Automatica 70 (2016) 239–248

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Multi-agent planning under local LTL specifications and event-based synchronization☆

Jana Tumova, Dimos V. Dimarogonas

School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden

ARTICLE INFO

Article history:
Received 3 March 2015
Received in revised form
3 March 2016
Accepted 18 March 2016
Available online 23 April 2016

Keywords:
Temporal logic
Finite state machines
Formal verification
Path planning
Synchronization
Decentralized control
Robot control

ABSTRACT

We study the problem of plan synthesis for multi-agent systems, to achieve complex, high-level, long-term goals that are assigned to each agent individually. As the agents might not be capable of satisfying their respective goals by themselves, requests for other agents' collaborations are a part of the task descriptions. We consider that each agent is modeled as a discrete state-transition system and its task specification takes a form of a linear temporal logic formula. A traditional automata-based approach to multi-agent plan synthesis from such specifications builds on centralized team planning and full team synchronization after each agents' discrete step, and thus suffers from extreme computational demands. We aim at reducing the computational complexity by decomposing the plan synthesis problem into finite horizon planning problems that are solved iteratively, upon the run of the agents. We introduce an event-based synchronization that allows our approach to efficiently adapt to different time durations of different agents' discrete steps. We discuss the correctness of the solution and find assumptions, under which the proposed iterative algorithm leads to provable eventual satisfaction of the desired specifications.

1. Introduction

In recent years, a considerable amount of attention has been devoted to synthesis of robot controllers for complex, high-level missions, such as "periodically survey regions *A*, *B*, *C*, in this order, while avoiding region *D*", specified as temporal logic formulas. Many of the suggested solutions to variants of this problem rely on a hierarchical procedure (Bhatia, Maly, Kavraki, & Vardi, 2011; Kloetzer & Belta, 2008; Kress-Gazit, Fainekos, & Pappas, 2009; Wongpiromsarn, Topcu, & Murray, 2010): First, the dynamics of the robotic system is abstracted into a finite transition system using e.g., sampling or cell decomposition methods. Second, leveraging ideas from formal verification, a discrete plan that meets the mission is synthesized. Third, the plan is translated into a controller for the original system. In this work, we focus on a multi-agent version of the above problem. We consider a heterogeneous team of robots, that are assigned a temporal logic mission each. As the robots may not be able to accomplish their missions without the help of the others, the specifications may contain requirements on the other team members' behavior. For instance, consider a warehouse solution with two mobile robots. A part of the first robot's mission is to load an object in region *A*, but it is not able to load it by itself. Therefore, the mission also includes a task for the second robot, to help loading. The goal of this paper is to efficiently synthesize a plan for each agent, such that each agent's mission is met. We follow the hierarchical approach to robot controller synthesis as outlined above and we narrow our attention to the second step of the approach, i.e., to generating discrete plans. The application of the algorithm that we propose is, however, not restricted to discrete systems: For the first step of the hierarchical approach, numerous methods for discrete modeling of robotic systems can be used (see, e.g., Kloetzer & Belta, 2008; Kress-Gazit et al., 2009; LaValle, 2006; Wongpiromsarn et al., 2010 and the references therein); for the third step, low-level controllers exist that can drive a robot from any position within a region to a goal region (see, e.g., Belta & Habets, 2006). The agents can, but do not have to, mutually synchronize after the execution of their respective discrete steps. The desired plans thus comprise not only of the agents' discrete steps to be taken, but also their synchronizations. Besides the satisfaction of all agents' missions, our goal is to avoid unnecessary synchronization in order to improve the team performance.

☆ This work was supported by the EU STREP RECONFIG, and by the H2020 ERC Starting Grant BUCOPHSYS. The material in this paper was partially presented at the 2014 American Control Conference, June 4–6, 2014, Portland, OR, USA. This paper was recommended for publication in revised form by Associate Editor Jan Komenda under the direction of Editor Christos G. Cassandras.

E-mail addresses: tumova@kth.se (J. Tumova), dimos@kth.se (D.V. Dimarogonas).

http://dx.doi.org/10.1016/j.automatica.2016.04.006
0005-1098/

As a mission specification language, we use Linear Temporal Logic (LTL), for its resemblance to natural language (Jing, Finucane, Raman, & Kress-Gazit, 2012), and expressive power. Here, we built LTL formulas over services, i.e., events of interest associated with execution of certain actions rather than over atomic propositions, i.e., inherent properties of the system states. Instead of evaluation of the specification as a conjunction of LTL formulas over the whole team behaviors, we propose the notion of satisfaction of an LTL formula from local perspective. This way, the problem of finding a collective team behavior is decomposed into several subproblems, enabling us to avoid the straightforward, but expensive fully centralized planning. The contribution of this paper can be summarized as the introduction of an efficient, iterative, finite horizon planning technique in the context of bottom-up plan synthesis for multi-agent systems from local LTL specifications. To our best knowledge, such an approach has not been taken to address the multi-agent LTL planning before. Our algorithm is adaptive in the sense that even if the real behavior of the team is not as planned due to unpredictable time durations of the agents' steps, the event-based synchronization and replanning still guarantees the satisfaction of all the tasks. This feature can be especially beneficial in heterogeneous multi-robot motion and task planning problems, where individual robots traverse their common environment at different speeds. This paper builds on our earlier work in Tumova and Dimarogonas (2014). In addition, it relaxes the assumption that the agents synchronize after every discrete step of theirs and introduces the event-based synchronization and replanning.

Multi-agent planning from temporal logic specification has been explored in several recent works. Planning from computational tree logic was considered in Quottrup, Bak, and Zamanabadi (2004), whereas in Kloetzer, Ding, and Belta (2011) and Loizou and Kyriakopoulos (2005), the authors focus on planning behavior of a team of robots from a single, global LTL specification. A fragment of LTL has been considered as a specification language for vehicle routing problems in Karaman and Frazzoli (2011), and a general reactivity LTL fragment has been used in Wiltsche, Ramponi, and Lygeros (2013). Decentralized control of a robotic team from local LTL specification with communication constraints is proposed in Filippidis, Dimarogonas, and Kyriakopoulos (2012). However, the specifications there are truly local and the agents do not impose any requirements on the other agents' behavior. Thus, the focus of the paper is significantly different to ours. As opposed to our approach, in Chen, Ding, Stefanescu, and Belta (2012) and Ulusoy, Smith, Ding, Belta, and Rus (2013), a top-down approach to LTL planning is considered; the team is given a global specification and an effort is made to decompose the formula into independent local specifications that can be treated separately for each robot. In Guo and Dimarogonas (2015), bottom-up planning from LTL specifications is considered, and a partially decentralized solution is proposed that takes into account only clusters of dependent agents instead of the whole group. A huge challenge of the previous approach is its extreme computational complexity, which we tackle in this work by applying receding horizon approach to multi-agent planning. Receding horizon approach was leveraged also in Wongpiromsarn et al. (2010) to cope with uncertain elements in an environment in single-robot motion planning. To guarantee the satisfaction of the formula, we use an attraction-type function that guides the individual agents towards a progress within a finite planning horizon; similar ideas were used in Ding, Belta, and Cassandras (2010) and Svorenova, Tumova, Barnat, and Cerna (2012) for a single-agent LTL planning to achieve a locally optimal behavior.

The rest of the paper is structured as follows. In Section 2, we fix the preliminaries. Section 3 introduces the problem and summarizes our approach. In Section 4, the details of the solution are provided. In Section 5, we provide analysis and discussion of the solution. We present simulation results in Section 6, and we conclude in Section 7.

2. Preliminaries

Given a set S, let 2^{S}, and S^{ω} denote the set of all subsets of S, and the set of all infinite sequences of elements of S, respectively. A finite or infinite sequence of elements of S is called a finite or infinite word over S, respectively. The ith element of a word w is denoted by $w(i)$. A subsequence of an infinite word $w = w(1)w(2)\ldots$ is a finite or infinite sequence of its elements $w(i_1)w(i_2)\ldots$, where $\forall 1 \leq j.\ 1 \leq i_j \leq i_{j+1}$. A factor of w is a continuous, finite or infinite, subsequence $w(i)w(i+1)\ldots$, where $1 \leq i$. A prefix of w is a finite factor starting at $w(1)$, and a suffix of w is an infinite factor. $\mathbb{N}$ and $\mathbb{R}_0^+$ denote positive integers and non-negative real numbers, respectively.

A *transition system (TS)* is a tuple $\mathcal{T} = (S, s_{init}, A, T)$, where S is a finite set of states; $s_{init} \in S$ is the initial state; A is a finite set of actions; and $T \subseteq S \times A \to S$ is a partial deterministic transition function. For simplicity, we denote a transition $T(s, \alpha) = s'$ by $s \xrightarrow{\alpha} s'$. A *trace* of $\mathcal{T}$ is an infinite alternating sequence of states and actions $\tau = s_1\alpha_1 s_2\alpha_2\ldots$, such that $s_1 = s_{init}$, and for all $i \geq 1$, $s_i \xrightarrow{\alpha_i} s_{i+1}$. A *trace fragment* $\hat{\tau}$ is a finite factor of a trace τ that begins and ends with a state.

A *linear temporal logic (LTL) formula* ϕ over the set of atomic propositions Π is defined inductively: (i) $\pi \in \Pi$ is a formula, and (ii) if ϕ_1 and ϕ_2 are formulas, then $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\mathsf{X}\phi_1$, $\phi_1 \mathsf{U} \phi_2$, $\mathsf{F}\phi_1$, and $\mathsf{G}\phi_1$ are each a formula, where $\neg$ and $\vee$ are standard Boolean connectives, and X, U, F, and G are temporal operators. The semantics of LTL are defined over infinite words over 2^{Π}. $\pi \in \Pi$ is satisfied on $w = \varpi_1\varpi_2\ldots$ if $\pi \in \varpi_1$. $\mathsf{X}\phi$ holds true if ϕ is satisfied on the word that begins in the next position ϖ_2, $\phi_1 \mathsf{U} \phi_2$ states that ϕ_1 has to be true until ϕ_2 becomes true, and $\mathsf{F}\phi$ and $\mathsf{G}\phi$ are true if ϕ holds on w eventually, and always, respectively. We denote the satisfaction of ϕ on a word w as $w \models \phi$. The set of all words accepted by an LTL formula ϕ is $\mathcal{L}(\phi)$. For full details see, e.g., Baier and Katoen (2008).

An *automaton* is a tuple $\mathcal{A} = (Q, q_{init}, \Sigma, \delta, F)$, where Q is a finite set of states; $q_{init} \in Q$ is the initial state; Σ is an input alphabet; $\delta \subseteq Q \times \Sigma \times Q$ is a non-deterministic transition relation; and F is an accepting condition. It is *deadlock-free* if $\forall q \in Q, \sigma \in \Sigma.\ \delta(q, \sigma) \neq \emptyset$. We define the set of states $\hat{\delta}^k(q)$ reachable from $q \in Q$ in exactly k steps inductively as $\hat{\delta}^0(q) = \{q\}$, and $\hat{\delta}^k(q) = \bigcup_{q' \in \hat{\delta}^{k-1}(q)} \{q'' \mid \exists \sigma \in \Sigma.\ (q', \sigma, q'') \in \delta\}, \forall k \geq 1$. A *Büchi automaton (BA)* is an automaton with the accepting condition $F \subseteq Q$. A *run* of the BA $\mathcal{B}$ from $q_1 \in Q$ *over* $w = \sigma_1\sigma_2\ldots \in \Sigma^{\omega}$ is a sequence $\rho = q_1 q_2\ldots$, such that $\forall i \geq 1.\ (q_i, \sigma_i, q_{i+1}) \in \delta$. A run ρ is *accepting* if it intersects F infinitely many times. A word w is *accepted* by $\mathcal{B}$ if there exists an accepting run over w from the state q_{init}. The set of all words accepted by $\mathcal{B}$ is $\mathcal{L}(\mathcal{B})$. Any automaton $(Q, q_{init}, \Sigma, \delta, F)$ can be viewed as a graph (V, E) with the vertexes $V = Q$ and the edges E given by δ in the expected way. The standard notation then applies: A *path* is a finite alternating sequence of states and transition labels $q_i\sigma_i q_{i+1}\ldots q_{k-1}\sigma_{k-1}q_k$, such that $\forall i \leq j < k.\ (q_j, \sigma_j, q_{j+1}) \in \delta$. $\mathrm{dist}(q, q')$ denotes the length of the *shortest path* between q and q', i.e., the minimal number of states in a path $q\ldots q'$. If no such path exists, then $\mathrm{dist}(q, q') = \infty$. If $q = q'$, then $\mathrm{dist}(q, q') = 0$. The shortest path can be computed using Dijkstra algorithm (see, e.g., Cormen, Stein, Rivest, & Leiserson, 2001).

3. Problem formulation and approach

Two general viewpoints can be taken in multi-agent planning: either the system acts as a team with a common goal, or the agents have their own, independent tasks. Although we permit each agent's task to involve requirements on the others, we adopt the second viewpoint; to decide whether the agents' tasks are met,