2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Robust Multi-Robot Optimal Path Planning with Temporal Logic Constraints

Alphan Ulusoy⋆ Stephen L. Smith† Xu Chu Ding‡ Calin Belta⋆

***Abstract*— In this paper we present a method for automatically planning robust optimal paths for a group of robots that satisfy a common high level mission specification. Each robot's motion in the environment is modeled as a weighted transition system, and the mission is given as a Linear Temporal Logic (LTL) formula over a set of propositions satisfied by the regions of the environment. In addition, an optimizing proposition must repeatedly be satisfied. The goal is to minimize the maximum time between satisfying instances of the optimizing proposition while ensuring that the LTL formula is satisfied even with uncertainty in the robots' traveling times. We characterize a class of LTL formulas that are robust to robot timing errors, for which we generate optimal paths if no timing errors are present, and we present bounds on the deviation from the optimal values in the presence of errors. We implement and experimentally evaluate our method considering a persistent monitoring task in a road network environment.**

I. Introduction

Temporal logics provide a powerful high-level language for specifying complex missions for groups of robots [1], [2], [3], [4], [5]. Their power lies in the wealth of tools from model checking [6], [7], which can be leveraged to generate robot paths satisfying desired mission specifications or produce counter-examples which prove that the mission is not possible. However, in robotics the goal is typically to plan paths that complete a mission in an optimal manner. In our earlier work [8] we considered Linear Temporal Logic (LTL) specifications, and a particular form of cost function, and provided a method for computing optimal robot paths for a single robot. We then extended this approach to multi-robot problems by utilizing timed automata [9].

The main difficulty in moving from a single robot to multiple robots is in allowing the robots to move asynchronously. In [10], the authors propose a method for decentralized motion of multiple robots by restricting the robots to take transitions synchronously. Once every robot has completed a transition, the robots synchronously make the next transition. While such an approach is effective for satisfying the LTL formula, it does not lend itself to optimizing the robot motion, since synchronization takes extra time. In [9] we approached this problem by modeling the group of robots in the environment as a timed automaton. This method allowed us to represent the relative position between robots which is necessary for optimizing the robot motion. After providing a bisimulation [11] of the infinite-dimensional timed automaton to a finite transition system we applied our results from [8] to compute an optimal run.

However, asynchronous motion of robots introduces issues in the robustness, and thus implementability of the multi-robot paths. Timed-automata rely heavily on the assumption that the clocks (or for robots, the speeds), are known exactly. If the clocks drift by even an infinitesimally small amount, then the reachability analysis developed for timed-automata is no longer correct [12], [13]. The intuition behind this is that if the robot speeds deviate from those used for planning, then robots can complete tasks in a different order than was specified in the plan. This switch in the order of events may result in the violation of the global mission specification.

The contribution of this paper is to present a method for generating paths for a group of robots satisfying general LTL formulas, which are robust to uncertainties in the speeds of robots, and which perform within a known bound of the optimal value. We focus on minimizing a cost function that captures the maximum time between satisfying instances of an *optimizing proposition*. The cost is motivated by problems in persistent monitoring and in pickup and delivery problems. We characterize the class of LTL formulas for which a robust solution exists. The characterization relies on using the concept of trace-closedness, which was first applied in multi-robot planning in [14]. For formulas in this class, we utilize a similar method as in [9] to generate robot plans. We then propose periodic synchronization of the robots to optimize the cost function in the presence of timing errors. We provide results from an implementation on a robotic test-bed, which shows the utility of the approach in practice. For simplicity of presentation, we assume that each robot moves among the vertices of an environment modeled as a graph. However, by using feedback controllers for facet reachability in polytopes [15] our method can be extended to robots with continuous dynamics traversing an environment with polytopic partitions.

Due to page constraints we omit all proofs of all results. An extended version of this paper can be found in [16].

II. Preliminaries

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. Moreover, we define $\Sigma^{\infty} = \Sigma^* \cup \Sigma^{\omega}$ and denote the empty string by $\emptyset$.

Definition II.1 (Transition System). *A (weighted) transition system (TS) is a tuple* $\mathbf{T} := (\mathcal{Q}_{\mathrm{T}}, q^0_{\mathrm{T}}, \delta_{\mathrm{T}}, \Pi_{\mathrm{T}}, \mathcal{L}_{\mathrm{T}}, w_{\mathrm{T}})$, *where (i)* $\mathcal{Q}_{\mathrm{T}}$ *is a finite set of states; (ii)* $q^0_{\mathrm{T}} \in \mathcal{Q}_{\mathrm{T}}$ *is the initial state; (iii)* $\delta_{\mathrm{T}} \subseteq \mathcal{Q}_{\mathrm{T}} \times \mathcal{Q}_{\mathrm{T}}$ *is the transition relation;*

This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, NSF CNS-0834260, NSF CNS-1035588 and NSERC.

⋆ Hybrid and Networked Systems Laboratory, Boston University, Boston, MA 02215 (alphan@bu.edu, cbelta@bu.edu)

† Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (stephen.smith@uwaterloo.ca)

‡ Embedded Systems and Networks, United Technologies Research Center, East Hartford, CT 06108 (dingx@utrc.utc.com)

(iv) Π_{T} *is a finite set of atomic propositions (observations); (v)* $\mathcal{L}_{\mathrm{T}} : \mathcal{Q}_{\mathrm{T}} \rightarrow 2^{\Pi_{\mathrm{T}}}$ *is a map giving the set of atomic propositions satisfied in a state; (vi)* $w_{\mathrm{T}} : \delta_T \rightarrow \mathbb{R}_{>0}$ *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 $q^0 = q_T^0$, $q^k \in \mathcal{Q}_{\mathrm{T}}$ and $(q^k, q^{k+1}) \in \delta_{\mathrm{T}}$ for all $k \geq 0$. A run generates an infinite word $\omega_{\mathrm{T}} = \mathcal{L}(q^0)\mathcal{L}(q^1)\ldots$ where $\mathcal{L}(q^k)$ is the set of atomic propositions satisfied at state q^k. To specify the mission of the group, we use LTL formulas over Π. We use the standard syntax and semantics defined in [17] and we follow the literal notation for the temporal operators $(\mathbf{G}, \mathbf{F}, \mathbf{X}, \mathcal{U})$. We say a run r_{T} satisfies ϕ if and only if the word generated by r_{T} satisfies ϕ.

Definition II.2 (Büchi Automaton). *A Büchi automaton is a tuple* $\mathbf{B} := (\mathcal{S}_{\mathrm{B}}, \mathcal{S}_{\mathrm{B}}^0, \Sigma_{\mathrm{B}}, \delta_{\mathrm{B}}, \mathcal{F}_{\mathrm{B}})$*, consisting of (i) a finite set of states* $\mathcal{S}_{\mathrm{B}}$*; (ii) a set of initial states* $\mathcal{S}_{\mathrm{B}}^0 \subseteq \mathcal{S}_{\mathrm{B}}$*; (iii) an input alphabet* Σ_{B}*; (iv) a non-deterministic transition relation* $\delta_{\mathrm{B}} \subseteq \mathcal{S}_{\mathrm{B}} \times \Sigma_{\mathrm{B}} \times \mathcal{S}_{\mathrm{B}}$*; (v) a set of accepting (final) states* $\mathcal{F}_{\mathrm{B}} \subseteq \mathcal{S}_{\mathrm{B}}$.

A *run* of $\mathbf{B}$ over an input word $\omega = \omega^0\omega^1 \ldots$ is a sequence $r_{\mathrm{B}} = s^0 s^1 \ldots$, such that $s^0 \in \mathcal{S}_{\mathrm{B}}^0$, and $(s^k, \omega^k, s^{k+1}) \in \delta_{\mathrm{B}}$, 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 $\mathcal{F}_{\mathrm{B}}$ 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 II.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.*

Definition II.4 (Language). *The set of all the words accepted by an automaton* $\mathbf{B}$ *is called the language recognized by the automaton and is denoted by* L_{B}.

Definition II.5 (Distribution). *Given a set* Σ*, the collection of subsets* $\Sigma_i \subseteq \Sigma$, $\forall\, i = 1, \ldots, m$ *is called a distribution of* Σ *if* $\cup_{i=1}^m \Sigma_i = \Sigma$.

Definition II.6 (Projection). *For a word* $\omega \in \Sigma^{\infty}$ *and a subset* $\Sigma_i \subseteq \Sigma$, $\omega \upharpoonright_{\Sigma_i}$ *denotes the projection of* ω *onto* Σ_i*, which is obtained by removing all the symbols in* ω *that are not in* Σ_i*. For a language* $L \subseteq \Sigma^{\infty}$ *and a subset* $\Sigma_i \subseteq \Sigma$, $L \upharpoonright_{\Sigma_i}$ *denotes the projection of L onto* Σ_i*, which is the set of projections of all words in L onto* Σ_i*, i.e.,* $\{\omega \upharpoonright_{\Sigma_i} \mid \omega \in L\}$.

Definition II.7 (Trace-Closed Language). *Given the distribution* $\{\Sigma_1, \ldots, \Sigma_m\}$ *of* Σ *and the words* $\omega, \omega' \in \Sigma^{\infty}$, ω' *is trace-equivalent to* ω*, denoted* $\omega' \sim \omega$*, iff their projections onto each one of the subsets in the given distribution are equal, i.e.,* $\omega \upharpoonright_{\Sigma_i} = \omega' \upharpoonright_{\Sigma_i}$ *for each* $i = 1, \ldots, m$*. For* $\{\Sigma_1, \ldots, \Sigma_m\}$*, the trace-equivalence class of* ω *is given by* $[\omega] = \{\omega' \in \Sigma^{\infty} \mid \omega' \upharpoonright_{\Sigma_i} = \omega \upharpoonright_{\Sigma_i} \forall i = 1, \ldots, m\}$*. Finally, a trace-closed language over* $\{\Sigma_1, \ldots, \Sigma_m\}$ *is a language L such that* $[\omega] \subseteq L$, $\forall\, \omega \in L$.

III. Problem Formulation and Approach

In this section we introduce the multi-robot path planning problem with temporal constraints, and we motivate the need for solutions that are robust to uncertain robot speeds.

A. Environment Model and Initial Formulation

Let $\mathcal{E} = (V, \rightarrow_{\mathcal{E}})$ be a graph, where V is the set of vertices and $\rightarrow_{\mathcal{E}} \subseteq V \times V$ is the set of edges. In this paper, $\mathcal{E}$ is the quotient graph of a partitioned environment, where V is a set of labels for the regions in the partition and $\rightarrow_{\mathcal{E}}$ is the corresponding adjacency relation (see Fig. 4).

Consider a team of m robots moving in an environment modeled by $\mathcal{E}$. Robot $i \in \{1, \ldots, m\}$ is modeled by a TS $\mathbf{T}_i = (\mathcal{Q}_i, q_i^0, \delta_i, \Pi_i, \mathcal{L}_i, w_i)$, where $\mathcal{Q}_i \subseteq V$; q_i^0 is the initial vertex of robot i; $\delta_i \subseteq \rightarrow_{\mathcal{E}}$ gives the motion capabilities of robot i; $\Pi_i \subseteq \Pi$ is the subset of propositions that can be satisfied by robot i such that $\{\Pi_1, \ldots, \Pi_m\}$ is a distribution of Π; $\mathcal{L}_i$ is a mapping from $\mathcal{Q}_i$ to 2^{Π_i} showing how the propositions are satisfied at vertices; $w_i(q, q')$ captures the time for robot i to go from vertex q to q', which we assume to be an integer. In this robotic model, robot i travels along the edges of $\mathbf{T}_i$, and spends zero time on the vertices. We assume that the robots are equipped with motion primitives which allow them to move from q to q' for each $(q, q') \in \delta_i$.

In our previous work [9], we considered multi-robot tasks specified by LTL formulae of the form $\phi := \varphi \wedge \mathbf{GF}\pi$ where φ can be any LTL formula over Π and $\pi \in \Pi$ is the atomic *optimizing proposition*. Our goal was to plan multi-robot paths that satisfy ϕ and minimize the maximum time between successive satisfying instances of π. For instance, in a persistent data gathering task, π may be assigned to upload regions, while φ can be used to specify rules (such as traffic rules) that must be obeyed at all times during the task [8].

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 the k^{th} symbol $\omega_i^k = \mathcal{L}_i(q_i^k)$ is satisfied at time t_i^k. Note that, as robots spend zero time on the vertices, each ω_i^k has a unique t_i^k which is the instant when robot i visits the corresponding vertex. 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 ascending order to obtain $\mathbb{T} := t^0 t^1, \ldots$. Then, we define $\omega_{team} = \omega_{team}^0 \omega_{team}^1 \ldots$ to be the word generated by the team of robots where the k^{th} symbol ω_{team}^k is the union of all propositions satisfied at time 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 the optimizing proposition π is satisfied for the k^{th} time by the team. Thus, the problem is that of synthesizing individual optimal runs for a team of robots so that ω_{team} satisfies ϕ and $\mathbb{T}^{\pi}$ minimizes

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

Since we consider LTL formulas containing $\mathbf{GF}\pi$, this optimization problem is always well-posed.

B. Robustness and Optimality in the Field

In this paper, we are interested in the implementability of our previous approach. Particularly, we consider the case where the actual value of $w_i(q, q')$ during deployment, denoted by $\tilde{w}_i(q,q')$, is a non-deterministic quantity in $[(1-\rho_i)w_i(q,q'),(1+\rho_i)w_i(q,q')]$ where ρ_i is the *deviation value* of robot i which is assumed to be given. 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 planned and actual values of some variable x.

Given the word ω_{team} that characterizes the planned run of the robotic team and the distribution $\{\Pi_1,\ldots,\Pi_m\}$, the *actual* word $\tilde{\omega}_{team}$ generated by the robotic team during its infinite asynchronous run in the field will be one of the trace equivalents of ω_{team}, *i.e.*, $\tilde{\omega}_{team} \in [\omega_{team}]$ due to the uncertainties in the traveling times of the robots. This leads to the definition of critical words.

Definition III.1 (Critical Words). *Given the language L_{B} of the Büchi automaton that corresponds to the LTL formula ϕ over Π, and given a distribution of Π, we define the word ω over Π to be a* critical word *if $\exists\, \tilde{\omega} \in [\omega]$ such that $\tilde{\omega} \notin L_{\mathrm{B}}$.*

Thus, we see that if the planned word is critical, then we may not satisfy the specification in the field. This can be formalized by noting that the optimal runs that satisfy ϕ are always in a prefix-suffix form [18], where the suffix cycle is repeated infinitely often. Using this observation and Def. III.1 we can formally define the words that can violate the LTL formula during the deployment of the robotic team.

Proposition III.2. *If the suffix cycle of the word ω_{team} is a critical word, then the correctness of the motion of the robotic team during its deployment cannot be guaranteed.*

We can also consider the field performance of the team in terms of the field value of the cost function (1). Using the same arguments presented in Prop. III.2 we can show that, the worst-case field value of (1) will be the minimum of $(\tilde{J}_1,\ldots,\tilde{J}_m)$ where $\tilde{J}_i$ is the maximum duration between successive satisfactions of π by robot i in the field. Thus, there is no benefit in executing the task with multiple robots, as the overall performance of the team will be limited by that of a single member.

C. Robust Problem Formulation

To maintain and characterize the field performance of the robotic team, we propose to use a synchronization protocol where robots can synchronize only when they are at the vertices of the environment. We assume that there is an atomic *synchronizing proposition* $\mathtt{Sync} \in \Pi$ and we consider multi-robot tasks specified using LTL formulas of the form

$$\phi_{sync} := \varphi \wedge \mathbf{GF}\pi \wedge \mathbf{GF}\mathtt{Sync}, \tag{2}$$

where φ can be any LTL formula over Π, π is the optimizing proposition and Sync is the special synchronizing proposition that is satisfied only when all members of the robotic team occupy vertices at the same time. We can now formulate the problem.

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

Note that the runs produced by a solution to Prob. III.3 are guaranteed not to violate ϕ_{sync} even if there is a mismatch between the weights $w_i(q,q')$ used for the solution of the problem and those observed in the field. 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.

D. Solution Outline

In [9], we showed that the joint behavior of a robotic team can be captured by a region automaton. A region automaton, as defined next, is a finite transition system that captures the relative positions of the members of the robotic team. This information is then used for computing optimal trajectories.

Definition III.4 (Region Automaton). *The region automaton $\mathbf{R}$ is a TS (Def. II.1) $\mathbf{R} := (\mathcal{Q}_{\mathrm{R}}, q^0_{\mathrm{R}}, \delta_{\mathrm{R}}, \Pi_{\mathrm{R}}, \mathcal{L}_{\mathrm{R}}, w_{\mathrm{R}})$, where (i) $\mathcal{Q}_{\mathrm{R}}$ 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 $\mathcal{Q}_i$ 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 the i^{th} element denotes the time elapsed since robot i left state q_i. (ii) q^0_{R} is the initial state that has zero-weight transitions to all those states in $\mathcal{Q}_{\mathrm{R}}$ 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 $(q^0_i,q'_i) \in \delta_i$. (iii) δ_{R} is the transition relation such that a transition from (q,r) to (q',r') exists if and only if $(q_i,q'_i),(q'_i,q''_i) \in \delta_i$ for all changed state pairs where the i^{th} element $q_iq'_i$ in q changes to $q'_iq''_i$ in q', $w_i(q_i,q'_i)-x_i$ 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 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'. (iv) $\Pi_{\mathrm{R}} = \cup_{i=1}^m \Pi_i$ is the set of propositions. (v) $\mathcal{L}_{\mathrm{R}} : \mathcal{Q}_{\mathrm{R}} \to 2^{\Pi_{\mathrm{R}}}$ is a map giving the set of atomic propositions satisfied in a state. For a state with $q=(q_1q'_1,\ldots,q_mq'_m)$, $\mathcal{L}_{\mathrm{R}}((q,r)) = \cup_{i=1}^m \mathcal{L}_i(q_i)$. (vi) $w_{\mathrm{R}} : \delta_R \to \mathbb{R}_{\geq 0}$ 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 $q'_iq''_i$ with a corresponding clock value of $x'_i=0$ in r'.*

Example III.5. *Fig. 2 illustrates the region automaton $\mathbf{R}$ that corresponds to the robots modeled with $\mathbf{T}_1$ and $\mathbf{T}_2$ given in Fig. 1. There is a transition from $((ba,bc),(0,0))$ to $((ba,cb),(1,0))$ with weight 1 in $\mathbf{R}$ because $(b,c) \in \delta_2$, $w_2(b,c)=1$, and $w_1(b,a) \neq 1$.*

Our solution to Problem III.3 can be outlined as follows: (i) We check if the LTL formula ϕ_{sync} is trace-closed guaranteeing that it will not be violated in the field (See Sec. IV-A); (ii) We prepare the serialized region automaton of the robotic team with synchronization points by modifying the output of our earlier algorithm OBTAIN-REGION-AUTOMATON [9] (See Sec. IV-B); (iii) We find optimal runs on individual $\mathbf{T}_i s$ using the OPTIMAL-RUN algorithm we previously developed in [18] and use a synchronization protocol to calculate an upper bound on the cost function (1) for given deviation values to obtain the solution to Prob. III.3 (See Sec. IV-C).

IV. Problem Solution

In this section, we explain each step of the solution to Prob. III.3 in detail. In the following, we use a simple example to illustrate ideas as we develop the theory for the general case. We present an experimental evaluation of our approach considering a more realistic scenario in Sec. V.

A. Trace-Closedness of the Original Formula

In the following, we say an LTL formula ϕ_{sync} is trace-closed if the language L_{B} of the corresponding Büchi automaton is trace-closed in the sense of Def. II.7.

Proposition IV.1. *If the general specification ϕ_{sync} is a trace-closed formula with respect to the distribution given by the robots' capabilities, then it will not be violated in the field due to uncertainties in the speeds of the robots.*

Thus, in order to guarantee correctness in the field, we first check that ϕ_{sync} is trace-closed using an algorithm adapted from [19]. However, as trace-closedness is not well-defined for words over 2^{Π}, we construct a Büchi automaton whose language L_{B} is over the set Π. Then, we proceed by obtaining the serialized region automaton with synchronization points where the Sync proposition is satisfied.

Example IV.2. *Fig. 1 illustrates a team of two robots that must satisfy an LTL formula in the form of* (2) *where* $\varphi = \mathbf{GF}\texttt{r1P} \wedge \mathbf{GF}\texttt{r2P}$, $\Pi_1 = \{\texttt{r1P}, \pi, \texttt{Sync}\}$, $\Pi_2 = \{\texttt{r2P}, \pi, \texttt{Sync}\}$*, and* $\Pi = \{\texttt{r1P}, \texttt{r2P}, \pi, \texttt{Sync}\}$.

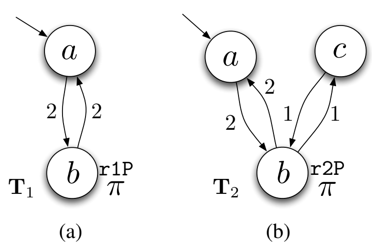


Fig. 1: TS's $\mathbf{T}_1$ and $\mathbf{T}_2$ of two robots in an environment with three vertices. Weights represent the traveling times between vertices. The propositions r1P, r2P and π are shown next to the vertices where they can be satisfied.

B. Serialized Region Automaton with Synchronization Points

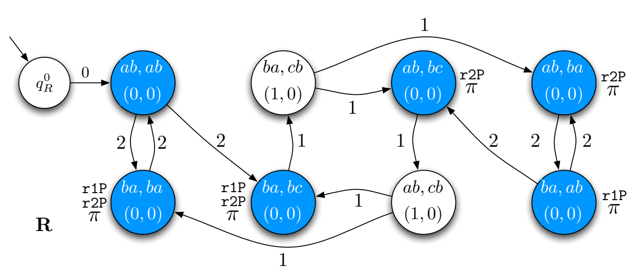


Fig. 2: Region automaton obtained using OBTAIN-REGION-AUTOMATON [9] that captures the joint behavior of the robotic team given in Fig. 1. Sync states where all robots occupy vertices are highlighted in blue.

If ϕ_{sync} is a trace-closed formula, we obtain the region automaton that captures the joint behavior of the robotic team using OBTAIN-REGION-AUTOMATON [9]. Next, using Alg. 1, we first add the special Sync proposition to the states where all robots occupy some vertex in their TS's simultaneously, *i.e.*, states with $r = (0, \ldots, 0)$. Note that, these are the states that will be used to calculate a bound on optimality when the robots are deployed in the field. We then expand the states where multiple propositions are satisfied simultaneously to obtain $\mathbf{R}_{ser}$ where at most one proposition is satisfied at each state. This ensures that languages of both $\mathbf{R}_{ser}$ and ϕ_{sync}'s Büchi automaton are over Π.

Example IV.2 Revisited. *Fig. 2 illustrates the region automaton* $\mathbf{R}$ *that captures the joint behavior of the team given in Fig. 1. The serialized region automaton with synchronization points* $\mathbf{R}_{ser}$ *that corresponds to* $\mathbf{R}$ *is given in Fig. 3.*

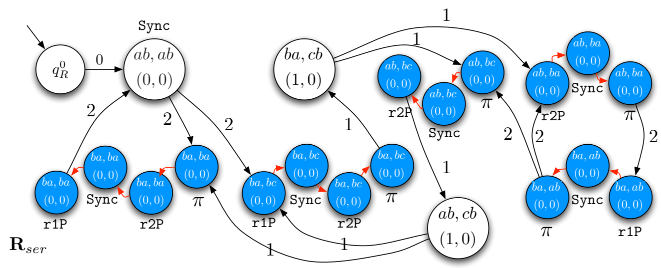


Fig. 3: Serialized region automaton with Sync states obtained by applying Alg. 1 to $\mathbf{R}$ in Fig. 2. New states introduced after serialization are highlighted in blue. Red arrows stand for zero-weight transitions.

Algorithm 1: SERIALIZE-REGION-AUTOMATON

Input: A region automaton $\mathbf{R}$ obtained using OBTAIN-REGION-AUTOMATON .
Output: $\mathbf{R}_{ser}$, the serialized region automaton with synchronization states, such that at most one proposition is satisfied at each state.

```
foreach State {q, r} in R do
    if r = (0, ..., 0) then
        Add Sync to propositions satisfied in {q, r}.
    k ⟵ Number of propositions satisfied in {q, r}.
    if k > 1 then
        propsTuple ← The tuple (p_1, ..., p_k) of
        propositions satisfied in {q, r}.
        Copy {q, r} k times to obtain {q, r}'_1, ..., {q, r}'_k.
        foreach i = 1, ..., k do
            L({q, r}'_i) ← propsTuple[i].
            if i < k then
                Add {q, r}'_i → {q, r}'_{i+1} to δ_R with zero
                weight.
        Re-direct all incoming transitions of {q, r} to
        {q, r}'_1.
        Originate all outgoing transitions of {q, r} from
        {q, r}'_k.
        Remove {q, r} from Q_R.
```

Remark IV.3. *Since ϕ_{sync} is trace-closed, the serialization can be done in any order. Since all possible orderings belong to the same trace-equivalent class, they do not affect the satisfaction of the formula.*

C. The Robust Optimal Run and the Optimality Bound

After obtaining the serialized region automaton $\mathbf{R}_{ser}$, we find an optimal run $r^{\star}_{\mathrm{R}}$ on $\mathbf{R}_{ser}$ that minimizes the cost function (1) using our earlier OPTIMAL-RUN algorithm [18]. The optimal run $r^{\star}_{\mathrm{R}}$ is always in a prefix-suffix form (Def. II.3). Furthermore, as $r^{\star}_{\mathrm{R}}$ satisfies ϕ_{sync}, it has at least one synchronization point in its suffix cycle, which we assume to start with a synchronization point.

Definition IV.4 (**Projection of a run on $\mathbf{R}_{ser}$ to $\mathbf{T}_i$s).** *Given the run $r_\mathrm{R} = \big((q_1^0 q_1^1, \ldots, q_m^0 q_m^1), (x_1^0, \ldots, x_m^0)\big)\big((q_1^1 q_1^2, \ldots, q_m^1 q_m^2), (x_1^1, \ldots, x_m^1)\big) \ldots$ on $\mathbf{R}_{ser}$ and the corresponding $\mathbb{T}$, we define its projection on $\mathbf{T}_i$ as run $r_i = q_i^0 q_i^1 \ldots$ for all $i = 1, \ldots, m$, where q_i^k only appears in r_i if $x_i^k = 0$ and $\mathbb{T}(k) \neq \mathbb{T}(k+1)$.*

In [9] we show that the individual runs r_i obtained by the projection in Def. IV.4 are equivalent to the region automaton run r_R in the sense that they produce the same word ω_{team}. Using Def. IV.4, we project the optimal run $r_\mathrm{R}^\star$ to individual $\mathbf{T}_i$s to obtain the set of optimal individual runs $\{r_1^\star, \ldots, r_m^\star\}$. As the robots execute their infinite runs in the field, they synchronize with each other at the synchronization point following the protocol given in Alg. 2 ensuring that they start each new suffix cycle in a synchronized way. Using this protocol, we can define a bound on optimality, *i.e.*, the value of the cost function (1) observed in the field, as given in the following proposition.

Algorithm 2: SYNC-RUN

```
Input: A run r_k of robot k in the prefix-suffix form
       with at least one synchronization point in its
       suffix cycle.
begin
    syncPoint ← First synchronization point in the
    suffix.
    teamFlags ← (0, ..., 0).
    while True do
        if syncMessage received from robot i then
            teamFlags[i] ← 1.
        if currentState = syncPoint then
            Stop
            Broadcast syncMessage.
            teamFlags[k] ← 1.
        if teamFlags = (1, ..., 1) then
            teamFlags ← (0, ..., 0).
            Continue executing r_k.
```

Proposition IV.5. *Suppose that each robot's deviation value is bounded by $\rho > 0$ (i.e., $\rho_i \leq \rho$ 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. 2 the field value of the cost satisfies $\overline{J(\mathbb{T}^\pi)} \leq J(\mathbb{T}^\pi) + \rho(J(\mathbb{T}^\pi) + 2d_s)$, where d_s is the planned duration of the suffix cycle.*

Example IV.2 Revisited. *Applying Alg.* OPTIMAL-RUN *[18] to $\mathbf{R}_{ser}$ given in Fig. 2 and the formula $\phi_{sync} := \mathbf{GF}\texttt{r1P} \wedge \mathbf{GF}\texttt{r2P} \wedge \mathbf{GF}\pi \wedge \mathbf{GF}\texttt{Sync}$ results in the optimal run with the prefix*

$\mathbb{T}$	*0*	*2*	*2*	*2*	*2*	*3*
$r_\mathrm{R}^\star$	*ab,ab (0,0)*	*ba,bc (0,0)*	*ba,bc (0,0)*	*ba,bc (0,0)*	*ba,bc (0,0)*	*ba,cb (1,0)*
$\mathcal{L}_\mathbf{R}(\cdot)$	`Sync`	`r1P`	`Sync`	`r2P`	π	

and the suffix cycle

$\mathbb{T}$	*4*	*4*	*4*	*6*	*6*	*6*
$r_\mathrm{R}^\star$	*ab,ba (0,0)*	*ab,ba (0,0)*	*ab,ba (0,0)*	*ba,ab (0,0)*	*ba,ab (0,0)*	*ba,ab (0,0)*
$\mathcal{L}_\mathbf{R}(\cdot)$	`r2P`	`Sync`	π	`r1P`	`Sync`	π

which will be repeated an infinite number of times. In the table above, the rows correspond to the times when transitions occur, the run $r_\mathrm{R}^\star$, and the satisfying atomic propositions, respectively. For this example, $\mathbb{T}^\pi = 2, 4, 6, 8, 10, \ldots$ and the cost as defined in (1) is $J(\mathbb{T}^\pi) = 2$. Furthermore, when the robotic team is deployed in the field, this cost is bounded from above by 2.5 for $\rho_1 = \rho_2 = 0.05$ as given by Prop. IV.5.

Applying Def. IV.4 to $r_\mathrm{R}^\star$ we have the individual runs as:

$\mathbb{T}$	*0*	*2*	*3*	*4*	*6*	*8*	*10*	$\ldots$
$r_1^\star$	*a*	*b*		*a*	*b*	*a*	*b*	$\ldots$
$r_2^\star$	*a*	*b*	*c*	*b*	*a*	*b*	*a*	$\ldots$

Note that, at time $t = 3$, the second robot has arrived at c while the first robot is still traveling from b to a, therefore the clock of the first robot is not zero at this time, i.e., $x_1 \neq 0$, and b does not appear in $r_1^\star$ at time $t = 3$.

We finally summarize our approach in Alg. 3, show that this algorithm indeed gives a solution to Prob. III.3 and analyze the overall complexity of our approach.

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

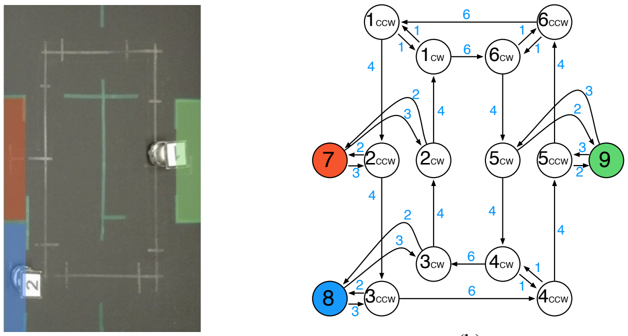
```
Input: m T_i's and a global LTL specification φ_sync of
       form (2).
Output: A set of robust optimal runs {r_1*, ..., r_m*} that
        satisfies φ_sync, minimizes (1), and the bound
        on the performance of the team in the field.
begin
    φ_sync := φ ∧ GFπ ∧ GFSync.
    if φ_sync is trace-closed then
        Obtain the region automaton R using
        OBTAIN-REGION-AUTOMATON [9].
        Obtain R_ser using
        SERIALIZE-REGION-AUTOMATON .
        Find the optimal run r_R* applying OPTIMAL-RUN
        [18] to R_ser and φ_sync.
        Obtain individual runs from r_R* using Def. IV.4.
        Find the bound on optimality as given in
        Prop. IV.5.
    else
        Abort.
```

Proposition IV.6. *Alg. 3 solves Prob. III.3.*

Proposition IV.7. *For the case where m identical robots are expected to satisfy an LTL specification ϕ in a common environment with Δ edges and a largest edge weight of W, the worst-case complexity of Alg. 3 is $O((\Delta \cdot W)^{3m} \cdot 2^{O(|\phi|)})$.*

V. IMPLEMENTATION AND CASE STUDIES

We implemented Alg. 3 in objective-C as the software package LTL ROBUST OPTIMAL MULTI-ROBOT PLANNER (LROMP) (available at `http://hyness.bu.edu/Software.html`). Following the steps detailed in Sec. IV, LROMP computes robust and optimal trajectories for robots performing persistent data gathering missions in a road network environment to obtain the solution to Prob. III.3.



(a) (b)

Fig. 4: (a) Road network used in the experiments (b) The model of the road network with weights shown in blue. 1 time unit in this model corresponds to 3 seconds. The red and blue regions are data gathering locations of robots 1 and 2, respectively and the green region is the common upload location. CW and CCW stand for clockwise and counter-clockwise, respectively.

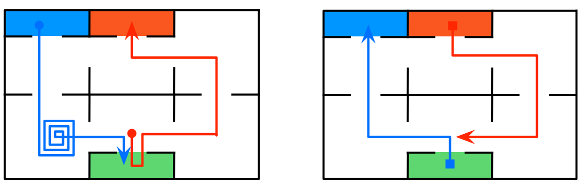


Fig. 5: Team trajectory used in the experiments. The red and blue regions are data gathering locations of robots 1 and 2, respectively and the green region is the common upload location. The circles on the left show the sync point, *i.e.*, the beginning of the suffix cycle, on the trajectories of the robots.

Fig. 4 illustrates our experimental platform, which is a road network consisting of roads, intersections, and task locations. The figure also shows the transition system that models the motion of the robots on this road network. In the following, the transition systems $\mathbf{T}_i$ are identical except for their initial states and propositions defined at states.

In our experiments, we consider a persistent monitoring task where two robots with deviation values of $\rho_1 = 0.09$, $\rho_2 = 0.04$ repeatedly gather and upload data and the maximum time in between any two data uploads must be minimized. We require robots 1 and 2 to gather data at 7 and 8 in Fig. 4, respectively and upload the data at 9. We define $\Pi = \{\texttt{R1Gather}, \texttt{R1Upload}, \texttt{R2Gather}, \texttt{R2Upload}, \texttt{Upload}, \texttt{Sync}\}$ and assign the atomic propositions as $\mathcal{L}_1(7) = \{\texttt{R1Gather}\}$, $\mathcal{L}_1(9) = \{\texttt{R1Upload}, \texttt{Upload}\}$, $\mathcal{L}_2(8) = \{\texttt{R2Gather}\}$, $\mathcal{L}_2(9) = \{\texttt{R2Upload}, \texttt{Upload}\}$ where Upload is set as the optimizing proposition π in (2). We forbid data uploads unless there is new data using $\varphi = \mathbf{G}(\texttt{R1Upload} \Rightarrow \mathbf{X}(\neg\texttt{R1Upload}\,\mathcal{U}\,\texttt{R1Gather})) \wedge \mathbf{G}(\texttt{R2Upload} \Rightarrow \mathbf{X}(\neg\texttt{R2Upload}\,\mathcal{U}\,\texttt{R2Gather}))$. Our overall LTL formula is $\phi_{sync} = \varphi \wedge \mathbf{G}\,\mathbf{F}\,\texttt{Upload} \wedge \mathbf{G}\,\mathbf{F}\,\texttt{Sync}$.

Running our algorithms on an iMac i5 quad-core computer, we obtain the robust optimal trajectory as illustrated in Fig. 5. The algorithm ran for 35 minutes, and $\mathbf{R}_{ser}$ had 5224 states. The value of the cost function was 57 seconds with an upper-bound of 82.65 seconds. This result was experimentally verified in our test-bed and the maximum time in between data uploads was measured to be 64 seconds during a run of 13 minutes. In order to demonstrate the effectiveness of our approach, we executed the same trajectory without synchronization. After approximately 6.5 minutes, the maximum time in between data uploads was measured to be 92 seconds, much worse than what is provided by our approach. Our video submission accompanying the paper displays the robot trajectories for both cases. It is interesting to note that, in the optimal solution the second robot spins between states 4_{CW} and 4_{CCW} (Figs. 4b, 5). This behavior is actually optimal as it decreases the maximum time between successive data uploads.

VI. Conclusions

In this paper we presented a method for planning robust optimal trajectories for a team of robots subject to temporal logic constraints. We considered trace-closed linear temporal logic specifications with optimizing and synchronizing propositions that must be repeatedly satisfied. Our method is robust to uncertainties in the traveling times of each robot, and thus has practical value in applications where multiple robots must perform a series of tasks collectively in a common environment.

References

[1] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 153–158.

[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[3] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Hybrid systems: Computation and Control*, Stockholm, Sweden, 2010, pp. 101–110.

[4] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Motion planning with hybrid dynamics and temporal goals," in *IEEE Conf. on Decision and Control*, 2010, pp. 1108–1115.

[5] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. LaValle, "Controlling wild bodies using linear temporal logic," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[6] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Logic in Computer Science*, 1986, pp. 322–331.

[7] G. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 279–295, 1997.

[8] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," *International Journal of Robotics Research*, vol. 30, pp. 1695–1708, Dec. 2011.

[9] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," pp. 3087–3092, Sep. 2011.

[10] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[11] R. Milner, *Communication and concurrency*. Prentice-Hall, 1989.

[12] A. Puri, "An undecidable problem for timed automata," *Discrete Event Dynamic Systems*, vol. 9, no. 2, pp. 135–146, 2000.

[13] ——, "Dynamical properties of timed automata," *Discrete Event Dynamic Systems*, vol. 10, no. 1-2, pp. 87–113, 2000.

[14] Y. Chen, X. C. Ding, and C. Belta, "A formal approach to the deployment of distributed robotic teams," *IEEE Transactions on Robotics*, 2011, to appear.

[15] L. C. G. J. M. Habets and J. H. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, pp. 21–35, 2004.

[16] A. Ulusoy, S. L. Smith, X. C. Ding, and C. Belta, "Robust multi-robot optimal path planning with temporal logic constraints," Feb 2012, available at http://arxiv.org/abs/1202.1307.

[17] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[18] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning under temporal logic constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Taipei, Taiwan, Oct. 2010, pp. 3288–3293.

[19] D. Peled, T. Wilke, and P. Wolper, "An algorithmic approach for checking closure properties of temporal logic specifications and omega-regular languages," *Theor. Comput. Sci.*, vol. 195, no. 2, pp. 183–203, 1998.