Hybrid Control of Multi-agent Systems with Contingent Temporal Tasks and Prescribed Formation Constraints

Meng Guo, Charalampos P. Bechlioulis, Kostas J. Kyriakopoulos and Dimos V. Dimarogonas

***Abstract*—In this paper, we present a distributed hybrid control strategy for multi-agent systems with contingent temporal tasks and prescribed formation constraints. Each agent is assigned a local task given as a Linear Temporal Logic (LTL) formula. Additionally, two commonly-seen kinds of cooperative robotic tasks, namely service and formation, are requested and exchanged among the agents in real-time. The service request is a short-term task provided by one agent to another. On the other hand, the formation request is a relative deployment requirement with predefined transient response imposed by an associated performance function. The proposed hybrid control strategy consists of four major components: (a) the contingent requests handling module, (b) the real-time events monitoring module, (c) the local discrete plan synthesis module, (d) the continuous control switching module, and it is shown that all local tasks and contingent service/formation requests are fulfilled. Finally, a simulated paradigm demonstrates the proposed control strategy.**

I. Introduction

In general, cooperative control of multi-agent systems focuses on designing local control protocols to achieve a single predefined global control objective, such as reference-tracking [1], consensus [2], or formation [3]. This work is motivated by the necessity to specify and achieve more structured and complex team behaviors than the aforementioned. Particularly, we adopt LTL formulas as appropriate descriptions of desired high-level goals. LTL allows the designer to rigorously specify various temporal tasks, such as periodic surveillance, sequencing, request-response and their combinations. Afterwards, formal verification-inspired methods may be used to synthesize a discrete plan over an abstraction of the state space, that guarantees the satisfaction of the specifications. Such a generic hierarchical approach has been formulated and widely employed during the last decades in single-agent as well as multi-agent settings. For instance, [4] proposes an automated framework to translate task specifications, expressed as LTL formulas, directly into a hybrid controller that drives a dynamical system towards achieving a task goal. Regarding multi-agent systems, LTL formulas have been used to specify complex high-level global tasks [5]–[9] and local tasks [10], [11]. In the first case, a global task that captures requirements on the overall team behavior is predefined. Then the motion of all agents is synthesized and coordinated in a centralized fashion. This top-down approach is exploited in [5], [9], where the global task is decomposed into smaller local tasks to be executed by the agents in a synchronized [5] or partially synchronized manner [9]. In the latter case, each agent is assigned a local task that favors a bottom-up approach. We adopt the second formulation here that agent has a locally-assigned task. However, in contrast to [10], these tasks are dependent due to contingent service and formation requests.

Service requests are of particular interest as they encapsulate the scenario where one agent needs another agent's assistance on a short-term task. The service requests can only be sent and confirmed among the agents online and are given as temporal formulas, which need to be satisfied by the confirmed recipient. This means that each agent has to react to the received service requests in a contingent way. Similar work on real-time reactivity for dynamical systems under temporal tasks can be found in [12], where sensory inputs are included in the General Reactivity GR(1) formulas to take into account possibly dynamical environments, as well as in [13], where similar techniques are applied for various single-robot and multi-robot applications. The local plans of each agent are synthesized off-line to handle all modeled changes in the environment, which is however not feasible in our formulation as the service and formation requests are exchanged in real-time under a non-predefined manner (i.e., the agents should adapt their local plans on-line and according to the actually confirmed service or formation requests).

Another commonly-seen multi-robot scenario appears when an agent requests from another one to form a relative coordination to accomplish a collaborative task. We denote such a requirement as a **formation request**, i.e., the confirmed pair of agents should converge to a desired relative configuration until the corresponding formation task is accomplished. Similar ideas of imposing formation constraints for multi-agent systems appeared in [7], where however a global formation task for the whole team was embodied. Although the formation control has been extensively studied for multi-agent systems, e.g., [3], [14], in this work, we enforce prescribed performance constraints on the transient response of the formation process. The prescribed performance control problem was originally studied for high-order MIMO

M. Guo is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA. {meng.guo@duke.edu}. D. V. Dimarogonas is with the ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden and with the KTH Centre for Autonomous Systems. {dimos@kth.se}. C. P. Bechlioulis and K. J. Kyriakopoulos are with the Control Systems Laboratory, School of Mechanical Engineering, National Technical University of Athens, Athens 15780, Greece. {chmpechl, kkyria@mail.ntua.gr}. This work was supported by the Swedish Research Council (VR), the ERC BUCOPHSYS and the EU STREP RECONFIG: FP7-ICT-600825.

nonlinear single-agent systems [15] and was recently extended to multi-agent systems in our earlier work [16]. However, we enhance this control technique here by: (i) considering transient performance specifications on the overall formation error, in contrast to the relative coordinates approach [16], thus simplifying the control design; and (ii) combining it with the high-level discrete planning such that various control modes are activated according to the real-time execution of the discrete plan.

This work is clearly motivated by applications where multi-robot systems are requested to exhibit complex collaborative behavior in cluttered environments. For instance, consider three robots denoted by $\mathfrak{R}_1$, $\mathfrak{R}_2$, $\mathfrak{R}_3$. Robot $\mathfrak{R}_1$ has limited motion and sensing capabilities. Hence, each time it meets $\mathfrak{R}_2$ it requests $\mathfrak{R}_2$ to perform a short-term coverage service task, e.g., visit two regions of interest in sequence. Moreover, whenever $\mathfrak{R}_1$ meets $\mathfrak{R}_3$, it requests $\mathfrak{R}_3$ to converge sufficiently fast to a relative formation to accomplish a cooperative task together, e.g., move a heavy object. Therefore, both service and formation requests are advantageous for specifying collaborative tasks in multi-agent systems. Towards this direction, we propose a hybrid control scheme that guarantees the fulfilment of high-level temporal tasks involving contingent service and formation requests. It consists of four major components: (i) the communication module that sends, receives and replies service and formation requests; (ii) the events monitoring module that detects real-time events; (iii) the discrete plan synthesis and adaptation module that incorporates contingent requests into local tasks, and (iv) the continuous controller switching module. The main contribution is twofold: (i) we present a novel way to incorporate contingent service and formation tasks in the local task specification, with predefined transient responses; (ii) the proposed plan adaptation algorithms that handle online updates of the task specifications and ensure both the current and past requests are satisfied in finite time.

The rest of the paper is organized as follows. Section II introduces the preliminaries. The problem is formulated in Section III. Section IV presents the solution. Section V demonstrates the simulation study. We conclude in Section VI.

II. Preliminaries

A. Linear Temporal Logic

The basic ingredients of an LTL formula are several boolean and temporal operators, and a set of Atomic Propositions (AP). In particular, LTL formulas are formed according to the following syntax [18]: $\varphi := \top \,|\, a \,|\, \neg\varphi \,|\, \varphi_1 \vee \varphi_2 \,|\, \varphi_1 \,\mathsf{U}\, \varphi_2$ where $a \in AP$, $\top \triangleq$ `True` and $\bot \triangleq$ `False`; $\neg$ is the "*negation*" operator; $\vee$ is the "*or*" operator; U is the "*until*" operator. Those basic operators can be used to derive other commonly used operators, such as $\wedge$ (*and*), $\Diamond$ (*eventually*), $\Box$ (*always*). Notice that contrary to the standard definition [18], the "*next*" operator is not allowed here.

The semantics of LTL is defined over *discrete-time* Boolean signals over the same AP, which is also called a word over AP. Particularly, given a discrete-time Boolean signal $w : \mathbb{Z} \to 2^{AP}$, $w(k) \subseteq AP$ is the set of propositions that signal w satisfies at discrete time k, $\forall k \in \mathbb{Z}$. Then, consider an LTL formula φ and a discrete-time Boolean signal w over AP; the satisfiability relation $(w,\, k) \models \varphi$, i.e., whether w satisfies φ at discrete time $k \in \mathbb{Z}$, is determined according to the following recursive definition: $(w,\, k) \models a \leftrightarrow a \in w(k)$; $(w,\, k) \models \neg\varphi \leftrightarrow (w,\, k) \not\models \varphi$; $(w,\, k) \models \varphi_1 \vee \varphi_2 \leftrightarrow (w,\, k) \models \varphi_1$ or $(w,\, k) \models \varphi_2$; $(w,\, k) \models \varphi_1 \,\mathsf{U}\, \varphi_2 \leftrightarrow \exists k_1 \geq k,\, (w,\, k_1) \models \varphi_2$ and $\forall k_2 \in [k,\, k_1),\, (w,\, k_2) \models \varphi_1$. Intuitively, $\varphi_1 \,\mathsf{U}\, \varphi_2$ requires that φ_1 must be true over w until φ_2 becomes true. We say that a formula φ is satisfied by w if $(w,\, 0) \models \varphi$ or $w \models \varphi$ for brevity.

Given an LTL formula φ over AP, the set of words that satisfies φ can alternatively be captured through a *Büchi automaton* $\mathcal{A}_\varphi = (Q, 2^{AP}, \delta, Q_0, F)$, where Q is a finite set of states; 2^{AP} is the allowed input alphabet; $\delta : Q \times 2^{AP} \to 2^Q$ is a transition function; $Q_0 \subseteq Q$ is a set of initial states; and $F \subseteq Q$ is a set of accepting states. An infinite run over an input word $w = w(0)w(1)\ldots$ is an infinite sequence of states $r = q_0 q_1 \ldots$ such that $q_0 \in Q_0$ and $q_{k+1} \in \delta(q_k, w(k))$, $\forall k \geq 1$. A run is accepting if it intersects with the accepting set F infinitely many times, and an input word w is accepted if there exists an accepting run over it. The Büchi automaton $\mathcal{A}_\varphi$ can be obtained by the translation algorithm [20], such that $\mathcal{A}_\varphi$ accepts exactly the words satisfying φ.

Moreover, one particular class of LTL is the syntactically co-safe LTL (sc-LTL) [21], which contains only the U and $\Diamond$ temporal operators and is written in positive normal form [23]. In contrast to general LTL, the satisfaction of an sc-LTL formula can be achieved in a finite time, i.e., each word satisfying an sc-LTL formula φ consists of a satisfying prefix that can be followed by an arbitrary suffix.

B. Real-time Temporal Logic

To analyze properties of real-time signals, we also consider a real-time extension of LTL, i.e., the real-time temporal logic (RTL), originally introduced in [22]. Its syntax is similar to LTL as introduced in Section II-A, but its semantics is defined over *continuous-time* Boolean signals. Consider such a signal $x : \mathbb{R}_{\geq 0} \to 2^{AP}$ over the atomic propositions AP, where $x(t) \subseteq AP$ is the set of propositions that x satisfies at time $t \geq 0$. Given an RTL formula φ over AP, the satisfiability relation $(x,\, t) \models \varphi$, i.e., whether signal x satisfies φ at time $t \geq 0$, is determined via the following recursive definition: $(x,\, t) \models a \leftrightarrow a \in x(t)$; $(x,\, t) \models \neg\varphi \leftrightarrow (x,\, t) \not\models \varphi$; $(x,\, t) \models \varphi_1 \vee \varphi_2 \leftrightarrow (x,\, t) \models \varphi_1$ or $(x,\, t) \models \varphi_2$; $(x,\, t) \models \varphi_1 \,\mathsf{U}\, \varphi_2 \leftrightarrow \exists t' \geq t,\, (x,\, t') \models \varphi_2$ and $\forall t'' \in (t,\, t'),\, (x,\, t'') \models \varphi_1$. We say that an RTL formula φ is satisfied by x if $(x,\, 0) \models \varphi$ or simply $x \models \varphi$. Moreover, similarly to sc-LTL, there exists a particular class of RTL, called sc-RTL that can be satisfied by a real-time Boolean signal in finite time. It only contains the U and $\Diamond$ temporal operators and is written in positive normal form [23]. Note that the RLT semantics does not include the "*next*" operator as it is not defined in continuous time [23]. Finally, since we consider RTL and LTL with the same syntax but interpreted with different semantics, we define the following correspondence between an RTL formula and an LTL formula:

Definition 1. Given an RTL formula φ, the associated LTL formula is denoted by $[\varphi]$, which has the same expression as φ

but is evaluated under the LTL semantics. ∎

For example, consider the RTL formula $\varphi_2^1 = \Box\Diamond(R_2 \wedge \Diamond R_5) \wedge \Box\neg R_0$ that will be demonstrated in the simulation section. Its associated LTL formula $[\varphi_2^1]$ has the same expression but is evaluated under different semantics, i.e., φ_2^1 is specified over continuous-time Boolean signals, while $[\varphi_2^1]$ over discrete-time Boolean signals.

III. Problem Formulation

A. System Description

We consider a team of N autonomous agents obeying the following single-integrator dynamics in a 2D workspace:

$$\dot{p}_i(t) = u_i(t), \tag{1}$$

where $p_i(t),\, u_i(t) \in \mathbb{R}^2$ denote the i-th agent's position and control input at time $t \geq 0$, $\forall i \in \mathcal{N} \triangleq \{1, 2, \cdots, N\}$. The agents are modeled as point masses without volume, hence inter-agent collisions are not considered. Moreover, all agents have a common sensing radius $r > 0$, i.e., agent i can only exchange information with another agent j if their relative distance satisfies $\|p_i(t) - p_j(t)\| \leq r$. We also denote by $\mathbf{p}_i(t)$ the trajectory of agent i during the time interval $[0,\, t)$. Similarly, $\mathbf{p}_i([t_1,\, t_2])$ stands for the trajectory segment during time $[t_1,\, t_2]$. Finally, each agent i has a predefined nonempty neighboring set $\mathcal{N}_i \subseteq \mathcal{N}$, $i \in \mathcal{N}$ and $j \in \mathcal{N}_i$ implies $i \in \mathcal{N}_j$, $\forall j \in \mathcal{N}_i$. Nevertheless, agent i can communicate with agent $j \in \mathcal{N}_i$ only if $\|p_i(t) - p_j(t)\| \leq r$, i.e., their relative distance is less than the communication radius.

Additionally, there exists a set $\Pi \triangleq \{\pi_1, \pi_2, \cdots, \pi_M\}$ of $M \geq 1$ regions of interest within the 2D workspace. These regions may be in various shapes, such as points [24], triangles [19], polygons [25]. We consider here the circular area around the points of interest:

$$\pi_\ell \triangleq \mathcal{B}(c_\ell, r_\ell) = \{p \in \mathbb{R}^2 | \|p - c_\ell\| \leq r_\ell\}, \tag{2}$$

for $\ell = 1, 2, \cdots, M$, where $c_\ell \in \mathbb{R}^2$ is the center and $r_\ell \geq r_{\min}$ is the radius, with $r_{\min}$ denoting the minimal radius. We assume that Π is included within a large circular region denoted by $\pi_B = \mathcal{B}(c_B,\, r_B)$, which stands for the allowed workspace. However, there is a set of *local* atomic propositions $R = \{R_\ell,\, \ell = 1, 2, \cdots, M\}$ representing the property fulfilled at each region. Hence, given the agent state $p_i(t)$ at time $t \geq 0$, it holds that $R_\ell(t) = \top$ if $p_i(t) \in \pi_\ell$ and $R_\ell(t) = \bot$ otherwise, for $\ell = 1, 2, \cdots, M$. Finally, it should be noted that the workspace, i.e., the location and property of the regions above, is fully-known and static to all agents.

B. Contingent Request

As mentioned earlier, the agents may exchange information directly with their neighbors when their relative distance is less than r. Particularly, we allow each agent to send to its neighbors two types of contingent requests:

(I) ***Service***: agent i requests its neighbor $j \in \mathcal{N}_i$ at time $t = t_0^{\mathrm{s}}$ to accomplish its own short-term service described by an sc-RTL formula $\varphi_{ij,t}^{\mathrm{s}}$ over R, which is assumed to be feasible for agent j. Notice that $\varphi_{ij,t}^{\mathrm{s}}$ is predefined for each neighbor j and may be different at various request times. This request is communicated by sending the formula $\varphi_{ij,t}^{\mathrm{s}}$ directly to agent j.

(II) ***Formation***: agent i requests its neighbor $j \in \mathcal{N}_i$ at time $t = t_0^{\mathrm{f}}$ to converge to a desired relative formation described by $c_{ij} \in \mathbb{R}^2$ with a predefined transient response imposed by the corresponding performance function $\rho_{ij}(t) : \mathbb{R}^+ \to \mathbb{R}^+$. This formation has to be kept until agent i accomplishes a short-term formation task described by an sc-RTL formula $\varphi_{ij,t}^{\mathrm{f}}$ over R and a release message is sent to agent j afterwards. The formation task $\varphi_{ij,t}^{\mathrm{f}}$ is also predefined for agent i and may be different at various request times. More specifically, the relative formation error is given by:

$$e_{ij}(t) \triangleq p_i(t) - p_j(t) - c_{ij}. \tag{3}$$

Let us also define $\mu_{ij} \triangleq e_{ij}^T e_{ij}$ as a scalar measure of the formation error and

$$\widehat{\mu}_{ij}(t) \triangleq \frac{\mu_{ij}(t)}{\rho_{ij}(t)} \tag{4}$$

as the normalized error over the associated performance function $\rho_{ij}(t)$, which is a smooth, bounded and strictly decreasing function of time. Here we adopt exponential functions

$$\rho_{ij}(t) \triangleq (\rho_{ij,t_0^{\mathrm{f}}} - \rho_{ij,\infty})\, \mathrm{e}^{-l_{ij}(t - t_0^{\mathrm{f}})} + \rho_{ij,\infty}, \tag{5}$$

where $l_{ij} > 0$ specifies the decreasing rate of $\rho_{ij}(t)$, $\rho_{ij,t_0^{\mathrm{f}}} > 0$ is the initial value of $\rho_{ij}(t)$ at time t_0^{f}, chosen such that $\rho_{ij,t_0^{\mathrm{f}}} > \mu_{ij}(t_0^{\mathrm{f}})$ and $\rho_{ij,\infty} > 0$ reflects the maximum allowed steady state error. This formation request is communicated by sending c_{ij} and the parameters of $\rho_{ij}(t)$ to agent j.

To monitor the formation performance, we also define a set of *controllable* atomic propositions $H_i \triangleq \{h_{ij},\, j \in \mathcal{N}_i\}$:

$$h_{ij}(t) \triangleq \begin{cases} \top & \text{if } \;\widehat{\mu}_{ij}(t) \in D_{ij} \equiv [0,\, 1), \\ \bot & \text{otherwise.} \end{cases} \tag{6}$$

Notice that agent j needs to satisfy the prescribed formation performance, i.e., $h_{ij}(t) = \top$ until it receives a release message from agent i.

Upon receiving either a service or formation request from agent i, we assume that agent j replies immediately to either confirm or deny it. Thus, we introduce a set of *observational* atomic propositions for each agent i: $O_i \triangleq \{o_{ij}^{\mathrm{s}},\, o_{ji}^{\mathrm{f}},\, j \in \mathcal{N}_i\}$, where $o_{ij}^{\mathrm{s}}(t) = \top$ if agent i confirms agent j's service request at time t; and $o_{ij}^{\mathrm{s}}(t) = \bot$ otherwise ($o_{ij}^{\mathrm{f}}(t)$ is defined similarly for formation requests). Additionally, we introduce a set of *releasing* atomic propositions: $Z_i \triangleq \{z_{ij},\, j \in \mathcal{N}_i\}$, where $z_{ij}(t) = \top$ if agent i sends a release message to its neighbor $j \in \mathcal{N}_i$ at time t; and $z_{ij}(t) = \bot$ otherwise.

C. Local Task Specification

Let us denote by $AP_i = R \cup O_i \cup H_i \cup Z_i$ the complete set of atomic propositions of each agent $i \in \mathcal{N}$. Each agent i is assigned a local task defined as an RTL formula φ_i over AP_i, which has the following structure:

$$\varphi_i \triangleq \varphi_i^{\mathrm{l}} \wedge \varphi_i^{\mathrm{s}} \wedge \varphi_i^{\mathrm{f}}. \tag{7}$$

Notice that φ_i^{l} describes a static task specification as a general RTL formula over R as the *local* task of agent i. Similarly, φ_i^{s}

is a contingent task specification regarding the *service* requests that agent i may receive from agent j with $i \in \mathcal{N}_j$:

$$\varphi_i^{\mathtt{s}} \triangleq \bigwedge_{j \in \mathcal{N}_i} \Box \, (o_{ij}^{\mathtt{s}} \to \varphi_{ji,t}^{\mathtt{s}}). \tag{8}$$

Hence, whenever agent i confirms a service request $\varphi_{ji,t}^{\mathtt{s}}$ from agent j, then it should fulfill this service task afterwards. Finally $\varphi_i^{\mathtt{f}}$ is the contingent task regarding the *formation* requests that agent i may send to its neighbor $j \in \mathcal{N}_i$:

$$\varphi_i^{\mathtt{f}} \triangleq \bigwedge_{j \in \mathcal{N}_i} \Box \Big(o_{ji}^{\mathtt{f}} \to \; \big(\varphi_{ij,t}^{\mathtt{f}} \wedge (h_{ij} \, \mathsf{U} \, z_{ij}) \big) \Big). \tag{9}$$

Thus, whenever a formation request from agent i is confirmed by agent j, the corresponding short-term formation task $\varphi_{ij,t}^{\mathtt{f}}$ should be fulfilled right afterwards and the formation controllable proposition h_{ij} should be always kept true, until a release message is sent to agent j. In other words, once agent j confirms a formation request from agent i, it should achieve the desired relative position with prescribed performance as imposed by the corresponding performance function $\rho_{ij}(t)$, until agent i has accomplished its formation task $\varphi_{ij,t}^{\mathtt{f}}$.

Remark 1. While the local task $\varphi_i^{\mathtt{l}}$ has been commonly-seen [25], [26] as a static task specification, the main novelty of this work lies in the contingent service task $\varphi_i^{\mathtt{s}}$ by (8) and the formation task $\varphi_i^{\mathtt{f}}$ introduced by (9) that have not been tackled for multi-agent systems in this context. Moreover, notice that all service and formation requests are exchanged online and hence cannot be known before the system starts. The validity of the observational and controllable propositions can only be determined in real-time. Thus, only the local task $\varphi_i^{\mathtt{l}}$ and not the complete specification φ_i is determined and assigned to agent i before the system initializes. ■

In the sequel we formulate the problem confronted here:

Problem 1. Consider a team of agents obeying the dynamics (1), with each member assigned a set of local task specifications defined in (7)-(9). The objective is to synthesize a distributed hybrid control protocol such that all RTL formulas φ_i are satisfied, $\forall i \in \mathcal{N}$. ■

IV. Main Results

The proposed hybrid control strategy consists of four major components as shown in Figure 1 (i.e., the communication, the event monitoring, the discrete planning and the hybrid control modules). In this section, we first present two continuous motion control schemes regarding the navigation and the formation tasks respectively. Subsequently we describe the real-time event monitoring module that handles critical events during the system operation as well as the established communication protocol among the agents regarding the contingent requests. Afterwards, we present the discrete plan synthesis and adaptation algorithms that handle the contingent service and formation requests. Finally, the overall hybrid control strategy is synthesized based on the aforementioned components and its correctness is formally proven.

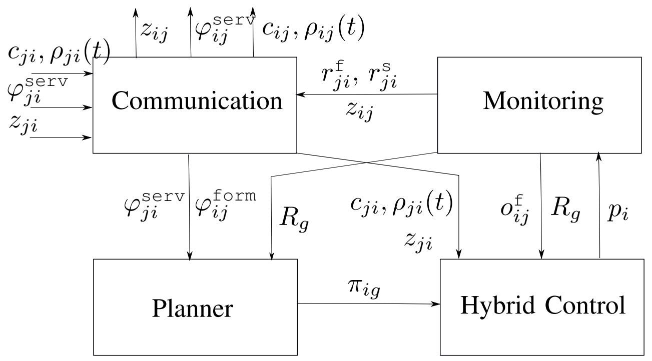


Figure 1. The overall structure of the hybrid control scheme. The arrows indicate the information flow and the communicated variables. A detailed description is given in Subsection IV-E.

A. Continuous Controller Design

In this subsection, we describe the continuous control design for two different control objectives, i.e., to navigate an agent from any initial position to its goal region, without crossing any undesirable regions; and to establish a desired formation with prescribed transient response.

1) Navigation Control: Recall that the set of circular regions of interest within the workspace boundary π_B is given by $\Pi = \{\pi_1, \pi_2, \cdots, \pi_M\}$. Since there are no explicit representations of "obstacles" and no fixed initial or goal regions, we denote by $\pi_g = \mathcal{B}(c_g, r_g) \in \Pi$ the goal region and $\pi_s = \mathcal{B}(c_s, r_s) \in \Pi$ the initial region. In this work, we rely on the navigation function approach proposed by Rimon and Koditschek in [24] to navigate agent i from π_s to π_g, without crossing other undesirable regions $\pi_j \in \Pi$ with $j \neq \{s, g\}$. In particular, the navigation function Φ_i is constructed as:

$$\Phi_i(p_i) \triangleq \frac{\gamma_g}{(\gamma_g^k + \beta_{gs})^{\frac{1}{k}}}, \tag{10}$$

where $k > 1$ is a design parameter, $p_i \in \mathbb{R}^2$, $\Phi_i \in [0, 1]$, $\gamma_g \triangleq \|p_i - c_g\|^2$ represents the attractive potential field to the desired position $c_g \in \mathbb{R}^2$ and $\beta_{gs} \triangleq \beta_B \prod_{j=1, j \neq s,g}^{M} \beta_j$ is the repulsive potential field by the workspace boundary and the undesirable regions that should be avoided, with $\beta_B \triangleq r_B^2 - \|p_i - c_B\|^2$ and $\beta_j \triangleq \|p_i - c_j\|^2 - r_j^2$. For brevity, we denote by $\Pi_{\text{avoid}} \triangleq \cup_{j=1, j \neq s,g}^{M} \pi_j$ the set of circular regions that should be avoided and by $\pi_B \backslash \Pi_{\text{avoid}}$ the free space. It is assumed that π_B and the sphere regions in Π satisfy the condition of a valid workspace in [24], i.e., $\pi_m \subset \pi_B$ and $\pi_m \cap \pi_n = \emptyset$, $\forall m, n \in \{1, 2, \cdots, M\}$ with $m \neq n$.

It has been proved in [24] that $\Phi_i(p_i)$ has only one global minimum at c_g and $M-2$ saddle points within the allowed free space with zero set-measure for a sufficiently large constant k. Hence, a feasible path within the free workspace that leads an agent from its initial position in π_s to its goal region π_g can be generated by following the negated gradient of $\Phi_i(p_i)$ or equivalently by adopting the subsequent control law:

$$u_i \, = \, -\nabla_{p_i} \Phi_i(p_i). \tag{11}$$

Based on [27], it is proven that $\gamma_g \to 0$ as $t \to \infty$ and $\beta_{gs} > 0$ holds, $\forall t \geq 0$. Moreover, a collision free path is ensured for almost any initial position in the free space (except a set of measure zero) to any goal region in the free space. Furthermore, the asymptotic stability of the aforementioned controller guarantees the convergence to the neighborhood of p_g (i.e., the goal region π_g) in finite time [27].

Lemma 1. *The navigation control law* (11) *remains always bounded, i.e., $\|u_i\| < u_{\max}$, where $u_{\max} > 0$ is finite.*

Proof. By computing the gradient of $\Phi_i(p_i)$ directly and owing to the fact that p_i remains within the allowed free space without crossing Π_{avoid}, all terms of $\nabla_{p_i}\Phi_i$ remain bounded and hence $\|u_i\| = \|\nabla_{p_i}\Phi_i\| \leq u_{\max}$ with an upper bound $u_{\max} > 0$ depending on the configuration of Π. ■

2) Formation Control with Prescribed Performance: As described in Section III-B, a contingent request may also involve a formation task that should be executed with prescribed performance (i.e., agent i requests agent $j \in \mathcal{N}_i$ to converge to a desired formation described by the relative position c_{ij} and the predefined transient response by the associated performance function $\rho_{ij}(t)$ defined in (5)). Moreover, this formation should be maintained until the associated short-term formation task $\varphi^{\text{f}}_{ij,t}$ is satisfied by agent i. Since agent i may send a formation request to more than one neighbors, we denote the set of neighbors that have confirmed its formation request by $\mathcal{N}^{\text{f}}_i \subseteq \mathcal{N}_i$. In the sequel, we propose the motion control scheme for both agent i and all agents $j \in \mathcal{N}^{\text{f}}_i$.

Notice that at the request time $t = t^{\text{f}}_0$ in case agent i lies inside a region $\pi_s \in \Pi$ then the associated navigation function can be constructed similarly to (10), i.e., $\Phi^{\text{f}}_i(p_i) \triangleq \frac{\gamma_g}{(\gamma_g^k + \beta_{gs})^{\frac{1}{k}}}$; otherwise, the repulsive potential β_{gs} should be slightly modified since all regions in Π except π_g should be treated as undesirable regions to be avoided, i.e., $\Phi^{\text{f}}_i(p_i) \triangleq \frac{\gamma_g}{(\gamma_g^k + \beta_g)^{\frac{1}{k}}}$, where $\beta_g \triangleq \beta_B \prod_{j=1, j\neq g}^{M} \beta_j$. The controller of agent i is:

$$u_i = -\nabla_{p_i} \Phi^{\text{f}}_i(p_i). \tag{12}$$

On the other hand, for each agent $j \in \mathcal{N}^{\text{f}}_i$ that has confirmed the formation request of agent i, we design a motion control law that ensures the convergence to the desired relative formation with prescribed performance as follows:

$$u_j = \frac{K_{ij}}{2} \frac{\varepsilon_{ij}}{\rho_{ij}(t)} e_{ij}, \tag{13}$$

where $K_{ij} > 0$ is a control gain; e_{ij} is the relative formation error from (3); $\rho_{ij}(t)$ is the performance function; and

$$\varepsilon_{ij} \triangleq \ln\Big(\frac{1}{1 - \widehat{\mu}_{ij}}\Big), \tag{14}$$

with $\widehat{\mu}_{ij}$ denoting the normalized formation error from (4).

Lemma 2. *Under the motion control laws* (12) *and* (13)*, initialized at time $t = t^f_0$, agent i will arrive to its goal region π_g within finite time $T_g > t^f_0$, while all agents $j \in \mathcal{N}^f_i$ that confirmed its formation request will satisfy the controllable proposition $h_{ij}(t) = \top$, $\forall t \in [t^f_0, T_g]$.*

Proof. Following similar analysis with Section IV-A1, we can show that agent i arrives its goal region π_g within finite time $T_g > t^{\text{f}}_0$, while avoiding the undesirable regions included in Π_{avoid}. Regarding the agents $j \in \mathcal{N}^{\text{f}}_i$, the dynamics of the normalized formation error is calculated as follows:

$$\dot{\widehat{\mu}}_{ij} = \frac{2\, e_{ij}^T u_{ij}\, \rho_{ij}(t) - e_{ij}^T e_{ij}\, \dot{\rho}_{ij}(t)}{\rho^2_{ij}(t)}$$

where $u_{ij}(t) = u_i(t) - u_j(t)$. Substituting the control input (13) and $e^T_{ij}(t)\, e_{ij}(t) \equiv \rho_{ij}(t)\, \widehat{\mu}_{ij}(t)$, we obtain:

$$\begin{aligned}\dot{\widehat{\mu}}_{ij} &= -\frac{\widehat{\mu}_{ij}}{\rho_{ij}(t)}\big[K_{ij}\,\varepsilon_{ij} + \dot{\rho}_{ij}(t)\big] + \frac{2\, e^T_{ij}\, u_i(t)}{\rho_{ij}(t)} \\ &= -\frac{\widehat{\mu}_{ij}}{\rho_{ij}(t)}\big[K_{ij} \ln\big(\frac{1}{1-\widehat{\mu}_{ij}}\big) + \dot{\rho}_{ij}(t)\big] + \frac{2\, e^T_{ij}\, u_i(t)}{\rho_{ij}(t)}.\end{aligned} \tag{15}$$

Since the right-hand side of (15) is continuous on t and locally Lipschitz on $\widehat{\mu}_{ij}$ over D_{ij}, as defined in (6), we may apply Theorem 54 of [17] (page 476) to deduce a maximal solution of (15) on a time interval $[t^{\text{f}}_0, \tau_{\max})$ such that $\widehat{\mu}_{ij}(t) \in D_{ij}$, $\forall t \in [t^{\text{f}}_0, \tau_{\max})$. Then consider the Lyapunov-like function:

$$V_{ij}(t) \triangleq \frac{1}{2}\varepsilon^2_{ij} \tag{16}$$

which owing to (14) is well-defined for all $t \in [t^{\text{f}}_0, \tau_{\max})$. Differentiating $V_{ij}(t)$ with respect to time, we get:

$$\dot{V}_{ij} = \varepsilon_{ij}\, \dot{\widehat{\mu}}_{ij} = \frac{\varepsilon_{ij}}{1 - \widehat{\mu}_{ij}}\, \dot{\widehat{\mu}}_{ij}. \tag{17}$$

Moreover, invoking (15), we arrive at:

$$\dot{V}_{ij} = -m_{ij}\Big[\widehat{\mu}_{ij}\big(K_{ij}\,\varepsilon_{ij} + \dot{\rho}_{ij}(t)\big) - 2\, e^T_{ij}\, u_i(t)\Big], \tag{18}$$

where $m_{ij}(t) = \frac{\varepsilon_{ij}}{(1-\widehat{\mu}_{ij})\,\rho_{ij}(t)} > 0$. Hence, since $\varepsilon_{ij} > 0$, $\rho_{ij}(t) > 0$ and $|e^T_{ij}\, u_i(t)| \leq \|e_{ij}\|\|u_i(t)\|$, we conclude that:

$$\begin{aligned}\dot{V}_{ij} &\leq -m_{ij}\Big[K_{ij}\,\widehat{\mu}_{ij}\,\varepsilon_{ij} + \widehat{\mu}_{ij}\,\dot{\rho}_{ij}(t) - 2\,\|e_{ij}\|\|u_i(t)\|\Big] \\ &\leq -m_{ij}\Big[K_{ij}\,\widehat{\mu}_{ij}\,\varepsilon_{ij} - |\dot{\rho}_{ij}(t)| - 2\sqrt{\rho_{ij}(t)}\,\|u_i(t)\|\Big],\end{aligned}$$

where we employed the fact that $0 \leq \widehat{\mu}_{ij} < 1$ and $\|e_{ij}\| = \sqrt{\mu_{ij}} = \sqrt{\widehat{\mu}_{ij}\,\rho_{ij}(t)} \leq \sqrt{\rho_{ij}(t)}$. Thus, invoking the inverse of (14) (i.e., $\widehat{\mu}_{ij} = 1 - \mathbf{e}^{-\varepsilon_{ij}}$), we conclude that $\dot{V}_{ij} < 0$ when

$$\begin{aligned}&\varepsilon_{ij}\big(1 - \mathbf{e}^{-\varepsilon_{ij}}\big) \\ &> \frac{\sup_{t\in[T_0,\,T_g)}\{|\dot{\rho}_{ij}(t)| + 2\sqrt{\rho_{ij}(t)}\,\|u_i(t)\|\}}{K_{ij}} \triangleq b_{ij}.\end{aligned} \tag{19}$$

Notice that b_{ij} is finite as $K_{ij} > 0$, $\dot{\rho}_{ij}(t)$, $\rho_{ij}(t)$ are bounded by construction and $\|u_i(t)\| < u_{\max}$ as proven in Lemma 1.

Consider now the smooth function $g(x) = x(1 - \mathbf{e}^{-x})$, which is monotonically increasing for $x > 0$ with $g(0) = 0$. Let $\varepsilon^\star_{ij}$ be a constant that satisfies $\varepsilon^\star_{ij}(1 - \mathbf{e}^{-\varepsilon^\star_{ij}}) = b_{ij}$, which exists and is unique owing to the monotonicity of $g(x)$. Hence, $\dot{V}_{ij} < 0$ when $\varepsilon_{ij} > \varepsilon^\star_{ij}$, from which we conclude that $\varepsilon_{ij}(t) < \max\{\varepsilon^\star_{ij},\, \varepsilon_{ij}(t^{\text{f}}_0)\} \triangleq \overline{\varepsilon}_{ij}$, $\forall t \in [t^{\text{f}}_0, \tau_{\max})$ and

$$\widehat{\mu}_{ij}(t) < 1 - \mathbf{e}^{-\overline{\varepsilon}_{ij}} \triangleq \overline{\widehat{\mu}}_{ij} < 1, \qquad \forall t \in [t^{\text{f}}_0, \tau_{\max}).$$

Thus, we deduce that $\widehat{\mu}_{ij}(t) \in [0,\, \overline{\widehat{\mu}}_{ij}] \triangleq D'_{ij}$, $\forall t \in [t^{\text{f}}_0, \tau_{\max})$, where D'_{ij} is a nonempty and compact subset of D_{ij}.

Finally, what remains to be shown is that $\tau_{\max}$ can be any time greater than t^{f}_0. Therefore, assume that $\tau_{\max} < \infty$. Since

$D'_{ij} \subset D_{ij}$ and $\widehat{\mu}_{ij}(t)$ is a maximal solution of (15) over $[t_0^{\mathsf{f}},\, \tau_{\max})$, Proposition C3.6 in [17] (page 481) dictates the existence of a time instant $t' \in [t_0^{\mathsf{f}},\, \tau_{\max})$ such that $\widehat{\mu}_{ij}(t) \notin D'_{ij}$, which contradicts with the fact that $\widehat{\mu}_{ij}(t) \in D'_{ij}$, $\forall t \in [t_0^{\mathsf{f}},\, \tau_{\max})$. Thus $\tau_{\max}$ can be extended to any time greater than t_0^{f} (i.e., $\tau_{\max}$ can be selected to be equal to T_g) and $\widehat{\mu}_{ij}(t) \in [0,\, \widehat{\underline{\mu}}_{ij}] \subset D_{ij}$, which ensures that $e_{ij}^T(t)e_{ij}(t) < \rho_{ij}(t)$ and consequently $h_{ij}(t) = \top$, $\forall t \in [T_0,\, T_g]$. ■

B. Real-time Event Monitoring Scheme

It is apparent that monitoring several real-time events plays a crucial role in the satisfaction of all tasks. For each agent $i \in \mathcal{N}$, four types of events need to be monitored online:

(a) *Region cross event*. It occurs when an agent enters or leaves a region in Π. More precisely, agent i enters the region $\pi_\ell \in \Pi$ at time $t_0 > 0$ if $p_i(t_0 - \delta_d) \notin \pi_\ell$ and $p_i(t_0) \in \pi_\ell$, where $\delta_d > 0$ is a design parameter as the dwell time [28]. Agent i leaves the region $\pi_\ell \in \Pi$ at time $t_0 > 0$ if $p_i(t_0 - \delta_d) \in \pi_\ell$ and $p_i(t_0) \notin \pi_\ell$. This event is directly related to the validity of proposition $R_\ell \in R$. δ_d can be enforced by setting the crossing event true only if agent i has been inside region π_ℓ for at least time period of length δ_d.

(b) *Request and reply event*. It occurs when agent i sends a service or formation request to its neighbor $j \in \mathcal{N}_i$ at time t_0 and agent j replies this request at the same time. Depending on the replies, we consider two cases: if agent j confirms this service request, the observational proposition $o_{ji}^{\mathsf{s}} \in O_i$ satisfies $o_{ji}^{\mathsf{s}}(t) = \top$, $\forall t \in [t_0,\, t_0 + \delta_d)$ and $o_{ji}^{\mathsf{s}}(t) = \bot$, $\forall t \in [t_0 + \delta_d,\, t_1]$, where $t_1 > t_0 + \delta_d$ is the next time instant when agent j receives a request from agent i; otherwise, if agent j denies this request, $o_{ji}^{\mathsf{s}}(t) = \bot$, $\forall t \in [t_0,\, t_1]$. The same rules apply to propositions related to formation requests $\{o_{ji}^{\mathsf{f}},\, \forall j \in \mathcal{N}_i\}$. In addition, if agent i sends a release message to its neighbor $j \in \mathcal{N}_i$ at t_0, then $z_{ij}(t) = \top$, $\forall t \in [t_0,\, t_0 + \delta_d)$ and $z_{ij}(t) = \bot$, $\forall t \in [t_0 + \delta_d,\, t_1]$, where t_1 is the next time instant when agent i sends a release message to agent j.

(c) *Service finish event*. It occurs when agent i has finished a service request that it has confirmed. For instance, suppose that $o_{ij}^{\mathsf{s}}(t_0) = \top$ and agent i confirms the service request $\varphi_{ji,t_0}^{\mathsf{s}}$ by agent j at time t_0. Then $\varphi_{ji,t_0}^{\mathsf{s}}$ is satisfied by agent i at time $t_f \geq t_0$, if its trajectory $\mathbf{p}_i([t_0,\, t_f])$ satisfies $\varphi_{ji,t_0}^{\mathsf{s}}$ on the basis of the semantics presented in Section II-B. Since all service tasks are feasible and can be finished in finite time, the finishing time $t_f > t_0$ is finite for each service request.

(d) *Formation finish event*. It occurs when agent i has finished a formation task that has been confirmed. For instance, suppose that $o_{ji}^{\mathsf{f}}(t_0) = \top$ and agent j confirms the formation request by agent i at time $t_0 > 0$. The corresponding formation task for agent i is given by $\varphi_{ij,t_0}^{\mathsf{f}}$. Then $\varphi_{ij,t_0}^{\mathsf{f}}$ is satisfied at time $t_f \geq t_0$ if the trajectory $\mathbf{p}_i([t_0,\, t_f])$ satisfies $\varphi_{ij,t_0}^{\mathsf{f}}$ on the basis of the semantics presented in Section II-B. Since all formation tasks are also assumed to be feasible in finite time, the time t_f is also finite for each formation request.

Summarizing, the events (a) and (b) can be easily monitored online. However it is not trivial to monitor the events (c) and (d) efficiently, especially when multiple service or formation requests occur. In this sense, since events (c) and (d) are closely related to the discrete plan synthesis and adaptation module in Section IV-D, more details will be given there.

C. Communication Protocol for Contingent Requests

Each agent $i \in \mathcal{N}$ will receive, send and confirm various contingent requests from its neighbors during the system operation. Recall that agent i can communicate with agent $j \in \mathcal{N}_i$ at time t only if $\|p_i(t) - p_j(t)\| \leq r$. Hence, let us denote by $r_{ji}^{s}(t) : \mathbb{R}^+ \to \mathbb{B}$ the continuous-time Boolean variable indicating whether agent i is serving a service request from agent $j \in \mathcal{N}_i$ at time $t > 0$. In addition, let $r_{ji}^{\mathsf{f}}(t) : \mathbb{R}^+ \to \mathbb{B}$ be the variable indicating whether agent i is serving a formation request from agent $j \in \mathcal{N}_i$ at time $t > 0$. Initially, $r_{ji}^{\mathsf{s}}(0) = \bot$ and $r_{ji}^{\mathsf{f}}(0) = \bot$, $\forall j \in \mathcal{N}_i$. Moreover, $r_{ii}^{\mathsf{f}}(t) : \mathbb{R}^+ \to \mathbb{B}$ indicates whether agent i is currently performing its own formation task, which is also initialized as false, i.e., $r_{ii}^{\mathsf{f}}(0) = \bot$.

Agent i may send the predefined service formula $\varphi_{ij,t}^{\mathsf{s}}$ to its neighbor $j \in \mathcal{N}_i$ at time $t > 0$ only if $r_{ij}^{\mathsf{s}}(t) = \bot$ (i.e., agent j is not still serving the previous service request from agent i). Similarly, agent i may send the formation request described by c_{ij} and $\rho_{ij}(t)$ at time $t > 0$, only if $r_{gi}^{\mathsf{f}} = \bot$, $\forall g \in \mathcal{N}_i$ (i.e., agent i itself is not currently serving any of its neighbor's formation request) and $r_{ij}^{\mathsf{f}}(t) = \bot$ (i.e., agent j is not still serving the previous formation request from agent i). After sending a request, agent i needs to wait for a reply from agent j. Once its request is confirmed by agent j, the observational proposition $o_{ij}^{\mathsf{s}}(t)$ or $o_{ij}^{\mathsf{f}}(t)$ is updated according to Section IV-B. In the same vein, other approaches like direct user triggering could also be incorporated.

On the other hand, whenever agent i receives a request from its neighbor $j \in \mathcal{N}_i$ at time $t > 0$, it checks first whether $r_{ii}^{\mathsf{f}}(t) = \top$ (i.e., agent i is currently executing its own formation task) or $r_{ji}^{\mathsf{f}}(t) = \top$, for any $j \in \mathcal{N}_i$ (i.e., it is already in formation with one of its neighbors). If so, agent i denies this request; otherwise, agent i confirms its request and starts serving it. In other words, when agent i receives multiple formation requests from its neighbors, it will reply to them according to the time they are received, i.e., in a first-come-first-serve manner. Moreover, if multiple service and formation requests are received *simultaneously*, then formation requests are prioritized over service requests and furthermore the service or formation requests from different neighbors are replied in the order of their identity numbers. Namely, the neighbor with larger identity number is replied to earlier.

D. Discrete Plan Synthesis and Adaptation

In this subsection, we show how to synthesize the initial discrete plan for each agent, based on its own local task, and how to adapt it online upon receiving contingent service and formation requests from neighboring agents.

1) Initial Discrete Plan Synthesis: At time $t = 0$, we assume that no requests have been sent yet, i.e., $o_{ij}^{\mathsf{s}}(0) = o_{ij}^{\mathsf{f}}(0) = \bot$, $\forall j \in \mathcal{N}_i$ and $\forall i \in \mathcal{N}$. As a result, only the static task φ_i^1 of the RTL formula φ_i defined in (7) is initially pursued. Hence, the initial plan synthesis aims at finding a discrete plan that satisfies φ_i^1. Our approach relies on the automaton-based model-checking algorithm [18]. Thus, we first define a Finite Transition System (FTS) as an abstraction of the agent's motion among the regions of interest.

Definition 2. The motion of agent i is abstracted by an FTS:

$$\mathcal{T}_i \triangleq (\Pi_i,\, \longrightarrow,\, R,\, L,\, \Pi_{i,0},\, W). \qquad (20)$$

where $\Pi_i \triangleq \{\pi_0\} \cup \Pi$ is the set of states with $\pi_0 \triangleq \mathcal{B}(p_i(0),\, 0)$; $\longrightarrow \triangleq (\Pi\times\Pi)\cup(\{\pi_0\}\times\Pi)$ denotes the transition relation; $L : \Pi_i \to 2^R$ with $L(\pi_\ell) \triangleq R_\ell,\, \forall \ell \in \{1, 2, \cdots, M\}$ is the labeling function; $\Pi_{i,0} \triangleq \{\pi_0\}$ is the initial state; and W : " $\longrightarrow$ " $\to \mathbb{R}^+$ computes the cost of each transition with $W(\pi_m, \pi_n) \triangleq \|c_m - c_n\|,\, \forall(\pi_m,\, \pi_n) \in\longrightarrow$. ■

Notice that only the local propositions of R are allowed in $\mathcal{T}_i$ and that the initial state π_0 is solely determined by the initial position of agent i. A path of $\mathcal{T}_i$ is given by $\tau = \pi_i^0\pi_i^1\pi_i^2\cdots$, where $(\pi_i^k,\, \pi_i^{k+1}) \in\longrightarrow,\, \forall k \geq 0$ and its associated trace, which corresponds to a discrete-time Boolean signal over R, is defined as the sequence of propositions that are true along the path, i.e., $\texttt{trace}(\tau) = L(\pi_i^0)L(\pi_i^1)\cdots$.

On the other hand, φ_i^1 is a general RTL formula over R with a corresponding LTL formula denoted by $[\varphi_i^1]$ based on Definition 1. Thus, following the notations defined in Section II-A, we may construct the NBA associated with $[\varphi_i^1]$ and denote it by $\mathcal{B}_{i,s} = (Q_s,\, 2^R,\, \delta_s,\, Q_{s,0},\, F_s)$. Subsequently, we may construct the weighted product automaton as follows:

Definition 3. The weighted product Büchi automaton $\mathcal{P}_i = \mathcal{T}_i \times \mathcal{B}_{i,s} = (Q_p',\, \delta_p',\, Q_{p,0}',\, F_p',\, W_p)$ is defined by $Q_p' = \Pi_i \times Q_s$ with $q_p' = \langle \pi,\, q\rangle \in Q_p',\, \forall \pi \in \Pi_i$ and $\forall q \in Q_s$; $\delta_p' : Q_p' \to 2^{Q_p'}$ with $\langle \pi_d,\, q_n\rangle \in \delta_s'(\langle \pi_c,\, q_m\rangle)$ if and only if $(\pi_c,\, \pi_d) \in\longrightarrow$ and $q_n \in \delta_s(q_m,\, L(\pi_c))$; $Q_{p,0}' = \Pi_{i,0} \times Q_{s,0}$ is the set of initial states; $F_p' = \Pi_i \times F_s$ is the set of accepting states; $W_p : \delta_p' \to \mathbb{R}^+$ with $W_p(\langle \pi_c,\, q_m\rangle,\, \langle \pi_d,\, q_n\rangle) = W(\pi_c,\, \pi_d),\, \forall(\langle \pi_c,\, q_m\rangle,\, \langle \pi_d,\, q_n\rangle) \in \delta_p'$. ■

After $\mathcal{P}_i$ is constructed, we search for one of its accepting runs denoted by R_i that has a prefix-suffix structure (i.e., $\mathsf{R}_i = \mathsf{R}_{i,\text{pre}}\, (\mathsf{R}_{i,\text{suf}})^\omega$) and minimizes the total cost defined by $\texttt{cost}(\mathsf{R}_i, \mathcal{P}_i) = \texttt{cost}(\mathsf{R}_{i,\text{pre}}) + \texttt{cost}(\mathsf{R}_{i,\text{suf}})$, where $\texttt{cost}(\mathsf{R}_{i,\text{pre}})$ and $\texttt{cost}(\mathsf{R}_{i,\text{suf}})$ are simply the accumulated weight of the transitions along the finite sequence of product states in $\mathsf{R}_{i,\text{pre}}$ and $\mathsf{R}_{i,\text{suf}}$. In this respect, since $\mathcal{P}_i$ may be viewed as a directed graph with initial and accepting states, a variation of the Dijkstra's shortest path algorithm can be used to find such an optimal accepting run. The worst-case computational complexity is $|Q_p'| \cdot \log(|\delta_p'|)$, where $|Q_p'|$, $|\delta_p'|$ are the number of states and transitions. For algorithmic details, we refer the readers to Algorithms 1 and 2 in [26]. Hence denoting by $\mathsf{R}_{i,0}$ this optimal accepting run and projecting it back onto Π_i yields the initial plan of agent i as $\tau_{i,0} = \mathsf{R}_{i,0}|_{\Pi_i}$. Notice that $\tau_{i,0}$ has the prefix-suffix structure and that the trace of $\tau_{i,0}$ satisfies $[\varphi_i^1]$ automatically [18]. In this way, the plan $\tau_{i,0}$ obtains the following sequence:

$$\tau_{i,0} = \pi_{i0}\pi_{i1}\cdots\big(\pi_{ik_i}\,\pi_{i(k_i+1)}\cdots\pi_{iK_i}\big)^\omega, \qquad (21)$$

with its trace given by $\texttt{trace}(\tau_{i,0}) = L(\pi_{i0})L(\pi_{i1})\cdots$, where $\pi_{i\ell} \in \Pi_i,\, \forall \ell = 1, 2, \cdots, K_i$; $\pi_{i0}\pi_{i1}\cdots\pi_{ik_i}$ is the plan prefix defined as a finite sequence of goal regions to reach; and $\pi_{ik_i}\pi_{i(k_i+1)}\cdots\pi_{iK_i}$ is the plan suffix, which is also finite but should be repeated infinitely often in order to satisfy $[\varphi_i^1]$.

If no contingent requests are sent or confirmed by agent i for all $t > 0$, its initial discrete plan $\tau_{i,0}$ should remain unchanged and be executed as follows. Initializing at $p_i(0)$, agent i moves to and enters region π_{i1} employing the motion controller described in Section IV-A1. Then, an event that agent i entered region π_{i1} should be detected. Afterwards, it leaves region π_{i1} and moves towards region π_{i2} under the same control scheme. Such procedure is repeated until it reaches π_{iK_i}, after which the next goal region is set to π_{ik_i} and continues to π_{iK_i} and back to π_{ik_i} again. In this way, the plan suffix is repeated infinitely often as $t \to \infty$. Thus, there exists an infinite sequence of time instants $0\, t_1^1\, t_1^2\, t_2^1\, t_2^2 \cdots t_k^1\, t_k^2 \cdots$, with $t_{k+1}^2 > t_{k+1}^1 > t_k^2 > t_k^1 > 0,\, \forall k \geq 1$, such that

$$p_i(t) \in \pi_{ik}, \quad \forall t \in [t_k^1,\, t_k^2),\ \forall k \geq 1, \qquad (22)$$

where $\pi_{ik} \triangleq \pi_{ik'}$ with $k' \triangleq \texttt{mod}(k - k_i,\, K_i - k_i) + k_i,\, \forall k > K_i$ ($\texttt{mod}$ is the modulo operation). Thus the trajectory $\mathbf{p}_i(t)$ intersects with the sequence of goal regions by $\tau_{i,0}$.

Theorem 1. *If no contingent requests are sent nor confirmed by agent i, i.e., $o_{ji}^{\mathsf{s}}(t) = o_{ij}^{\mathsf{f}}(t) = \bot,\, \forall j \in \mathcal{N}_i$ and $\forall t \geq 0$, then the resulting trajectory $\mathbf{p}_i(t)$ of $\tau_{i,0}$ satisfies the φ_i.*

Proof. If $o_{ji}^{\mathsf{s}}(t) = o_{ij}^{\mathsf{f}}(t) = \bot,\, \forall j \in \mathcal{N}_i$ and $\forall t \geq 0$, the service task φ_i^{s} and formation task φ_i^{f} defined in (8) and (9) may be ignored owing to the semantics of the implication operator. Thus, φ_i is equivalent to φ_i^1 and $[\varphi_i]$ is equivalent to $[\varphi_i^1]$. Notice also that φ_i^1 is specified over R and therefore depends solely on the agent's trajectory. Moreover, we have shown that the trace of $\tau_{i,0}$ satisfies the LTL formula $[\varphi_i^1]$ as well as that the trajectory $\mathbf{p}_i(t)$ satisfies (22). Hence, what we need further to show is that $\mathbf{p}_i(t)$ satisfies the RTL formula φ_i^1. Let us define $w = \texttt{trace}(\tau_{i,0})$, which is a discrete-time Boolean signal over R that $(w,\, 0) \models [\varphi_i^1]$.

- If $(w,\, 0) \models [R_\ell]$ for $R_\ell \in R$, i.e., agent i starts from region π_ℓ, then (22) guarantees that there exists a time interval $[0,\, t_1)$ with $t_1 > 0$ such that $p_i(t) \in \pi_\ell,\, \forall t \in [0,\, t_1)$. Thus, invoking the RTL semantics we conclude that $(\mathbf{p}_i,\, 0) \models R_\ell$. Similar arguments may also apply for $[\neg R_\ell]$ with $R_\ell \in R$.
- If $(w,\, 0) \models [R_{\ell_1} \vee R_{\ell_2}]$ for $R_{\ell_1},\, R_{\ell_2} \in R$, i.e., agent i starts either from region π_{ℓ_1} or π_{ℓ_2}, then (22) guarantees that there exists a time interval $[0,\, t_1)$ with $t_1 > 0$ such that $p_i(t) \in \pi_{\ell_1}$ or $p_i(t) \in \pi_{\ell_2},\, \forall t \in [0,\, t_1)$. Thus, invoking the RTL semantics we also conclude that $(\mathbf{p}_i,\, 0) \models R_{\ell_1} \vee R_{\ell_2}$.
- If $(w,\, 0) \models [R_{\ell_1}\mathsf{U}R_{\ell_2}]$ for $R_{\ell_1},\, R_{\ell_2} \in R$, i.e., agent i stays at region π_{ℓ_1} before it moves to region π_{ℓ_2}, then (22) guarantees that there exists a time interval $[0,\, t_1)$ with $t_1 > 0$ such that $p_i(t) \in \pi_{\ell_1}\ \forall t \in [0,\, t_1)$ and a subsequent one $[t_1,\, t_2)$ with $t_2 > t_1$ such that $p_i(t) \in \pi_{\ell_2},\, \forall t \in [t_1,\, t_2)$. Thus, invoking the RTL semantics we also conclude that $(\mathbf{p}_i,\, 0) \models R_{\ell_1}\mathsf{U}R_{\ell_2}$. Similar arguments also apply inductively to other operators like $\Diamond$, $\to$ and $\Box$. Thus, since $(\texttt{trace}(\tau_{i,0}),\, 0) \models [\varphi_i]$, we conclude that $(\mathbf{p}_i(t),\, 0) \models \varphi_i$. ■

The above results are valid only if *no* contingent requests are exchanged, which however is not the case here. Thus, in the sequel we study how service or formation requests should be handled by adapting the local discrete plans.

2) Event-based Discrete Plan Adaptation: Initially, we show how to handle contingent service requests. Based on the previous analysis, an agent may confirm and serve multiple service requests simultaneously. In such case, agent i needs to satisfy three types of tasks: (i) the static local task; (ii) all the service requests received so far that have not been satisfied; and (iii) the newly-received service request. These tasks need to be treated differently since the first type should be satisfied by the whole trajectory from $t = 0$; the second type by the trajectory starting from the time the service requests are confirmed; the third type by the future trajectory. Thus, it is important to keep track of how much the local task as well as the past and current service requests have been satisfied. In that respect, we consider three different cases:

Case I: Suppose the first service request $\varphi^{\mathsf{s}}_{ji,t_j}$ received by agent i at time $t_j > 0$ was sent by agent j. Agent i needs to incorporate $\varphi^{\mathsf{s}}_{ji,t_j}$ into its static task specification φ^1_i and update its current plan $\tau_{i,0}$ to satisfy this request. As mentioned earlier, the service task specification defined in (7) requires $\varphi^{\mathsf{s}}_{ji,t_j}$ to be satisfied within a finite time. The associated LTL formula is denoted by $[\varphi^{\mathsf{s}}_{ji,t_j}]$ and $\mathcal{B}_{[\varphi^{\mathsf{s}}_{ji,t_j}]} = (Q_j,\, 2^R,\, \delta_j,\, Q_{j,0},\, F_j)$ is the corresponding NBA, with similar notation to Section II-A. Recall that $\mathcal{B}_{[\varphi^1_i]} = (Q_s,\, 2^R,\, \delta_s,\, Q_{s,0},\, F_s)$ is the NBA associated with $[\varphi^1_i]$ and assume that at time t_j the product state of agent i in $\mathcal{P}'_i$ is $q'_{p,t_j} \in Q'_p$ and the associated Büchi state in $\mathcal{B}_{[\varphi^1_i]}$ is $q_{s,t_j} = q'_{p,t_j}|_{Q_s}$. Thus, we define the request-prioritized and layered intersection of $[\varphi^{\mathsf{s}}_{ji,t_j}]$ and $[\varphi^1_i]$ below:

Definition 4. The intersection of $\mathcal{B}_{[\varphi^{\mathsf{s}}_{ji,t_j}]}$ and $\mathcal{B}_{[\varphi^1_i]}$ is an NBA:

$$\mathcal{A}_{[\varphi_i]} = (Q,\, 2^R,\, \delta,\, Q_0,\, F), \tag{23}$$

where $Q = Q_j \times Q_s \times \{1,2\}$; $Q_0 = Q_{j,0} \times \{q_{s,t_j}\} \times \{1\}$; $F = F_j \times F_s \times \{2\}$; $\delta : Q \times 2^R \to 2^Q$, with $\langle \check{q}_j,\, \check{q}_s,\, \check{c}\rangle \in \delta(\langle q_j,\, q_s,\, c\rangle,\, l)$ when the following conditions hold: (i) $\langle q_j,\, q_s,\, c\rangle$, $\langle \check{q}_j,\, \check{q}_s,\, \check{c}\rangle \in Q$; (ii) $\check{q}_j \in \delta_j(q_j,\, l)$ and $\check{q}_s \in \delta_s(q_s,\, l)$; (iii) $q_j \notin F_j$ and $\check{c} = c = 1$; *or* $q_j \in F_j$, $c = 1$ and $\check{c} = 2$; *or* $\check{c} = c = 2$. ■

The aforementioned definition is different from the conventional way of computing intersections of Büchi automata [18] owing to the fact that $[\varphi^1_i]$ is a general LTL formula that should be satisfied at $t = 0$ and $[\varphi^{\mathsf{s}}_{ji,t_j}]$ is an sc-LTL formula that should be satisfied after it is received. In particular, $\mathcal{A}_{[\varphi_i]}$ has two layers and it transits from the first layer to the second only if it reaches F_j (i.e., $[\varphi^{\mathsf{s}}_{ji,t_j}]$ is satisfied); afterwards, it stays at the second layer in order to satisfy $[\varphi^1_i]$. In the following lemma, we prove the correctness of Definition 4 by showing that $\mathcal{A}_{[\varphi_i]}$ accepts all words that satisfy both $[\varphi^1_i]$ and $[\varphi^{\mathsf{s}}_{ji,t_j}]$.

Lemma 3. *If there exists an ω-word $w \in R^\omega$ such that $w \models [\varphi^{\mathsf{s}}_{ji,t_j}]$ and $w \models [\varphi^1_i]$, then $\varphi^{\mathsf{s}}_{ji,t_j}$ and φ^1_i are mutually feasible, and $\mathcal{A}_{[\varphi_i]}$ has at least one accepting run.*

Proof. Owing to the fact that $w \models [\varphi^{\mathsf{s}}_{ji,t_j}]$, at least one of the resulting runs of w in $\mathcal{B}_{[\varphi^{\mathsf{s}}_{ji,t_j}]}$ is an accepting run, denoted by $r_j = q_{j,0}q_{j,1}\cdots q_{j,K_j}(q_{j,K_j})^\omega$, where $q_{j,0}q_{j,1}\cdots q_{j,K_j}$ is a finite sequence from an initial state $q_{j,0} \in Q_{j,0}$ to an accepting state $q_{j,K_j} \in F_j$ and $(q_{j,K_j})^\omega$ is a repetitive suffix over q_{j,K_j}. Such argument holds true since all accepting states of $\mathcal{B}_{[\varphi^{\mathsf{s}}_{ji,t_j}]}$ have a self-cycle that accepts any input alphabet (see Remark 4.31 of [18]). On the other hand, given that $w \models [\varphi^1_i]$, then w results in at least one accepting run of $\mathcal{B}_{[\varphi^1_i]}$, denoted by $r_s = q_{s,0}q_{s,1}\cdots(q_{s,K_f}\, q_{s,K_f+1}q_{s,K_f+2}\cdots q_{s,K_f+K_s})^\omega$ that starts from an initial state $q_{s,0} \in Q_{s,0}$ to an accepting state $q_{s,K_f} \in F_s$. We assume $K_f > K_j$. If not so, the suffix of r_s can be extended by repeating itself until $K_f > K_j$. In this sense, we can easily verify that a resulting run of w in $\mathcal{A}_{[\varphi_i]}$ can be constructed as $r = (q_{j,0}, q_{s,0}, 1)\cdots(q_{1,K_j}, q_{s,K_j}, 1)(q_{1,K_j}, q_{s,K_j+1}, 2)(q_{1,K_j}, q_{s,K_j+2}, 2)(q_{1,K_j}, q_{s,K_j+3}, 2)\cdots((q_{1,K_j}, q_{s,K_f}, 2)(q_{1,K_j}, q_{s,K_f+1}, , 2)\cdots(q_{1,K_j}, q_{s,K_f+K_s}, , 2))^\omega$. Since $(q_{1,0}, q_{s,0}, 1) \in Q_0$ and the state within the repetitive suffix $(q_{1,K_j}, q_{s,K_f}, 2) \in F$, then by definition r is an accepting run of $\mathcal{A}_{[\varphi_i]}$ and hence $\mathcal{A}_{[\varphi_i]}$ accepts w. ■

From the proof aforementioned, it can be seen that r reaches the accepting states in F_j that satisfy $[\varphi^{\mathsf{s}}_{ji,t_j}]$ before it repeats the suffix that satisfies $[\varphi^1_i]$. Thus, the service request has a higher priority than the static specification and will be satisfied earlier. Moreover, the layered structure of $\mathcal{A}_{[\varphi_i]}$ allows us to track the satisfaction of $[\varphi^1_i]$ and $[\varphi^{\mathsf{s}}_{ji,t_j}]$ separately.

Case II: Suppose that agent i has confirmed m service requests $[\varphi^{\mathsf{s}}_{gi,t_g}]$ at time t_g from agent g, $\forall g \in \mathcal{N}^{\mathsf{s}}_i \subseteq \mathcal{N}_i$. Without loss of generality, we can re-order the neighbors in $\mathcal{N}^{\mathsf{s}}_i$ by $\{1, 2, \cdots, m\} \triangleq \mathcal{N}^{\mathsf{s}}_i$, according to the time their service request was received, e.g., $[\varphi^{\mathsf{s}}_{1i,t_1}]$ denotes the earliest and $[\varphi^{\mathsf{s}}_{mi,t_m}]$ denotes the latest received request. Furthermore, let $\mathcal{B}_{[\varphi^{\mathsf{s}}_{gi,t_g}]} = (Q_g,\, 2^R,\, \delta_g,\, Q_{g,0},\, \mathcal{F}_g)$ be the NBA associated with $[\varphi^{\mathsf{s}}_{gi,t_g}]$, $\forall g \in \mathcal{N}^{\mathsf{s}}_i$, with similar notations to Section II-A. Additionally, assume that at time t_m the corresponding product state of agent i is $q'_{p,t_g} \in Q'_p$. In that respect, its associated Büchi state in $\mathcal{B}_{[\varphi^1_i]}$ is $q_{s,t_m} = q'_{p,t_m}|_{Q_s}$ and the associated Büchi state in each $\mathcal{B}_{[\varphi^{\mathsf{s}}_{gi,t_g}]}$ is $q_{g,t_m} = q'_{p,t_m}|_{Q_g}$, $\forall g \in \mathcal{N}^{\mathsf{s}}_i$. Thus, we may define the request-prioritized intersection of $[\varphi^{\mathsf{s}}_{gi,t_g}]$, $\forall g \in \mathcal{N}^{\mathsf{s}}_i$ and $[\varphi^1_i]$ as follows:

Definition 5. The intersection of $\mathcal{B}_{[\varphi^{\mathsf{s}}_{gi,t_g}]}$, $\forall g \in \mathcal{N}^{\mathsf{s}}_i$ and $\mathcal{B}_{[\varphi^1_i]}$ is an NBA defined by:

$$\mathcal{A}_{[\varphi_i]} = (Q,\, 2^R,\, \delta,\, Q_0,\, \mathcal{F}), \tag{24}$$

where $Q = Q_1 \times Q_2 \cdots \times Q_m \times Q_s \times \{1,\, 2,\, \cdots, m{+}1\}$; $Q_0 = \{q_{1,t_m}\} \times \{q_{2,t_m}\} \cdots \times \{q_{m-1,t_m}\} \times Q_{m,0} \times \{q_{s,t_m}\} \times \{1\}$; $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \cdots \times \mathcal{F}_m \times \mathcal{F}_{s,0} \times \{m{+}1\}$; $\delta : Q \times 2^R \to 2^Q$, with $\langle \check{q}_1,\, \check{q}_2,\, \cdots, \check{q}_m,\, \check{q}_s,\, \check{c}\rangle \in \delta(\langle q_1,\, q_2,\, \cdots, q_m,\, q_s,\, c\rangle,\, l)$ when the following conditions hold: (i) $\langle q_1,\, q_2,\, \cdots, q_m,\, q_s,\, c\rangle$, $\langle \check{q}_1,\, \check{q}_2,\, \cdots, \check{q}_m,\, \check{q}_s,\, \check{c}\rangle \in Q$; (ii) $\check{q}_j \in \delta_j(q_j,\, l)$, $\forall j \in \mathcal{N}^{\mathsf{s}}_i$; and $\check{q}_s \in \delta_s(q_s,\, l)$; (iii) $q_c \notin \mathcal{F}_c$ and $\check{c} = c$; *or* $q_c \in \mathcal{F}_c$ and $\check{c} = c + 1$; *or* $\check{c} = c = m + 1$. ■

Notice that $\mathcal{A}_{[\varphi_i]}$ has $m{+}1$ layers and transits to the $(c{+}1)$-th layer only if the set of accepting states F_c is reached for $c = 1, 2, \cdots, m$ (i.e., $[\varphi^{\mathsf{s}}_{ci,t_c}]$ is satisfied), and then stays at the $(m{+}1)$-th layer in order to satisfy $\mathcal{B}_{[\varphi^1_i]}$. The definition of Q_0 ensures that the past progress of serving $[\varphi^1_i]$ and $[\varphi^{\mathsf{s}}_{gi,t_g}]$ for $g \in \mathcal{N}^{\mathsf{s}}_i$ is preserved, while the definition of δ allows us to keep track of such progress separately. Clearly, $\mathcal{A}_{[\varphi_i]}$ has $|Q_1|\cdot|Q_2|\cdots|Q_m|\cdot|Q_s|\cdot(m{+}1)$ states and maximally $|\delta_1|\cdot|\delta_2|\cdots|\delta_m|\cdot|\delta_s|\cdot(m{+}1)$ transitions. Finally, the correctness of the aforementioned definition relies on the following lemma.

Lemma 4. *If there exists an ω-word $w \in R^\omega$ such that $w \models [\varphi^{\mathrm{s}}_{gi,t_g}]$, $\forall g \in \mathcal{N}_i^s$ and $w \models [\varphi_i^1]$, then the received service tasks $\varphi^{\mathrm{s}}_{gi,t_g}$, $\forall g \in \mathcal{N}_i^s$ and φ_i^1 are mutually feasible, and $\mathcal{A}_{[\varphi_i]}$ has at least one accepting run.*

Proof. Following the same line of proof with Lemma 3, we may construct an accepting run of $\mathcal{A}_{[\varphi_i]}$ by employing the resulting runs of w over $[\varphi_i^1]$ and $[\varphi^{\mathrm{s}}_{gi,t_g}]$, $\forall g \in \mathcal{N}_i^{\mathrm{s}}$. Similarly to the one constructed in Lemma 3, such accepting run reaches an accepting state of $\mathcal{B}_{[\varphi^{\mathrm{s}}_{2i,t_2}]}$ first, transits to the second layer, reaches an accepting state of $\mathcal{B}_{[\varphi^{\mathrm{s}}_{2i,t_2}]}$, transits to the third layer and so on. This process is repeated until it reaches the $(m+1)_{th}$ layer and stays there. Afterwards its suffix is repeated infinitely often to satisfy $[\varphi_i^1]$. Thus, $\mathcal{A}_{[\varphi_i]}$ accepts the common words of $[\varphi_i^1]$ and all the service requests $[\varphi^{\mathrm{s}}_{gi,t_g}]$, $\forall g \in \mathcal{N}_i^{\mathrm{s}}$. Finally notice that $\mathcal{A}_{[\varphi_i]}$ is updated recursively by Definition 5, whenever agent i confirms new service requests. ■

Case III: Assume that agent i has confirmed a service request $\varphi^{\mathrm{s}}_{li,t_l^1}$ from its neighbor $l \in \mathcal{N}_i$ at time $t_l^1 > 0$ and accomplished it at $t_l^2 > t_l^1$. Afterwards, at time $t_l^3 > t_l^2$, agent i confirms a new service request $\varphi^{\mathrm{s}}_{li,t_l^3}$ from the same neighbor l. In the sequel, we discuss how we should adjust $\mathcal{A}_{[\varphi_i]}$ in order to remove the old request $\varphi^{\mathrm{s}}_{li,t_l^1}$ and incorporate the new request $\varphi^{\mathrm{s}}_{li,t_l^3}$. Let us denote by $\mathcal{N}_i^{\mathrm{s}}$ the set of neighbors, whose service requests agent i has to satisfy as well as by $\mathcal{A}_{[\varphi_i]}$ the associated intersection NBA. Since the service request from agent l has changed from $\varphi^{\mathrm{s}}_{li,t_l^1}$ to $\varphi^{\mathrm{s}}_{li,t_l^3}$, $\mathcal{A}_{[\varphi_i]}$ needs to be updated as follows. First, the service requests need to be re-organized according to the time they were received. Hence, the service request of agent l should be assigned the index $m = |\mathcal{N}_i^{\mathrm{s}}|$ as it was the latest received one. Subsequently, given the product state q'_{p,t_l^3} of agent i at time t_l^3, the Büchi state associated with each $[\varphi^{\mathrm{s}}_{gi,t_g}]$ is derived by $q'_{p,t_l^3}|_{Q'_g}$, $\forall g \in \mathcal{N}_i^{\mathrm{s}}$ and $g \neq l$. Thus, $\mathcal{A}_{[\varphi_i]}$ can be recomputed by Definition 5, with the service request of agent l being moved to the m-th layer. The rest of the details are similar to **Case II**.

Given the updated intersection automaton $\mathcal{A}_{[\varphi_i]}$ from **Cases I-III**, the corresponding product automaton $\mathcal{P}_i$ should be updated following Definition 3. Thus, we search for an accepting run of the updated $\mathcal{P}_i$ that minimizes the cost function mentioned in Section IV-D1, of which the computational complexity is derived analogously with the updated $\mathcal{P}_i$. This accepting run always exists due to the assumption that the local task and the received service tasks are mutually feasible for each agent $i \in \mathcal{N}$. Subsequently, this accepting run can be projected onto Π, thus yielding the updated discrete plan of agent i. Finally, notice that the local plan should be adapted whenever $\mathcal{A}_{[\varphi_i]}$ is updated in any of the cases above.

In the sequel, we prove the correctness of the proposed discrete plan adaptation approach. In this respect, assume that agent i receives a service request $\varphi^{\mathrm{s}}_{gi,t_g}$ from its neighbor $g \in \mathcal{N}_i^{\mathrm{s}}$ at time $t_g > 0$. After that, at time $t > t_g > 0$, agent i has crossed the sequence of goal regions $\pi_{i,1}\pi_{i,2}\cdots\pi_{i,K}$ during $[t_g,\, t]$, which is uniquely determined by its discrete plan $\tau_i([t_g,\, t])$, where $\pi_{i,k} \in \Pi$, $\forall i = 1, 2, \cdots, K$. In this way the corresponding trace is defined by $\texttt{trace}_i([t_g,\, t]) = L_i(\pi_{i,0})L_i(\pi_{i,1})\cdots L_i(\pi_{i,K})$. Furthermore, let $q'_{i,t} \in Q'$ be the corresponding product state in $\mathcal{P}_i$ at time t.

Theorem 2. *Given that $\varphi^{\mathrm{s}}_{gi,t_g}$, $\forall g \in \mathcal{N}_i^s$ and φ_i^1 are mutually feasible, there exists a finite time $t > t_g$ such that $\mathbf{p}_i([t_g,\, t])$ satisfies the RTL formula $\varphi^{\mathrm{s}}_{gi,t_g}$, for each agent $g \in \mathcal{N}_i^s$.*

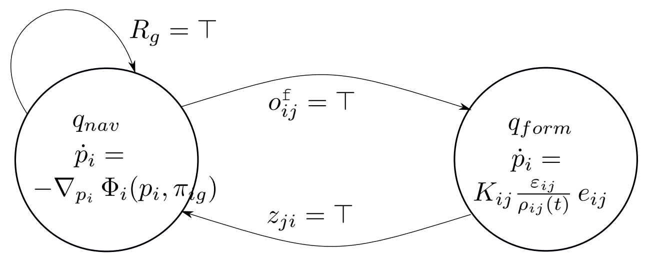
Proof. Lemma 4 ensures that the intersection automaton $\mathcal{A}_{[\varphi_i]}$ accepts the satisfying words of $[\varphi^{\mathrm{s}}_{gi,t_g}]$. Irrespectively of the layer that $[\varphi^{\mathrm{s}}_{gi,t_g}]$ is in $\mathcal{A}_{[\varphi_i]}$, since all service requests are co-safe, there exists a finite time $t > t_g$ when $q'_{i,t}|_Q$ reaches the accepting state F_g. Furthermore, from the definition of $\mathcal{A}_{[\varphi_i]}$, we know that if $q_g \in F_g$ for any $g \in \mathcal{N}_i^{\mathrm{s}}$, then the past trace during $[t_g,\, t]$ (i.e., $\texttt{trace}_i([t_g,\, t])$) results in an accepting run of $\mathcal{B}_{[\varphi^{\mathrm{s}}_{gi,t_g}]}$ from the initial state to the accepting state q_g. Thus, $\texttt{trace}_i([t_g,\, t])$ satisfies $[\varphi^{\mathrm{s}}_{gi,t_g}]$. Finally, it can be easily deduced, following the proof of Theorem 1, that the resulting trajectory $\mathbf{p}_i([t_g,\, t])$ satisfies the RTL formula $\varphi^{\mathrm{s}}_{gi,t_g}$. ■

The proof above provides an efficient way for agent i to monitor the satisfaction of each request $\varphi^{\mathrm{s}}_{gi,t_g}$, $\forall g \in \mathcal{N}_i^{\mathrm{s}}$. Notice that given the projection $q_{i,t} = q'_{i,t}|_Q$, if $q_{i,t}|_{Q_g} \in F_g$ for any $g \in \mathcal{N}_i^{\mathrm{s}}$, then $[\varphi^{\mathrm{s}}_{gi,t_g}]$ is satisfied by $\texttt{trace}_i([t_g,\, t])$. This plays an important role for the event monitoring scheme and the communication protocol in Sections IV-B and IV-C.

Finally, while serving its service requests, agent i may send formation requests to its neighbors in $\mathcal{N}_i$ that may be confirmed. In this case, assume that agent $j \in \mathcal{N}_i$ confirms a formation request at time t_j. The formation between agents i and j should be kept until the associated formation task $\varphi^{\mathrm{f}}_{ij,t_j}$ is fulfilled by agent i. Namely, agent i first needs to incorporate $\varphi^{\mathrm{f}}_{ij,t_j}$ into φ_i, employing the same procedure with the case of new service requests described previously. Specifically, the NBA associated with $[\varphi^{\mathrm{f}}_{ij,t_j}]$ is computed as $\mathcal{B}_{[\varphi^{\mathrm{f}}_{ij,t_j}]}$ and the intersection automaton $\mathcal{A}_{[\varphi_i]}$ is recomputed by adding one extra layer in $\mathcal{B}_{[\varphi^{\mathrm{f}}_{ij,t_j}]}$ as in Definition 5. Thus, similar to Theorem 2, we assume that the formation task $\varphi^{\mathrm{f}}_{ij,t_j}$ is mutually feasible with the local task and the received service tasks. Then, the resulting intersection automaton always have an accepting run. Similarly $\varphi^{\mathrm{f}}_{ij,t_j}$ will be satisfied at a finite time, after which, agent i sends a release message to agent j and then both intersection automata are updated.

Theorem 3. *Given that φ^{f}_{ij,t_j} is mutually feasible with the local task φ_i^1 and the service tasks φ^{s}_{gi,t_g}, $\forall g \in \mathcal{N}_i^s$ for agent $i \in \mathcal{N}$, there exists a finite time $t^{\mathrm{f}} > t_j$ such that: (i) the trajectory $\mathbf{p}_i([t_j,\, t^{\mathrm{f}}])$ of agent i satisfies the formation task φ^{f}_{ij,t_j} and (ii) agent j satisfies the controllable proposition $h_{ij}(t) = \top$, $\forall t \in [t_j,\, t^{f})$.*

Proof. The first part can be proved similarly to Theorem 2. In particular, the newly-confirmed formation task is incorporated into $\mathcal{A}_{[\varphi_i]}$. Since it is co-safe and mutually-feasible with the local task and the received service tasks, there exists a finite time $t^{\mathrm{f}} > t_j$ such that $q'_{i,t}|_Q$ reaches the accepting states



$R_g = \top$
q_{nav}
$\dot{p}_i = -\nabla_{p_i}\Phi_i(p_i,\pi_{ig})$
$o^{\mathrm{f}}_{ij} = \top$
$z_{ji} = \top$
q_{form}
$\dot{p}_i = K_{ij}\frac{\varepsilon_{ij}}{\rho_{ij}(t)}e_{ij}$

Figure 2. The hybrid controller module. The arrows indicate the discrete transitions labeled by the associated guards for agent $i \in \mathcal{N}$.

of $\mathcal{B}_{\varphi^{\mathrm{f}}_{ij,t_j}}$ and thus $\mathtt{trace}_i([t_j,\, t^{\mathrm{f}}])$ satisfies $[\varphi^{\mathrm{f}}_{ij,t_j}]$. Subsequently, it can be shown as in Theorem 1 that the resulting trajectory $\mathbf{p}_i([t_g,\, t^{\mathrm{f}}])$ satisfies the RTL formula $\varphi^{\mathrm{f}}_{ij,t_j}$. At the same time, a confirmation message is sent from agent i to j. Secondly, Lemma 2 indicates that the formation control law (13) guarantees that agent j will converge to the desired relative formation with prescribed performance, i.e., the controllable proposition $h_{ij}(t) = \top$, $\forall t \in [t_j,\, t^{\mathrm{f}}]$, until a release signal is received at time t and $z_{ij}(t) = \top$. ■

E. Overall Structure

Now we formalize the overall hybrid control architecture, based on the aforementioned analysis. As depicted in Figure 1, the architecture is organized in four interconnected modules that run concurrently in real time. It includes the communication module from Section IV-C, the discrete planner module from Section IV-D1, the event monitoring module from Section IV-B and the hybrid control module.

Hybrid control module. Agent i is in charge of switching between the navigation and formation control modes. As shown in Figure. 2, the hybrid control automaton is defined as a tuple $H_{cont} \triangleq (Q,\, P,\, \mathrm{Init},\, f,\, D,\, O,\, E,\, G)$, where $Q = \{q_{nav},\, q_{form}\}$ is the set of discrete states; $P \subseteq \pi_{i0}$ is the continuous state; $\mathrm{Init} = q_{nav} \times p_i(0)$ is the initial state; $f(q_{nav},\, p_i) = -\nabla_{q_i}\Phi_i(p_i, \pi_{ig})$ is the continuous dynamics (10) within the discrete state q_{nav}, where π_{ig} is the goal region imposed by the discrete planner; $f(q_{form},\, p_i) = K_{ji}\,\varepsilon_{ji}\, e_{ji}/\rho_{ji}(t)$ is the continuous dynamics (13) within the discrete state q_{form}, where c_{ji}, $\rho_{ji}(t)$ are the formation request conveyed via the communication module; $D(q_{nav}) = D(q_{form}) \subseteq \pi_{i0}$ are the domains of the continuous state; $O = \{o_1,\, o_2,\, o_3\}$ is the set of external events, where o_1 is "$R_g = \top$", o_2 is "$o^{\mathrm{f}}_{ij} = \top$", and o_3 is "$z_{ji} = \top$"; $E = \{(q_{nav},\, q_{nav}),\, (q_{nav},\, q_{form}),\, (q_{form},\, q_{nav})\}$ involves the allowed discrete transition edges; $G(q_{nav},\, q_{nav}) = o_1$, $G(q_{nav},\, q_{form}) = o_2$, $G(q_{form},\, q_{nav}) = o_3$ indicate the guards that should hold over each discrete transition.

Theorem 4. *Given the hybrid control architecture above and that all static tasks and service/formation requests are mutually feasible, then each local task φ_i is satisfied by the trajectory $p_i(t)$ of agent i as $t \to \infty$, $\forall i \in \mathcal{N}$.*

Proof. The local task φ_i consists of three parts: φ^{l}_i, φ^{s}_i and φ^{f}_i. Firstly, Theorem 1 guarantees that if no service or formation requests are confirmed by agent i, then its trajectory will satisfy the local task φ^{l}_i. Subsequently, since φ^{l}_i, φ^{s}_i and φ^{f}_i are assumed to be mutually feasible, we show that they are all fulfilled when agent i exchanges requests with its neighbors. In case agent i confirms a service request $\varphi^{\mathrm{s}}_{gi,\,t_g}$ from agent g, then Theorem 2 ensures that there exists a finite time $t^s_i > t_g$ such that the trajectory $\mathbf{p}_i([t_g,\, t^s_i])$ satisfies $\varphi^{\mathrm{s}}_{gi,\,t_g}$. Moreover, the same holds for all service requests confirmed by agent i, which implies that φ^{s}_i is fulfilled. Alternatively, if a formation request $\varphi^{\mathrm{f}}_{gi,\,t_g}$ is confirmed, then Theorem 3 ensures that there exists a finite time $t^f_i > t_g$ such that the trajectory $\mathbf{p}_g([t_g,\, t^f_i])$ of agent g fulfills the formation task $\varphi^{\mathrm{f}}_{gi,t_g}$ in finite time $t > t_g$ and meanwhile the formation between agents i and g satisfies the desired performance specifications during $[t_g,\, t^f_i]$, i.e., the controllable propositions $h_{ij}(t) = \top$, $\forall t \in [t_g,\, t^f_i]$ and $\forall j \in \mathcal{N}^{\mathrm{f}}_i$. Furthermore, since the same holds for all formation requests confirmed by agent i, it implies that φ^{f}_i is fulfilled. As a consequence, all three parts of φ_i are fulfilled and thus φ_i is satisfied for each agent $i \in \mathcal{N}$. ■

Remark 2. Notice that this hybrid controller will not exhibit *Zeno* behavior since agent i will stay within the region for at least the dwell time δ_d defined in Section IV-B and a formation is requested only after the current formation is finished. ■

V. Case Study

In this section, we present a simulated paradigm of four autonomous robots with heterogeneous capabilities. The proposed algorithms were implemented in Python 2.7 and all simulations were carried out on a desktop computer (3.06 GHz Duo CPU with 8GB of RAM).

A. Workspace and Task Description

The area of size $4m \times 4m$ is shown in Figure 3, within which there are seven cyclic regions $\pi_0, \pi_1, \cdots, \pi_6$ of interest

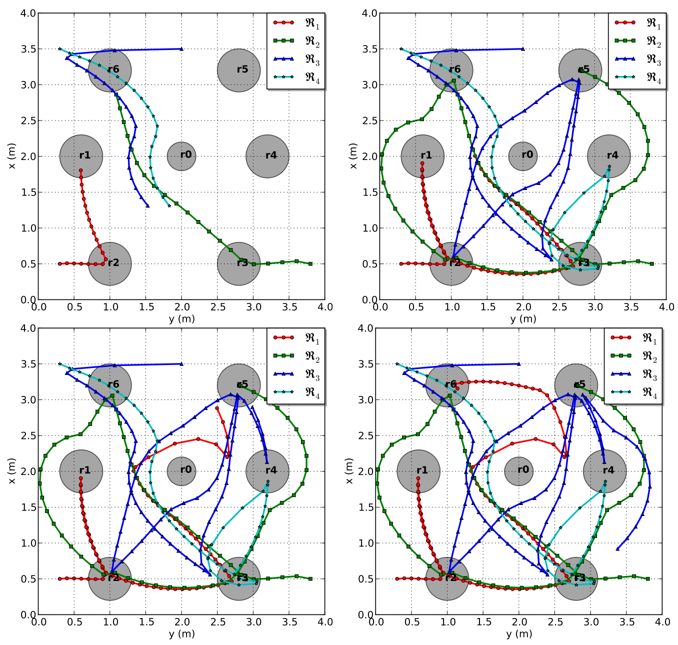


Figure 3. Snapshots of the agent's trajectories at $4.2s$, $10.5s$, $17s$ and $20s$. Descriptions of these events can be found in Section V-B.

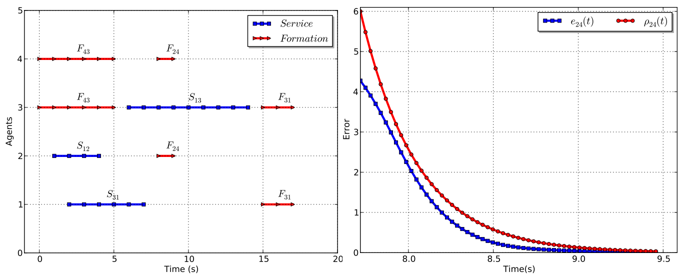


Figure 4. Left: the service/formation tasks each agent is engaged. The label S_{ij} indicates that agent j is carrying out a service request from agent i (similarly, F_{ij} for formation request). Right: evolution of the formation errors and the performance function, for the formation task $\varphi_{24}^{\mathsf{f}}$ during $[7.7s,\ 9.4s]$.

with various radius. They are labeled by the local propositions $R_0, R_1, \cdots, R_6$. Moreover, the workspace is bounded by the circle $\mathcal{B}([2,2],\ 2.35)$. Additionally, let us denote by $\mathfrak{R}_1$, $\mathfrak{R}_2$, $\mathfrak{R}_3$, $\mathfrak{R}_4$ the team of four agents that satisfy the single integrator dynamics (1). Their neighboring sets are defined as follows: $\mathcal{N}_1 = \{\mathfrak{R}_2,\ \mathfrak{R}_3\}$, $\mathcal{N}_2 = \{\mathfrak{R}_1,\ \mathfrak{R}_4\}$, $\mathcal{N}_3 = \{\mathfrak{R}_1,\ \mathfrak{R}_4\}$, $\mathcal{N}_4 = \{\mathfrak{R}_2,\ \mathfrak{R}_3\}$, representing a ring graph. The agents start from $(0.3,\ 0.5)$, $(3.8,\ 0.5)$, $(2.0,\ 3.5)$ and $(0.3,\ 3.5)$, respectively and the communication radius is uniformly set to $1m$.

For agent $\mathfrak{R}_1$, the local task $\varphi_1^{\mathsf{l}} = \big(\Diamond(R_2 \wedge \Diamond R_5)\big) \wedge \big(\Box\Diamond R_3 \wedge \Box\Diamond R_6\big)$ requires that it first visits region π_2, then π_5 and surveils over regions π_3 and π_6. Its predefined service requests concern $\mathfrak{R}_2$ (i.e., $\varphi_{12}^{\mathsf{s}} = \Diamond R_6$) and $\mathfrak{R}_3$ (i.e., $\varphi_{13}^{\mathsf{s}} = \Diamond(R_2 \wedge \Diamond R_5)$), while there are no formation requests to either $\mathfrak{R}_2$ or $\mathfrak{R}_3$. Agent $\mathfrak{R}_2$ has the local task $\varphi_2^{\mathsf{l}} = \Box\Diamond(R_2 \wedge \Diamond R_5) \wedge \Box\neg R_0$ to surveil over regions π_2 and then π_5, and avoid π_0 all the time. Moreover, there are no service requests to $\mathfrak{R}_1$ or $\mathfrak{R}_4$, while the formation request to $\mathfrak{R}_4$ is described by $c_{24} = (-0.5,\ 0)$, $\rho_{24,0} = 6$, $\rho_{24,\infty} = 0.0001$ and $l_{24} = 3$, under the formation task $\varphi_{24}^{\mathsf{f}} = \Diamond R_2$. Agent $\mathfrak{R}_3$ has the local task $\varphi_3^{\mathsf{l}} = (\Diamond R_3) \wedge (\Diamond R_4) \wedge (\Diamond\Box R_5) \wedge (\Box\neg R_0)$, i.e., it should visit regions π_3, π_4 and π_5 in any order, and avoid π_0 all the time. The predefined service request to $\mathfrak{R}_1$ is given by $\varphi_{31}^{\mathsf{s}} = \Diamond(R_1 \wedge \Diamond R_2)$ and the formation request to $\mathfrak{R}_1$ is given by $c_{31} = (0.5,\ 0)$, $\rho_{31,0} = 7$, $\rho_{31,\infty} = 0.0001$ and $l_{31} = 4$, under the formation task $\varphi_{31}^{\mathsf{f}} = \Diamond R_5$. There are no service or formation requests to $\mathfrak{R}_4$. Finally, agent $\mathfrak{R}_4$ has the local task $\varphi_4^{\mathsf{l}} = \Box\Diamond R_3 \wedge \Box\Diamond R_4$ to surveil regions π_3 and π_4. There are no service requests to $\mathfrak{R}_2$ or $\mathfrak{R}_3$, while the formation request to $\mathfrak{R}_3$ is defined as $c_{43} = (0.3,\ 0)$, $\rho_{43,0} = 7$, $\rho_{43,\infty} = 0.0001$ and $l_{43} = 10$ under the formation task $\varphi_{43}^{\mathsf{f}} = \Diamond R_3$.

B. Simulation Results

The system was simulated for $20s$ when the agents are at $(1.1, 3.2)$, $(2.7, 3.3)$, $(3.4, 0.9)$ and $(2.7, 0.5)$, respectively. Notice that all service and formation requests which are defined earlier were exchanged and accomplished at $17.1s$. An accompanying video can be found in [29].

More specifically, at $t = 0$, the initial synthesis of the discrete plan is done locally by each agent. The initial plan of $\mathfrak{R}_1$ is given by $\tau_1 = \pi_2\pi_5\pi_3\pi_6(\pi_0\pi_3)^\omega$, the initial plan of $\mathfrak{R}_2$ is given by $\tau_2 = \pi_3(\pi_2\pi_5)^\omega$, the initial plan of $\mathfrak{R}_3$ is given by $\tau_3 = \pi_5\pi_4\pi_3(\pi_5)^\omega$ and the initial plan of $\mathfrak{R}_4$ is given by $\tau_4 = \pi_6(\pi_4\pi_3)^\omega$. It can be easily verified that all satisfy the respective local tasks. After the system initialized, the predefined formation and service tasks were performed by the agents as shown in the left plot of Figure 4. Snapshots of the agents' trajectories at key time instants are shown in Figure 3. In particular, at $t = 0.1s$, $\mathfrak{R}_3$ receives $\mathfrak{R}_4$'s formation request and executes it until $t = 5.5s$, when $\mathfrak{R}_4$ finishes the formation task $\Diamond R_3$ by reaching π_3. The evolution of the formation error along with the corresponding performance function is shown in the right plot of Figure 4. At $0.6s$, $\mathfrak{R}_2$ confirms the service request $\Diamond R_6$ from $\mathfrak{R}_1$ and changes its local plan to visit π_6 first, which is done at $4.2s$. At the same time, $\mathfrak{R}_1$ confirms service request $\Diamond(R_1 \wedge \Diamond R_2)$ from $\mathfrak{R}_3$ and changes its local plan to visit π_1 and then π_2, which is done at $7.1s$. During $[7.7s,\ 9.4s]$, $\mathfrak{R}_4$ carries out a relative formation task with $\mathfrak{R}_2$ until $\mathfrak{R}_2$ finishes the formation task $\Diamond R_2$ by reaching π_2. Then $\mathfrak{R}_3$ confirms the service request $\Diamond(R_2 \wedge \Diamond R_5)$ from $\mathfrak{R}_1$ at $5.5s$. This service is accomplished at $14.8s$ by $\mathfrak{R}_3$'s detour to π_2 first and π_5 afterwards. Finally, $\mathfrak{R}_1$ establishes a relative formation with $\mathfrak{R}_3$ at $14.8s$ until agent $\mathfrak{R}_3$ finishes the formation task $\Diamond R_5$ by reaching π_5 at $17.1s$. By then, all predefined contingent tasks were fulfilled and no further contingent service or formation tasks will be exchanged. Subsequently, each agent continues executing its local plan to fulfill its local task. Note that since some agents have an infinite plan, we simulate the system until $20s$. Finally, regarding the formation requests $\varphi_{43}^{\mathsf{f}}$ during $[0s,\ 5.5s]$, $\varphi_{24}^{\mathsf{f}}$ during $[7.7s,\ 9.4s]$ and $\varphi_{31}^{\mathsf{f}}$ during $[14.8s,\ 17.1s]$, notice that the evolution of the formation errors, as depicted in Figure 4, meets the predefined performance specifications until the associated formation task is fulfilled.

VI. Conclusions

We presented a hybrid control strategy for multi-agent systems with contingent temporal tasks and prescribed formation constraints that fulfills all local high-level temporal tasks and contingent service/formation requests. Future research efforts will address physically interacting cooperative tasks and more complex agent dynamics with non-zero agent volume.

References

[1] S. G. Loizou, K. J. Kyriakopoulos. Closed loop navigation for multiple non-holonomic vehicles. *IEEE International Conference on Robotics and Automation*, 3: 4240-4245, 2003.

[2] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. *American Control Conference (ACC)*, 1859–1864, 2005.

[3] M. Ji and M. B. Egerstedt. Distributed coordination control of multi-agent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4): 693–703, 2007.

[4] G. E. Fainekos, S. G. Loizou, and G. J. Pappas, Translating temporal logic to controller specifications, *IEEE Conference on Decision and Control(CDC)*, 899–904, 2006.

[5] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics*, 28(1):158–171, 2012.

[6] S. Karaman and E. Frazzoli. Vehicle routing with temporal logic specifications: Applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control*, 21:1372–1395, 2011.

[7] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on ltl specifications. *IEEE Conference on Decision and Control(CDC)*, 153–158, 2004.

[8] S. G. Loizou and K. J. Kyriakopoulos. Automated planning of motion tasks for multi-robot systems. *IEEE Conference on Decision and Control (CDC)*, 78–83, 2005.

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

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

[11] J. Tumova and D. V. Dimarogonas. A receding horizon approach to multi-agent planning from LTL specifications. *American Control Conference (ACC)*, 1775–1780, 2014.

[12] H. Kress-Gazit, G. E. Fainekos, G. J. Pappas. Where's Waldo? Sensor-based temporal logic motion planning. *IEEE International Conference on Robotics and Automation*, 3116-3121, 2007.

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

[14] P. Ogren, M. Egerstedt, and X. Hu. A control Lyapunov function approach to multi-agent coordination. *IEEE Conference on Decision and Control(CDC)*, 1150–1155, 2001.

[15] C. P. Bechlioulis and G. A. Rovithakis. Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance. *IEEE Transactions on Automatic Control*, 53(9): 2090-2099, 2008.

[16] C. P. Bechlioulis, K. J. Kyriakopoulos. Robust model-free formation control with prescribed performance and connectivity maintenance for nonlinear multi-agent systems. *IEEE Conference on Decision and Control (CDC)*, 2014.

[17] E. D. Sontag. *Mathematical Control Theory*. Springer, 1998.

[18] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[19] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, G. J. Pappas. Symbolic planning and control of robot motion. *IEEE Robotics and Automation Magazine*, 14: 61-71, 2007.

[20] P. Gastin, D. Oddoux. Fast LTL to Büchi automaton translation. *International Conference on Computer Aided Verification (CAV'01)*, 2001.

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

[22] M. Reynolds. Continuous temporal models. *Australian joint conference on artificial intelligence*, 414425. Springer, 2001.

[23] G. E. Fainekos, A. Girard, H. Kress-Gazit, G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2): 343–352, 2009.

[24] D. E. Koditschek, E. Rimon. Robot navigation functions on manifolds with boundary. *Advances Appl. Math.*, 11:412-442, 1990.

[25] A. Bhatia, L. E. Kavraki, M. Y. Vardi. Sampling-based motion planning with temporal goals. *IEEE International Conference on Robotics and Automation*, 2010.

[26] M. Guo and D. V. Dimarogonas. Reconfiguration in motion planning of single- and multi-agent systems under infeasible local LTL specifications. *IEEE Conference on Decision and Control(CDC)*, 2758-2763, 2013.

[27] S. G. Loizou, A. Jadbabaie. Density Functions for Navigation Function Based Systems. *IEEE Conference on Decision and Control*, 2006.

[28] K. H. Johansson, M. Egerstedt, J. Lygeros, S. Sastry. On the regularization of Zeno hybrid automata. *Systems and Control Letters*, 38(3), 141-150, 1999.

[29] Simulation Video. `https://vimeo.com/138463775`

Meng Guo received his M.Sc. degree in System, Control and Robotics in 2011 and Ph.D. degree in Electrical Engineering in 2016, both from KTH Royal Institute of Technology, Sweden. Currently he is a postdoc associate at the Department of Mechanical Engineering and Materials Science, Duke University, USA. His main research interest includes distributed motion and task planning of multi-agent systems and formal control synthesis.

Charalampos P. Bechlioulis was born in Arta, Greece, in 1983. He is currently a postdoctoral researcher in the Control Systems Laboratory at the School of Mechanical Engineering of the National Technical University of Athens. He received a diploma in electrical and computer engineering in 2006 (first in his class), a bachelor of science in mathematics in 2011 (second in his class) and a Ph.D. in electrical and computer engineering in 2011, all from the Aristotle University of Thessaloniki, Thessaloniki, Greece. His research interests include nonlinear control with prescribed performance, system identification, control of robotic vehicles, multi-agent systems and object grasping. He has authored more than 55 papers in scientific journals, conference proceedings and book chapters.

Kostas J. Kyriakopoulos was born in Athens, Greece (1962). He received the Mechanical Eng. Dipl., with Honours, from the National Technical University of Athens (NTUA), Greece in 1985 and MS & Ph.D. degrees in Computer & Systems Eng. at Rensselaer Polytechnic Institute (RPI), Troy, NY, in 1987 and 1991, respectively. Between 1988-91 he did research at the NASA Cntr for Intelligent Robotic Systems for Space Exploration while between 1991-93 he was a Research Assistant Professor at the Electrical, Computer and Systems Eng. Dept. of RPI and the New York State Cntr for Advanced Technology in Automation & Robotics. Since December 1994 he has been with Mechanical Eng. Dept at NTUA where he serves as a Professor and Director of the Graduate Program on Automation Systems. His research interests are in the areas of: (i) Nonlinear Control applications to Sensor Based Motion Planning & Control of Robotic multi-agent systems: Manipulators & Vehicles (Mobile, Marine, Aerial) and (ii) Neuro-Robotics.

Dimos V. Dimarogonas was born in Athens, Greece, in 1978. He received the Diploma in Electrical and Computer Engineering in 2001 and the Ph.D. in Mechanical Engineering in 2007, both from the National Technical University of Athens (NTUA), Greece. Between May 2007 and February 2009, he was a Postdoctoral Researcher at the Automatic Control Laboratory, School of Electrical Engineering, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. Between February 2009 and March 2010, he was a Postdoctoral Associate at the Laboratory for Information and Decision Systems (LIDS) at the Massachusetts Institute of Technology (MIT), Boston, MA, USA. He is currently an Associate Professor at the Automatic Control Laboratory, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. His current research interests include Multi-Agent Systems, Hybrid Systems and Control, Robot Navigation and Networked Control. He was awarded a Docent in Automatic Control from KTH in 2012. He serves in the Editorial Board of Automatica, the IEEE Transactions on Automation Science and Engineering and the IET Control Theory and Applications and is a member of IEEE and the Technical Chamber of Greece. He received an ERC Starting Grant from the European Commission for the proposal BUCOPHSYS in 2014 and was awarded a Wallenberg Academy Fellow grant in 2015.