R_2 working in a workspace shown in Fig. 4, which is divided into four regions of interest depicted by four different colors. Each region is further divided into several sub-regions as shown in the figure. Robots R_1 and R_2 start from sub-region A and E respectively. At each instant, each robot can move to one of its adjacent sub-regions without crossing obstacles, which are black parts in the figure, or to stay in its current sub-region. Furthermore, when the robot moving from one sub-region to another, one unit of cost incurs if these two sub-regions belong to the same region and two units of cost incur when the two sub-regions belong to different regions. There is no cost if the robot chooses to stay in its current sub-region.

Planning Objectives: We use atomic proposition μ^i to denote that robot R_i is in the sub-region μ, where $\mu \in \{A, B, C, D, E, F, G, H\}$. The objective of robot R_1 is to first visit sub-region B before visiting C infinitely often. The objective of robot R_2 is to visit sub-region D infinitely often. Therefore, the task formula is

$$\varphi = (\neg C^1 U B^1) \wedge \square\lozenge C^1 \wedge \square\lozenge D^2.$$

Observation Model and Security Constraint: We assume that whenever each robot finishes its movement to a new sub-region, it needs to communicate to the central control station to report its current status. The communication services are provided by four base stations; each of them covers all sub-regions in one of the four regions. We assume that there is an intruder that knows whether a robot is using a specific base station, but does not know the specific sub-region within the service of the base station. In other words, whenever the robot makes a movement, the intruder knows which region (color) it belongs to but cannot specify the precise sub-region. We assume that sub-region B is a secret state for which the system does not want the intruder to know either some robots *have been to secret state* or *which one* has been to the secret sub-region B. This may because, for instance, having been to the secret state means subsequent work may involve secret things while locking in the only suspect target will expose this target to danger.

Generated Plan: The above setting can also be modeled by Problem 3.1 which is the same as Case Study 1 and 2. The output function maps each sub-region to the color of the region it belongs to. The secret state is sub-region B and the above security constraints fit Type I and II security. Therefore the proposed planning Algorithm 1 in Section 4 can be implemented in this case and the results are shown in Fig. 4. Specifically, the robot team takes the plan

$$P = (A, E) \rightarrow (A, F) \rightarrow (B, G) \rightarrow [(C, D)]^{\omega}.$$

This plan is not the least costly without security consideration. However, it is the optimal one among all secure plans. Specifically, for this plan, we can find another plan $P' = P'' = (A, E) \rightarrow (F, F) \rightarrow (G, B) \rightarrow [(D, C)]^{\omega}$, such that the privacy constraints will be satisfied. Therefore, from the finite observation $(\text{green}, \text{purple}) \rightarrow (\text{green}, \text{green}) \rightarrow (\text{red}, \text{red}) \rightarrow (\text{blue}, \text{blue})$, the intruder can neither infer whether the robot R_1 has been to the secret sub-region nor suspect robot R_1 only.

For the purpose of comparison, we still apply the standard LTL planning algorithm in Guo and Dimarogonas (2015) and obtain optimal plan $(A, E) \rightarrow (B, H) \rightarrow [(C, D)]^{\omega}$. Note that the cost of this plan is seven, which is smaller than that of our synthesized plan. However, similar to the previous case study, we see that this plan is still not secure because only R_1 has been to the red region and the intruder will only suspect it, which violates Type II security. Our algorithm synthesizes a plan with a higher cost but ensures security of the system.

7. Conclusion

In this paper, we solved an optimal security-preserving temporal logic task planning problem for multi-agent systems. Two types of conditions were proposed to capture different security requirements. An effective algorithm was proposed based on the product of multiple-labeling-GTS and the Büchi automaton to ensure the proposed security conditions. Simulation results as well as real-world experiments were provided to illustrate the proposed results. The complexity of proposed algorithm is polynomial in the size of the GTS but is exponential in the number of robots. In the future, we would like to explore recent developed scalable task planning methods, such as sampling-based approach (Kantaros & Zavlanos, 2018, 2020) and learning-based approach (Cai et al., 2020; Hasanbeig et al., 2019), to further mitigate the computational complexity of the security-preserving planning algorithm.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Baier, C., & Katoen, J. (2008). *Principles of model checking*. MIT Press.

Basile, F., Chiacchio, P., & Di Marino, E. (2019). An auction-based approach to control automated warehouses using smart vehicles. *Control Engineering Practice, 90*, 285–300.

Belta, C., Yordanov, B., & Gol, E. (2017). *Formal methods for discrete-time dynamical systems (vol. 89)*. Springer.

Cai, M., Peng, H., Li, Z., & Kan, Z. (2020). Learning-based probabilistic LTL motion planning with environment and motion uncertainties. *IEEE Transactions on Automatic Control*.

Ding, D., Han, Q., Ge, X., & Wang, J. (2020). Secure state estimation and control of cyber-physical systems: A survey. *IEEE Transactions on Systems Man and Cybernetics: Systems*, *51*(1), 176–190.

Fanti, M. P., Mangini, A. M., Pedroncelli, G., & Ukovich, W. (2018). A decentralized control strategy for the coordination of AGV systems. *Control Engineering Practice*, *70*, 86–97.

Gastin, P., & Oddoux, D. (2001). Fast LTL to Büchi automata translation. In *International conference on computer aided verification* (pp. 53–65). Springer.

Goryca, J., & Hill, R. (2013). Formal synthesis of supervisory control software for multiple robot systems. In *American control conference* (pp. 125–131). IEEE.

Guo, M., & Dimarogonas, D. (2015). Multi-agent plan reconfiguration under local LTL specifications. *International Journal of Robotics Research*, *34*(2), 218–235.

Hadjicostis, C. (2018). Trajectory planning under current-state opacity constraints. *IFAC-PapersOnLine*, *51*(7), 337–342.

Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G. J., & Lee, I. (2019). Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *58th conference on decision and control* (pp. 5338–5343). IEEE.

Hill, R., & Lafortune, S. (2017). Scaling the formal synthesis of supervisory control software for multiple robot systems. In *American control conference* (pp. 3840–3847). IEEE.

Ji, Y., Yin, X., & Lafortune, S. (2019). Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, *108*, Article 108476.

Ju, C., & Son, H. (2019). Modeling and control of heterogeneous agricultural field robots based on Ramadge–Wonham theory. *IEEE Robotics and Automation Letters*, *5*(1), 48–55.

Kantaros, Y., Malencia, M., Kumar, V., & Pappas, G. (2020). Reactive temporal logic planning for multiple robots in unknown environments. In *IEEE international conference on robotics and automation* (pp. 11479–11485). IEEE.

Kantaros, Y., & Zavlanos, M. (2018). Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. *IEEE Transactions on Automatic Control*, *64*(5), 1916–1931.

Kantaros, Y., & Zavlanos, M. (2020). Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *International Journal of Robotics Research*, *39*(7), 812–836.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, *30*(7), 846–894.

Kloetzer, M., & Belta, C. (2007). Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, *23*(2), 320–330.

Kloetzer, M., & Belta, C. (2009). Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, *26*(1), 48–61.

Kloetzer, M., & Mahulea, C. (2020). Path planning for robotic teams based on LTL specifications and Petri net models. *Discrete Event Dynamic Systems*, *30*(1), 55–79.

Kress-Gazit, H., Fainekos, G., & Pappas, G. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, *25*(6), 1370–1381.

Kress-Gazit, H., Lahijanian, M., & Raman, V. (2018). Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control Robotics and Autonomous Systems*.

Lafortune, S., Lin, F., & Hadjicostis, C. (2018). On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, *45*, 257–266.

LaValle, S. (2006). *Planning algorithms*. Cambridge University Press.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, *47*(3), 496–503.

Ma, Z., & Cai, K. (2021). Optimal secret protections in discrete-event systems. *IEEE Transactions on Automatic Control*.

Mahulea, C., & Kloetzer, M. (2017). Robot planning based on boolean specifications using Petri net models. *IEEE Transactions on Automatic Control*, *63*(7), 2218–2225.

Mahulea, C., Kloetzer, M., & González, R. (2020). *Path planning of cooperative mobile robots using discrete event models*. John Wiley & Sons.

Moarref, S., & Kress-Gazit, H. (2020). Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. *Autonomous Robots*, *44*(3), 585–600.

Oh, Y., Cho, K., Choi, Y., & Oh, S. (2020). Chance-constrained multi-layered sampling-based path planning for temporal logic-based missions. *IEEE Transactions on Automatic Control*.

O'Kane, J., & Shell, D. (2015). Automatic design of discreet discrete filters. In *IEEE international conference on robotics and automation* (pp. 353–360). IEEE.

Plaku, E., & Karaman, S. (2016). Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications*, *29*(1), 151–162.

Raman, V., & Kress-Gazit, H. (2014). Synthesis for multi-robot controllers with interleaved motion. In *IEEE international conference on robotics and automation* (pp. 4316–4321). IEEE.

Saboori, A., & Hadjicostis, C. (2011). Coverage analysis of mobile agent trajectory via state-based opacity formulations. *Control Engineering Practice*, *19*(9), 967–977.

Tatsumoto, Y., Shiraishi, M., Cai, K., & Lin, Z. (2018). Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. *Control Engineering Practice*, *81*, 97–104.

Tong, Y., Li, Z., Seatzu, C., & Giua, A. (2018). Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems*, *28*(2), 161–182.

Ulusoy, A., Smith, S., Ding, X., Belta, C., & Rus, D. (2013). Optimality and robustness in multi-robot path planning with temporal logic constraints. *International Journal of Robotics Research*, *32*(8), 889–911.

Wang, Y., Nalluri, S., & Pajic, M. (2020). Hyperproperties for robotics: Planning via hyperLTL. In *IEEE international conference on robotics and automation* (pp. 8462–8468). IEEE.

Wu, Y.-C., Sankararaman, K. A., & Lafortune, S. (2014). Ensuring privacy in location-based services: An approach based on opacity enforcement. *IFAC Proceedings Volumes*, *47*(2), 33–38.

Xing, B., Dai, J., & Liu, S. (2016). Enforcement of opacity security properties for ship information system. *International Journal of Naval Architecture and Ocean Engineering*, *8*(5), 423–433.

Yang, S., Yin, X., Li, S., & Zamani, M. (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. In *59th IEEE conference on decision and control* (pp. 4460–4466). IEEE.