Optimal Multi-Robot Path Planning with LTL Constraints: Guaranteeing Correctness Through Synchronization

Alphan Ulusoy, Stephen L. Smith, and Calin Belta

Abstract In this paper, we consider the automated planning of optimal paths for a robotic team satisfying a high level mission specification. Each robot in the team is modeled as a weighted transition system where the weights have associated deviation values that capture the non-determinism in the traveling times of the robot during its deployment. The mission is given as a Linear Temporal Logic (LTL) formula over a set of propositions satisfied at the regions of the environment. Additionally, we have an optimizing proposition capturing some particular task that must be repeatedly completed by the team. The goal is to minimize the maximum time between successive satisfying instances of the optimizing proposition while guaranteeing that the mission is satisfied even under non-deterministic traveling times. Our method relies on the communication capabilities of the robots to guarantee correctness and maintain performance during deployment. After computing a set of optimal satisfying paths for the members of the team, we also compute a set of synchronization sequences for each robot to ensure that the LTL formula is never violated during deployment. We implement and experimentally evaluate our method considering a persistent monitoring task in a road network environment.

1 Introduction

Temporal logics [5], such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), are extensions of propositional logic that can capture time. Even though temporal logics have been used in model checking of finite systems [1] for quite some time, they have gained popularity as a means for specifying complex mission requirements in path planning and control synthesis problems only

Alphan Ulusoy, Calin Belta
Boston University, Boston, MA, USA e-mail: alphan@bu.edu, cbelta@bu.edu

Stephen L. Smith
University of Waterloo, Waterloo, ON, Canada e-mail: stephen.smith@uwaterloo.ca

recently [15, 13, 21]. Existing work on path planning and control synthesis concentrates on LTL specifications for finite state systems, which may be abstractions of their infinite counterparts [15, 18]. Particularly, given the system model and the mission specification expressed in some temporal logic, satisfying paths and corresponding control strategies can be computed automatically through a search of the state space for deterministic [9], non-deterministic [16, 18, 13, 10] and probabilistic systems [2, 11, 4].

However, more often than not, there are multiple paths that can satisfy a given mission specification. In that case, one generally wants to be able to pick the path that is superior to others with respect to some metric, such as safety, speed, cost, etc. In our previous work, we focused on mission specifications given in LTL along with a particular cost function, and proposed an automated method for finding optimal robot paths that satisfy the mission and minimize the cost function for a single robot [14]. Next, we extended this approach to multi-robot teams by utilizing an abstraction based on timed automata [20]. Then, we proposed a robust method that could accomodate uncertainties in the traveling times of robots with limited communication capabilities [19].

Extending the optimal path planning problem from a single robot to multiple robots is not trivial, as the joint asynchronous motion of all members of the team must be captured in a finite model. In [9], the authors propose a method for decentralized motion of multiple robots subject to LTL specifications. Their method, however, results in sub-optimal performance as it requires the robots to travel synchronously, blocking the execution of the mission before each transition until all robots are synchronized. The vehicle routing problem (VRP) [17] and its extensions to more general classes of temporal constraints [7, 8] also deal with finding optimal satisfying paths for a given specification. In [8], the authors consider optimal vehicle routing with metric temporal logic specifications by converting the problem to a mixed integer linear program (MILP). However, their method does not apply to the missions where robots must repeatedly complete some task, as it does not allow for specifications of the form "always eventually". Furthermore, none of these methods are robust to timing errors that can occur during deployment, as they rely on the robots' ability to follow generated trajectories exactly for satisfaction of the mission specification.

In [20], we proposed a method that uses timed automata to capture the joint asynchronous motion of the members of the robotic team in the environment. After providing a bisimulation [12] of an infinite-dimensional timed automaton to a finite dimensional transition system, we applied our results from [14] to compute an optimal satisfying run. However, multi-robot paths found using this method are implementable only if the traveling times of the robots during deployment exactly match the traveling times used for planning. Otherwise, the order of events may switch resulting in the violation of the mission specification during deployment. In [19], we addressed this issue for robots operating under communication constraints that limit their communication capabilities to a subset of regions. We showed that a trace-closed mission specification will never be violated due to uncertainties in the

speeds of the robots. Then, we proposed a synchronization protocol to maintain and characterize the field performance of the robotic team.

The methods given in [20] and [19] are actually two extremes: In [20], the robots can follow the generated trajectories exactly and do not communicate at all, while in [19] the robots' traveling times during deployment deviate from those used in planning, and they cannot communicate freely. In this paper, we address the middle between these two extremes: the robots cannot follow the generated trajectories exactly, but they can communicate regardless of their positions in the environment. Thus, after obtaining an optimal satisfying run of the team, we compute synchronization sequences that leverage the communication capabilities of the robots to robustify the planned trajectory against deviations in traveling times.

The main contribution of this paper is threefold. First, we provide an algorithm to capture the joint asynchronous behavior of a team of robots modeled as transition systems in a single transition system. This team transition system is provably more compact than the approach based on timed automata that we previously proposed in [20]. Second, for a satisfying run made up of a finite length prefix and an infinite length cyclic suffix, we propose a synchronization protocol and an algorithm to compute synchronization sequences that guarantee correctness under non-deterministic traveling times that may be observed during deployment. Finally, we provide an automated framework that leverages these two methods along with the OPTIMAL-RUN algorithm previously proposed in [14] to solve the multi-robot optimal path planning problem with robustness guarantees. Our experiments show that the computed runs and synchronization sequences indeed provide robustness to uncertainties in traveling times that may occur during the deployment of the team.

The rest of the paper is organized as follows: In Sec. 2, we provide some definitions and preliminaries in formal methods. In Sec. 3, we formulate the optimal and robust multi-robot path planning problem and give an outline of our approach. We provide a complete solution to this problem in Sec. 4. In Sec. 5, we present experiments involving a team of robots performing a persistent surveillance mission in a road network environment. Finally, in Sec. 6, we conclude with final remarks.

2 Preliminaries

In this section, we introduce the notations that we use in the rest of the paper and briefly review some concepts related to automata theory, LTL, and formal verification. For a more rigorous treatment of these topics, we refer the interested reader to [3, 6, 1] and references therein.

For a set Σ, we use $|\Sigma|$, 2^{Σ}, Σ^*, and Σ^ω to denote its cardinality, power set, set of finite words, and set of infinite words, respectively. We define $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ and denote the empty string by $\emptyset$.

Definition 1 (Transition System). A (weighted) transition system (TS) is a tuple $\mathbf{T} := (\mathscr{Q}_\mathrm{T}, q^0_\mathrm{T}, \delta_\mathrm{T}, \Pi_\mathrm{T}, \mathscr{L}_\mathrm{T}, w_\mathrm{T})$, where

1. $\mathscr{Q}_\mathrm{T}$ is a finite set of states;
2. $q^0_\mathrm{T} \in \mathscr{Q}_\mathrm{T}$ is the initial state;
3. $\delta_\mathrm{T} \subseteq \mathscr{Q}_\mathrm{T} \times \mathscr{Q}_\mathrm{T}$ is the transition relation;
4. Π_T is a finite set of atomic propositions;
5. $\mathscr{L}_\mathrm{T} : \mathscr{Q}_\mathrm{T} \to 2^{\Pi_\mathrm{T}}$ is a map giving the set of atomic propositions satisfied in a state;
6. $w_\mathrm{T} : \delta_T \to \mathbb{R}_{>0}$ is a map that assigns a positive weight to each transition.

We define a run of $\mathbf{T}$ as an infinite sequence of states $r_\mathrm{T} = q^0, q^1, \ldots$ such that $q^0 = q^0_T$, $q^k \in \mathscr{Q}_\mathrm{T}$ and $(q^k, q^{k+1}) \in \delta_\mathrm{T}$ for all $k \geq 0$. A run generates an infinite word $\omega_\mathrm{T} = \mathscr{L}(q^0), \mathscr{L}(q^1), \ldots$ where $\mathscr{L}(q^k)$ is the set of atomic propositions satisfied at state q^k.

In this work, we consider mission specifications expressed in Linear Temporal Logic (LTL) [1, 3]. Informally, an LTL formula over the set Π of atomic propositions may contain boolean operators $\neg$ (negation), $\vee$ (disjunction) and $\wedge$ (conjunction), and temporal operators $\mathbf{X}$ (next), $\mathscr{U}$ (until), $\mathbf{F}$ (eventually) and $\mathbf{G}$ (globally/always). LTL formulas are interpreted over infinite words (generated by the transition system $\mathbf{T}$ from Def. 1). For instance, $\mathbf{X}\mathtt{p}$ states that at the next position of a word, proposition $\mathtt{p}$ is true. The formula $\mathtt{p}_1 \mathscr{U} \mathtt{p}_2$ states that there is a future position of the word when proposition $\mathtt{p}_2$ is true, and proposition $\mathtt{p}_1$ is true at least until $\mathtt{p}_2$ is true. The formula $\mathbf{G}\mathtt{p}$ states that $\mathtt{p}$ is true at all positions of the word; the formula $\mathbf{F}\mathtt{p}$ states that $\mathtt{p}$ eventually becomes true in the word. More expressivity can be achieved by combining the temporal and boolean operators. We say a run r_T satisfies ϕ if and only if the word generated by r_T satisfies ϕ. An LTL formula ϕ over a set Π can be represented by a *Büchi automaton*, which is defined next.

Definition 2 (Büchi Automaton). A Büchi automaton is a tuple $\mathbf{B} := (\mathscr{Q}_\mathrm{B}, \mathscr{Q}^0_\mathrm{B}, \Sigma_\mathrm{B}, \delta_\mathrm{B}, \mathscr{F}_\mathrm{B})$, consisting of

1. a finite set of states $\mathscr{Q}_\mathrm{B}$;
2. a set of initial states $\mathscr{Q}^0_\mathrm{B} \subseteq \mathscr{Q}_\mathrm{B}$;
3. an input alphabet Σ_B;
4. a non-deterministic transition relation $\delta_\mathrm{B} \subseteq \mathscr{Q}_\mathrm{B} \times \Sigma_\mathrm{B} \times \mathscr{Q}_\mathrm{B}$;
5. a set of accepting (final) states $\mathscr{F}_\mathrm{B} \subseteq \mathscr{Q}_\mathrm{B}$.

A *run* of $\mathbf{B}$ over an input word $\omega = \omega^0, \omega^1, \ldots$ is a sequence $r_\mathrm{B} = q^0, q^1, \ldots$, such that $q^0 \in \mathscr{Q}^0_\mathrm{B}$, and $(q^k, \omega^k, q^{k+1}) \in \delta_\mathrm{B}$, for all $k \geq 0$. A Büchi automaton $\mathbf{B}$ accepts a word over Σ_B if and only if at least one of the corresponding runs intersects with $\mathscr{F}_\mathrm{B}$ infinitely many times. For any LTL formula ϕ over a set Π, one can construct a Büchi automaton with input alphabet $\Sigma_\mathrm{B} = 2^\Pi$ accepting all and only words over 2^Π that satisfy ϕ.

Definition 3 (Prefix-Suffix Structure). A prefix of a run is a finite path from an initial state to a state q. A periodic suffix is an infinite run originating at the state q reached by the prefix, and periodically repeating a finite path, which we call the suffix cycle, originating and ending at q, and containing no other occurrence of q. A run is in prefix-suffix form if it consists of a prefix followed by a periodic suffix.

3 Problem Formulation and Approach

In this section we introduce the multi-robot path planning problem with temporal constraints for robots with uncertain, but bounded traveling times. Let

$$\mathscr{E} = (V, \rightarrow_{\mathscr{E}}) \tag{1}$$

be a graph, where V is the set of vertices and $\rightarrow_{\mathscr{E}} \subseteq V \times V$ is the set of edges. We consider $\mathscr{E}$ as the quotient graph of a partitioned environment, where V is the set of labels of the regions in the environment and $\rightarrow_{\mathscr{E}}$ is the corresponding adjacency relation. For instance, V can be a set of labels for the roads, intersections, and buildings in an urban-like environment and $\rightarrow_{\mathscr{E}}$ can given the connections in between (see Fig. 5(a)).

Consider a team of m robots moving in an environment modeled by $\mathscr{E}$. The motion capabilities of robot $i, i = 1, \ldots, m$ are represented by a TS $\mathbf{T}_i = (\mathscr{Q}_i, q_i^0, \delta_i, \Pi_i, \mathscr{L}_i, w_i)$, where $\mathscr{Q}_i \subseteq V$; q_i^0 is the initial vertex of robot i; $\delta_i \subseteq \rightarrow_{\mathscr{E}}$ is a relation modeling the capability of robot i to move among the vertices; $\Pi_i \subseteq \Pi$ is the subset of propositions that can be satisfied by robot i; $\mathscr{L}_i$ is a mapping from $\mathscr{Q}_i$ to 2^{Π_i} showing how the propositions are satisfied at vertices; and $w_i(q, q')$ is the average time for robot i to go from vertex q to q', which we assume to be an integer. However, due to the uncertainty in the traveling times of the robots, the actual value of $w_i(q, q')$ observed during deployment, which we denote by $\tilde{w}_i(q, q')$, is a non-deterministic quantity that lies in the interval $[\underline{\rho_i} w_i(q, q'), \overline{\rho_i} w_i(q, q')]$ where $\underline{\rho_i}, \overline{\rho_i}$ are the lower and upper *deviation values* of robot i, respectively. We further assume that lower and upper deviation values $\underline{\rho_i}$ and $\overline{\rho_i}$ of each robot i are known a priori and $0 < \underline{\rho_i} < 1 < \overline{\rho_i}$. In this model, robot i travels along the edges of $\mathbf{T}_i$, and spends zero time on the vertices. We also assume that the robots are equipped with motion primitives that allow them to deterministically move from q to q' for each $(q, q') \in \delta_i$, even though the time it takes to reach from q to q' is non-deterministic within a given interval. In the following, we use the expression "*in the field*" to refer to the non-deterministic traveling times that occur during deployment, and use x and $\tilde{x}$ to denote the planned and actual values, respectively of some variable x.

We consider the case where the robotic team has a mission in which some particular task must be repeatedly completed and the maximum time in between successive completions of this task must be minimized. For instance, in a persistent data gathering mission, the global mission could be *keep gathering data while obeying traffic rules at all times*, and the repeating task could be *gathering data*. For this example, the robots would operate according to the mission specification while ensuring that the maximum time in between any two successive data gatherings is minimized. Consequently, we assume that there is an *optimizing proposition* $\pi \in \Pi$ that corresponds to this particular repeating task and consider multi-robot missions specified by LTL formulae of the form

$$\phi := \varphi \wedge \mathbf{GF}\pi, \tag{2}$$

where φ can be any LTL formula over Π, and $\mathbf{GF}\pi$ means that the proposition π must be repeatedly satisfied. Our aim is to plan multi-robot paths that satisfy the mission ϕ and minimize the maximum time in between successive satisfying instances of π.

To state this problem formally, we assume that each run $r_i = q_i^0, q_i^1, \ldots$ of $\mathbf{T}_i$ (robot i) starts at $t = 0$ and generates a word $\omega_i = \omega_i^0, \omega_i^1, \ldots$ and a corresponding sequence of time instances $\mathbb{T}_i := t_i^0, t_i^1, \ldots$ such that $\omega_i^k = \mathscr{L}_i(q_i^k)$ is satisfied at t_i^k. To define the behavior of the team as a whole, we consider the sequences $\mathbb{T}_i$ as sets and take the union $\bigcup_{i=1}^m \mathbb{T}_i$ and order this set in an ascending order to obtain the sequence $\mathbb{T} := t^0, t^1, \ldots$. Then, we define $\omega_{team} = \omega_{team}^0, \omega_{team}^1, \ldots$ to be the word generated by the team of robots where ω_{team}^k is the union of all propositions satisfied at t^k. Finally, we define the infinite sequence $\mathbb{T}^\pi = \mathbb{T}^\pi(1), \mathbb{T}^\pi(2), \ldots$ where $\mathbb{T}^\pi(k)$ stands for the time instance when π is satisfied for the k^{th} time by the team and define the cost function

$$J(\mathbb{T}^\pi) = \limsup_{i \to +\infty} \left(\mathbb{T}^\pi(i+1) - \mathbb{T}^\pi(i)\right). \tag{3}$$

Thus, the problem becomes that of finding an optimal run of the team that satisfies ϕ and minimizes (3). However, the non-determinism in traveling times imposes two additional difficulties which directly follow from Prop. 3.2 in [19]: First, if the traveling times observed during deployment deviate from those used in planning, then there exist missions that will be violated in the field. Second, the worst case performance of the robotic team during deployment in terms of Eq. 3 will be limited by that of a single member.

To guarantee correctness in the field, and limit the deviation of the performance of the team from the planned optimal run during deployment, we propose periodic synchronization of the robots. Using this synchronization protocol, robots synchronize with each other according to pre-computed synchronization sequences $s_i, i = 1, \ldots, m$ as they execute their runs $r_i, i = 1, \ldots, m$ in the field. We can now formulate the problem.

Problem 1. Given a team of m robots modeled as transition systems $\mathbf{T}_i$, $i = 1, \ldots, m$, and an LTL formula ϕ over Π in the form (2), synthesize individual runs r_i and synchronization sequences s_i for each robot such that $\mathbb{T}^\pi$ minimizes the cost function (3), and $\tilde{\omega}_{team}$, *i.e.*, the word observed in the field, satisfies ϕ.

Note that, if we think of $\tilde{w}_i(q, q')$ as independent random variables with expected value $w_i(q, q')$, then our objective in Prob. 1 is equivalent to minimizing the expected time between successive satisfactions of π while ensuring that ϕ is never violated. Since $\tilde{\omega}_{team}$ observed in the field is likely to be sub-optimal, we will also seek to bound the deviation from optimality in the field. As we consider LTL formulas containing $\mathbf{GF}\pi$, this optimization problem is always well-posed.

Our solution to Problem 1 can be outlined as follows:

1. We obtain the team transition system $\mathbf{T}$ that captures the joint asynchronous behavior of the members of the team (See Sec. 4.1);

2. We find an optimal satisfying run $r^{\star}_{team}$ on $\mathbf{T}$ using the OPTIMAL-RUN algorithm we previously developed in [14] and obtain individual optimal runs $r^{\star}_i, i = 1,\ldots,m$ (See Sec. 4.2);
3. We generate the synchronization sequences $s_i, i = 1,\ldots,m$ to guarantee correctness in the field and calculate an upper bound on the field value of the cost function (3) (See Sec. 4.3).

4 Problem Solution

In this section, we describe each step of our solution to Prob. 1 in detail with the help of a simple illustrative example. We present our experimental results in Sec. 5.

4.1 Obtaining the Team Transition System

In [20], we showed that the joint asynchronous behavior of a robotic team modeled as m transition systems $\mathbf{T}_i, i = 1,\ldots,m$ (Def. 1) can be captured using a region automaton. A region automaton, as given in the following definition from [19], is a finite transition system that keeps track of the relative positions of the robots as they move asynchronously in the environment.

Definition 4 (Region Automaton). The region automaton $\mathbf{R}$ is a TS (Def. 1) $\mathbf{R} := (\mathscr{Q}_{\mathrm{R}}, q^0_{\mathrm{R}}, \delta_{\mathrm{R}}, \Pi_{\mathrm{R}}, \mathscr{L}_{\mathrm{R}}, w_{\mathrm{R}})$, where

1. $\mathscr{Q}_{\mathrm{R}}$ is the set of states of the form (q,r) such that
   a. q is a tuple of state pairs $(q_1q'_1,\ldots,q_mq'_m)$ where the i^{th} element $q_iq'_i$ is a source-target state pair from $\mathscr{Q}_i$ of $\mathbf{T}_i$ meaning robot i is currently on its way from q_i to q'_i, and
   b. r is a tuple of clock values $(x_1,\ldots,x_m)$ where the i^{th} element denotes the time elapsed since robot i left state q_i.
2. $q^0_{\mathrm{R}} \subseteq \mathscr{Q}_{\mathrm{R}}$ is the set of initial states with $r = (0,\ldots,0)$ and $q = (q^0_1q'_1,\ldots,q^0_mq'_m)$ such that q^0_i is the initial state of $\mathbf{T}_i$ and $(q^0_i,q'_i) \in \delta_i$.
3. δ_{R} is the transition relation such that a transition from (q,r) to (q',r') exists if and only if
   a. $(q_i,q'_i),(q'_i,q''_i) \in \delta_i$ for all changed state pairs where the i^{th} element $q_iq'_i$ in q changes to $q'_iq''_i$ in q',
   b. $w_i(q_i,q'_i) - x_i$ of all changed state pairs are equal to each other and are strictly smaller than those of unchanged state pairs, and
   c. for all changed state pairs corresponding x'_i in r' becomes $x'_i = 0$ and all other clock values in r are incremented by $w_i(q_i,q'_i) - x_i$ in r'.
4. $\Pi_{\mathrm{R}} = \cup_{i=1}^{m} \Pi_i$ is the set of propositions;

5. $\mathscr{L}_{\mathrm{R}} : \mathscr{Q}_{\mathrm{R}} \to 2^{\Pi_{\mathrm{R}}}$ is a map giving the set of atomic propositions satisfied in a state. For a state (q,r), $\mathscr{L}_{\mathrm{R}}((q,r)) = \cup_{i|x_i=0}\mathscr{L}_i(q_i)$;
6. $w_{\mathrm{R}} : \delta_R \to \mathbb{R}_{\geq 0}$ is a map that assigns a non-negative weight to each transition such that $w_{\mathrm{R}}((q,r),(q',r')) = w_i(q_i,q_i') - x_i$ for each state pair that has changed from q_iq_i' to $q_i'q_i''$ with a corresponding clock value of $x_i' = 0$ in r'.

Example 1. *Fig. 1 illustrates the TS's of two robots that are expected to satisfy the mission* $\phi := \mathbf{G}(\mathtt{p}_1 \Rightarrow \mathbf{X}(\neg \mathtt{p}_1 \mathscr{U} \mathtt{p}_3)) \wedge \mathbf{GF}\pi$*, where* $\Pi_1 = \{\mathtt{p}_1, \pi\}$*,* $\Pi_2 = \{\mathtt{p}_2, \mathtt{p}_3, \pi\}$*, and* $\Pi = \{\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3, \pi\}$*. The region automaton* $\mathbf{R}$ *that models the robots is given in Fig. 2.*

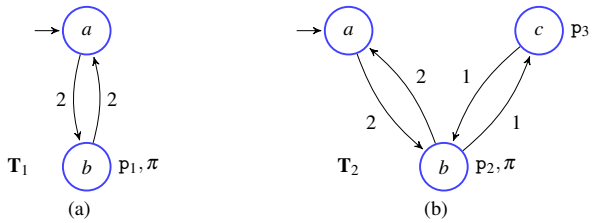


Fig. 1 Figs. (a) and (b) show the transition systems $\mathbf{T}_1$ and $\mathbf{T}_2$ of two robots in an environment with three vertices. The states of the transition systems correspond to vertices $\{a,b,c\}$ and the edges represent the motion capabilities of each robot. The weights of the edges represent the traveling times between any two vertices. The propositions $\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3$ and π are shown next to the vertices where they can be satisfied by the robots.

However, as a region automaton encodes the directions of travel of the robots as opposed to their locations, it typically contains redundant states, and thus can typically be reduced to a smaller size. The following example illustrates this fact.

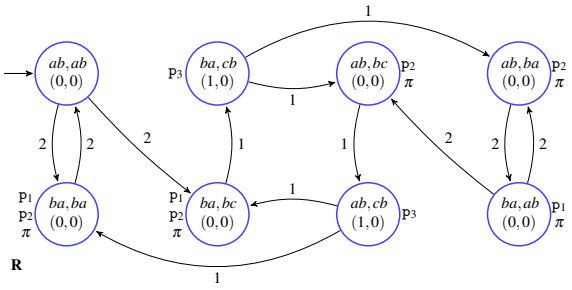


Fig. 2 The finite state region automaton capturing the joint behavior of two robots in 9 states. In a circle representing a state (q,r), the first line is q and the second line is r.

Example 1 Revisited. *State* $((ab,bc),(0,0))$ *of the region automaton* $\mathbf{R}$ *given in Fig. 2 is equivalent to the state* $((ab,ba),(0,0))$ *in the sense that both robots satisfy the same propositions and the positions of both robots are the same at both states,*

i.e., robot 1 is at a and robot 2 is at b. These two states differ only in the future direction of travel of the second robot, i.e., robot 2 travels towards c in the first state whereas it travels towards a in the second state. This information, however, is redundant as it can be obtained just by looking at the next state of the team in any given run.

Motivated by this observation, we define a binary relation $\mathscr{R}$ to reduce the region automaton $\mathbf{R}$ to a smaller team transition system $\mathbf{T}$.

Definition 5 (Binary Relation $\mathscr{R}$). Binary relation $\mathscr{R} = \{(s,t) | s \in \mathscr{Q}_\mathrm{R}, t \in \mathscr{Q}_\mathrm{T}\}$ is a mapping between the states of $\mathbf{R}$ and $\mathbf{T}$ that maps a state $s = ((q_1 q_1', \ldots, q_m q_m'), (x_1, \ldots, x_m))$ in $\mathscr{Q}_\mathrm{R}$ to a state $t = (t_1, \ldots, t_m)$ in $\mathscr{Q}_\mathrm{T}$, where $t_i = q_i$ if $x_i = 0$ and $t_i = q_i q_i' x_i$ if $x_i > 0$. Note that, $x_i = 0$ for at least one $i \in \{1, \ldots, m\}$. We refer to a state $t_i \in \mathscr{Q}_\mathrm{T}$ of the form $q_i q_i' x_i$ as a traveling state as it captures the instant where robot i has traveled from q_i to q_i' for x_i time units.

Given a region automaton $\mathbf{R}$, we can obtain the corresponding team transition system $\mathbf{T}$ using the binary relation $\mathscr{R}$ and the following procedure.

Procedure 1 (Obtaining T from R) *Using $\mathscr{R}$ we construct the team transition system $\mathbf{T}$ from the region automaton $\mathbf{R}$ as follows:*

1. *For each $s \in \mathscr{Q}_\mathrm{R}$ we define the corresponding $t \in \mathscr{Q}_\mathrm{T}$ as given in Def. 5 such that $(s,t) \in \mathscr{R}$.*
2. *We set $\mathscr{L}_\mathrm{T}(t) = \mathscr{L}_\mathrm{R}(s)$. Note that, each s that corresponds to a given t has the same set of propositions due to the way $\mathbf{R}$ is constructed (Def. 4) [20].*
3. *For each s corresponding to a given t, we define the corresponding transitions originating from t in $\mathbf{T}$ such that $\exists (t,t') \in \delta_\mathrm{T} \forall (s,s') \in \delta_\mathrm{R}$ where $(s,t) \in \mathscr{R}$ and $(s',t') \in \mathscr{R}$.*
4. *We mark a state t in $\mathscr{Q}_\mathrm{T}$ as the initial state of $\mathbf{T}$ if the corresponding s is an initial state in $\mathscr{Q}_\mathrm{R}$. Note that, all states that correspond to a given t are either in q_R^0 altogether or none of them are in q_R^0.*

The following proposition shows that the team transition system $\mathbf{T}$ obtained using Proc. 1 and the corresponding region automaton $\mathbf{R}$ are bisimulation equivalent, *i.e.*, there exists a binary relation between the states and the transitions of $\mathbf{R}$ and $\mathbf{T}$ such that they behave in the same way [1].

Proposition 1 (Bisimulation Equivalence). *The team transition system $\mathbf{T}$ obtained using Proc. 1 and the region automaton $\mathbf{R}$ are bisimulation equivalent, i.e., $\mathbf{R} \sim \mathbf{T}$, and $\mathscr{R}$ is a bisimulation relation for $\mathbf{R}$ and $\mathbf{T}$.*

Proof. In the following, we use $Post(s)$ to denote the set of states that can be reached from state s after taking a single transition out of s. For any $(s,t) \in \mathscr{R}$ where $s \in \mathscr{Q}_\mathrm{R}$ and $t \in \mathscr{Q}_\mathrm{T}$, it holds that $\mathscr{L}(s) = \mathscr{L}(t)$. Furthermore, for any $(s,t) \in \mathscr{R}$ it also holds by construction that $\forall s' \in Post(s), \exists t' \in Post(t) | (s',t') \in \mathscr{R}$ and $\forall t' \in Post(t), \exists s' \in Post(s) | (s',t') \in \mathscr{R}$. Finally, we also have $\forall s \in q_\mathrm{R}^0$, $\exists t \in q_\mathrm{T}^0 | (s,t) \in \mathscr{R}$ and $\forall t \in q_\mathrm{T}^0, \exists s \in q_\mathrm{R}^0 | (s,t) \in \mathscr{R}$. Therefore, $\mathbf{R}$ and $\mathbf{T}$ are bisimulation equivalent, *i.e.*, $\mathbf{R} \sim \mathbf{T}$, and $\mathscr{R}$ is a bisimulation relation for $\mathbf{R}$ and $\mathbf{T}$. ■

Example 1 Revisited. *Using $\mathscr{R}$ we construct* $\mathbf{T}$ *(Fig. 3) that captures the joint asynchronous behavior of the team in 6 states whereas the corresponding region automaton* $\mathbf{R}$ *had 9 states. A state labeled (a,b) means robot 1 is at region a and robot 2 is at region b, whereas a state labeled $(ba1,c)$ means robot 1 traveled from b to a for 1 time unit and robot 2 is at c.*

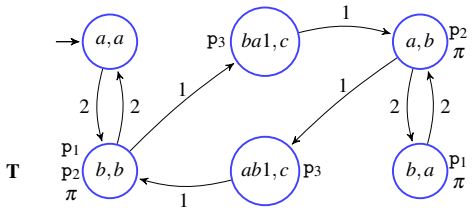


Fig. 3 The team transition system capturing the joint behavior of two robots in 6 states.

In [20] we showed that the number of states $|\mathscr{Q}_{\mathrm{R}}|$ of the region automaton $\mathbf{R}$ that models the m TSs $\mathbf{T}_i, i = 1,\ldots,m$ is bounded by $(\prod_{i=1}^{m}|\delta_i|)\,(\prod_{i=1}^{m}W_i - \prod_{i=1}^{m}(W_i-1))$, where $|\delta_i|$ is the number of transitions in the TS $\mathbf{T}_i$ of robot i and W_i is maximum weight of any transition in $\mathbf{T}_i$. The following proposition provides a bound on the number of states $|\mathscr{Q}_{\mathrm{T}}|$ of $\mathbf{T}$ and shows that it is indeed significantly smaller than the bound on $|\mathscr{Q}_{\mathrm{R}}|$.

Proposition 2. *The number of states $|\mathscr{Q}_{\mathrm{T}}|$ of* $\mathbf{T}$ *is bounded by*

$$\prod_{i=1}^{m}|\mathscr{Q}_i| + (W-1)\prod_{i=1}^{m}|\delta_i| \tag{4}$$

where W is the largest edge weight in all TS's.

Proof. The first term in (4) is the maximum number of states that we can have in the Cartesian product of $T_i, i = 1,\ldots,m$. The second term in (4) is an upper-bound on the number of traveling states (Def. 5) that we can define as we construct $\mathbf{T}$. Here, $\prod_{i=1}^{m}|\delta_i|$ is the maximum number of transitions that we can have in the Cartesian product of $\mathbf{T}_i$s and $(W-1)$ is the upper bound on the number of new traveling states per transition. Thus, $|\mathscr{Q}_{\mathrm{T}}|$ is bounded by the sum of these two terms as given in (4). ■

Finally, we note that the states of $\mathbf{T}$ corresponds to the instants where at least one member of the team has completed a transition in its individual TS and is currently at a vertex while other robots may still be traveling. Using this fact, one can construct $\mathbf{T}$ directly by using a depth first search that runs in parallel on the TS's of the individual members of the team as given in Alg. 1.

Alg. 1 is essentially a recursive depth first search (lines 4 – 17) that starts at the initial state of the team transition system $\mathbf{T}$ (line 3). The initial state q_{T}^{0} of $\mathbf{T}$ is defined as the tuple of the initial states of the m $\mathbf{T}_i$s (line 2). Given a state q of $\mathbf{T}$, the function `dfsT` first generates all possible tuples of transitions that can be

Algorithm 1: CONSTRUCT-TEAM-TS

Input: $(\mathbf{T}_1, \ldots, \mathbf{T}_m)$.
Output: Corresponding team transition system $\mathbf{T}$.

1 $q_{\mathrm{T}}^0 := (q_1^0, \ldots, q_m^0)$, where q_i^0 is the initial state of $\mathbf{T}_i$.
2 **dfsT**(q_{T}^0).

3 **Function dfsT**(state tuple $q \in \mathscr{Q}_{\mathrm{T}}$)

4 $q[i]$ is the i^{th} element of state tuple $q \in \mathscr{Q}_{\mathrm{T}}$.
5 t_i is a transition of $\mathbf{T}_i, i = 1, \ldots, m$, such that $t_i \in \{(q[i], q_i') | (q[i], q_i') \in \delta_i\}$ if $q[i] \in \mathscr{Q}_i$. Else if $q[i] = q_i q_i' x_i$, then $t_i = (q_i, q_i')$.
6 $T := (t_1, \ldots, t_m)$ is a tuple of such transitions.
7 $\mathscr{T}$ is the set of all such transition tuples at q.
8 **foreach** *transition tuple* $T \in \mathscr{T}$ **do**
9 $w \leftarrow$ Shortest time until a robot is at a vertex while the transitions in T are being taken.
10 Find the q' that corresponds to this new state of the team using $\mathscr{R}$.
11 **if** $q' \notin \mathscr{Q}_{\mathrm{T}}$ **then**
12 Add state q' to $\mathscr{Q}_{\mathrm{T}}$.
13 Set $\mathscr{L}(q') = \cup_{i | q[i] \in \mathscr{Q}_i} \mathscr{L}(q[i])$.
14 Add (q, q') to δ_{T} with weight w.
15 Continue search from q': **dfsT**(q').
16 **else if** $(q, q') \notin \delta_{\mathrm{T}}$ **then**
17 Add (q, q') to δ_{T} with weight w.

taken at the current states of the m TSs (lines 4 – 7). The current state of TS $\mathbf{T}_i$ is given by the i^{th} element $q[i]$ of the current state q of the $\mathbf{T}$. At line 5 of Alg. 1, we consider all possible transitions out of the current states of all TSs $\mathbf{T}_i, i = 1, \ldots, m$. If $q[i] \in \mathscr{Q}_i$, *i.e.*, $q[i]$ is a regular state of $\mathbf{T}_i$, then all transitions going out of this state in $\mathbf{T}_i$ will be considered in the transition tuples that we will construct. Else, $q[i]$ is a traveling state of $\mathbf{T}_i$ of the form $q_i q_i' x_i$, and the only transition that can be taken is the one that is being taken, *i.e.*, the transition from q_i to q_i'. Then, we construct the set of all possible tuples of transitions that can be taken at the current states of the m TSs (lines 6–7) and process each tuple one by one (lines 8–17). In a transition tuple T, the i^{th} element gives the transition that can be taken at the current state of $\mathbf{T}_i$. In lines 9–10, we find the next instant where at least one transition in T has been completed and the next state q' of $\mathbf{T}$ that has been reached. If q' is a new state (lines 11 – 15), we accordingly add it to $\mathscr{Q}_{\mathrm{T}}$ and define its propositions. Then, we add the transition that has just been completed to δ_{T} and continue our search from this new state q'. Else, we add the transition that has just been completed to δ_{T} if required and proceed to the next transition tuple in $\mathscr{T}$. The algorithm concludes when all states and transitions of $\mathbf{T}$ have been discovered.

Remark 2 (Comparison with Naive Construction). *One can avoid going through Alg. 1 and capture the joint behavior of the team by discretizing each transition in $\mathbf{T}_i, i = 1, \ldots, m$ to unit-length edges and taking the synchronous product of these m $\mathbf{T}_i$'s. This approach, however, yields a much larger model whose state count is*

bounded by

$$\prod_{i=1}^{m}\left(|\mathscr{Q}_i|+\sum_{(q,q')\in\delta_i} w_i(q,q')-|\delta_i|\right).$$

For the case where we have m identical robots in an environment with Q vertices, Δ edges and a uniform edge weight of W, the above given bound is $O((Q+\Delta W)^m)$, whereas the bound given by Prop. 2 is $O(Q^m+\Delta^m W)$.

4.2 Obtaining Optimal Satisfying Runs and Transition Systems with Traveling States

After constructing $\mathbf{T}$ that models the team, we use Alg. OPTIMAL-RUN from [14] to obtain an optimal run $r^\star_{team}$ on $\mathbf{T}$ that minimizes the cost function (3). The optimal run $r^\star_{team}$ is always in prefix-suffix form, consisting of a finite sequence of states of $\mathbf{T}$ (prefix), followed by infinite repetitions of another finite sequence of states of $\mathbf{T}$ (suffix) as given in Def. 3.

Example 1 Revisited. *For the example we have shown, running Alg.* OPTIMAL-RUN *[14] on* $\mathbf{T}$ *given in Fig. 3 for the formula* $\phi=\mathbf{G}(\mathrm{p}_1\Rightarrow\mathbf{X}(\neg\mathrm{p}_1\,\mathscr{U}\,\mathrm{p}_3))\wedge\mathbf{GF}\pi$ *results in the optimal run*

$\mathbb{T}$	0	2	3	4	5	6	...
$r^\star_{team}$	a,a	b,b	ba1,c	a,b	ab1,c	b,b	...
$\mathscr{L}_{\mathbf{T}}(\cdot)$	$\emptyset$	$\mathrm{p}_1,\mathrm{p}_2,\pi$	p_3	p_2,π	p_3	$\mathrm{p}_1,\mathrm{p}_2,\pi$	...

where the first row shows when transitions occur, the second row corresponds the run $r^\star_{team}$, and the last row shows the satisfying atomic propositions. For this run, $(a,a),(b,b)$ is the finite prefix and $(ba1,c),(a,b),(ab1,c),(b,b)$ is the suffix cycle, which will be repeated infinite number of times. Also, the time sequence $\mathbb{T}^\pi$ of satisfaction of π is $\mathbb{T}^\pi=2,4,6,8,\ldots$ and the cost as defined in (3) *is $J(\mathbb{T}^\pi)=2$.*

Since $\mathbf{T}$ captures the asynchronous motion of the robots, the optimal satisfying run $r^\star_{team}$ on $\mathbf{T}$ may contain some traveling states which do not appear in the individual TSs $\mathbf{T}_i, i=1,\ldots,m$ that we started with. But we cannot ignore such traveling states either, as each one of them is a candidate synchronization point for the corresponding robot as we discuss in Sec. 4.3. Instead, we insert those traveling states into the run $r^\star_{team}$ and the individual TSs so that the robots will be able to synchronize with each other at those points if needed. In the following, we use $q^k[i]$ to denote the i^{th} element of the k^{th} state tuple in $r^\star_{team}$, which is also the state of robot i at that position of $r^\star_{team}$. As given in Def. 5, a traveling state of robot i has the form $q_iq'_ix_i$. First, we construct the set $\mathscr{S}=\{q^k[i]\mid q^k[i]=q_iq'_ix_i\,\forall k,i\}$ of all traveling states that appear in $r^\star_{team}$. Then, we check each pair $q^k[i]q^{k+1}[i]$ in $r^\star_{team}$ for all i and k to see if the corresponding transition skips any of the traveling states in $\mathscr{S}$. In between all those pairs that skip some traveling state in $\mathscr{S}$, we insert a new state tuple q^{new} consisting

of the appropriate traveling states. Notice that, some elements of q^{new} may not be in $\mathscr{S}$. If there are any such new traveling states, we add them to the set $\mathscr{S}$ and repeat until the set $\mathscr{S}$ stops changing. Next, we add each traveling state in $\mathscr{S}$ to its corresponding TS $\mathbf{T}_i$. Then, we break the weight of the corresponding original transition from q_i to q_i' into segments so that robot i visits the new traveling states of the form $q_i q_i' x_i\ x_i$ time-units after leaving q_i for q_i'. Finally, using the following definition, we project the optimal satisfying run $r^\star_{team}$ down to individual robots $\mathbf{T}_i, i = 1,\ldots,m$ to obtain individual optimal satisfying runs $r^\star_i, i = 1,\ldots,m$.

Definition 6 (Projection of a Run on T to $\mathbf{T}_i$'s). Given a run r_{team} on $\mathbf{T}$ where $r_{team} = q^0, q^1, \ldots$, we define its projection on $\mathbf{T}_i$ as run $r_i = q_i^0, q_i^1, \ldots$ for all $i = 1,\ldots,m$, such that such that $q_i^k = q^k[i]$ where $q^k[i]$ is the i^{th} element of tuple q^k.

It can be easily seen that the set of runs $r_i, i = 1,\ldots,m$ obtained from r_{team} using Def. 6 and the run r_{team} on $\mathbf{T}$ indeed correspond to each other: The projection given in Def. 6 simply breaks down a sequence of tuples of states into a tuple of sequences of states, while preserving the order of the states. Thus, the word ω and the time sequence $\mathbb{T}$ generated by $r_i, i = 1,\ldots,m$ are exactly the word ω_{team} and the time sequence $\mathbb{T}_{team}$ generated by r_{team}. Moreover, if run r_{team} is in prefix-suffix form, all individual runs r_i projected from r_{team} are also in prefix-suffix form. Therefore, the individual runs projected from the optimal run $r^\star_{team}$ are always in prefix-suffix form.

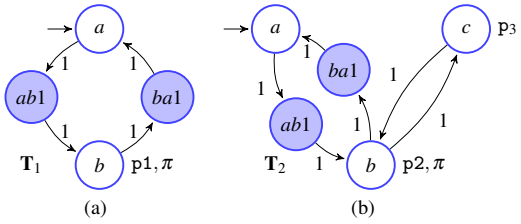


Fig. 4 Figs. (a) and (b) show the TSs with new traveling states that correspond to the optimal run $r^\star_{team}$ that we computed for Ex. 1. The new traveling states $ab1$ and $ba1$ of the TSs are highlighted in blue.

Example 1 Revisited. *For this example, after inserting the traveling states to $r^\star_{team}$, we have*

$\mathbb{T}$	0	1	2	3	4	5	6	...
$r^\star_{team}$	a,a	ab1,ab1	b,b	ba1,c	a,b	ab1,c	b,b	...
$\mathscr{L}_{\mathbf{T}}(\cdot)$	$\emptyset$	$\emptyset$	$\mathtt{p}_1, \mathtt{p}_2, \pi$	$\mathtt{p}_3$	$\mathtt{p}_2, \pi$	$\mathtt{p}_3$	$\mathtt{p}_1, \mathtt{p}_2, \pi$	...

Fig. 4 illustrates the corresponding TSs with new traveling states $ab1$ and $ba1$ highlighted in blue. For the optimal run $r^\star_{team}$ we obtained for this example, we have runs of individual robots from Def. 6 as $r^\star_1 = a, ab1, b, ba1, a, ab1, b, ba1, a, ab1, \ldots$ and $r^\star_2 = a, ab1, b, c, b, c, b, c, b, c, \ldots$.

4.3 Guaranteeing Correctness through Synchronization and the Optimality Bound

As the robots execute their infinite runs in the field, they synchronize with each other according to the synchronization sequences that we generate using Alg. 2. The synchronization sequence s_i of robot i is an infinite sequence of pairs of sets. The k^{th} element of s_i, denoted by s_i^k, corresponds to the k^{th} element q_i^k of $r_i^\star$. Each s_i^k is a tuple of two sets of robots: $s_i^k = (s_{i,wait}^k, s_{i,notify}^k)$, where $s_{i,wait}^k$ and $s_{i,notify}^k$ are the *wait-set* and *notify-set* of s_i^k, respectively. The *wait-set* of s_i^k is the set of robots that robot i must wait for at state q_i^k before satisfying its propositions and proceeding to the next state q_i^{k+1} in $r_i^\star$. The *notify-set* of s_i^k is the set of robots that robot i must notify as soon as it reaches state q_i^k. As we discussed earlier in Sec. 4.2, the optimal run $r_{team}^\star$ of the team and the individual optimal runs $r_i^\star, i = 1,\ldots,m$ of the robots are always in prefix-suffix form (Def. 3). Consequently, individual synchronization sequences s_i of the robots are also in prefix-suffix form.

Algorithm 2: SYNC-SEQ

Input: Individual optimal runs of the robots $\{r_1^\star,\ldots,r_m^\star\}$, Büchi automaton $\mathbf{B}_{\neg\phi}$ that corresponds to $\neg\phi$.
Output: Synchronization sequence for each robot $\{s_1,\ldots,s_m\}$.

1 $\mathscr{I} = \{1,\ldots,m\}$.
2 $beg \leftarrow$ beginning of suffix cycle.
3 $end \leftarrow$ end of suffix cycle.
4 Initialize each s_i so that all robots wait for and notify each other at every position of their runs.
5 **foreach** $k = 1,\ldots,end$ **do**
6 **foreach** $i \in \mathscr{I}$ **do**
7 **if** $k \neq 1$ *and* $k \neq beg$ **then**
8 **foreach** $j \in \mathscr{I} \setminus i$ **do**
9 Remove j from $s_{i,wait}^k$.
10 Remove i from $s_{j,notify}^k$.
11 Construct the TS $\mathbf{W}$ that generates every possible $\tilde{\omega}_{team}$.
12 **if** *the language of* $\mathbf{B}_{\neg\phi} \times \mathbf{W}$ *is not empty* **then**
13 Add j back to $s_{i,wait}^k$.
14 Add i back to $s_{j,wait}^k$.
15 Rest of each s_i is an infinite repetition of its suffix-cycle, i.e. $s_i^{beg},\ldots,s_i^{end}$.

Alg. 2 is essentially a loop (lines 5 – 14) that computes wait-sets and notify-sets for each position of the runs of the robots to guarantee correctness in the field. Initially, synchronization sequences are set so that the robots wait for and notify all other robots at every position of their runs (line 4). At line 7 of Alg. 2, if k is the first position of the runs, we do not modify this initial value of s_i^k. This ensures that all robots start executing their runs in a synchronized way. Also, if k is the beginning of

the suffix cycle, we again keep this initial value of s_i^k so that all robots synchronize with each other globally at the beginning of each suffix cycle. This lets us define a bound on optimality, *i.e.*, the value of the cost function (3) observed in the field, as given in Prop. 3. For all other positions of the runs, we try to shrink the wait-set and notify-set of each s_i^k so that communication effort is reduced while we can still guarantee correctness in the field (lines 9 – 14). To this end, we consider each one of the robots in robot i's k^{th} wait-set, *i.e.*, $s_{i,wait}^k$, one by one. After removing some robot j from the $s_{i,wait}^k$, we also remove robot i from $s_{j,notify}^k$ accordingly (lines 10–12). Then, given the $\underline{\rho_i}$ and $\overline{\rho_i}$ values of the robots, we construct the TS $\mathbf{W}$ that generates all possible words $\tilde{\omega}_{team}$ that can be observed in the field due to the uncertainties in the traveling times. Next, we check if the language of the product $\mathbf{B}_{\neg\phi} \times \mathbf{W}$ is empty or not, where $\mathbf{B}_{\neg\phi}$ is the Büchi automaton corresponding to the negation of the LTL formula ϕ (line 12). If the language of the product is empty, then robot i indeed does not need to wait for robot j at the k^{th} position of its run. Thus, we keep the new values of $s_{i,wait}^k$ and $s_{j,notify}^k$. Else, we restore $s_{i,wait}^k$ and notify-set of $s_{j,notify}^k$ to their previous values (lines 13–14) and proceed with the next robot in $s_{i,wait}^k$. Once every robot in $s_{i,wait}^k$ is considered, we proceed with the next robot in the team, and eventually next position of the run. Notice that, the synchronization sequences generated by Alg. 2 are free from any dead-locks as lines 9 – 10, and lines 13 – 14 ensure that if some robot i waits for robot j at position k, then robot j notifies robot i at position k, *i.e.*, $j \in s_{i,wait}^k \iff i \in s_{j,notify}^k \; \forall\, i,j,k$. As the synchronization sequences of the robots are in prefix-suffix form and the robots synchronize with each other globally at the beginning of each suffix (line 8), at line 15, we define the rest of each synchronization sequence as an infinite repetition of its first suffix-cycle that we have just generated. For a prefix of length p and a suffix cycle of length s, the complexity of Alg. 2 is $O((p+s)m^2L)$ where m is the number of robots and L is the complexity of constructing $W \times B_{\neg\phi}$ and checking emptiness of its language at each iteration. The synchronization protocol that the robots follow in the field is given in Alg. 3.

Algorithm 3: SYNC-RUN

Input: The run r_i and synchronization sequence s_i of robot i .

1 $k \leftarrow 0$.
2 **while** *True* **do**
3 **Notify** all robots in $s_{i,notify}^k$.
4 **Wait** until notification messages of all robots in $s_{i,wait}^k$ are received.
5 Make transition to r_i^{k+1} after satisfying the propositions at r_i^k.
6 $k \leftarrow k+1$.

The following proposition slightly extends the result of Prop. 4.5 in [19] by considering unequal lower and upper deviation values.

Proposition 3. *Suppose that each robot's deviation values are bounded by $\underline{\rho}$ and $\overline{\rho}$ where $\overline{\rho} \geq \underline{\rho} > 0$ (i.e., $\underline{\rho_i} \geq \underline{\rho}$ and $\overline{\rho_i} \leq \overline{\rho}$ for all robots i), and let $J(\mathbb{T}^\pi)$ be the cost of the planned robot paths. Then, if the robots follow the protocol given in Alg. 3 the field value of the cost satisfies*

$$\overline{J(\mathbb{T}^\pi)} \leq J(\mathbb{T}^\pi)\overline{\rho} + d_s(\overline{\rho} - \underline{\rho})$$

where d_s is the planned duration of the suffix cycle.

Example 1 Revisited. *For the example we have shown throughout this section, we obtain the following individual optimal runs and synchronization sequences.*

$\mathbb{T}$	*0*	*2*	*3*	*4*	*5*	*6*	...
$r_1^\star$	*a*	*b*	*ba1*	*a*	*ab1*	*b*	...
s_1	$(\{2\},\{2\})$	$(\{\},\{\})$	$(\{2\},\{2\})$	$(\{\},\{\})$	$(\{\},\{\})$	$(\{\},\{\})$	...
$\mathscr{L}_1(.)$		p_1, π				p_1, π	...
$r_2^\star$	*a*	*b*	*c*	*b*	*c*	*b*	...
s_2	$(\{1\},\{1\})$	$(\{\},\{\})$	$(\{1\},\{1\})$	$(\{\},\{\})$	$(\{\},\{\})$	$(\{\},\{\})$	...
$\mathscr{L}_2(.)$		p_2, π	p_3	p_2, π	p_3	p_2, π	...

In a line corresponding to a synchronization sequence s_i, first and second elements of the tuple at position k are $s_{i,wait}^k$ and $s_{i,notify}^k$, respectively.

We finally summarize our approach in Alg. 4 and show that this algorithm indeed solves Prob. 1.

Proposition 4. *Alg. 4 solves Prob. 1.*

Proof. Note that Alg. 4 combines all steps outlined in this section. The planned word ω_{team} generated by the entire team satisfies ϕ, and minimizes (3), as shown in [14]. The synchronization sequences guarantee correctness in the field by ensuring that the $\tilde{\omega}_{team}$ generated in the field never violates ϕ for given deviation values. Therefore, $\{r_1^\star, \ldots, r_m^\star\}$ and $\{s_1, \ldots, s_m\}$ as obtained from Alg. 4 is a solution to Prob. 1. ■

Algorithm 4: ROBUST-MULTI-ROBOT-OPTIMAL-RUN

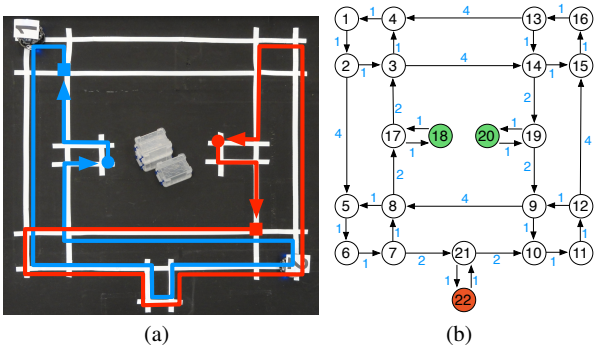
Input: m $\mathbf{T}_i$'s, corresponding deviation values, and a global LTL specification ϕ of the form (2).

Output: A set of optimal runs $\{r_1^\star, \ldots, r_m^\star\}$ that satisfies ϕ and minimizes (3), a set of synchronization sequences $\{s_1, \ldots, s_m\}$ that guarantees correctness in the field, and the bound on the performance of the team in the field.

1 Construct the team transition system $\mathbf{T}$ using Alg. 1.
2 Find an optimal run $r_{team}^\star$ on $\mathbf{T}$ using OPTIMAL-RUN [14].
3 Insert new traveling states to TSs according to $r_{team}^\star$ (See. Sec. 4.2).
4 Obtain individual runs $\{r_1^\star, \ldots, r_m^\star\}$ using Def. 6.
5 Generate synchronization sequences $\{s_1, \ldots, s_m\}$ using SYNC-SEQ (Alg. 2).
6 Find the bound on optimality as given in Prop. 3.

5 Implementation and Case-Study

We implemented Alg. 4 as a python module and used it to plan optimal satisfying paths and synchronization sequences for the scenario that we consider in this section. Our experimental platform (Fig. 5(a)) is a road network comprising roads, intersections and task locations. Fig. 5(b) illustrates the model that captures the motion of the robots on this platform, where 1 time unit corresponds to 1.574 seconds.



(a) (b)

Fig. 5 Fig. (a) shows our experimental platform. The squares and the circles on the trajectories of the robots represent the beginning of the suffix cycle and sync. points, respectively. Fig. (b) illustrates the TS that models the robots. The green and red regions are data gather and upload locations, respectively.

In our experiments, we consider a persistent surveillance task involving two robots with deviation values $\overline{\rho_1} = \overline{\rho_2} = 1.05$ and $\underline{\rho_1} = \underline{\rho_2} = 0.95$. The building in the middle of the platform in Fig. 5(a) is our surveillance target. We define the set of propositions $\Pi = \{\texttt{R1Gather18}, \texttt{R1Gather20}, \texttt{R2Gather18}, \texttt{R2Gather20}, \texttt{R1Gather}, \texttt{R2Gather}, \texttt{R1Upload}, \texttt{R2Upload}, \texttt{Gather}\}$ and assign them as $\mathscr{L}_1(18) = \{\texttt{R1Gather18}, \texttt{R1Gather}, \texttt{Gather}\}$, $\mathscr{L}_2(18) = \{\texttt{R2Gather18}, \texttt{R2Gather}, \texttt{Gather}\}$, $\mathscr{L}_1(20) = \{\texttt{R1Gather20}, \texttt{R1Gather}, \texttt{Gather}\}$, $\mathscr{L}_2(20) = \{\texttt{R2Gather20}, \texttt{R2Gather}, \texttt{Gather}\}$, $\mathscr{L}_1(22) = \{\texttt{R1Upload}\}$ and $\mathscr{L}_2(22) = \{\texttt{R2Upload}\}$. The main objective is to keep gathering data while minimizing the maximum time between successive gathers. We require the robots to gather data in a synchronous manner at data gather locations 18 and 20 while ensuring that they do not gather data at the same place at the same time. We also require the robots to upload their data at upload location 22 before their next data gather. We express these requirements in LTL in the form of (2) as

$$\begin{aligned}\phi =& \mathbf{G}\left(\texttt{R1gather} \Rightarrow \mathbf{X}(\neg\texttt{R1gather}\ \mathscr{U}\ \texttt{R1upload})\right) \wedge \mathbf{G}\left(\texttt{R2gather} \Rightarrow\right.\\ & \left.\mathbf{X}(\neg\texttt{R2gather}\ \mathscr{U}\ \texttt{R2upload})\right) \wedge \mathbf{G}\left((\texttt{R1Gather18} \Rightarrow \texttt{R2Gather20}) \wedge\right.\\ & (\texttt{R1gather20} \Rightarrow \texttt{R2gather18}) \wedge (\texttt{R2gather18} \Rightarrow \texttt{R1gather20}) \wedge\\ & \left.(\texttt{R2gather20} \Rightarrow \texttt{R1gather18})\right) \wedge \mathbf{GF}\,\texttt{Gather},\end{aligned}$$

where Gather is set as the optimizing proposition.

Fig. 5(a) illustrates the solution we obtain using our algorithm. Using an iMac i5 quad-core computer, it took our implementation 10 minutes to compute the optimal

runs and synchronization sequences of the robots. The planned value of the cost function was 44.072 seconds (28 time units) with an upper bound of 50.683 seconds (32.2 time units) seconds. We deployed our robots in our experimental platform to demonstrate and verify the result. The maximum time between any two successive data uploads was measured to be 48 seconds. The video available at `http://hyness.bu.edu/dars_2012.mov` demonstrates the execution of this run by the robots.

6 Conclusion

In this paper we presented an automated method for planning optimal paths for a robotic team subject to temporal logic constraints expressed in LTL. The robots that we consider have bounded non-deterministic traveling times characterized by robot specific deviation values. We first compute a set of optimal satisfying paths for the members of the team. Then, leveraging the communication capabilities of the robots, we also compute a set of synchronization sequences for each robot to ensure that the LTL formula is never violated during deployment. Our experiments show that our method has practical value in scenarios where the traveling times of the robots during deployment deviate from those used in planning.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
2. Bianco, A., Alfaro, L.D.: Model checking of probabilistic and nondeterministic systems. pp. 499–513. Springer-Verlag (1995)
3. Clarke, E.M., Peled, D., Grumberg, O.: Model checking. MIT Press (1999)
4. Ding, X.C., Smith, S.L., Belta, C., Rus, D.: Mdp optimal control under temporal logic constraints. In: IEEE Conf. on Decision and Control, pp. 532–538. Orlando, FL (2011)
5. Emerson, E.A.: Temporal and modal logic. In: J. van Leeuwen (ed.) Handbook of Theoretical Computer Science: Formal Models and Semantics, vol. B, pp. 995–1072. North-Holland Pub. Co./MIT Press (1990)
6. Hopcroft, J., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison Wesley (2007)
7. Karaman, S., Frazzoli, E.: Complex mission optimization for multiple-uavs using linear temporal logic. In: American Control Conference, pp. 2003–2009. Seattle, WA (2008)
8. Karaman, S., Frazzoli, E.: Vehicle routing problem with metric temporal logic specifications. In: IEEE Conf. on Decision and Control, pp. 3953–3958. Cancún, México (2008)
9. Kloetzer, M., Belta, C.: Automatic deployment of distributed teams of robots from temporal logic specifications. IEEE Transactions on Robotics **26**(1), 48–61 (2010)
10. Kress-Gazit, H., Fainekos, G., Pappas, G.J.: Where's waldo? sensor-based temporal logic motion planning. In: IEEE Intl. Conf. on Robotics and Automation, pp. 3116–3121 (2007)
11. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic symbolic model checking with prism: A hybrid approach. In: International Journal on Software Tools for Technology Transfer, pp. 52–66. Springer (2002)
12. Milner, R.: Communication and concurrency. Prentice-Hall (1989)

13. M.Kloetzer, Belta, C.: Dealing with non-determinism in symbolic control. In: M. Egerstedt, B. Mishra (eds.) Hybrid Systems: Computation and Control: 11th International Workshop, Lecture Notes in Computer Science, pp. 287–300. Springer Berlin / Heidelberg (2008)
14. Smith, S.L., Tůmová, J., Belta, C., Rus, D.: Optimal path planning for surveillance with temporal logic constraints. Intl. Journal of Robotics Research **30**(14), 1695–1708 (2011)
15. Tabuada, P., Pappas, G.J.: Linear time logic control of discrete-time linear systems. IEEE Transactions on Automatic Control **51**(12), 1862–1877 (2006)
16. Thomas, W.: Infinite games and verification. In: Intl. Conf. on Computer Aided Verification, pp. 58–64 (2002)
17. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM (2001)
18. Tumova, J., Yordanov, B., Belta, C., Cerna, I., Barnat, J.: A symbolic approach to controlling piecewise affine systems. In: IEEE Conf. on Decision and Control, pp. 4230–4235. Atlanta, GA (2010)
19. Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C.: Robust multi-robot optimal path planning with temporal logic constraints. In: IEEE Intl. Conf. on Robotics and Automation, pp. 4693–4698. St. Paul, MN, USA (2012)
20. Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C., Rus, D.: Optimal path planning for surveillance with temporal logic constraints. In: IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems, pp. 3087–3092. San Francisco, CA, USA (2011)
21. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon control for temporal logic specifications. In: Hybrid systems: Computation and Control, pp. 101–110. Stockholm, Sweden (2010)