Temporal Task Planning in Wirelessly Connected Environments with Unknown Channel Quality

Meng Guo and Michael M. Zavlanos

Abstract— **We consider a mobile robot tasked with gathering data in an environment of interest and transmitting these data to a data center. The task is specified as a high-level Linear Temporal Logic (LTL) formula that captures the data to be gathered at various regions in the workspace. The robot has a limited buffer to store the data, which needs to be transmitted to the data center before the buffer overflows. Communication between the robot and the data center is through a dedicated wireless network to which the robot can upload data with rates that are uncertain and unknown. In this case, most existing methods based on dynamic programming can not be applied due to the lack of an accurate model. To address this challenge, we propose here an actor-critic reinforcement learning algorithm where the task execution, workspace exploration, and parameterized-policy learning are all performed online and simultaneously. The derived motion and communication control strategy satisfies the buffer constraints and is reactive to the uncertainty in the wireless transmission rate. The overall complexity and performance of our method is compared in simulation to static solutions that search for constrained shortest paths, and to existing learning algorithms that rely on the construction of the product automaton.**

I. Introduction

Autonomous robots have been proposed for a variety of applications ranging from transportation, and healthcare, to agriculture. In many of these applications the robots are required to collect information about their environment or the status of their tasks and send it to a user that can respond to possible situations. Communication between robots and users can take place through dedicated wireless networks to which the robots are able to connect. Then, the goal is to transmit information to the users in a timely and reliable way. Recognizing that the wireless channel is uncertain and that it is almost impossible to maintain reliable communications for all time, in this paper we propose a new definition of reliability that depends on the ability of the robot to transfer a prescribed amount of data to the user over a desired interval of time, while satisfying buffer size constraints. The tasks assigned to the robot are specified as Linear Temporal Logic (LTL) formulas [1]. There has been a large amount of work focusing on motion planning under high-level temporal tasks for a single robot [2], [3]. This framework has also been extended to multi-robot systems; see [4], [5].

Communication among robots has typically relied on proximity graphs in which case the communication problem becomes equivalent to preservation of topological connectivity [6]–[8]. Relevant is also recent work on intermittent communication control that allows the robots to temporarily disconnect from the network to accomplish their tasks and then meet periodically to communicate [9], [10]. While simple, the above graph-based models do not capture tangible and reliable communication. Instead, here we employ more realistic communication models that take into account path loss, shadowing, and multi-path fading as well as optimal routing decisions for desired information rates, as discussed in [11]–[14]. However, most of the aforementioned work does not address complex high-level tasks specified as temporal logic formulas.

Due to uncertainty in the wireless channel quality, the rate at which the robot can upload data to the network at different locations in the workspace becomes a random variable with a distribution that is often unknown. Planning in such uncertain systems is usually modeled using Markov Decision Processes (MDP). If the MDP model is fully-known (particularly the states and the transition probabilities between them), then dynamic programming techniques can be applied to construct an optimal control policy [15]. There has also been work that controls MDPs to satisfy LTL specifications; e.g., see [16]–[18]. However, if the transition probabilities are unknown, as in this work, the above methods can not be directly applied; instead, learning methods are needed to learn the optimal policy and the MDP model at the same time. Here, we rely on a reinforcement learning (RL) algorithm that is a combination of the dynamic learning framework (see Chapter 9 of [19]) and the least-squares temporal difference (LSTD) method of the actor-critic type proposed in [20]–[22].

The main contribution of this work lies in the co-design of a robot motion and communication controller for high-level data-gathering tasks in realistic wireless communication network. The proposed algorithm does not assume perfect knowledge of the wireless network model and the resulting control strategy is reactive to the network quality during run time. The computational efficiency and performance of the proposed algorithm is validated via extensive numerical simulations. To the best of our knowledge, this is the first work that addresses high-level data-gathering tasks under both buffer constraints and realistic wireless communication networks. At the same time, this is the first approach that combines reinforcement learning with motion and communication control synthesis from high-level LTL specifications when the system model is unknown.

The most related work is [22], which applies the same LSTD actor-critic algorithm to maximize the probability that a MDP satisfies a given LTL specification. This work differs from [22] in three ways: (i) [22] assumes that the MDP

The authors are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA. Emails: `meng.guo, michael.zavlanos@duke.edu`. This work is supported in part by the NSF awards CNS #1261828 and CNS #1302284.

model is known; (ii) [22] performs the learning over the product automaton between the MDP and the Deterministic Rabin Automaton (DRA) associated with the task; (iii) the suffix behavior of the system is not considered in [22]. Since in our case the MDP model is unknown, the product automaton cannot be constructed. Instead, we first synthesize a discrete plan in the reduced space of robot positions that is assumed known, and then use this discrete plan to guide the learning process in the joint space of robot positions and communication variables that are unknown. Another recent work [23] proposes a probably approximately correct MDP (PAC-MDP) methodology that estimates the MDP model via a limited number of samples, which guarantees a threshold on the optimality of the resulting policy. However, RL algorithms are not explored there.

II. Preliminaries on LTL

Atomic propositions are Boolean variables that can be either true or false. The ingredients of an LTL formula are a set of atomic propositions AP and several boolean and temporal operators, which are specified according to the following syntax [1]: $\varphi ::= \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq$ `True`, $p \in AP$ and $\bigcirc$ (*next*), U (*until*). $\bot \triangleq \neg\top$. For brevity, we omit the derivations of other useful operators like $\Box$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*). We refer the readers to Chapter 5 of [1] for a full definition of LTL syntax and semantics.

III. Problem Formulation

Consider a mobile robot that satisfies the first-order dynamics: $\dot{p}(t) = u(t)$, where $p(t) = [x(t),\, y(t)] \in \mathbb{R}^2$ and $u(t) = [v_x(t),\, v_y(t)]$ are the robot's position and velocity at time $t \geq 0$. The workspace is a bounded 2D area $\mathcal{W} \subset \mathbb{R}^2$, within which there are static clusters of obstacles $\mathcal{O} \subset \mathcal{W}$ and the free space $\mathcal{F} = \mathcal{W} \backslash \mathcal{O}$.

A. Data-gathering Tasks

The task assigned to the robot is to gather data in regions of interest. Denote by $\Pi = \{\pi_1, \cdots, \pi_M\}$ these regions, where $\pi_\ell \subset \mathcal{F}$, $\forall \ell = 1, \cdots, M$ and $M \geq 1$. Moreover, there is a set of data-gathering actions that the robot can perform at these regions, denoted by $G = \{g_0, g_1, \cdots, g_K\}$, where g_k means that "type-k data is gathered", $\forall k = 1, 2, \cdots, K$ and $K \geq 1$. By default, g_0 means doing nothing. The cost to perform each action is given by a function $Z : G \to \mathbb{R}^+$.

With a slight abuse of notation, we denote the set of atomic propositions by $AP = \{\pi_\ell \wedge g_k, \forall \pi_\ell \in \Pi, \forall g_k \in G\}$, where $\pi_\ell \wedge g_k$ stands for "the robot gathers type-k data at region π_ℓ". We can specify a high-level data-gathering task over AP, denoted by φ, following the LTL semantics in Section II. LTL formulas allow us to specify data-gathering tasks of finite or *infinite* executions.

Example 1: A mobile robot can first record video (g_1) at the entrance (π_2), then infinitely often record temperature (g_2) at the lab (π_1) and sample LiDAR pointclouds (g_3) at the storage (π_3). This task can be specified as the LTL formula: $\varphi = \Diamond\big((\pi_2 \wedge g_1) \wedge \Diamond(\Box\Diamond((\pi_1 \wedge g_2) \wedge \Box\Diamond((\pi_3 \wedge g_3)))\big)$. ■

The robot's *trace* is defined as the sequence of regions the robot has reached and actions it has performed there, i.e., $\textbf{Trace}_R = \{\pi_{\ell_1}, g_{k_1}\}\{\pi_{\ell_2}, g_{k_2}\}\cdots$, where $\pi_{\ell_i} \wedge g_{k_i} \in AP$, $\forall \ell_i, k_i \geq 1$. Given the infinite trace $\textbf{Trace}_R$, it can be verified whether the robot's behavior satisfies the task by checking if $\textbf{Trace}_R \models \varphi$, where $\models$ is the satisfying relation [1].

B. Wireless Communication Network

Along with gathering data, the robot is also tasked with transmitting this data to a data center. This data transfer happens through a wireless network deployed in the workspace, whose set of nodes $\mathcal{K} \triangleq \mathcal{K}_0 \cup \mathcal{K}_r \cup \mathcal{K}_s$ consists of the data center $\mathcal{K}_0 \triangleq \{0\}$, $K-1$ relay nodes $\mathcal{K}_r \triangleq \{1, \cdots, K-1\}$, and the source robot $\mathcal{K}_s \triangleq \{K\}$, where $K \geq 2$. The data center and all relay nodes are *stationary* at positions $p_k \in \mathcal{W}$, $\forall k \in \mathcal{K}_0 \cup \mathcal{K}_r$, while the coordinates of the robot are $p_K(t) = p(t)$, $\forall t \geq 0$. In this network, point-to-point connectivity is modeled through a rate function $R_{kk'}$ that determines the amount of information that is transmitted from node $k \in \mathcal{K}$ and is successfully received by node $k' \in \mathcal{K}$. Then, the data gathered by the robot is transmitted and routed to the data center in a multi-hop fashion via the relays. The routing decisions are captured by the variables $T_{kk'}(t) \in [0,\, 1]$, $\forall k, k' \in \mathcal{K}$, that represent the fraction of time that node k selects node k' as its recipient. Since $T_{kk'}(t)$ represent time slot shares, they also satisfy $\sum_{k' \in \mathcal{K}} T_{kk'}(t) \leq 1$, $\forall k \in \mathcal{K}$. Moreover, we assume that: (i) no self-transmission is allowed, i.e., $T_{kk} = 0$, $\forall k \in \mathcal{K}$; (ii) the data center only receives data, i.e., $T_{0k'} = 0$, $\forall k' \in \mathcal{K}$; and (iii) the source robot only sends data, i.e., $T_{k'K} = 0$, $\forall k' \in \mathcal{K}$.

Given routing decisions, the direct transmission rate from node k to node k' at time t can be defined as $\varrho_{kk'}(t) \triangleq T_{kk'}(t) R_{kk'}$. Between their generation or arrival from another node, and transmission, data are stored in a queue. Let $d_k(t) \geq 0$ denote the size of the queue at node k at time t. Then, the rate of change of the queue at node k is

$$\dot{d}_k(t) = \varrho_k^{\text{in}}(t) - \varrho_k^{\text{out}}(t),\ \forall k \in \mathcal{K}, \tag{1}$$

where $\varrho_k^{\text{in}}(t) = \varrho_k^{\text{gen}}(t) + \sum_{k' \in \mathcal{K}} \varrho_{k'k}(t)$ and $\varrho_k^{\text{out}} = \sum_{k' \in \mathcal{K}} \varrho_{kk'}(t)$ are the total rates at which data *arrive* at and *leave* node k, $\forall k \in \mathcal{K}_r$, and $\varrho_k^{\text{gen}}(t)$ is the rate of data generated at node k at time t. The queue sizes $d_k(t)$ and generated data $\varrho_k^{\text{gen}}(t)$ satisfy the following assumptions:

Assumption 1: (i) The queue size for all relay nodes is zero for all time, i.e., $d_k(t) = 0$, $\forall k \in \mathcal{K}_0 \cup \mathcal{K}_r$ and $\forall t \geq 0$; and (ii) Only the source robot generates data, i.e., $\varrho_K^{\text{gen}}(t) \geq 0$ and $\varrho_k^{\text{gen}}(t) = 0$, $\forall k \in \mathcal{K}_0 \cup \mathcal{K}_r$. ■

Note that Assumption 1(i) can be ensured by selecting appropriate routing decisions that satisfy (1), while Assumption 1(ii) is due the data-gathering tasks for the robot only.

C. Robot Buffer Dynamics and Constraints

Moreover, the robot has a *limited buffer* to store the gathered data. Let $\overline{B} > 0$ denote the maximum buffer size, so that it should hold that $d_K(t) \leq \overline{B}$, $\forall t \geq 0$. In this paper, we separate data generation from transmission. Specifically, we make the following assumption.

Assumption 2: For all time $t \geq 0$ when $\varrho_K^{\text{gen}}(t) > 0$, it holds that $T_{kk'}(t) = 0$ for all $k, k' \in \mathcal{K}$. ■

Under Assumption 2 and recalling equation (1), during the data generation phase, the queue size at the robot evolves as

$$\dot{d}_K(t) = \varrho_K^{\text{in}}(t) - \varrho_K^{\text{out}}(t) = \varrho_K^{\text{gen}}(t) \triangleq D(g_k)\delta(t), \quad (2)$$

where $D : G \to \mathbb{Z}^+$ specifies the data units gathered by taking action g_k, as discussed in Section III-A, and $\delta(t)$ is the unit impulse function, so that data generation is considered instantaneous. Integrating (2), we have that $d_K(t^+) = d_K(t^-) + D(g_k)$, where $d_K(t^-)$ and $d_K(t^+)$ are the stored units *before* and *after* action g_k is performed at $t \geq 0$. If $d_K(t^+) > \overline{B}$, then this action g_k can *not* be performed as the buffer will overflow. We assume that $D(g_k) \leq \overline{B}$, $\forall g_k \in G$ so that any action can be performed if the buffer is empty. Similarly, we can obtain the rate of data leaving the robot's buffer during the transmission phase, as

$$\dot{d}_K(t) = \varrho_K^{\text{in}}(t) - \varrho_K^{\text{out}}(t) = -\varrho_K^{\text{out}}(t). \quad (3)$$

D. Problem Statement

The considered problem is to determine motion and communication controllers for a mobile robot so that its LTL tasks are satisfied, the generated data is successfully transmitted to the data center, and the total distance traveled is minimized. Formally, this problem is defined as:

$$\begin{aligned} \mathbf{min}_{u(t),\mathbf{T}(t)} \quad & \int_0^{\overline{\tau}} \|p(\tau)\| d\tau - \int_0^{\overline{\tau}} \mathbb{E}[\varrho_K^{\text{out}}(p(\tau), \mathbf{T}(\tau))] d\tau \\ \textbf{s.t.} \quad & \textbf{Trace}_R \models \varphi, \\ & \varrho_K^{\text{out}}(t), \mathbf{T}(t) \;\text{ satisfies (1)}, \\ & d_K(t) \;\text{ satisfies (2), (3)}, \\ & \dot{p}(t) = u(t) \;\text{ and }\; p(t) \in \mathcal{F},\; \forall t \geq 0, \end{aligned} \quad (4)$$

where $\overline{\tau}$ is the total execution time; $u(t)$ is the control input; $\mathbf{T}(t) \triangleq \{T_{kk'}(t), \forall k, k' \in \mathcal{K}\}$ is the set of routing decisions.

Note that LTL tasks such as those described in Example 1 can be executed indefinitely. In this case, $\overline{\tau}$ represents the time required to execute the plan prefix and single plan suffix (which are defined in the sequel). Note also that in (4), along with minimizing the distance traveled, the robot also needs to maximize the cumulative rate at which it transmits data to the network. While different objectives are possible to select $\mathbf{T}(t)$, the proposed objective allows for faster data transmission to the data center. Finally, in practice, the rate functions $R_{kk'}(t)$ in (1) depend on the signal-to-noise ratio of the transmission channel, which is *uncertain* [11] and can only be estimated during run time. Thus, the rates $R_{kk'}(t)$ become random variables and so is the uploading rate $\varrho_K^{\text{out}}(t)$ of the robot which appears in (4) in the expectation operator; same holds for (1) and (3).

The main challenge in solving problem (4) lies in: (i) addressing the joint optimization in the space of robot positions and communication variables subject to the LTL task; and (ii) the fact that, the rate functions $R_{kk'}(t)$ are random and unknown. We tackle this challenge by decomposing the high-level planning from communication routing. Then we employ RL to synthesize the motion and communication control strategy simultaneously, so that the high-level plan can be successfully executed while optimizing the total cost.

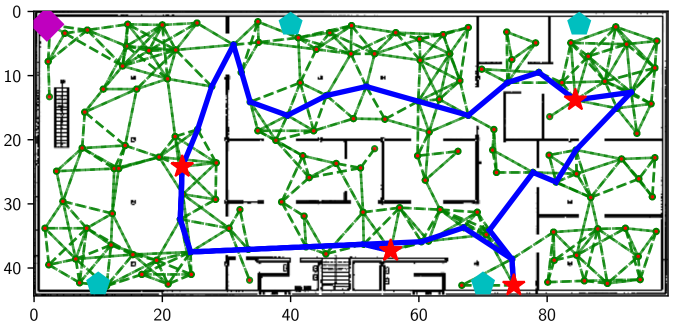

Fig. 1: Example of the roadmap (in dashed green lines) within an office workspace. The initial discrete plan (in blue) for the task considered in Section VI is derived by following Section IV-A.2.

IV. Motion and Communication Control

In this section, we present the motion and communication control modules that are essential to solve Problem 4. We first discuss the discrete plan synthesis algorithm for the local task; second, we describe how the plan can be decomposed into segments and then combined with the wireless routing; and lastly, we describe the two-mode LSTD actor-critic RL algorithm that learns the system model and a control policy.

A. Discrete Task Planning

In this section, we show how to build an abstraction of the robot motion and then how to use this abstraction to synthesize the initial discrete plan, which satisfies the task.

1) Abstraction of Robot Motion: First, a roadmap is constructed to model the robot motion within the workspace avoiding collision with the obstacles. In this work, we rely on the sampling-based method [24] to construct a deterministic roadmap. Denote this roadmap by $\mathcal{M} = (P, E)$, where $P \subset \mathcal{F}$ is the set of waypoints and $E \subseteq P \times P$ is the set of edges. $\mathcal{M}$ is constructed as follows. Initially, $P = \{p_0\}$ where $p_0 \in \mathcal{F}$ is the robot's initial position and $E = \emptyset$. Then, we first uniformly sample the freespace $\mathcal{F}$ to obtain p_r and find the nearest waypoint $p_n \in P$ to p_r. If p_r can be reached from p_n within the sampling time $T_s > 0$ via a straight-line path without colliding with the obstacles, then we add p_r to P and (p_n, p_r) to E. Otherwise, we resample $\mathcal{F}$ for a new point p_r. Given a point p_r, we find every existing waypoint $p_s \in P$ that can reach p_r within time T_s and add (p_r, p_s) to E. Finally we repeat the above process until the maximum number of waypoints is reached.

Given $\mathcal{M}$, we can construct a finite transition system (FTS) to abstract the robot motion among different regions in the workspace, which is denoted by $\mathcal{T} \triangleq (\Pi, \to, \Pi_0, C)$, where $\Pi \subset \mathcal{F}$ is the set of regions of interest, $\to \subseteq \Pi \times \Pi$ denotes the transition relation, $\Pi_0 \in \Pi$ is the initial region, $C : \text{``} \to \text{''} \to \mathbb{R}^+$ approximates the duration of each transition. Particularly, consider two regions $\pi_s, \pi_f \in \Pi$. Denote by $p_s, p_f \in P$ the *closest* waypoints to the center of π_s

and π_f, respectively. Then $(\pi_s, \pi_f) \in \rightarrow$ if there exists a path in $\mathcal{M}$ starting from p_s and ending at p_f *without* crossing any other other region $\pi_\ell \in \Pi$ with $\ell \neq s, f$. Denote the shortest path by $\Gamma_{sf} = p_s p_{s+1} \cdots p_f$. Then, $C(\pi_s, \pi_f) \triangleq |f - s| \cdot T_s$ is the time to traverse Γ_{sf}, $\forall (\pi_s, \pi_f) \in \rightarrow$.

Then the *complete* robot motion and action model is defined by $\mathcal{R} = (\Xi, \rightarrow_\mathcal{R}, AP, L_\mathcal{R}, \xi_0, T_\mathcal{R})$, where $\Xi = \Pi \times G$ is the set of states; $\rightarrow_\mathcal{R} \subseteq \Xi \times \Xi$ is the transition relation so that $(\langle \pi_s, g_\ell \rangle, \langle \pi_f, g_k \rangle) \in \rightarrow_\mathcal{R}$ if (i) $\langle \pi_s, \pi_f \rangle \in \rightarrow$ and $g_k = g_0$, or (ii) $\pi_s = \pi_f$ and $g_\ell, g_k \in G$; AP are the atomic propositions; $L_\mathcal{R}(\langle \pi_s, g_\ell \rangle) = \{\pi_s, g_\ell\}$, $\forall \langle \pi_s, g_\ell \rangle \in \Xi$ is the labeling function; $\xi_0 = \langle \Pi_0, g_0 \rangle$ is the initial state; and $T_\mathcal{R}(\langle \pi_s, g_\ell \rangle, \langle \pi_f, g_k \rangle) = C(\pi_s, \pi_f) + Z(g_\ell)$, $\forall (\langle \pi_s, g_\ell \rangle, \langle \pi_f, g_k \rangle) \in \rightarrow_\mathcal{R}$ is the cost function.

2) Robot Motion and Action Plan: Given $\mathcal{R}$, we can apply the method proposed in our earlier work [5] to synthesize a discrete motion and action plan that has a prefix-suffix structure and a minimal total cost (with respect to the cost function $T_\mathcal{R}$ above). We omit the algorithmic details here due to limited space. This discrete plan has the form

$$\Xi \triangleq \xi_0 \, \xi_1 \cdots \xi_{L-1} \left[\xi_L \, \xi_{L+1} \cdots \xi_H \right]^\omega, \qquad (5)$$

where $\xi_\ell \in \Xi$, $\forall \ell = 0, 1, \cdots, H$. Note that the suffix part $\xi_L \, \xi_{L+1} \cdots \xi_H$ is repeated infinitely often. Namely, Ξ is a sequence of states that the robot follows to satisfy task φ, as the trace of Ξ satisfies φ automatically [5]. An example of the synthesized plan is shown in Figure 1. However, the discrete plan Ξ does *not* account for the buffer constraints. Thus an associated communication protocol needs to be designed for the successful execution of Ξ.

B. Plan Decomposition and Critical Segments

As the robot follows Ξ, it collects data which needs to be uploaded to the network before its buffer overflows. Let

$$\vartheta \triangleq (\xi_{s_0}, \xi_{g_0})(\xi_{s_1}, \xi_{g_1}) \cdots \Big[(\xi_{s_C}, \xi_{g_C}) \cdots (\xi_{s_F}, \xi_{g_F}) \Big]^\omega, \qquad (6)$$

be a subsequence of Ξ that contains states where the robot needs to *empty* its buffer while executing Ξ, where $\xi_{s_i}, \xi_{g_i} \in \Xi$ are the start and goal states of each segment, $\forall s_i, g_i \geq 0$ and $s_0 < g_0 \leq s_1 < g_1 \leq \cdots \leq s_F < g_F$. We call $(\xi_{s_i}, \xi_{g_i}) \in \vartheta$ the *critical segments* of Ξ.

Given Ξ and $\overline{B}$, ϑ is constructed as follows: (i) First, s_0, g_0 are the smallest indices such that $0 < \sum_{\ell=0}^{s_0} D(g_\ell) \leq \overline{B}$, and $\sum_{\ell=0}^{g_0} D(g_\ell) > \overline{B}$, where $\xi_\ell = \langle \pi_\ell, g_\ell \rangle \in \Xi$. Namely, the robot buffer is less than or equal to $\overline{B}$ up to ξ_{s_0}, but it will overflow *after* performing the data-gathering action at ξ_{g_0}. Thus it is critical that the robot uploads the data in its buffer within that segment. (ii) Second, s_{i+1}, g_{i+1} can be derived iteratively, such that $g_i \leq s_{i+1} < g_{i+1}$ are the smallest indices such that $0 < \sum_{\ell=g_i}^{s_{i+1}} D(g_\ell) \leq \overline{B}$, and $\sum_{\ell=g_i}^{g_{i+1}} D(g_\ell) > \overline{B}$, which holds for all $i \in [0, F]$ except for $i = C, F$, where C is the smallest index such that $g_{C-1} \geq L$ and F is the smallest index such that $g_F \geq H$, where the indices L, H are given by the first and last states of the suffix in Ξ of (5). (iii) Last, $\xi_{g_{C-1}} \triangleq \xi_L$ and $\xi_{g_F} \triangleq \xi_L$. Namely, the repetitive suffix of ϑ is achieved by enforcing an empty buffer each time the suffix of Ξ is executed.

C. Motion and Communication Policy Synthesis

In this section, we present how to synthesize a randomized control policy such that a critical segment in (6) can be executed and the robot's buffer is *emptied*. The method is based on the actor-critic RL algorithm [19] and the resulting control policy is learned and improved during run time. Specifically, we address the following problem.

Problem 1: Given a critical segment (ξ_{s_i}, ξ_{g_i}) in (6), design a motion and communication policy such that the robot starts from ξ_{s_i} with buffer $b \leq \overline{B}$ and reaches ξ_{g_i} with buffer 0, with a minimum expected travel distance. ■

1) Joint Motion and Communication Space: We model the joint motion and communication space as a Markov Decision Process (MDP) with *unknown* transition probabilities. Particularly, it is defined as a 4-tuple:

$$\chi \triangleq (\mathcal{S}, \mathcal{A}, \delta, R), \qquad (7)$$

where: (i) The set of states is given by $\mathcal{S} = P \times \mathcal{D}$ where P is the set of waypoints from $\mathcal{M}$ and $\mathcal{D} = \{0, \iota, 2\iota, \cdots, I\iota\}$ is a user-defined quantization of the buffer size with $I \in \mathbb{N}$ the largest integer such that $I\iota \leq \overline{B}$ and $\iota > 0$. Therefore, state $\langle p_n, d_k \rangle \in \mathcal{S}$ means that the robot is at waypoint p_n with d_k units of data in its buffer, $\forall p_n \in \mathcal{P}$ and $\forall d_k \in \mathcal{D}$; (ii) The function of admissible actions $\mathcal{A} : \mathcal{S} \rightarrow 2^A$ is given by $\mathcal{A}(\langle p_n, d_k \rangle) = \{a_{p_n p_m}, \forall (p_n, p_m) \in E\}$, $\forall s = \langle p_n, d_k \rangle \in \mathcal{S}$, where $a_{p_n p_m}$ stands for "moving from p_n to p_m" and A is the set of all allowed actions. (iii) The transition probabilities are given by the function $\delta : \mathcal{S} \times A \times \mathcal{S} \rightarrow [0, 1]$ such that $\delta(s, a_s, s')$ is the probability of transitioning from s to s' via action $a_s \in \mathcal{A}(s)$, $\forall s, s' \in \mathcal{S}$. In particular, consider $s = \langle p_n, d_k \rangle$, action $a_{p_n p_m} \in \mathcal{A}(s)$, and $s' = \langle p_m, d_\ell \rangle$. Then, it holds that

$$\delta(s, a_{p_n p_m}, s') \triangleq Pr\Big(\iota \leq \zeta(p_n, p_m) \leq d_l - d_k \Big), \qquad (8)$$

where $\zeta(\cdot) = \int_{t_n}^{t_m} \overline{\varrho}_K^{\text{out}}(p(\tau), \mathbf{T}(\tau)) d\tau$ is a function that returns the total amount of data that can be uploaded by the robot to the network while moving from waypoint p_n to p_m following $\mathcal{M}$ (during time t_n to t_m) and $\overline{\varrho}_K^{\text{out}}(p(t), \mathbf{T}(t))$ is the optimal solution to the optimization problem

$$\textbf{max}_{\mathbf{T}(t)} \; \varrho_K^{\text{out}}(p_K(t), \mathbf{T}(t)), \quad \textbf{s.t.} \;\; \text{constraints by (1)}, \qquad (9)$$

for $t \in [t_n, t_m]$. Given the channel rates $R_{kk'}(t)$ and robot position $p(t)$ at time $t \in [t_n, t_m]$, problem (9) is linear and can be solved efficiently via a Linear Program (LP) solver, such as Gurobi [25]. For example, the data-uploading rate within an office workspace given a wireless network is shown in Figure 2; (iv) The reward function $R : \mathcal{S} \times A \rightarrow \mathbb{R}_{\geq 0}$ satisfies $R(s, a) = 0$, $\forall a \in \mathcal{A}(s)$ if $s = s_0 \triangleq \langle p_{s_i}, I\iota \rangle$, $R(s, a) = 1$, $\forall a \in \mathcal{A}(s)$ if $s = s_g \triangleq \langle p_{g_i}, 0 \rangle$, and $R(s, a) = -1$ otherwise, where s_0, s_g are the initial and goal states, and p_{s_i}, p_{g_i} are waypoints of ξ_{s_i}, ξ_{g_i}. Namely, a positive reward is received only at the goal state, while a negative reward is received at *all* intermediate states in order to minimize the total distance to reach the goal state.

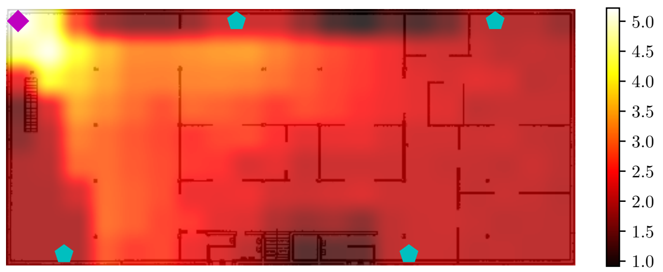

Fig. 2: Heatmap of the data-uploading rate (in "MB/s") within an office workspace used in simulation Section VI, where the data center (purple diamond) and relay nodes (green pentagons) are also shown. The transmission rate at each location is derived by (9).

Remark 1: The function $\zeta(p_n, p_m)$ is *unknown* because the channel rates $R_{kk'}$ are unknown and so is the distribution of uploading rates $\overline{\varrho}_K^{\text{out}}(\cdot)$ along the trajectory from waypoint p_n to p_m by (9). Therefore, the rates $\overline{\varrho}_K^{\text{out}}(\cdot)$ can only be determined empirically during run time. ■

Remark 2: The communication problem (9) is defined here as a proxy to the second objective in problem (4). Determining robot waypoints associated with maximum uploading rates as per (9) is a sufficient condition for solving (4). ■

2) Two-mode LSTD Actor-Critic RL Algorithm: Standard dynamic programming techniques [15] can not be applied directly here since the exact model of χ is unknown. Instead we employ RL algorithms [19], [21] so that the robot can actively explore its environment, learn the model, and determine the optimal control policy, all *at the same time*. In the sequel, we denote by $\tilde{\chi}$ the current estimate of χ that is known by the robot, and by $\tilde{\delta}$ the estimate of the transition probability. Moreover, $\tilde{\delta}$ can be initialized based on empirical data of the uploading rate across the workspace.

We consider the actor-critic RL algorithm proposed in [19] and particularly the LSTD type proposed in [20]–[22]. It consists of two parts: the *actor* that controls the system according to the current policy and the *critic* that oversees the actor's actions and more importantly the received reward in order to evaluate how the policy can be improved. The policy employed by the actor is a *parameterized* and *randomized stationary* policy given by (instead of a lookup table):

$$\eta_{\boldsymbol{\theta}}(s,\, a) \triangleq \frac{e^{h(\boldsymbol{\theta},s,a)}}{\sum_{b\in A(s)} e^{h(\boldsymbol{\theta},s,b)}} \tag{10}$$

where $\boldsymbol{\theta} \in \mathbb{R}^2$ is the parameter; $\eta_{\boldsymbol{\theta}}(s,\, a) \in [0,1]$ is the probability of applying action $a \in A(s)$ at state $s \in \mathcal{S}$ using the Boltzmann distribution [19]; $h(\boldsymbol{\theta}, s, a) \in \mathbb{R}$ is the estimated *score* of applying action a at state s given the current value of $\boldsymbol{\theta}$:

$$h(\boldsymbol{\theta}, s, a) \triangleq \boldsymbol{\theta}^{\intercal} \cdot \boldsymbol{w}(s, a) \tag{11}$$

where $\boldsymbol{w}(s,a) \in \mathbb{R}^2$ is the estimated *feature* vector of the state-action pair (s, a) within $\tilde{\chi}$, defined as

$$\boldsymbol{w}(s,a) \triangleq \begin{bmatrix} \nu_p(s,s_g) - \sum_{s'} \tilde{\delta}(s,a,s')\,\nu_p(s',s_g) \\ \nu_d(s,s_g) - \sum_{s'} \tilde{\delta}(s,a,s')\,\nu_d(s',s_g) \end{bmatrix}, \tag{12}$$

where $\nu_p(s, s_g) \triangleq \texttt{dijkstra}(\mathcal{M}, p, p_g) \geq 0$ is the *distance* feature measured by the length of the shortest path from waypoint p to p_g in the roadmap $\mathcal{M}$; and $\nu_d(s, s_g) \triangleq \|d_g - d\| \geq 0$ is the *data* feature measured as the difference in the data size, where $s = \langle p, d\rangle$ and $s_g = \langle p_g, d_g\rangle$, $\forall s, s_g \in \mathcal{S}$. Note that both ν_p and ν_d can be computed easily given the estimated model $\tilde{\chi}$. Thus, given a value of $\boldsymbol{\theta}$, for any state $s \in \mathcal{S}$ and any allowed action $a \in A(s)$, $\eta_{\boldsymbol{\theta}}(s,\, a)$ can be computed on-the-fly, without the need of a table to store it. On the other hand, the critic maintains an estimate of the *parameterized* state-action value function $Q(s, a)$ defined as

$$Q^r_{\boldsymbol{\theta}}(s,a) \triangleq \boldsymbol{\phi}^{\intercal}_{\boldsymbol{\theta}}(s,a) \cdot \boldsymbol{r}, \tag{13}$$

where $\boldsymbol{r} \in \mathbb{R}^2$ is the parameter and $\boldsymbol{\phi}_{\boldsymbol{\theta}}(s, a) \in \mathbb{R}^2$ is the gradient of the function $\ln\big(\eta_{\boldsymbol{\theta}}(s,\, a)\big)$ with respect to $\boldsymbol{\theta}$, i.e.,

$$\begin{aligned} \boldsymbol{\phi}_{\boldsymbol{\theta}}(s,a) &\triangleq \nabla_{\boldsymbol{\theta}} \ln\big(\eta_{\boldsymbol{\theta}}(s,\, a)\big) \\ &= \boldsymbol{w}(s,a) - \textstyle\sum_{b\in A(s)} \eta_{\boldsymbol{\theta}}(s,\, b)\, \boldsymbol{w}(s,b), \end{aligned} \tag{14}$$

where the second equation is derived directly from (11) and (12). The reason that $Q^r_{\boldsymbol{\theta}}(s,a)$ is designed to have the structure above is closely related to the actor-critic update algorithm below. Specifically, denote by $\boldsymbol{\theta}_k$, $\boldsymbol{r}_k$ the value of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{r}$ at time k, by s_k the state at which robot is and by a_k the action robot takes at time k, $\forall k \geq 0$. Then, at time $k \geq 0$, the actor updates $\boldsymbol{\theta}_k$ as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta_k\, \boldsymbol{r}_k^{\intercal} \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(s_k, a_k) \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(s_{k+1}, a_{k+1}), \tag{15}$$

where $\boldsymbol{\theta}_0$ is the initial value and $\beta_k > 0$ is a time-varying step size. On the other hand, the critic updates $\boldsymbol{r}_k$ as

$$\begin{aligned} \boldsymbol{z}_{k+1} &= \lambda\, \boldsymbol{z}_k + \boldsymbol{\phi}_{\boldsymbol{\theta}_k}(s_k, a_k), \\ \boldsymbol{b}_{k+1} &= \boldsymbol{b}_k + \gamma_k \big[R(s_k,a_k)\boldsymbol{z}_k - \boldsymbol{b}_k\big], \\ \boldsymbol{A}_{k+1} &= \boldsymbol{A}_k + \gamma_k \big[\boldsymbol{z}_k \big(\boldsymbol{\phi}^{\intercal}_{\boldsymbol{\theta}_k}(s_{k+1}, a_{k+1}) \\ &\qquad\qquad - \boldsymbol{\phi}^{\intercal}_{\boldsymbol{\theta}_k}(s_k, a_k)\big) - \boldsymbol{A}_k\big], \\ \boldsymbol{r}_{k+1} &= -\boldsymbol{A}_k^{-1} \boldsymbol{b}_k, \end{aligned} \tag{16}$$

where $\boldsymbol{z}_k \in \mathbb{R}^2$, $\boldsymbol{b}_k \in \mathbb{R}^2$, $\boldsymbol{A}_k \in \mathbb{R}^{2\times 2}$ are internal variables initialized as $\boldsymbol{A}_0 = \boldsymbol{I}_2$, $\boldsymbol{z}_0 = \boldsymbol{b}_0 = \boldsymbol{r}_0 = [0, 0]$, $\lambda \in (0, 1)$ is a design parameter, and $\gamma_k > 0$ is another time-varying step size. Note that $R(s_k, a_k)$ is the *reward* obtained by the robot for performing action a_k at s_k from (7).

The proposed RL algorithm is shown in Alg. 1. It can be run in two different modes: (i) the *direct* mode where the robot learns from *real* experience by interacting with the environment, and (ii) the *indirect* mode where the robot learns from *simulated* experience. Specifically, in the case of direct learning (as shown in Lines 3-7), starting from the initial state $s = \langle p_n, d_k\rangle$, the robot chooses its action a_s by following the randomized policy $\eta_{\boldsymbol{\theta}}(s, a)$ (to move from p_n to p_m). While executing the action a_s, it uploads data according to the optimal routing decisions $\mathbf{T}^\star(p_n)$ obtained by solving problem (9). After the robot reaches waypoint p_m, it updates its state as $s = \langle p_m, d_h\rangle$, where $d_h \in \mathcal{D}$ is the amount of data *left* in the robot's buffer. Then, $\tilde{\delta}$ is updated given the transition $(\langle p_n, d_k\rangle, \langle p_m, d_h\rangle)$. More importantly, the policy parameter $\boldsymbol{\theta}$ is updated as in (15).

Algorithm 1: Policy Synthesis for One Critical Segment

```
   Input: $(\xi_{s_i}, \xi_{g_i}) \in \vartheta$, optional $\boldsymbol{\theta}_{s_i g_i}$, $\tilde{\chi}_0$
1  Initialize variables in (15) and (16).
2  if direct mode then                    // Direct learning
3      Choose randomized action $a_s$ by $\eta_{\boldsymbol{\theta}}(s, a)$ in (10).
4      Transmit data with the routing parameter $\mathbf{T}^\star(t)$ via
       solving optimization (9).          // Communication
5      Update state $s$ after action $a_s$ is done, based on the
       amount of data $b(t)$ in buffer.
6      Update $\tilde{\delta}$ of $\tilde{\chi}$, and update $\boldsymbol{\theta}$ via (15)-(16).
7      Repeat lines 3-6 until $s = s_g$.
8  else                                   // Indirect learning
9      Choose randomized action $a_s$ by $\eta_{\boldsymbol{\theta}}(s, a)$ in (10).
10     Determine the next state $s'$ directly via $\tilde{\chi}$.
11     Update $\boldsymbol{\theta}$ via (15)-(16) and set $s = s'$.
12     Repeat lines 9-11 until desired number of episodes.
13 return $\tilde{\chi}$, $\boldsymbol{\theta}_{s_i g_i} = \boldsymbol{\theta}$.
```

Similar procedure applies to the case of indirect learning (as shown in Lines 9-12), where the main difference is that instead of moving to p_m and measuring the uploaded data, the robot simulates the next state based on $\tilde{\delta}$. As mentioned in Chapter 9 of [19], indirect methods often make fuller use of a limited amount of experience and thus achieve a better policy with fewer environmental interactions. On the other hand, direct methods refine the policy using actual data and are not affected by bias in the estimated model.

Theorem 1: For each $(\xi_{s_i}, \xi_{g_i}) \in \vartheta$, assuming that the real model χ is static, then Alg. 1 converges to the neighborhood of the locally optimal $\boldsymbol{\theta}^\star$ after a finite repetition of Ξ, under the appropriate choices of the step-size $\{\gamma_k\}$, $\{\beta_k\}$ and λ.

Proof: The proof follows directly from Theorem III of [20] and Theorem 3.13 of [22]. Given that the choices of $\{\gamma_k\}$, $\{\beta_k\}$ satisfy Assumption C of [20] and λ is sufficiently close to 1, $\boldsymbol{\theta}$ converges to the neighborhood of the locally optimal $\boldsymbol{\theta}^\star$ such that no improvement larger than a given bound can be made regarding the expected total reward. The same argument holds for each critical segment in ϑ. A detailed proof is omitted due to limited space. ■

V. The Integrated System

In this section, we integrate the components described in Section IV into a complete system. Then we discuss the performance and computational complexity of our method.

A. Complete Plan Execution and Task Satisfaction

The integrated algorithm alternates between the execution of the discrete plan Ξ and the online policy learning for *each* critical segment. By guaranteeing that each critical segment can be successfully executed, we also ensure the successful execution of Ξ and thus the satisfaction of task φ. More importantly, we fuse the two-mode LSTD actor-critic RL algorithm with the high-level plan execution.

Firstly, the discrete plan Ξ and the sequence of critical segments ϑ are synthesized. Then starting from the first element ξ_ℓ of Ξ for $\ell = 0$, the robot moves towards $\xi_{\ell+1}$. There are two cases: (i) If $\xi_\ell = \langle \pi_s, g_l \rangle$ does not belong to a critical segment of ϑ, then the robot does not need to transmit data and can move directly to the next state $\xi_{\ell+1} = \langle \pi_f, g_k \rangle$ in Ξ by following the shortest path from π_s to π_f in $\mathcal{M}$. Also the robot needs to perform the action g_k at π_f to gather data. (ii) If ξ_ℓ corresponds to a critical segment $(\xi_\ell, \xi_{\ell+1}) \in \vartheta$, the robot needs to empty its buffer before reaching $\xi_{\ell+1}$. Then, Alg. 1 is called first in the *indirect* mode for a desired number of episodes to improve $\boldsymbol{\theta}_{s_i g_i}$. Afterwards, Alg. 1 is called in the *direct* mode to update $\tilde{\chi}$ and $\boldsymbol{\theta}_{s_i g_i}$.

It is worth mentioning that the policy parameter $\boldsymbol{\theta}_{s_i g_i}$ is saved for *each* critical segment $(\xi_{s_i}, \xi_{g_i}) \in \vartheta$ and used as the initial $\boldsymbol{\theta}$ in Alg. 1 each time the robot traverses (ξ_{s_i}, ξ_{g_i}). Furthermore, each time the robot runs Alg. 1 for the segment (ξ_{s_i}, ξ_{g_i}), not only the policy parameter $\boldsymbol{\theta}_{s_i g_i}$ but also the model $\tilde{\chi}$ are improved. The model $\tilde{\chi}$ can be used in Alg. 1 as the updated model for *all* critical segments, by only changing the reward function (given the initial and final states of the current critical segment). The above process repeats itself for all consecutive segments $(\xi_\ell, \xi_{\ell+1}) \in \Xi$ in (6) until the system is stopped. Notice that over time, while the robot keeps executing the plan and exploring the workspace, the model $\tilde{\chi}$ becomes accurate in terms of its ability to represent the real world. Consequently, the control policy for each critical segment $(\xi_{s_i}, \xi_{g_i}) \in \vartheta$, i.e., the parameter $\boldsymbol{\theta}_{s_i g_i}$, will converge to a steady value as shown in Theorem 1.

Theorem 2: (I) $\textbf{Trace}_R \models \varphi$; (II) $\varrho_K^{\text{out}}(t), \mathbf{T}(t)$ satisfies (1), $\forall t \geq 0$; (III) $b(t)$ satisfies (2), (3), $\forall t \geq 0$.

Proof: The first part is ensured by the correctness of the discrete plan Ξ and the model-checking process. The second and third parts are guaranteed by the construction of the critical segments ϑ and Alg. 1. Particularly, while executing Ξ, the robot buffer size increases when the robot performs a data-gathering action and is emptied when the robot learns the dynamic policy for each critical segment. ■

Note that the proposed approach *does not* necessarily provide the optimal solution of (4), as the discrete plan Ξ is synthesized without full knowledge of the wireless network.

B. Computational Complexity

The size of the roadmap $\mathcal{M}$ depends on the number of waypoints. Then the size of $\mathcal{R}$ is polynomial in $|\mathcal{M}|$ and the robot action model is typically small. The algorithm in [5] to construct Ξ has complexity $\mathcal{O}(|\mathcal{R}| \cdot 2^{2^{|\varphi|}})$, i.e., polynomial in the size of $\mathcal{R}$ and double-exponential in the length of φ [1]. Lastly, as also discussed in [22]. The learning space of Alg. 1 lies in $\mathbb{R}^2$, independent of the size of χ. The complete complexity is linear to the length of the discrete plan Ξ, particularly the number of critical segments in ϑ.

VI. Case Study

This section presents simulation results for the proposed approach. All algorithms are implemented in Python 2.7 and are available in [26]. The simulations are carried out on a laptop (3.06GHz Duo CPU and 8GB of RAM).

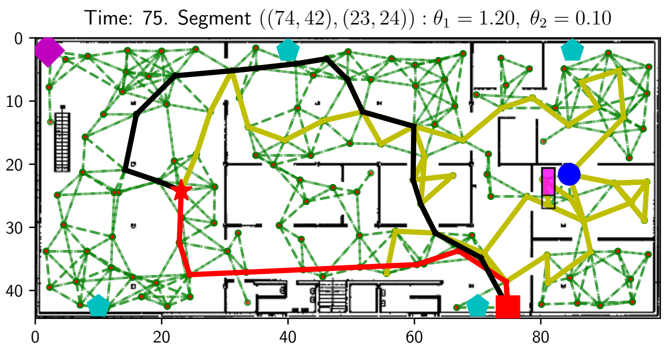
Time: 75. Segment $((74,42),(23,24))$: $\theta_1 = 1.20$, $\theta_2 = 0.10$

Fig. 3: Snapshot of the simulation at time $75s$. The robot (in filled blue circle) is learning the policy for the critical segment $((74, 42), (23, 24))$. The robot's past trajectory is shown in yellow, while the absolute shortest path is in red and the constrained shortest path is in black. The robot's buffer is shown as the filled bar (in magenta). Simulation videos can be found at [27].

A. System Description

A robot that satisfies the first-order dynamics is deployed in the workspace shown in Figure 3. Its maximum velocity is set to $1m/s$. It has buffer size of 80 data units. The workspace has size $100m \times 45m$ and is partitioned into regions that resemble an office environment. The control sampling time is chosen to be $1s$. The wireless network consists of one data center located at $(2, 2)$, four relay nodes located at $(85, 2)$,$(10, 42)$,$(70, 42)$,$(40, 2)$, respectively, and the robot. Furthermore, we use the model discussed in [11] to estimate the channel quality: $P_{kk'} \sim L_0 - 10n \cdot \log\left(\|p_k - p_{k'}\|\right) - W(p_k, p_{k'}) - F_{kk'}$, where $p_k, p_{k'} \in \mathcal{W}$ are the positions of nodes $k, k' \in \mathcal{K}$; $L_0 > 0$ denotes the measured power at a distance $d_0 > 0$ from the source, n is a path loss exponent, $W(p_k, p_{k'})$ is a non-smooth function that models shadowing effects, and $F_{kk'}$ is a Gaussian random variable that models fading effects. Using the signal strength, we can generate samples of the packet error rate and thus the transmission rate $R_{kk'}$ [11]. We set $L_0 = 10$, $n = 0.1$ and the effect of W, F is represented by a 40% uniform uncertainty.

The robot is capable of performing three actions (g_1, g_2, g_3) to collect type-1, type-2, and type-3 data over four regions of interest r_1, r_2, r_3, r_4, located at $(55, 37)$, $(23, 24)$, $(84, 13)$, $(75, 42)$, as shown in Figure 1. Moreover, the actions g_1, g_2, g_3 gather 40, 50, 70 units of data, respectively. Initially, the robot starts from r_1. The robot is tasked with infinitely often gathering type-2 data at region r_2 and type-1 data at region r_1, visiting region r_3 before gathering type-3 data at region r_4, and then gathering type-1 data at region r_2, i.e., $\varphi = (\square\Diamond(r_2 \wedge g_2)) \wedge (\square\Diamond(r_1 \wedge g_1)) \wedge \square(\Diamond((r_3 \wedge g_0) \wedge \Diamond(r_4 \wedge g_3))) \wedge (\square\Diamond(r_2 \wedge g_1))$. It can be seen that none of the two data-gathering actions can be performed consecutively by the robot without emptying its buffer first.

B. Simulation Results

Following Section V, first, a roadmap of 150 nodes and 810 edges is constructed as shown in Figures 1-3. Then it took $0.6s$ for the synthesis algorithm in [5] to find the discrete plan Ξ, of which the repetitive suffix is $[r_1 g_1 r_2 g_2 r_3 g_0 r_4 g_3 r_1 g_1]$ and shown in Figure 1. Lastly, the sequence of critical segments is derived as $\vartheta = [((r_1, g_1), (r_2, g_0)), ((r_2, g_2), (r_4, g_0)), ((r_4, g_3), (r_2, g_0)),$ $((r_2, g_1), (r_1, g_0))]^\omega$, which contains *four* repeated critical segments. For each critical segment, we initialize the RL algorithm by $\boldsymbol{\theta} = [1.0,\ 0.1]$, $\iota = 0.5$, $\lambda = 0.99$ and the step size $\gamma = 1/k$, $\beta = 1/k^2$, for $k = 1, \cdots, K$, where K is the number of times that the suffix of Ξ is repeated. The initial model $\tilde{\chi}_0$ is derived by (7), which has 12150 nodes and 65610 transitions. Then Alg. 1 is called for 10 episodes of indirect learning and 1 episode of direct learning, for each critical segment in ϑ. We perform 30 executions of Ξ and every critical segment is executed once during each execution. It took $95s$ in total and the evolution of $\boldsymbol{\theta}$ for two critical segments is shown in Figure 4, which shows that $\boldsymbol{\theta}$ converges to a local optimal as proven in Theorem 1. Note that a different final value of $\boldsymbol{\theta}$ is reached for different segments. For instance, $\boldsymbol{\theta}$ converges to $[2.13, 0.09]$ for segment $((r_1, g_1), (r_2, g_0))$; and to $[2.56, 0.14]$ for segment $((r_2, g_2), (r_4, g_0))$. Furthermore, given the final values of $\boldsymbol{\theta}$, we performed 200 Monte-Carlo simulations of the robot executing the plan Ξ by following the policies in (10). The distribution of the total cost of the resulting trajectories is shown in Figure 5. The simulation videos of the first and final learning process are available at [27].

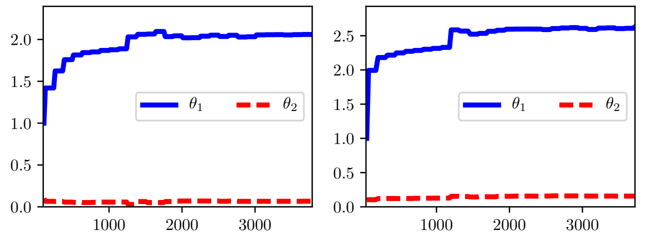

Fig. 4: Evolution of $\boldsymbol{\theta}$ for two critical segments in ϑ, after 30 repetitions of the suffix of the discrete plan Ξ.

Approach		MDP χ Size	Time	Iterations	Cost
Proposed		(1.2e4, 6.5e4)	2.5 mins	120	186
Product-	LP	(5.2e5, 3.1e6)	6.8 hrs	1	133
based	RL	(5.2e5, 3.1e6)	20.3 hrs	30	157

TABLE I: Comparison between the proposed method and the product-based approach via [17] (based on LP) and [22] (based on Actor-critic RL), if the model is *fully known*, regarding the computational complexity and performance. Note $a\text{e}b \triangleq a \times 10^b$.

C. Comparisons

1) Product-based Approach: As mentioned in Section V-B, assuming that the system model χ is *fully known*, the product-based approach proposed in [17] (based on Linear Programming) and in [22] (based on Actor-critic RL) requires first the full construction of motion and action model χ (which has 12482 states and 191242 edges) and the Deterministic Robin Automaton (DRA) $\mathcal{A}_\varphi$ associated with φ [1] (which has 42 states, 210 edges and 2 accepting

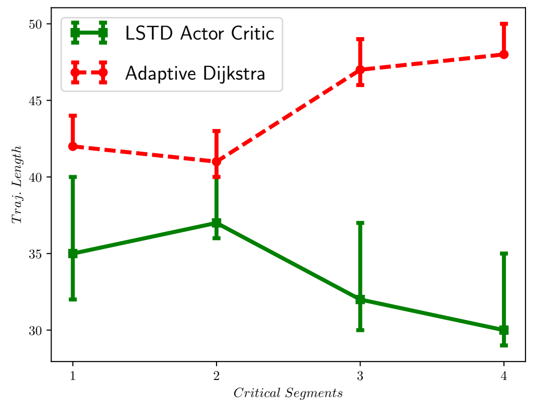

Fig. 5: Distribution of the total cost for each critical segment under 200 Monte-Carlo simulations, by the proposed approach and the adaptive shortest path strategy described in Section VI-C.2.

pairs). Then, the product automaton between χ and $\mathcal{A}_\varphi$ can be constructed (which has 5.2×10^5 states and 3.1×10^6 edges and 2 accepting pairs). The resulting policy drives the system to reach and remain within the accepting maximum end components (AMEC). Table I compares the complexity and performance of these three approaches. It can be seen that while the average total cost via the product-based approach is smaller than our approach, the computational time and memory complexity is much higher due to the enormous size of the product automaton and the complexity of evaluating the feature vector for each state as proposed in [22]. Implementation details can be found in [26].

2) Adaptive Shortest Path Strategy: One simple control strategy to overcome uncertainties in the communication network is via the adaptive shortest path. Namely, the robot follows the shortest path to reach a goal state in $\tilde{\chi}$. Each time it reaches a new state, it checks if its current buffer size is less than the expected size at this state. If not, it waits there and uploads data until this condition holds. This strategy ensures the robot buffer will never overflow, but can be quite inefficient when the uploading rate has large uncertainty and particularly falls below the expected value. We performed 200 Monte-Carlo simulations of the robot executing Ξ by this strategy. The distribution of total cost for each critical segment is shown in Figure 5. It can be seen that the proposed approach reduces greatly the control effort for each critical segment and thus the whole plan.

VII. Conclusion and Future Work

In this work we propose a joint robot motion and communication control strategy that guarantees the satisfaction of a data-gathering task subject to buffer constraint and wireless network routing constraint. The task execution, workspace exploration and parameterized control policy learning are all performed online and simultaneously. Compared with existing work, it does not assume a fully-known environment model and avoids constructing the product automaton. Future work involves multiple-robot systems.

References

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

[2] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[3] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 2689–2696.

[4] 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," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[5] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[6] M. Ji and M. B. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness." *Georgia Institute of Technology*, 2007.

[7] M. Schuresko and J. Cortés, "Distributed motion constraints for algebraic connectivity of robotic networks," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 99–126, 2009.

[8] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[9] Y. Kantaros and M. M. Zavlanos, "Simultaneous intermittent communication control and path optimization in networks of mobile robots," in *Decision and Control (CDC), IEEE Conference on*, 2016, pp. 1794–1799.

[10] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2017. To appear.

[11] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE personal communications*, vol. 6, no. 2, pp. 46–55, 1999.

[12] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3–18, 2013.

[13] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.

[14] U. Ali, Y. Yan, Y. Mostofi, and Y. Wardi, "An optimal control approach for communication and motion co-optimization in realistic fading environments," in *IEEE American Control Conference (ACC)*, 2015, pp. 2930–2935.

[15] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[16] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *Automatic Control, IEEE Transactions on*, vol. 59, no. 5, pp. 1244–1257, 2014.

[17] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *Symposium on Automated Technology for Verification and Analysis*. Springer, 2012, pp. 317–332.

[18] R. Dimitrova, J. Fu, and U. Topcu, "Robust optimal policies for markov decision processes with safety-threshold constraints," in *Decision and Control (CDC), IEEE Conference on*, 2016.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[20] P. Pennesi and I. C. Paschalidis, "A distributed actor-critic algorithm and applications to mobile sensor network coordination problems," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 492–497, 2010.

[21] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[22] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, "Temporal logic motion control using actor–critic methods," *The International Journal of Robotics Research*, p. 0278, 2015.

[23] J. Fu and U. Topcu, "Probably approximately correct MDP learning and control with temporal logic constraints," *arXiv preprint arXiv:1404.7073*, 2014.

[24] S. M. LaValle, *Planning algorithms*. Cambridge university, 2006.

[25] Gurobi, `https://www.gurobi.com/`.

[26] ac_ltl_wsn, `https://github.com/MengGuo/ac_ltl_wsn`.

[27] Videos, `https://vimeo.com/205452793`, `205443075`.