Control Engineering Practice 81 (2018) 97–104

Contents lists available at ScienceDirect

Control Engineering Practice

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

Application of online supervisory control of discrete-event systems to multi-robot warehouse automation

Yuta Tatsumoto [a,*], Masahiro Shiraishi [a], Kai Cai [a], Zhiyun Lin [b]

[a] Department of Electrical and Information Engineering, Osaka City University, Japan
[b] Hangzhou Dianzi University, China

ARTICLE INFO

Keywords:
Supervisory control
Discrete-event systems
Warehouse automation
Online control

ABSTRACT

In this paper we present an online supervisory control approach, based on limited lookahead policy, that is amenable for the control of multi-agent discrete-event systems. We then apply this online control scheme to model and control a warehouse automation system served by multiple mobile robots; the effectiveness of this scheme is demonstrated through a case study. Moreover, we build an experiment testbed for testing the validity of our proposed method with implementation on real robots.

1. Introduction

Supervisory control theory of discrete-event systems (DES) was first proposed by Ramadge and Wonham in the 1980s (Ramadge & Wonham, 1987), with the aim to formalizing general (high-level) control principles for a wide range of application domains. In this theory, DES are modeled as finite-state automata, and their behaviors represented by regular languages. The control feature is that certain events (or state transitions) can be disabled by an external supervisor to enforce a desired behavior. This feature leads to the fundamental concept of *language controllability*, which determines the existence of a supervisor that suitably disables a series of events in order to satisfy an imposed control specification. For a comprehensive account of supervisory control theory, the reader is referred to Wonham and Cai (2018); also see Wonham, Cai, and Rudie (2018) for a recent historical overview of the theory.

While supervisory control of DES is theoretically sound, it is often computationally infeasible for practical systems that are large and complex, due to the notorious problem of state explosion (Gohari & Wonham, 2000). Indeed, the computational complexity of synthesizing a supervisor is exponential in the number of plant components. In addition, systems that are time-varying or subject to unknown changes are also unsuitable to be dealt with by supervisory control theory, because a supervisor is synthesized offline and consequently cannot account for changes in operation.

To address large and time-varying DES, online supervisory control based on *limited lookahead policy* was proposed in Chung, Lafortune, and Lin (1992, 1993, 1994). This control scheme generates at the current state a limited-step-ahead projection of the plant's behavior, and determines based on the projected behavior the next control action to satisfy an imposed specification. A new projection is generated after each occurrence of event, so time-varying changes in the system can be taken into account in this scheme. In particular, Chung et al. (1993, 1994) presented recursive strategies to make the online supervisory synthesis more efficient. Other extensions of the online supervisory control can be found in Boroomand and Hashtrudi-Zad (2013), Hadj-Alouane, Lafortune, and Lin (1996), Kumar, Cheung, and Marcus (1998) and Winacott and Rudie (2009). In Hadj-Alouane et al. (1996), online supervisory control of partially observed DES was addressed where some occurrences of events are unobservable. In Kumar et al. (1998), a variant method was reported for generating a limited-step-ahead projection of the plant's behavior. In Winacott and Rudie (2009), online control of probabilistic DES was investigated where event occurrences are random accordingly to given probabilities. Finally in Boroomand and Hashtrudi-Zad (2013), online control was extended to deal with robust supervisory control where the plant model is uncertain. Common in all the online supervisory control methods above (Boroomand & Hashtrudi-Zad, 2013; Chung et al., 1992, 1993, 1994; Hadj-Alouane et al., 1996; Kumar et al., 1998; Winacott & Rudie, 2009), the plant model is assumed to be given or already computed.

This paper adapts the online supervisory control approach to the case of multi-agent DES, i.e. plant consisting of multiple components, and applies the approach to warehouse automation systems served by multiple mobile robots. In recent years warehouse automation has received significant interest from both academia and industries, due to

* Corresponding author.
E-mail address: tatsumoto@c.info.eng.osaka-cu.ac.jp (Y. Tatsumoto).

https://doi.org/10.1016/j.conengprac.2018.09.003
Received 19 March 2018; Received in revised form 3 July 2018; Accepted 5 September 2018

its central role in the rapidly growing e-commerce, supply chain, and material handling enterprise. A landmark of such examples is Kiva systems (Wurman, D'Andrea, & Mountz, 2008), which dispatches hundreds of robots to serve the logistics in Amazon's distribution centers.

The contributions of this paper are threefold. First, we present a key modification to the existing online supervisory control approach such that the approach is amenable to multi-agent DES. The approach in Boroomand and Hashtrudi-Zad (2013), Chung et al. (1992, 1993, 1994) , Hadj-Alouane et al. (1996), Kumar et al. (1998) and Winacott and Rudie (2009) is not suitable for multi-agent DES because it assumes that the plant model is already given or can be computed, and then generates a limited-step-ahead projection of the plant model. For multi-agent DES, however, the plant model is the synchronous product of the component agents, which itself may not be feasibly computable particularly when the number of agents is large. To circumvent this problem, we propose to first generate limited-step-ahead projections of each agent model, and then compute the synchronous product of the projected agent models. This is computationally more efficient than the existing approaches in Boroomand and Hashtrudi-Zad (2013), Chung et al. (1992, 1993, 1994), Hadj-Alouane et al. (1996), Kumar et al. (1998) and Winacott and Rudie (2009). Second, we show how a warehouse automation system served by multiple robots can be modeled as a multi-agent DES, and demonstrate through a case study the effectiveness of applying the adapted online supervisory control approach to control the automation system. Finally, we build a testbed, consisting of multiple LEGO MINDSTORMS EV3 robots (https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313), that serves to experimentally test the validity of our proposed online supervisory control approach for multi-agent DES.

We note that Goryca and Hill (2013), Hill and Lafortune (2016, 2017), Lopes, Trenkwalder, Leal, Dodd, and Groß(2016), Su (2013), Su and Lennartson (2017), Su and Lin (2013) and Ware and Su (2016) also studied supervisory control for multi-agent DES. In Ware and Su (2016), the authors presented a method that assigns distinct priorities to all agents and then incrementally synthesizes supervisors for solving a scheduling problem. This method is efficient, but need not find a supervisor for scheduling even when a schedule exists. In Lopes et al. (2016), the authors applied supervisory control methods to achieving various cooperative behaviors of swarm robots (including segregation, aggregation, object clustering, and formations), and demonstrated their results on real multi-robot systems. In Goryca and Hill (2013) and Hill and Lafortune (2016, 2017), the authors proposed an abstraction-based method to efficiently synthesize supervisors for multi-agent DES, and implemented the method into a software whose efficiency was tested through a multi-robot planning case study. Finally in Su (2013), Su and Lennartson (2017) and Su and Lin (2013), the authors proposed a scalable control design for a type of multi-agent DES, where an "agent" was not just a plant component, but indeed a plant of its own including an imposed specification. The "agents" were instantiated from a template; for the template, under certain conditions, an algorithm was proposed to design a supervisor whose instantiation was shown to work for each "agent". In all the references above, the proposed control synthesis methods are offline; although these methods addressed in one way or another the issue of computational efficiency, they cannot handle multi-agent DES that are time-varying and subject to unknown changes. By contrast, our proposed method for multi-agent DES is online, and therefore not only reduces computational effort, but also deals with dynamic changes in the system.

The rest of this paper is organized as follows. In Section 2 we introduce the adapted online supervisory control of multi-agent DES. In Section 3 we model a warehouse automation system by multi-agent DES, and apply online supervisory control for its control. Moreover, we present a testbed for experimental validation of online supervisory control for warehouse automation. Finally in Section 4 we state our conclusions.

2. Online supervisory control of multi-agent DES

In standard supervisory control (Ramadge & Wonham, 1987; Wonham and Cai, 2018), the plant to be controlled is modeled by a finite state *automaton*

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is the finite state set, $q_0 \in Q$ the *initial state*, $Q_m \subseteq Q$ the set of *marker states*, Σ the finite *event set*, and $\delta : Q \times \Sigma \to Q$ the (partial) *state transition function*. Letting Σ^* denote the set of all finite-length strings of events in Σ, we extend δ such that $\delta : Q \times \Sigma^* \to Q$ and write $\delta(q, s)!$ to mean that $\delta(q, s)$ is defined.

The closed behavior of $\mathbf{G}$ is the set of all strings that can be generated by $\mathbf{G}$:

$$L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\}.$$

On the other hand, the *marked behavior* of $\mathbf{G}$ is the subset of strings that can reach a marker state:

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G}).$$

$\mathbf{G}$ is *nonblocking* if $L(\mathbf{G}) = \overline{L_m(\mathbf{G})}$ ($\overline{\cdot}$ means *prefix closure*), namely every string in the closed behavior may be completed to a string in the marked behavior.

The event set Σ of $\mathbf{G}$ is partitioned into a subset Σ_c of *controllable* events and a subset Σ_u of *uncontrollable* events. A language $E \subseteq \Sigma^*$ is said to be *controllable* (with respect to $\mathbf{G}$) if $\overline{E}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{E}$, i.e.

$$(\forall s \in \Sigma^*, \forall \sigma \in \Sigma) s \in \overline{E}, \sigma \in \Sigma_u, s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in \overline{E}.$$

Let $K \subseteq L_m(\mathbf{G})$ be a specification language imposed on the plant $\mathbf{G}$. Denote by $C(K)$ the family of controllable sublanguages of K, i.e.

$$C(K) := \{K' \subseteq K \mid \overline{K'}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K'}\}.$$

Then the supremal controllable sublanguage of K exists and is given by $\sup C(K) = \cup\{K' \mid K' \in C(K)\}$. Let **SUP** be a (nonblocking) automaton such that $L_m(\mathbf{SUP}) = \sup C(K)$. We call **SUP** the supervisor for plant $\mathbf{G}$ that enforces $\sup C(K) \subseteq L_m(\mathbf{G})$. The control action of **SUP** after an arbitrary string $s \in L(\mathbf{G})$ is to enable the events in

$$\gamma := \{\sigma \in \Sigma_u \mid s\sigma \in L(\mathbf{G})\} \cup \{\sigma \in \Sigma_c \mid s\sigma \in L(\mathbf{SUP})\}.$$

To introduce online supervisory control based on limited lookahead policy, we need an operation that 'truncate' an automaton from a specified state to limited step ahead. Given an automaton $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, a state $q \in Q$, and the step number $N \geq 1$, the 'truncation' operation $f^N(\mathbf{G}, q)$ on $\mathbf{G}$ at state q defines a new automaton

$$f^N(\mathbf{G}, q) = \mathbf{G}^N(q) := (Q^N, \Sigma^N, \delta^N, q_0^N, Q_m^N)$$

where

$$\begin{aligned} Q^N &= \{q' \in Q \mid (\exists s \in \Sigma^*) q' = \delta(q, s) \ \&\ |s| \leq N\} \\ \Sigma^N &= \{\sigma \in \Sigma \mid (\exists q' \in Q^N) \delta(q', \sigma)! \ \&\ \delta(q', \sigma) \in Q^N\} \\ \delta^N &= \{(q_1, \sigma, q_2) \mid q_1, q_2 \in Q^N \ \&\ \sigma \in \Sigma^N \ \&\ \delta(q_1, \sigma) = q_2\} \\ q_0^N &= q \\ Q_m^N &= Q_m \cap Q^N. \end{aligned}$$

In the above definition of Q^N, $|s|$ denotes the *length* of string s. Note also that the marker state set Q_m^N may be empty.

In a multi-agent DES, the plant consists of $n(> 1)$ component agents, where each agent $k(\in \{1, \ldots, n\})$ is modeled by a finite state automaton $\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{k,0}, Q_{k,m})$. The plant model $\mathbf{G}$ is the *synchronous product* (Wonham and Cai, 2018) of the n component agents, written $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m) = \mathbf{G}_1 \| \cdots \| \mathbf{G}_n$. The event set Σ_k of each agent $\mathbf{G}_k$ is partitioned into a controllable subset and an uncontrollable subset, i.e. $\Sigma_k = \Sigma_{k,c} \cup \Sigma_{k,u}$; hence $\Sigma_c = \cup_{k\in\{1,\ldots,n\}} \Sigma_{k,c}$, $\Sigma_u = \cup_{k\in\{1,\ldots,n\}} \Sigma_{k,u}$, and $\Sigma = \cup_{k\in\{1,\ldots,n\}} \Sigma_k$.