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. 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 temporal relations. 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

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

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

M.A. Hsieh and G. Chirikjian (eds.), *Distributed Autonomous Robotic Systems,*
Springer Tracts in Advanced Robotics 104,
DOI: 10.1007/978-3-642-55146-8_24,

complex mission requirements in path planning and control synthesis problems only recently [14, 12, 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 [14, 17]. Particularly, given the system model and some temporal logic formula, satisfying paths and corresponding control strategies can be computed automatically through a search of the state space for deterministic [9], non-deterministic [15, 17, 12, 10] and probabilistic systems [2, 11, 4].

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) [16] 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 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 [13]. Next, we extended this approach to multi-robot teams by utilizing an abstraction based on timed automata [20]. 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. Then, we proposed a robust method that could accomodate uncertainties in the traveling times of robots with limited communication capabilities [19]. 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 uses these two methods along with the OPTIMAL-RUN algorithm previously proposed in [13] to solve the multi-robot optimal path planning problem with robustness guarantees.

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. In Sec. 6, we conclude with final remarks. Due to page constraints we omit the proofs of all results. The proofs are contained in an extended version available online [18].

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) $w_{\mathrm{T}} : \delta_T \to \mathbb{N}_{>0}$ is a map that assigns a positive integer 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 as Linear Temporal Logic (LTL) formulas over Π using the standard syntax and semantics defined in [1]. We follow the literal notation for temporal operators ($\mathbf{G}, \mathbf{F}, \mathbf{X}, \mathscr{U}$). We say a run r_{T} satisfies ϕ if and only if the word generated by r_{T} satisfies ϕ.

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 $\mathscr{F}_B$ infinitely many times. For any LTL formula ϕ over a set Π, one can construct a Büchi automaton with input alphabet $\Sigma_B = 2^\Pi$ 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. 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 logic constraints for robots with uncertain, but bounded traveling times. Let $\mathscr{E} = (V, \rightarrow_{\mathscr{E}})$ be a directed 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 and $\rightarrow_{\mathscr{E}}$ is the corresponding adjacency relation.

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 modeled 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 $w_i(q,q')$ gives the *nominal* time for robot i to go from vertex q to q', which we assume to be a positive integer. However, due to the uncertainty in the traveling times of the robots, the *actual* time it takes for robot i to go from q to q', 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 predetermined lower and upper *deviation values* of robot i that satisfy $\overline{0} < \underline{\rho_i} \leq 1 \leq \overline{\rho_i}$. 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 uncertain. In the following, we use the expression "*in the field*" to refer to the model with uncertain traveling times, and use x and $\tilde{x}$ to denote the *nominal* and *actual* values 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 between successive completions of this task must be minimized. For instance, in a persistent surveillance 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 between successive data gatherings is minimized. Consequently, we assume that there is an *optimizing proposition* $\pi \in \Pi$ that corresponds to this

repeating task and consider multi-robot missions specified by LTL formulae of the form

$$\phi := \varphi \wedge \mathbf{GF}\pi, \tag{1}$$

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 ϕ and minimize the maximum time 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 interpret 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 $\mathbb{T}^\pi(k)$ is the time instance when π is satisfied for the k^{th} time by the team and define the cost function

$$J(\mathbb{T}^\pi) = \limsup_{i \to +\infty} (\mathbb{T}^\pi(i+1) - \mathbb{T}^\pi(i)). \tag{2}$$

The form of the cost function given in Eq. (2) is motivated by persistent surveillance missions, where one is interested in the long-term behavior of the team. Given a sequence $\mathbb{T}^\pi$ corresponding to some run of the team, the cost function in Eq. (2) captures the maximum time between satisfying instances of π once the team behavior reaches a steady-state, which we achieve in finite time as we will discuss in Sec. 4.2. Thus, the problem becomes that of finding an optimal run of the team that satisfies ϕ and minimizes (2). 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. (2) 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 (1), synthesize individual runs r_i and synchronization sequences s_i for each robot such that $\mathbb{T}^\pi$ minimizes the cost function (2), and $\tilde{\omega}_{team}$, *i.e.*, the word observed in the field, satisfies ϕ.

Note that our aim in Prob. 1 is to find a run that is optimal under nominal values while ensuring that ϕ is never violated in the field. Since $\tilde{\mathbb{T}}^\pi$, *i.e.*, the sequence of instants at which π is satisfied during deployment, 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.

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 $\mathscr{Q}_\mathrm{R}$ is the set of states of the form (q,r) such that 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 r is a tuple of clock values $(x_1,\ldots,x_m)$ where $x_i \in \mathbb{N}$ denotes the time elapsed since robot i left state q_i. $q^0_\mathrm{R} \subseteq \mathscr{Q}_\mathrm{R}$ 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 $(q^0_i,q'_i) \in \delta_i$. δ_R is the transition relation such that a transition from (q,r) to (q',r') where the i^{th} state pair $q_iq'_i$ and the i^{th} clock value x_i in (q,r) change to $q'_iq''_i$ and x'_i in (q',r') exists if and only if $(q_i,q'_i),(q'_i,q''_i) \in \delta_i$ for all changed state pairs, $w_i(q_i,q'_i) - x_i$ of all changed state pairs are equal to each other and are strictly smaller than those of unchanged state pairs, and for all changed state pairs, the 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'. $\Pi_\mathrm{R} = \cup_{i=1}^m \Pi_i$ is the set of propositions. $\mathscr{L}_\mathrm{R} : \mathscr{Q}_\mathrm{R} \to 2^{\Pi_\mathrm{R}}$ is a map giving the set of atomic propositions satisfied in a state. For a state (q,r), $\mathscr{L}_\mathrm{R}((q,r)) = \cup_{i|x_i=0}\mathscr{L}_i(q_i)$. $w_\mathrm{R} : \delta_R \to \mathbb{N}_{>0}$ is a map that assigns a positive integer 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 $q'_iq''_i$ 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* $\phi := \mathbf{G}(\mathrm{p}_1 \Rightarrow \mathbf{X}(\neg \mathrm{p}_1 \mathscr{U} \mathrm{p}_3)) \wedge \mathbf{GF}\pi$, *where* $\Pi_1 = \{\mathrm{p}_1, \pi\}$, $\Pi_2 = \{\mathrm{p}_2, \mathrm{p}_3, \pi\}$, *and* $\Pi = \{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3, \pi\}$. *We also have* $\overline{\rho_1} = \overline{\rho_2} = 1.1$ *and* $\underline{\rho_1} = \underline{\rho_2} = 0.9$. *The region automaton* $\mathbf{R}$ *that models the robots is given in Fig. 2.*

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.

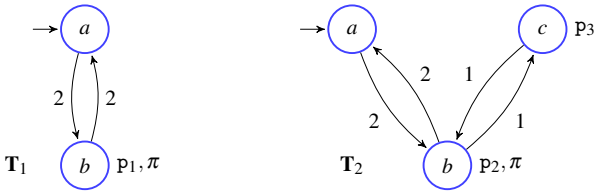

Fig. 1 Transition systems $\mathbf{T}_1$ and $\mathbf{T}_2$ of two robots in an environment with three vertices. The states of the transition systems correspond to vertices $\{a,b,c\}$ and the edges represent the motion capabilities of each robot. The weights of the edges represent the traveling times 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.

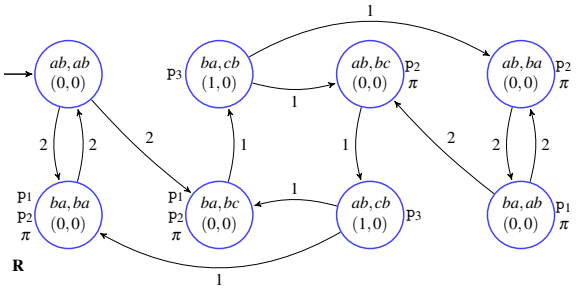

Fig. 2 The finite state region automaton capturing the joint behavior of two robots in 9 states. In a circle representing a state (q,r), the first line is q and the second line is r.

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 **R** *is constructed (Def. 4) [20]. (3) For each s corresponding to a given t, we define the corresponding transitions originating from t in* **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* **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 **T** obtained using Proc. 1 and the corresponding region automaton **R** are bisimulation equivalent, *i.e.*, there exists a binary relation between the states and the transitions of **R** and **T** such that they behave in the same way [1].

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

Example 1 Revisited. *Using $\mathscr{R}$, we construct* **T** *(Fig. 3) that captures the joint asynchronous behavior of the team in 6 states whereas the corresponding region automaton* **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.*

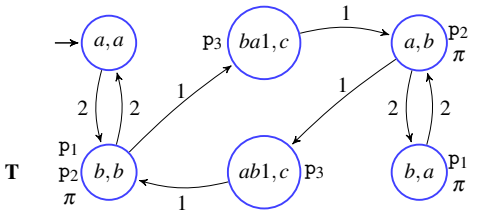

Fig. 3 The team transition system capturing the joint behavior of two robots in 6 states

In [20] we showed that the number of states $|\mathscr{Q}_\mathrm{R}|$ of the region automaton **R** that models m robots $\mathbf{T}_i, i = 1,\ldots,m$ 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 **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* **T** *is bounded by $\prod_{i=1}^m |\mathscr{Q}_i| + (W - 1)\prod_{i=1}^m |\delta_i|$ where W is the largest edge weight in all TS's.*

Finally, we note that the states of **T** correspond to the instants where at least one robot 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 **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. A detailed discussion on Alg. 1 can be found in [18].

Algorithm 1. CONSTRUCT-TEAM-TS

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

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

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

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

4.2 Obtaining Optimal Satisfying Runs and Transition Systems with Traveling States

After constructing $\mathbf{T}$ that models the team, we use OPTIMAL-RUN from [13] to obtain an optimal satisfying run $r_{team}^{\star}$ on $\mathbf{T}$ that minimizes the cost function (2) and satisfies ϕ. The optimal run $r_{team}^{\star}$ 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. *Running Alg.* OPTIMAL-RUN *[13] 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 with the prefix* $(a,a),(b,b)$ *and the suffix cycle* $(ba1,c),(a,b),(ab1,c),(b,b)$*, which will be repeated indefinitely. The cost as defined in* (2) *is* $J(\mathbb{T}^{\pi}) = 2$.

Since $\mathbf{T}$ captures the asynchronous motion of the robots, the optimal satisfying run $r_{team}^{\star}$ 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 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 $q_i q'_i x_i$. First, we construct the set $\mathscr{S} = \{(i, q^k[i]) \mid q^k[i] = q_i q'_i x_i \, \forall k, i\}$ of all traveling states that appear in $r^\star_{team}$. Elements of $\mathscr{S}$ are tuples where the second element is a traveling state and the first element gives the transition system this new traveling state will be added to. Next, we construct the set $\mathscr{T} = \{(i, (q^k[i], q^{k+1}[i]), x) \mid ((i, q^k[i]) \in \mathscr{S}) \vee ((i, q^{k+1}[i]) \in \mathscr{S}), x = w_{\mathrm{T}}(q^k, q^{k+1}) \, \forall k, i\}$ of all transitions that involve any of the traveling states in $r^\star_{team}$. Elements of $\mathscr{T}$ are triplets where the second element is a transition, the third element is the weight of this transition, and the first element shows the transition system that this new transition will be added to. Then, we add the traveling states in $\mathscr{S}$ and the transitions in $\mathscr{T}$ to their corresponding transition systems. 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 $q_i^k = q^k[i]$ where $q^k[i]$ is the i^{th} element of tuple q^k.

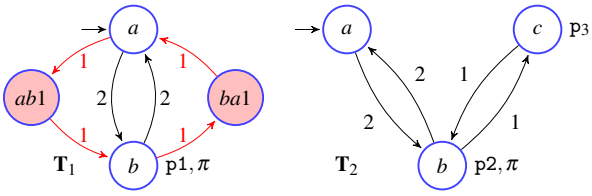

Fig. 4 Transition systems with new traveling states that correspond to the optimal run $r^\star_{team}$ that we computed for Ex. 1. The new traveling states and transitions of $\mathbf{T}_1$ are highlighted in red.

Example 1 Revisited. *For this example, we have* $\mathscr{S} = \{(1, ab1), (1, ba1)\}$ *and* $\mathscr{T} = \{(1, (a, ab1), 1), (1, (ab1, b), 1), (1, (b, ba1), 1), (1, (ba1, a), 1)\}$. *Fig. 4 illustrates the corresponding TSs with new traveling states and transitions highlighted in red. Using Def. 6, we obtain the runs of the individual robots as* $r^\star_1 = a, b, ba1, a, ab1, b, ba1, a, ab1, \ldots$ *and* $r^\star_2 = a, b, c, b, c, b, c, b, c, \ldots$.

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: $s_i^k = (s^k_{i,wait}, s^k_{i,notify})$, where $s^k_{i,wait}$ and $s^k_{i,notify}$ 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. A detailed discussion on Alg. 2 can be found in [18].

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  I = {1,...,m}.
2  beg ← beginning of suffix cycle.
3  end ← 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 = 0,...,end do
6      foreach i ∈ I do
7          if k ≠ 0 and k ≠ beg then
8              foreach j ∈ I \ i do
9                  Remove j from s^k_{i,wait}.
10                 Remove i from s^k_{j,notify}.
11                 Construct the TS W that generates every possible ω̃_team.
12                 if the language of B_{¬φ} × W is not empty then
13                     Add j back to s^k_{i,wait}.
14                     Add i back to s^k_{j,notify}.
15 Rest of each s_i is an infinite repetition of its suffix-cycle, i.e. s_i^beg,...,s_i^end.
```

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* $\overline{\rho} \geq 1 \geq \underline{\rho} > 0$ *(i.e.,* $\underline{\rho_i} \geq \underline{\rho}$ *and* $\overline{\rho_i} \leq \overline{\rho}$ *for each robot i). Let* $J(\mathbb{T}^\pi)$ *be the cost of the planned robot paths and let* $J(\tilde{\mathbb{T}}^\pi)$ *be the actual value of the cost observed during deployment. Then, if the robots use the synchronization sequences generated by Alg. 2, the field value of the cost satisfies* $J(\tilde{\mathbb{T}}^\pi) \leq J(\mathbb{T}^\pi)\overline{\rho} + d_s(\overline{\rho} - \underline{\rho})$ *where* d_s *is the planned duration of the suffix cycle.*

Example 1 Revisited. *Using Alg. 2, we obtain the following individual synchronization sequences:* $s_1 = (\{2\},\{2\}),(\{\},\{\}),(\{2\},\{2\}),(\{\},\{\}),(\{\},\{\}),(\{\},\{\}),\ldots$, *and* $s_2 = (\{1\},\{1\}),(\{\},\{\}),(\{1\},\{1\}),(\{\},\{\}),(\{\},\{\}),(\{\},\{\}),\ldots$. *The elements*

of the k^{th} pair in the synchronization sequences correspond to $s^k_{i,wait}$ and $s^k_{i,notify}$, respectively. Also, from Prop. 3, the field value of the cost function is bounded from above by 3 for $\overline{\rho_1} = \overline{\rho_2} = 1.1$ and $\underline{\rho_1} = \underline{\rho_2} = 0.9$.

We finally summarize our approach in Alg. 3 and show that this algorithm indeed solves Prob. 1. We discuss the complexity of our approach in Rem. 2.

Proposition 4. *Alg. 3 solves Prob. 1.*

Remark 2 (Computational Complexity). *The main drawback of our approach is its computational complexity, which is exponential in the number of robots (due to generation of the team transition system and the synchronization sequences) and in the length of the LTL formula (due to the conversion to a Büchi automaton). This cost, however, is justified by the globally optimal runs that our approach computes, and in practice, we can solve fairly large problems.*

Algorithm 3. ROBUST-MULTI-ROBOT-OPTIMAL-RUN

Input: m transition systems $\mathbf{T}_i, i = 1,\ldots,m$, corresponding deviation values, and a global LTL specification ϕ of the form (1).

Output: A set of optimal runs $\{r^\star_1,\ldots,r^\star_m\}$ that satisfies ϕ and minimizes (2), 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 $r^\star_{team}$ on $\mathbf{T}$ using OPTIMAL-RUN [13].
3 Insert new traveling states to TSs according to $r^\star_{team}$ (See. Sec. 4.2).
4 Obtain individual runs $\{r^\star_1,\ldots,r^\star_m\}$ using Def. 6.
5 Generate synchronization sequences $\{s_1,\ldots,s_m\}$ using Alg. 2.
6 Find the bound on optimality as given in Prop. 3.

5 Implementation and Case-Study

We implemented Alg. 3 as a python module (available at `http://hyness.bu.edu/lomap/`) 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.

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 $\Pi = \{$`R1Gather18`, `R1Gather20`, `R2Gather18`, `R2Gather20`, `R1Gather`, `R2Gather`, `R1Upload`, `R2Upload`, `Gather`$\}$ and assign them as $\mathscr{L}_1(18) = \{$`Gather`, `R1Gather18`, `R1Gather`$\}$, $\mathscr{L}_2(18) = \{$`Gather`, `R2Gather18`, `R2Gather`$\}$, $\mathscr{L}_1(20) = \{$`Gather`, `R1Gather20`, `R1Gather`$\}$, $\mathscr{L}_2(20) = \{$`Gather`, `R2Gather20`, `R2Gather`$\}$, $\mathscr{L}_1(22) = \{$`R1Upload`$\}$ and $\mathscr{L}_2(22) = \{$`R2Upload`$\}$.

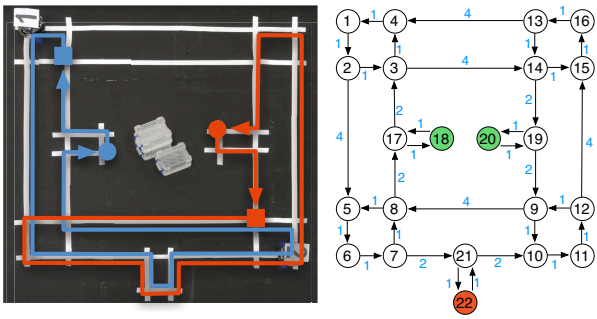

Fig. 5 Left figure shows our experimental platform. The squares and the circles on the trajectories of the robots represent the beginning of the suffix cycle and sync. points, respectively. Right figure illustrates the TS that models the robots. The green and red regions are data gather and upload locations,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

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 (1) as

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

where `Gather` is set as the optimizing proposition.

Fig. 5(a) illustrates the solution which took our algorithm approximately 20 seconds to compute on an iMac i5 quad-core computer. 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/lomap/dars2012.mov` demonstrates the execution of this run by the robots.

6 Conclusion

In this paper we present an automated method for planning optimal paths for a robotic team subject to a Linear Temporal Logic formula. The robots that we consider have bounded non-deterministic traveling times. We first compute a set of optimal satisfying paths for the members of the team. Then, we compute synchronization sequences for the robots to guarantee correctness 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.

Acknowledgements. This work was supported in part by ONR MURI N00014-09-1051, NSF CNS-0834260, and NSERC.

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. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026, pp. 499–513. Springer, Heidelberg (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, Orlando, FL, pp. 532–538 (2011)
5. Emerson, E.A.: Temporal and Modal Logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science: Formal Models and Semantics, 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, Seattle, WA, pp. 2003–2009 (2008)
8. Karaman, S., Frazzoli, E.: Vehicle Routing Problem with Metric Temporal Logic Specifications. In: IEEE Conf. on Decision and Control, Cancún, México, pp. 3953–3958 (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. International Journal on Software Tools for Technology Transfer, 52–66 (2002)
12. Kloetzer, M., Belta, C.: Dealing with Non-Determinism in Symbolic Control. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 287–300. Springer, Heidelberg (2008)
13. 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)
14. Tabuada, P., Pappas, G.J.: Linear Time Logic Control of Discrete-Time Linear Systems. IEEE Transactions on Automatic Control 51(12), 1862–1877 (2006)
15. Thomas, W.: Infinite Games and Verification. In: Intl. Conf. on Computer Aided Verification, pp. 58–64 (2002)
16. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM (2001)
17. 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, Atlanta, GA, pp. 4230–4235 (2010)
18. Ulusoy, A., Smith, S.L., Belta, C.: Optimal Multi-Robot Path Planning with LTL Constraints: Guaranteeing Correctness Through Synchronization (2012), `http://arxiv.org/abs/1207.2415`

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, St. Paul, MN, USA, pp. 4693–4698 (2012)
20. Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C., Rus, D.: Optimal Multi-Robot Path Planning with Temporal Logic Constraints. In: IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems, San Francisco, CA, USA, pp. 3087–3092 (2011)
21. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding Horizon Control for Temporal Logic Specifications. In: Hybrid Systems: Computation and Control, Stockholm, Sweden, pp. 101–110 (2010)